Dual, Incorporated Technical Report 9602.024

Final Report

(21 February 1996 - 20 March 1998)

Integrated Data Visualization

and

Virtual Reality Tool

Dated 30 March 1998

Submitted by:
David A. Dryer, Ph.D.
Principal Investigator
Olatokunbo T. Fakinlede
Lead Software Engineer

TABLE OF CONTENTS

I. INTRODUCTION

This draft final report is prepared under NASA Contract NAS5-33215 to document and summarize the results of the entire contract work. This project is a Phase II effort of the Small Business Innovation Research (SBIR) Topic 07.05 of NASA solicitation 94.1. This final report includes sections on project results, conclusions, recommendations, as well as a separate discussion of Phase III commercialization plans. Additionally, there is a system user guide, technical documentation on translator software, and commented copies of source code listings in standard personal computer (PC) floppy disk format. A completed Report Documentation Page (NASA Form 1626) is included as the final page of the report.

The Integrated Data Visualization and Virtual Reality Tool (IDVVRT) Phase II effort was for the design and development of an innovative Data Visualization Environment Tool (DVET) for NASA engineers and scientists, enabling them to visualize complex multi-dimensional and multivariate data in a virtual environment. The objectives of the project were to: (1) demonstrate the transfer and manipulation of standard engineering data in a virtual world (2) demonstrate the effects of design and changes using finite element analysis tools (3) determine the training and engineering design and analysis effectiveness of the visualization system. These objectives were successfully accomplished.

A description of significant events which have occurred in the project follows. The Phase II contract was awarded on 20 February 1996, with Dual, Incorporated's (DUAL's) acknowledgment of receipt on 5 March 1996. Further guidance was received from the Contracting Officer's Technical Representative (COTR) during a 19 March orientation meeting. Principal Investigator duties transitioned from Hank Okraski to David Dryer starting on 6 May 1996, due to Mr. Okraski's promotion to Vice President. Mr. Okraski will provide overall direction and commercialization oversight as program director. A planning conference was held with Clark Atlanta University (CAU) and the University of Central Florida (UCF) on 13 and 14 May 1996 at DUAL's Lake Mary facility. Mr. Dryer visited CAU on 7 Jun 96 to further refine the CAU scope of work with all CAU team members and view CAU test bed resources. Letters were received from both UCF and CAU in Jun 96, confirming that purchase orders from DUAL were in place. Mr. Dryer visited NASA GSFC on 25 - 27 Jun 96 for initial meetings and an assessment of the NASA FEA environment. Dr. Corso visited NASA GSFC on 6 Sep 96 for further observation of the NASA FEA environment. An initial Finite Element Analysis (FEA) and visualization workshop was held at UCF on 11 Oct 96. NASA finite element model (FEM) input files were imported and manipulated in candidate immersive environments at CAU in Oct 96. In Nov 96, a primary software development environment was selected and an initial DVET prototype developed which includes FEM animation functionality. The FEM data translation software, called FEM2VR, was also enhanced to handle limited FEM output data and translation to Virtual Reality Markup Language (VRML) 1.1. Mr. Dryer visited CAU on 20-22 Nov 96 to further assess the CAU test bed and refine the prototyping schedule and direction. An initial DVET prototype was established in Nov 96 at CAU using dVISE by Division, Inc. as the selected development environment. A virtual environment (VE) laboratory was

established at DUAL in Dec 96 and one of its uses will be to develop and manage configuration for DVET software releases. DVET prototype development at CAU and DUAL continued in Jan 97 and an Hyper Text Markup Language (HTML) FEA questionnaire was finalized for use in obtaining industry engineer feedback. In Feb 97, software training was conducted at CAU, prototype development continued at CAU and UCF, and a paper concerning the project was accepted for presentation at the International Training and Education Conference (ITEC) 97. The further development of the CAU testbed DVET prototype was the focus of project work in Mar and Apr 97. The ITEC 97 paper was presented on 22 Apr 97 and was well received. A demonstration of initial DVET testbed functionality at NASA-GSFC occurred on 29 Apr 97. Addressing feedback from the 29 Apr NASA demonstration and transfer of coding effort to DUAL occurred in May 97. DVET code development occurred in Jun 97 to extend the DVET FEM dynamic model architecture and functionality for DVET Prototype Release 1. In July 97, extensive DVET code development continued to prepare for a DVET Prototype Release 1 in Aug 97. Also, a second FEA and visualization workshop was held with industry and academic representation at UCF on 11 July 97. In August 97, DVET Prototype Release 1 was completed and DVET Prototype Release 2 modifications and enhancements were started. A demonstration of DVET Prototype Release 1, with some Release 2 functionality occurred on 29 Aug 97 at CAU and was attended by Tim Carnahan, NASA COTR. Many NASA suggestions for enhancements were incorporated into the DVET software and hardware development plan. Subsequently, in the Sep 97 reporting period, software engineering activities continued to enhance DVET functionality towards Release 2. The DVET experimental plan for usability assessment at CAU was also finalized. In Oct 97, DVET functionality was enhanced towards DVET Release 2, including additional visualization capability, improved data structures, and headtracking device integration. Evaluators for the DVET prototype effectiveness assessment were also identified. In November 1997, a DVET Windows NT (WinNT) System was demonstrated at the DUAL booth in Bldg.2 at NASA Johnson Space Center's Inspection '97 on the 12 - 14th. A meeting was held with Division, Inc. to discuss commercialization and further development of DVET. In December 1997, DVET software and hardware development continued towards DVET Release 2, primarily in the areas of enhancing the FEM2VR module and porting DVET to a SGI system at DUAL. In January 1998, a graphical user interface was developed for the FEM2VR module of WinNT DVET. Visualization of critical FEA boundary conditions has been completed. Users can now toggle selected buttons and either visualize or hide the load and/or the constraint cases. February activities included a final project demonstration at NASA-GSFC attended by Tim Carnahan, NASA COTR, and NASA-GSFC management and engineering personnel. During this presentation, DVET version 2.0 functionality and potential collaborative enhancements were demonstrated. March 1998 activities included refining software documentation, draft final report preparation, and DVET code preparation for delivery.

II. RESULTS

This section will summarize significant project results starting with most recent activities. Whenever possible, monthly report submissions will be referenced where results have already been documented and submitted to NASA.

A. Final Project Demonstration - 17 February 1998

In February 1998, DUAL presented a final demonstration of Integrated Data Visualization and Virtual Reality Tool Phase II SBIR project at NASA-Goddard Space Flight Center (GSFC). The demonstration took place on February 17 from 0900-1200 in the Skybox conference room at Building 28, NASA-GSFC. This demonstration was presented by David Dryer and Ola Fakinlede from DUAL. Key attendees included Hank Okraski, DUAL Senior V.P., Research and Technology; Tim Carnahan, NASA-GSFC Contracting Officer Technical Representative for this project; Mr. Brodeur, NASA-GSFC Code 540; John Decker, NASA-GSFC Code 542; and Bill Hayden, NASA-GSFC Code 542. Other NASA-GSFC attendees included Jeffery Hosler, Code 588; Steve Maher, Code 935; Drew Jones, Code 543; Scott Gordon, Code 542; Debbie Wheeler, Code 542; Sandra Irish, Code 542; and Matt Brandt, Code 588. The demonstration went well. NASA-GSFC personnel obtained a detailed view of this Phase II effort and were able to see and experience DUAL's Data Visualization Environment Tool (DVET) system which was developed under this project. Demonstration briefing slides are attached as Appendix A. The following describes key aspects of this demonstration and associated trip activities.

Demonstration Setup - February 16, 1998. Setup for this demonstration involved configuring two immersive DVET systems and establishing a local area network (LAN) between these systems. The first DVET system was a Windows NT (WinNT) platform with a Polhemus InsideTrak headtracker and a Virtual Research V8 Head Mounted Display (HMD). The second DVET system was a Silicon Graphics (SGI) O2 System with an Ascension Flock of Birds headtracker and a Virtual Research V6 HMD. The LAN was established using a network hub. All demonstration equipment, with the exception of a monitor for the SGI O2 platform was shipped from DUAL to facilitate setup and compatibility between DVET software and immersive demonstration hardware.

Demonstration - February 17, 1998. The following describes significant events and comments during the demonstration. After the project was briefed, WinNT DVET and SGI DVET were demonstrated. Finite Element Models (FEMs) used for the demonstration included NASA's Next Generation Space Telescope FEM and a NASA optical mirror model to ensure the attendees saw DVET used in their engineering domain. Demonstration attendees were all given the chance to try DVET and most did try the system.

The following items relating to DVET functionality were discussed. Users thought the visual filtering of FEM output data with the DVET interactive color scale was an innovative feature. The ability to animate the FEM, while interactively navigating around and inside the model received very positive comments. Also, users were impressed with the crisp 640x480 HMD resolution of the Virtual Research V8. The DVET "floating"

3D toolbox widgets were discussed as opposed to widgets "fixed" to the headtracked viewpoint orientation. Benefits of floating menus include the user's ability to change viewpoint and not have a menu display constantly blocking the view of the FEM. However, slider and other widgets can be harder to select and manipulate due to headtracking movements. On WinNT DVET, the user can address this problem by temporarily disabling headtracking to make fine slider widget adjustments. The blue text coloring was questioned for legibility, but this was only a problem in the projector view of DVET. Users could clearly see text with the Virtual Research V8 and V6 HMDs. The blue text coloring is used in the current DVET color scheme so that text is legible and does not blend in with the black background or any color scale colors. A faster or adjustable "fly mode" speed was requested. A minor problem of one HMD failure was addressed by switching the other "back up" HMD between systems. The final demonstration also included a limited prototype of a collaborative networked DVET running between the two systems, which showed the tremendous potential for DVET collaborative enhancements.

The following were comments on desired directions to take with DVET. The ability to use non-immersive stereo shutter glasses is desirable and should be included in DVET configurations. The integration of other engineering output with FEA output is desirable, including optical paths and thermal output. Audio cues and data sonification are desirable. Comments concerning the limited collaborative DVET demonstration were very positive. The leader-follower ability to go to another user's viewpoint is very useful. A future architecture of collaborative DVET might need a super server SGI "master" linked to multiple remote "slave" subordinate users. The use of web-based JAVA was mentioned as a development environment for collaborative visualization, but JAVA is currently not integrated with virtual environment software development toolkits (SDKs) which have the dynamic functionality that DVET requires.

There were many suggestions on potential Phase III opportunities to further develop and commercialize DVET. John Decker approved submitting an estimate for installing a DVET system within Code 542. Mr. Decker also offered assistance in making contact with Next Generation Space Telescope (NGST) project team members to explore integrating DVET with the NGST design engineering effort. Tim Carnahan and Bill Hayden suggested approaching NASA Langley, who is developing the future design environment. Bill Hayden is involved with an Integrated Synthesis Vision Team at NASA-GSFC which could have future need of immersive tools for high level modeling. The opportunity also existing to develop SBIR topics for enhancements to DVET and these should be looked on favorably, since they are extending existing successful NASA SBIR work.

The demonstration appeared to be well received. DUAL was able to show the result of their NASA Phase II SBIR project and solicit interest from NASA-GSFC in Phase III funding opportunities,

B. DVET FEM Translation Documentation

Documentation concerning the three FEM translators developed as part of this project was completed in Feb 98 and this documentation is attached as Appendix B. The three translators are 1) FEMAP to DXF 2) FEMAP to VRML and 3) From FEMAP to DVET.

The source code concerning these translators is contained in Appendix E, DVET Source Code Listings.

C. DVET System User Guide

A draft DVET System User Guide has been developed and is attached as Appendix C to this report. This system user guide describes the DVET graphical user interfaces used for FEM translation and DVET view, data, and visualization interactions.

D. CAU DVET Software Usability Study

This study was an evaluation of the usability of the Data Visualization Environment Tool (DVET) software. The DVET software presents a graphical representation of a solid object model, permits the user to manipulate the object, as well as, load levels and threshold values. Data were collected from five engineers to assess the usability of the software. The evaluation was performed on the menu tree and the general appearance of the graphical interface. Specific recommendations were made based on the findings. This study is contained in the Jan 98 monthly report.

E. Initial DVET Functionality And Capability Document

An initial description of DVET functionality and capabilities was provided as an information and marketing tool. This document also provided a basis for the draft DVET User's Guide. This document is contained in the Jan 98 monthly report.

F. HCI Questionnaire Input

This is a documentation of key FEA problem areas and suggestions taken from industry engineers (mainly NASA), who have submitted FEA questionnaire feedback. It is contained in the Apr 97 monthly report

G. International Training and Education Conference (ITEC 97) Paper

A paper concerning the project was accepted for presentation at the International Training and Education Conference. This paper, entitled "The Use of Synthetic Environments and Visualization for Finite Element Analysis" is contained in the Feb 97 monthly report.

H. HTML FEA questionnaire

An HTML FEA questionnaire was finalized for use in obtaining industry engineer feedback. This questionnaire is contained in the Jan 97 report.

I. Advanced Material Systems Review

Materials capabilities of NASTRAN and similar FEM application packages were reviewed and design features identified that may be added to the specification of the DVET system. This analysis is contained in the Dec 96 report.

J. DVET Software Evaluation using the Quality Function Description Matrix

An assessment was conducted of the product characteristics of the candidate software for use in the DVET testbed, based on a quantitative weighted list of user requirements. The basic approach used in this study is that of Quality Function Methodology or House

of Quality. This technique is used to assess the key product characteristics against the user requirements. This analysis is contained in the Nov 96 report.

K. HCI Task Analysis Draft Report

This report provided an overview of the major human-computer interaction issues involved in finite element modeling (FEM). An overview of the human-computer interaction issues addressed in this report highlight major concerns within reviewed FEM programs. These issues should be addressed within any modification of the existing programs or any new programs. This analysis is contained in the Sep 96 report.

L. Characterization of Critical Activities in the FEA Validation and Interpretation (V&I) Process

This analysis characterized critical activities in the FEA validation and interpretation (V&I) process. This strawman characterization received further input at GSFC and from other potential DVET users. A description of this initial task analysis characterization is contained in the Jun 96 monthly report.

M. Literature Review

This review and synthesis included the following research areas: general virtual reality, finite element analysis (FE), VE physiological effects, VE simulation & training, virtual prototyping / structural design, VE interaction techniques, scientific visualization, VE human-computer interaction (HCI), human cognition and perception, and virtual and graphical information processing. This review is contained in the Mar 96 through Jul 96 monthly eports.

III. PHASE III COMMERCIALIZATION PLANS

A. Commercialization Strategy

Commercialization opportunities for DVET are being pursued as a two axis strategy. One axis is to win Phase III NASA and other government project funding to install DVET at selected government installations, such as NASA-GSFC, and further tailor and integrate DVET towards specific project needs. The other axis is to develop DVET commercial partnerships and target engineering market areas that are early adopters of promising new technology. Now that DVET has transitioned from a concept to actual software practice, demonstrations of the product are now being conducted for targeted customer leads. The follow sections describe commercialization initiatives by DUAL concerning these two areas.

Government project initiatives

NASA-Goddard Space Flight Center (GSFC)

Tim Carnahan requested during the final demonstration that he would like to see a DVET installation at NASA-GSFC. He commented that DVET can be a vehicle to spur the procurement of VR hardware infrastructure at NASA-GSFC, which is needed to purchase VR software and to fit in with Dan Goldin's vision of using VR to "simulate and visualize our engineering processes in real-time with full interactive control....We can

take it a step further...into a high fidelity...high information content...distributed...virtual environment" (Tool of the Future Presentation, Jan 1998). As previously mentioned, John Decker approved submitting an estimate for installing a DVET system within Code 542, supporting Tim Carnahan's request. It was suggested to implement and support a three month evaluation period at NASA-GSFC as part of this installation. This would enable DVET and engineering virtual reality (VR) in general to receive greater exposure at NASA-GSFC. Mr. Decker also offered assistance in making contact with Next Generation Space Telescope (NGST) project team members to explore integrating DVET with the NGST design engineering effort. Through NASA-GSFC management points of contact, DUAL is attempting to demonstrate DVET to NGST project members and explore how DVET can address NGST engineering project needs.

NASA-Johnson Space Center (JSC)

As part of NASA Inspection '97 at NASA-JSC, DUAL made initial contact with members of the Integrated Design Environment (IDE). This advanced programs project has goals of creating an integrated collaborative design environment which makes use of advanced visualization and user interaction techniques. A letter was sent to IDE team members by Hank Okraski, DVET project director on 18 November 1997, forwarding information on DUAL's Data Visualization Environment Tool (DVET) and requested further dialog concerning the possibilities of joining the Integrated Design Environment (IDE) team. Unfortunately, it appears from the NASA-JSC response that this project's funding has been delayed for this year:

> Dear Mr. Okraski;
> I appreciate your interest in our HEDS IDE project. However it appears that the IDE project we spoke about is not going to be funded this year as cost overruns from the International Space Station project are consuming advanced programs funding. However if this situation changes I will contact you and schedule a demonstration of your capabilities.
> Regards,
>
> David Fletcher
> Engineer - NASA/JSC
> Advanced Development Office - EX2
> (281) 244-5136 (Phone)
> (281) 244-7478 (FAX)

However, DUAL will maintain contact with IDE team members and continue to pursue this opportunity when project funding becomes available.

NASA-Marshall Space Flight Center (MSFC)

Dual and Division, Inc. are teaming to demonstrate DVET to NASA-MSFC personnel involved in the High Fidelity Simulation of a Typical Multidisciplinary Device Project. Project members have requested a demonstration of DVET/Division software in Mar 98. This project's goals are to develop a capability for a "cutting-edge" high fidelity, cross-discipline, real-time simulation for integrated design concepts. The project wants to develop or use a simulation architecture which enables dynamic visualization of simulation model output, including thermal and stress outputs. Also desired is the

ability to navigate "inside" the visualizations. DVET can address these needs. Joe Hale, NASA-MSFC has been the main POC for this demonstration coordination.

NASA-Langley Research Center (LRC)

DUAL is planning to approach NASA-LRC as suggested by Tim Carnahan and Bill Hayden. Langley is developing a future design environment for NASA which could potentially make use of DVET immersive engineering concepts. Other relevant concepts being investigated are the Smart Assembler, which uses SGI inventor format and uses intelligent synthesis. DUAL plans to obtain NASA-LRC future design environment points of contact and obtain a letter of introduction from Bill Hayden for appropriate Langley personnel.

Navy/Army Ballistic Effects

Potential extensions of DVET functionality for live fire testing and training are being pursued with U.S. Army and U.S. Navy agencies. Tim Carnahan submitted a letter stating the opportunity for live fire testing technology transfer of DVET to Mr. James O'Bryon, Office of the Secretary of Defense, Operational Test and Evaluation, Live Fire Testing, which is attached as Appendix D. Also, the U.S. Navy has requested a description of this concept for potential project integration or SBIR topic use. This Navy concept description is shown below:

> TITLE: Visualization of Weapons effects for Training and Test and Evaluation
>
> OBJECTIVE: Develop a low cost system enabling visualization of weapons effects for training, and test and evaluation activities.
>
> DESCRIPTION: There is a need within the Navy and DOD to develop tools with which to model, present, and visualize weapons effects. The ability to navigate around, in and through simulated target combat systems would afford engineers a unique and powerful analysis tool. Coupled with physics-based models, visualization in this manner would provide the user with accurate, perceptible displays of information not readily understood in raw data format.
>
> In distributed training and simulation, battlefield casualty predictions are often based upon a probability of weapon hit and kill, assigned based upon various forms of empirical data or heuristics. Using visualization, this raw data could be converted to provide meaningful feedback to planners and warfighter personnel for better understanding of system effectiveness concerning tactics and weaponry.
>
> The Navy desires to develop an ability for a person to observe, from any aspect angle, the effects of a weapon impact on a ship, aircraft, or land based target. The visualization can be outside or within the target as desired by the individual. In this manner a person can observe the penetration of the weapon, the fragmentation, stress and heat distribution, and be able to assess the damage to the equipment and personnel.

Both Army and Navy live fire testing opportunities are being pursued and could result in funded work to extend DVET.

Commercial initiatives

Southern Technology Applications Center (STAC)

In Feb 98, DUAL met with Dave Sapuppo, Area Director with STAC. STAC has capability and experience in transferring federal, university, and private sector technologies to successful commercial products. Dave was given a briefing and demonstration of DVET and made the following significant comments during the meeting.

- It is typical for slow progress in commercialization of innovative technology until the product is able to be demonstrated or "reduced to practice." Then commercialization initiatives advance faster due to the customer's ability to see and evaluate the product in their own domain. It appears that DVET is just now being "reduced to practice" and is at a critical juncture for commercialization.
- Timing is everything. Avoid "hyping" the product before the product is ready. Otherwise, the product will disappoint and not meet expectations.
- Industry is looking for "faster, better, cheaper" engineering solutions and asks what is their return on investment (ROE) for purchasing tools, such as DVET.
- STAC can provide services for commercial product assessment of DVET, including target customer feedback from a good sampling of market segment and reaction to price points for product configurations.
- STAC can also help in identifying target users. One potential market segment is the rapid prototype service business who are typically "early adopters" of innovative tools to sell their service and more effectively conduct service tasks, such as FEA.

STAC is going to provide an estimate on commercialization services. DUAL's priorities for commercialization services include identifying market segments and target users which have good potential for using DVET and obtaining potential customer feedback on DVET in terms of strengths, weaknesses, and desired improvements.

Academic Institutions

Both the University of Central Florida (UCF) and Clark-Atlanta University (CAU) have shown interest in further use of DVET for educational use. DUAL will investigate options with UCF and CAU, if desired, for installation and support of commercial DVET at these universities for integration into FEA curricula. A cost effective strategy for these institutions will be pursued as collaborative team members, with discounts for academic pricing.

Division, Inc.

After a DVET demonstration and meeting with Division, Inc. in Nov 97, there was strong interest from Division sales personnel, including Will Siembor, Vice President, Eastern Area to explore commercialization of DVET FEM translators for use in Division releases and in potential teaming with Ansys, Inc. on Virtual FEM visualization. Division and Ansys could give DUAL access to the automotive design industry, where both companies are established. DUAL has supplied an initial video to Alan Barclay, Southeast Regional Sales Manager and is planning a DVET non-immersive

demonstration to include in Division's software promotions. A meeting with DUAL, Division and Ansys is still being pursued to discuss potential partnerships.

Other FEA-related Software Companies

DUAL is investigating relationships with other established FEA companies (e.g., MSC) to use DVET as an immersive FEA module. Also, DUAL is investigating licensing and partnering issues with Enterprise Software, Inc. - makers of FEMAP software, since DVET currently uses the FEMAP neutral file format for FEM translation. The licensing of FEMAP converters from other FEM formats to FEMAP neutral file format is being investigated as well as partnering to obtain early release information on future file format changes.

An early Phase II initiative to team with Engineering Animation, Inc. (EAI) which showed early promise was not realized. Apparently, EAI did not want to conduct immersive engineering development as an early enabler of this technology without proven commercial success.

B. DVET Product Dissemination Activities

Phase II DVET Dissemination Activites

An incremental strategy of disseminating DVET concepts and demonstrating DVET functionality was pursued during Phase II with the following visibility highlights:

International Training and Education Conference (ITEC) 97 - Apr 97

A paper concerning the project was accepted for presentation at the International Training and Education Conference (ITEC) 97. The ITEC 97 paper was presented on 22 Apr 97 and was well received. This presentation to a simulation and engineering audience gave the project some international exposure.

Paris Air Show - May 97

As part of a separately funded DUAL marketing plan, DVET Version 1.0 was demonstrated at the Paris Air Show in May 97, along with DUAL aerospace-related technologies. As a result of this exposure, DUAL received some further requests for information concerning DVET development. These contacts were relayed to the DUAL project director and principal investigator.

UCF FEA Visualization Short Courses -

Two short course workshops were held which taught FEA and VE basics. The target participants in these short courses were industry designers who were not experts in FE. The intent of these workshops was to introduce DVET concepts and receive industry feedback on these concepts and DVET prototypes. The first short course was held on 11 Oct 96 which covered DVET design concepts, but could not include an actual DVET demonstration. The second workshop on finite element visualization using VR was held on 11 Jul 97. The workshop was attended by seven engineers from industry and academia, including representation from the United Space Alliance, Westinghouse Power Generation, and UCF faculty. The current prototype of DVET on the DUAL Windows NT platform was demonstrated and used by workshop participants in a non-immersive mode. Also, static FEM models were translated into DXF format and

immersively presented using Sense8 WorldToolKit on Windows NT with DUAL's Virtual Research V6 Head Mounted Display and Polhemus InsideTrak headtracker. Comments from participants in both short courses were very positive, with all participants rating the potential of virtual environments in FEA as high. Feedback on DVET from these industry and academic engineers was used throughout the Phase II prototype development process

NASA Inspection '97 - Nov 97

During NASA's Inspection '97 (12-14 Nov 97), DUAL showcased the Integrated Data Visualization and Virtual Reality Tool project in Bldg. 2 at Johnson Space Center. Building 2 contained technology demonstrations from other NASA centers, including GSFC, and associated contractors. DVET on the WinNT platform was immersively demonstrated and received great interest over the three days. Don Friedman, GSFC SBIR Office, was able to see the system during this period.

Planned Phase III DVET Marketing Outlets

The following marketing outlets are planned as part of Phase III commercialization.

A DUAL DVET internet site, linked through the DUAL home page and other appropriate agencies and search engines, will provide the latest information on DVET releases, features, and contact information for ordering and pricing inquiries.

Hank Okraski, DUAL's project director for this SBIR, is also Chairman of the Board of Directors for the National Center for Simulation. DUAL will make the DVET product known to members of this organization, which contains about 100 members from industry, academia, and government.

STAC has the ability to maintain and disseminate DVET information as part of their federal technology transfer mission. STAC marketing techniques include electronic and paper media.

For qualified opportunities, DVET systems will be shown at engineering-related demonstrations, including engineering trade shows and research venues.

C. DVET Return on Investment (ROE) Justifications

Potential industry and government customers of DVET are concerned primarily with where the return on investment (ROE) is in using DVET over existing FEA visualization systems. The following list provides some initial rationale for ROE using DVET version 2.0:

- Ease of use - In user testing at CAU, all participants found the software to be relatively easy to learn and use. The toolbox widgets were simple to understand and were easy to manipulation with the mouse interface.
- More efficient detection of critical FEM output in complex models - In user testing at CAU, all participants were able to identify critical FEA output regions directly on the geometry with very little difficulty. This is due in part to the interactive set of visualization tools in DVET, which include a unique visual filtering capability. Users can visually filter out non-critical node and element geometry and just focus on critical element locations and associated textual information.

- More efficient, intuitive navigation - Due to the combination of orientation headtracking and mouse navigation, the user is able to more efficiently conduct exploratory navigation of FEMs. Natural headtracked orientation replaces clumsy hand manipulation tasks involved in orientation adjustments with non-immersive systems.
- Integrated FEM information display - DVET is an integrated information visualization environment with direct manipulation of FEM geometry and direct recall of associated output textual information. This reduces the analyst's cognitive load when compare to current analysis systems which force the user to switch between separate 2D text and non-immersive 3D graphic screens.
- Potential for collaborative environments - DVET was designed with future collaborative functionality in mind. While already containing limited Local Area Network and Wide Area Network functionality, future versions will enable more dynamic interactions to be accessible and shared between remote locations.

D. Initial DVET Product Configurations and Pricing Strategy

Commercial pricing of a hardware/software system configuration, similar to WinNT DVET prototype is projected to be $40K, with computer workstation. Commercial pricing of software-only system is projected to be approximately $10K. Initial DVET product options are contained in the matrix below and can be configured for customer needs. These configurations will be revised based on customer needs and immersive VR technology advancements.

DVET Configuration Type	Typical CPU Requirements	Peripherals
High End (Immersive/Non-immersive)	SGI Octane & above WinNT Intergraph Workstation	Virtual Research V8 HMD Polhemus InsideTrak (WinNT)/Ascension Flock of Birds (SGI) Crystal Eyes Stereo Glasses
Medium (Immersive/Non-immersive)	SGI O2 WinNT w/ OpenGL Graphics Card	Virtual Research V6 HMD Polhemus InsideTrak (WinNT)/Ascension Flock of Birds (SGI) Crystal Eyes Stereo Glasses
Low End (Non-Immersive)	SGI O2 WinNT w/ OpenGL Graphics Card	Crystal Eyes Stereo Glasses

Instructional/ Custom DVET FEM visualizations (CD Rom)	Multimedia SGI/PC w/ CD ROM	

E. Potential Marketing Channels

Potential distribution channels for DVET include one or more of the following options:

- DUAL directly marketing DVET to customers through an internal or outsourced software marketing and sales organization. This marketing could also take the form of online internet marketing and distribution with appropriate security protections.
- DUAL partnering with an established FEA or other software company and using that company's software marketing and sales organization.
- DUAL licensing DVET software technology to established FEA or other software companies to integrated into their existing engineering software products.

The most viable configuration of marketing channels will be established by DUAL based on establishing strategic partnerships with interested government and commercial organizations.

IV. CONCLUSIONS

The following conclusions are a combination of project feedback from the Contracting Officer Technical Representative (COTR), Tim Carnahan during and after the Feb 98 final demonstration and assessments of project activities by DUAL team members. Overall, the project was successful in achieving its objectives. The objectives of the project were to: (1) demonstrate the transfer and manipulation of standard engineering data in a virtual world (2) demonstrate the effects of design and changes using finite element analysis tools (3) determine the training and engineering design and analysis effectiveness of the visualization system. As previously stated, these objectives were successfully accomplished. Through the development of FEM translators, standard FEM data was translated into static and dynamic virtual environment objects. Through development of a Data Visualization Environment Tool (DVET) and integration of FEM translation software, FEM input and output data was able to be immersively manipulated in the virtual world. This virtual world manipulation included viewpoint, data, and visualization interactions which provided an intuitive engineering analysis tool for FEM interpretation and validation tasks. Through the use of interactive color scale and geometry widgets, as well as dynamic animation techniques, the effects of design and changes using finite element models were demonstrated. Finally, using human-computer interaction analysis and user testing, the training and engineering design and analysis effectiveness of the DVET visualization system was assessed.

This project was intended to be an software investigation and development of virtual reality (VR) tools for engineering analysis and was not designed to improve or enhance virtual environment hardware peripherals. One drawback of this software development focus was having to deal with and adapt to the current generation of virtual reality

devices. In the view of project team members and the COTR, the development of cost effective VR hardware peripherals is lagging behind in areas including head mounted displays and head tracking devices. This, in turn, slows down customer willingness to buy into VR systems due to some high cost hardware items. It is hoped that cost effective, high resolution VR displays are on the horizon and that organizations, such as NASA invest more in VR hardware infrastructure and research upon seeing the immersive benefits of systems such as DVET.

For DUAL, this Phase II SBIR has been the foundation of a research and development initiative in visualization and virtual reality, which has already brought dividends in terms of subsequent Phase I SBIR awards and other directly funded activities. DUAL looks forward to entering into Phase III commercialization of products developed under this contract and further research partnerships with NASA.

V. RECOMMENDATIONS

The following recommendations are summarized from this project's final demonstration feedback and commercialization analysis at DUAL.

DUAL will submit an estimate to NASA-GSFC to install the DVET system for evaluation and exposure of NASA engineers to an engineering VR tool. DUAL will propose to provide technical support and FEM model testing services to NASA to further test DVET with NASA models and help ensure successful user interactions with the system.

DUAL will continue work with VR hardware peripheral manufactures to help drive the development of cost effective VR equipment, especially in the area of HMDs. HMD requirements for engineering applications which surfaced during this project included lighter weight, lower cost, and ability to change the opacity of the view to "see through" to the real world at selected times during task performance.

Project members recommend that NASA build up their immersive VR infrastructure, so the NASA director's vision of immersive engineering design can start being realized.

Since this project resulted in a successful implementation of an immersive engineering tool, it is recommended that DVET be integrated into one or more NASA integrated design projects. Final demonstration feedback reinforced this recommendation for DUAL (and advocates of DUAL's technology at NASA) to aggressively market DVET for such a project.

In terms of highest payoff industry commercialization, NASA and other external sources have recommended that DUAL continue dialog with FEA software vendors for potential partnerships and further develop and refine DVET as a commercial product line for industry and academia.

APPENDIX A -

DVET FINAL DEMONSTRATION BRIEFING SLIDES

Integrated Data Visualization and Virtual Reality Tool Project - SBIR Phase II

NASA Final Demonstration
17 February 1998

February 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

Integrated Data Visualization and Virtual Reality Tool Demonstration SBIR Phase II 17 February 1998

TIM CARNAHAN INTRODUCTION TO DUAL SBIR	9:00 - 9:05
NTEGRATED DATA VISUALIZATION AND VIRTUAL REALITY TOOL PROJECT OVERVIEW	9:05 - 9:15
DATA VISUALIZATION ENVIRONMENT TOOL (DVET) INTRODUCTION	9:15 - 9:30
WIN NT DVET DEMONSTRATION: SOLID BEAM STATIC FEM EXAMPLE OPTICAL MIRROR STATIC FEM EXAMPLE	9:30 - 10:15
DVET HANDS ON/BREAK	10:15 -10:45
SGI DVET DEMONSTRATION: NEXT GENERATION SPACE TELESCOPE (NGST) DYNAMIC FEM EXAMPLE DVET COLLABORATIVE NETWORKING PROTOTYPE DEMONSTRATION	10:45 -11:15
DVET HANDS ON	11:15 -11:45
FOLLOW ON PROJECT/COMMERCIALIZATION OPPORTUNITIES	11:45 -12:00

February 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

Demonstration Objectives

- Project Summary
- Demonstration of Dual Inc's Data Visualization Environment Tool (DVET) 2.0
- Hands On Sessions with DVET SGI and WinNT Systems
- Commercialization Opportunities and Discussion
- Obtain NASA Feedback

Project Objectives

- Demonstrate the transfer and manipulation of standard engineering data in a virtual world
- Demonstrate the effects of design and changes using finite element analysis tools
- Determine the training and engineering design and analysis effectiveness of the visualization system

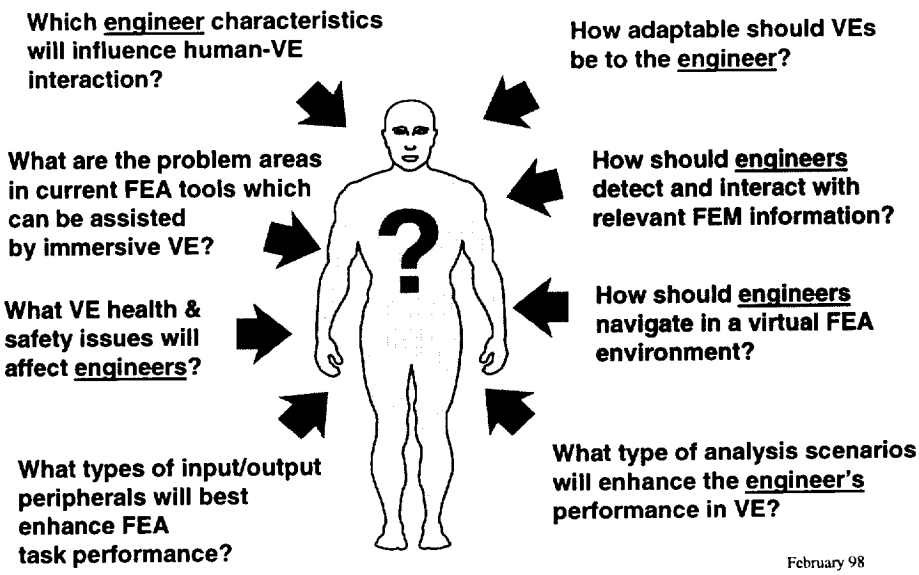
Project Status

- End of 24 month Phase II SBIR
- Preparing final draft report due 4 March 98
- DVET deliverable due 20 March 98
- Demonstrating DVET 2.0 System and pursuing follow-on NASA project and commercialization opportunities

Cross Platform VR Technology Strategy

- Computing Hardware
  - DUAL
    - PC Pentium Win NT workstations
    - SGI O2 loaned workstation
    - Plans for SGI Octane to further pursue Phase III opportunities
  - CAU
    - SGI Crimson Reality Engine
    - Other SGI workstations
- VR Hardware
  - DUAL
    - Virtual Research V6 HMD
    - Virtual Research V8 HMD
    - Polhemus InsideTrak Headtracker
    - Ascension Flock of Birds
  - CAU
    - Ascension Flock of Birds
    - Dataglove

Issues in Applying VE to FEA



February 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

Requirements Analysis

- Visualization
  - Dynamic Representation
  - Filtering for Entity of Interest
  - Annotation
- Navigation
  - Locational Metaphors
  - Path Definitions
- User Interaction
  - Multiple User Roles
  - Functional Metaphors

February 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

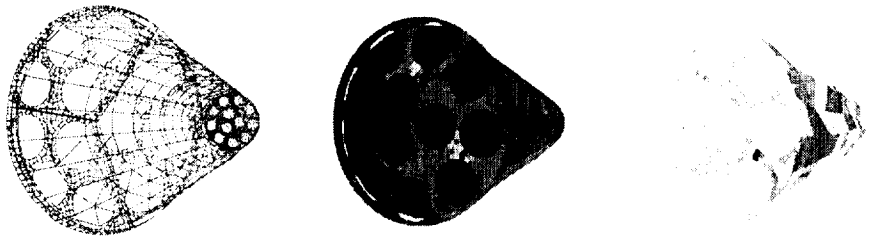
Initial Prototype

- Visualization
  - Stress based color mapped static DXF geometry
  - Displacement based deformed DXF geometries
  - Static colormap
- Navigation
  - Orbital scan mode
  - Independent user fly mode
- User Interaction
  - Immersive Tool Box (GUI)
  - No data filtering

DVET Version 1.0

- Visualization
  - Vertex coloring based on NASTRAN stress data
  - Displacement based dynamic geometry
- Navigation
  - Orbital scan mode
  - Independent user fly mode
- User Interaction
  - User of Immersive Toolbox (GUI)
  - Sliders for data filtering
  - Safety Margin and Load Level

Critical FEA Information Detection



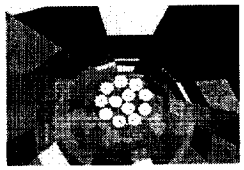
- FEM Data Rendering
  - Colored Mesh Detection
  - Geometry Detection
- Use of Transparency
  - Highlight Relevant "Domain Semantics"
  - Reduces Visual Clutter

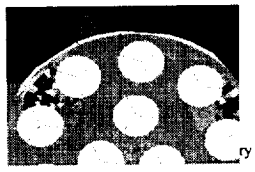
February 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

FEA VE Navigational Modes

- Types
  - Model Navigation
  - Analysis Process
- User Control
  - Passive Grand Tour
  - Guided Tour
  - Total User Control





ry 98

Dual Incorporated, Clark Atlanta University, University of Central Florida

WIN NT DVET DEMONSTRATION

- SOLID BEAM STATIC FEM EXAMPLE
  - Views/Navigation
  - Data Manipulation
  - Visualization
- NASA OPTICAL MIRROR STATIC FEM EXAMPLE

DVET 2.0 Views/Navigation

- Interactive horizontal and vertical orbit of FEM
- Preset Home, Left, Right, Back, and Isometric Viewpoints
- Navigation modes of fly (default) and hyper/instantaneous modes
- Navigation to a close offset distance from the currently selected FEM node
- User-defined viewpoints
- Independent user fly mode

DVET 2.0 Data Manipulation

- "On the fly" switching between 5 Nodal and 5 Elemental Sets
- Load Cases
- Constraints
- FEMAP neutral file to DVET
- FEMAP neutral file to DXF (separate program)
- FEMAP neutral file to VRML 1.0 (separate program)

DVET 2.0 Visualization

- Load Percentage slider control
- Geometry Exaggeration slider control
- Switchable Data Text Display
- Disable/enable headtracking "on the fly" (WinNT)
- Switchable Mesh display
- Dynamic and Static Mesh modes
- Animation of FEM deformations/FEA output data
- Animation Sawtooth and Ramp modes

DVET 2.0 Visualization (cont)

- Dynamic FEM element or node coloring based on FEA static or modal output data
- Dynamic FEM node geometry based on FEA output displacement data
- Interactive color scale for output data
- Extract FEA element/node data using interactive pointing
- Color Scale Threshold slider

Interaction peripheral support

- Immersive Headmounted Display (HMD) headtracking orientation and mouse cursor translation navigation on SGI and WinNT platform
- Non-immersive 2D mouse navigation interface on SGI and WinNT platforms
- Assessments of dataglove and joystick interactions conducted

- NASA Optical Mirror Static FEM Example

- DVET Hands On/Break

- SGI DVET DEMONSTRATION:
  - NEXT GENERATION SPACE TELESCOPE (NGST) DYNAMIC FEM EXAMPLE

DVET Collaborative Networking Initial Prototype Demonstration

- Limited prototype to show DVET potential in this area

- Multiple user manipulation of
  - Immersive tools
  - Data probe
  - Viewpoints

- Guided wayfinding and data presentation

- Potential for FEA collaborative analysis and guided training

Key Suggested Phase III DVET Extensions

- Stereoscopic viewing
- Collaborative networked DVET over LAN/WAN
- Voice recognition for toolbox commands
- Data sonification for data probe nodal values
- Integration of FEM with CAD, thermal, and optical model outputs
- Extend FEM data import capability
- Enhanced additive FEM output (e.g. dx, dy, dz by g factors)
- FEM model displacements influencing CAD model displacements

Follow On Project/Commercialization Opportunities

- Commercial
  - Strong interest from Division, Inc. in developing teaming with Ansys
  - Possible relationship with Enterprise (FEMAP)
  - Pursuing relationships with established FEA companies (e.g., MSC) to use DVET as immersive FEA module
- Government
  - Demo request from NASA Marshall Integrated Simulation Design Project
  - Interest from NASA Integrated Design Environment (IDE) - (Funding reduced)
  - NASA-GSFC Opportunities?
  - Pursuing Navy/Army Ballistic Effects DVET extension

Discussion/Wrap Up

- Feedback on NASA-GSFC Management Interest for DVET
- VE Visualization Potential
  - Dimensional Richness of 3D Attributes
  - Interactive, Multi-Sensory Experiential Benefits
  - Hardware Performance/Cost Increase
- Engineering VE Is Evolving
  - Issues Include Visual HMD Resolution, Cost, Comfort of Peripherals, Safety, & User Acceptance
  - Potential of Improved Information Processing and Interface Navigation for Analysis
  - Commercialization Options (Phase III Project Use) for DVET

APPENDIX B -

DVET FEM TRANSLATION DOCUMENTATION

INTRODUCTION

This document highlights the task formally presented to UCF, regarding DVET. The major task for UCF has been to write translators converting finite element data, both input and output, to formats which can be read by virtual reality systems. We refer to these translators as FEM2VR. The responsibility for using, and as needed writing or modifying, virtual reality visualization software and dealing with the human-computer interface has resided with Dual Inc and Clark Atlanta University.

The rationale for this project is very simple. The response of complex structures modeled by finite elements is difficult to cope with, in large part because the visual field is so cluttered. Owing to its interactive, manipulative, and immersive nature, virtual reality software provides the analyst and designer performing finite element analysis with critical additional degrees of freedom, including the position and orientation of the viewpoint, lighting, textures, etc. Furthermore, it allows elements to be "turned off" in response to queries, to call the analyst's attention to the most important information such as elements with high stresses. We believe that a prototype of system showing such benefits has been developed in this project.

SUMMARY OF ACCOMPLISHMENTS AND POTENTIAL FUTURE WORK

The source code has been transferred to DUAL on an ongoing basis throughout the project. Three translators written by UCF are detailed below:

From FEMAP to DXF

From FEMAP to VRML

From FEMAP to Dvise.

A number of large structural models were obtained from GSFC and successfully visualized. They included a satellite model (HESI), an instrument package (MOLA), and a space telescope model (NGST). Two workshops were held at UCF to introduce the evolving system to industry and government engineering, with favorable reviews from the participants.

In the first section below, the translators are described and documented, in addition to comments incorporated in the codes.

FEMAP is a commercial software system, used at GSFC, which can serve as a pre- and post-processor for a large number of commercial finite element code, and its file structure is well designed. Comparable systems include PATRAN, HYPERMESH, I-DEAS, GEOSTAR and others. Unfortunately, finite element data structures are not standardized. However, thanks to their common origin in NASTRAN, they are usually very similar. It is believed that the current translators can be extended with modest effort to other FEM IO systems.

One hope that we had throughout is that it would enable FEM visualization via the Internet. For this purpose we developed a translator to VRML Ver 2. Unfortunately, the massive amounts of data involved,

the file size restrictions of current browsers and the immaturity of the VRML standard do not yet permit Internet visualization.

We believe that the translators have some commercial potential, and would have much more potential if extended to more FEM IO systems. We likewise believe that the DVET visualization tool , incorporating the translators and the virtual reality system based on Dvise, has commercial potential. Currently, this potential is greatest on workstation platforms such as SGI computers. With rapid advances in technology, it should likewise have great potential on microcomputer platforms.

DOCUMENTATION OF TRANSLATORS

The paragraphs below document three translators developed at UCF as required by the contract with DUAL. The source code has been separately transmitted to DUAL on an ongoing basis throughout the period of performance, and is extensively commented.

Three codes NEU_DVISE.CPP, NEU_DXF.CPP, and NEUDXFVRML.CPP have been developed to convert the FEM data (including both input and output data) from FEMAP neutral file format to DVISE, DXF, and VRML respectively. All these codes are written in C language. In the code, several data arrays are generated to store FEM model information such as nodal coordinates, the connectivity table for elements, loads, the boundary conditions, as well as output data including various stresses, strains, nodal forces, etc. In general, we use several simple two-dimensional triangular elements and quadrilateral elements to represent complicated solid elements such as brick elements, wedge elements, etc. In order to save memory and enhance rendering speed, internal surfaces between solid elements are filtered under the user's option. In DXF file format, 3DFACE elements are used for shell/plate elements and line elements are used for beam/rod/bar elements. In VRML format, commands are used to represent the shell/plate elements and beam/bar/rod elements. In the DXF format, colors are used to describe the FEM output such as stress values. The colors are associated with layers and layers are attributes of elements. Only a very limited number of colors are available in the DXF file format. It is almost impossible to make color in the FEM output change smoothly across elements. In VRML format, the color of an element is composed of three quantities in the diffusion color array. These quantities can be changed continuously in the range of [0,1]. In VRML, the FEM output can be displayed more smoothly across elements than that in DXF format. Because of memory size limitation, most current available VRML viewers still are unable to handle large FEM models with thousands of polygons. In Dvise, colors can be dynamically interpolated in each element. Much better visualization can be realized in Dvise using its very powerful render engine.

The converters for DXF and VRML are pretty straightforward. Some comments are already written in the corresponding C codes. So, they will not be discussed further here.

The key task for the converter NEU_DVISE is correctly reading FEMAP neutral file and properly arranging and storing the data in memory for later visualization. It is necessary for the C code to be reliable for different FEM models, different type analysis, different element types, load and boundary condition. The current code consists of three different major parts as well as a simple user interface.

The simple user interface has been upgraded to a graphical user interface. The GUI allows you to enter the name of the FEMAP file that is to be modeled. There is also a sequence of more prompt windows the request information regarding whether or not to load into memory constraint and load information, the

various nodal and elemental output data sets can also be selected, including the mode that should be analyzed. A prompt also is displayed asking whether or not to filter the internal surface of the model.

The first part is designed to read data from the FEMAP neutral file. FEMAP neutral file format consists of various data blocks. Each data block has a special ID number. In ASCII format, each data block starts and ends by "-1". Actually only a few data blocks are of interest for our purpose. They are the title block, the node information block, the element information block, the load information block, the boundary condition block, the element type block, the material block, and the output data block. In the current code, the FEMAP neutral file is read twice. The first time is to check the file, to count the numbers of node, elements, load sets, etc. and to record the pointers to important data blocks. With the information, appropriate memory addresses can be allocated for various data arrays such as element array, node array, load array, etc. The second time, the code does not read the whole FEMAP neutral file. It only reads the necessary information for visualization purposes by directly starting from the pointers recorded in the first time reading.

The second part of the code filters the internal surfaces from the solid elements. In current VR systems, a solid object is represented by its surfaces. So we decompose the solid elements into several surfaces. As an example, a 3D brick element can be decomposed into six shell elements to represent its six surfaces. For a complicated FEM model, there are several thousand solid elements. For this decomposition, the number of surface elements will be increased by a factor of six. Even a very powerful computer is still very challenged to handle such a huge numbers of polygons. We set up an option switch. Let the user decide if the internal surfaces between solid elements are retained or not. The internal surfaces can be filtered and only the elements on the exterior surfaces of the structures or bodies are left when the filter switch is on. The algorithm for searching exterior surface elements is based on a very simple fact that an exterior element does not share all three (triangle) or four (quadrilateral) nodes with any other element. So we just match elements to see if they share the same nodes. If two elements have the same three or four nodes, we can set a flag to one, meaning internal surface elements.

In dVISE, a line is invisible. So for beam/bar/rod elements, we artificially displace both end nodes with very small displacements and generate two extra nodes. In this way, we create a four node element for each 1D element and make it visible in the dVISE system.

The third part is to sort the data into arrays for convenience for later usage in dVISE and to output the data to temporary files for debugging. In the current code, the following data arrays are created.

```
struct NODE_DATA
        {
        long int A;
        double x;
        double y;
        double z;
        double dx;
        double dy;
        double dz;
        double output_data[5];
        int H;
        }
```

In this *nodal data structure*, the long integer A is the ID number of the node in the original FEMAP model. x, y, z, dx, dy, and dz are its global coordinates and displacements. Output_data are quantities from FEM analysis results. Here displacements and five nodal output_data are case dependent quantities reflecting user selection. The integer H is a flag to indicate if it is an internal node in a solid model.

```
struct ELEMENT_DATA
        {
        long int A;
        long int B[4];
        double C[5];
        int D;
        int E;
        }
```

In this *element data structure*, the long integer A is the ID number of the element in the original FEMAP model. Long integer data array B stores the four nodal ID numbers of the element. The real number data array C is used to store five sets of elemental output data. Integer D indicates an element property number. The flag E is an index. It will be zero for an exterior element and one for an internal element in a solid model.

```
struct ELEMENT_PROPERTY
{
int A;
int H;
double B[100];
}
```

In this *element property data structure*, integer A is an index for element type such as a triangle element or a beam, etc. Integer H is an index for the material ID number. The real number data array B stores the properties associated with this element type. The meaning of each data item is defined following the definitions in FEMAP manual.

```
struct MATERIAL
{
int A;
char title[25];
double Young_Modulus[3];
double Shear_Modulus[3];
double Poisson_Ratio[3];
double GMatrix[21];
double alpha[6];
double k[6];
double thermal_cap, density, damping, temperature;
double tension_limit[2];
double comp_limit[2];
double shear_limit;
}
```

The *material data structure* consists of the title and the mechanical and thermal properties of the material.

```
struct CONSTRAINT
{
int A;
char B[25];
long int NUM;
fpos_t file_constraint;
long int *ID;
```

```
int *INDEX;
}
```

The *constraint data structure* is different from nodal data or element data. The constraint is case dependent. The same models may have different constraint sets. In this data array, integer A is the ID number. B is string for name of the constraint. The long integer NUM indicates the total number of nodes associated with this constraint set. The pointer file_constraint points to the start point of the constraint data in the FEMAP neutral file. The long integer pointer ID is used for dynamic allocation of memory to store the ID numbers of nodes associated with this constraint set, while the integer pointer INDEX is used for dynamic allocation of memory for the constraint type index corresponding to the nodes.

```
struct COORDINATE
{
int A;
int B;
int C;
char D[25];
double E[3];
double F[3];
}
```

The *coordinate system data structure* consists of the ID number A (in the original FEMAP file), the ID number of the coordinate in which the coordinate system is defined, the index C for the coordinate type, the string D for name of the coordinate system, and real number arrays for the origin and rotation related to the coordinate system in which it is defined.

```
struct LOAD
{
int SET_ID;
char NAME[25];
fpos_t load_file, nt_file, et_file;
long int NUM,NT_NUM,ET_NUM;
long int *ID,*NT_ID,*ET_ID;
int *TYPE,*FACE;
double *VALUE,*NT_VALUE,*ET_VALUE;
}
```

In this *load data structure*, the SET_ID is the ID number of the load set. The string NAME is the name. Load_file, nt_file, and et_file are pointers for the data in the FEMAP neutral file. Load_file points to the load start point of vectorial mechanical and thermal load data (except temperature), nt_file to scalar nodal temperature, and et_file to scalar elemental temperature in the FEMAP neutral file. NUM, NT_NUM, and ET_NUM are the total numbers of vectorial loads, scalar nodal loads and elemental temperatures respectively. Pointers ID, NT_ID, and ET_ID point to the locations for storage of nodal IDs for vector loads, scalar nodal temperature, and elemental temperature loads. For vectorial mechanical and thermal loads, the pointers TYPE and FACE point the locations where the index of the load type and face of element or direction of node in which the load is imposed are stored. Finally the pointers VALUE, NT_VALUE, and ET_VALUE point to the locations where the load value, nodal temperature, and elemental temperature are stored in dynamic memory.

APPENDIX C -

DVET SYSTEM USER GUIDE

Data Visualization Environment Tool

Users Guide

Introduction

This document is intended to provide a detailed explanation of the usage of the functions and buttons associated with the Data Visualization Environment Tool (DVET). A step by step walkthrough will be shown accompanied with the necessary graphics. The documents will be divided into sections:

1. Graphical User Interface
2. View Button
3. Data Button
4. Visual Button

Graphical User Interface

The Graphical User Interface was developed for the purpose of making it easy to input the various information into DVET that was required for it's operation.

1. The first of such prompts requests you provide the name of the FEMAP file to be modeled. The file that read in must be in the local directory. If the file entered does not exist the prompt will be re-displayed until an existing file is entered. You may exit from the system if you choose by selecting the cancel button on this prompt.

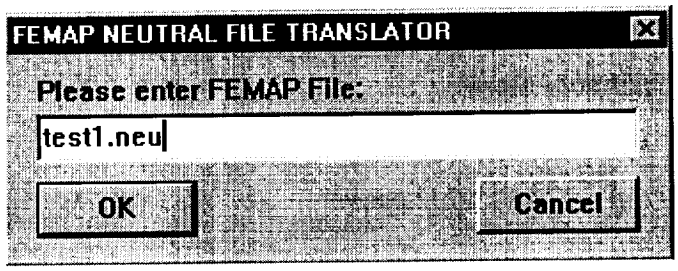


Figure 1. Prompt requesting FEMAP file name.

2. The second window prompts for loading of constraint information into memory.

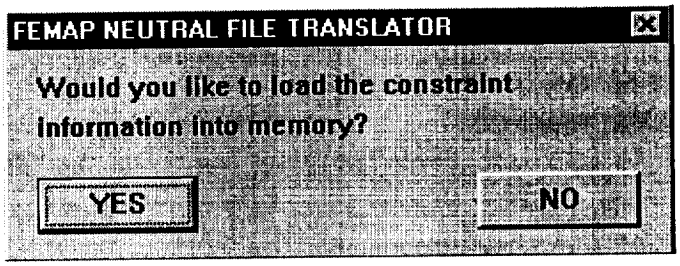


Figure 2. Load constraint information into memory.

3. If yes is pressed on the above window another window is displayed prompting for a selection of a constraint set.

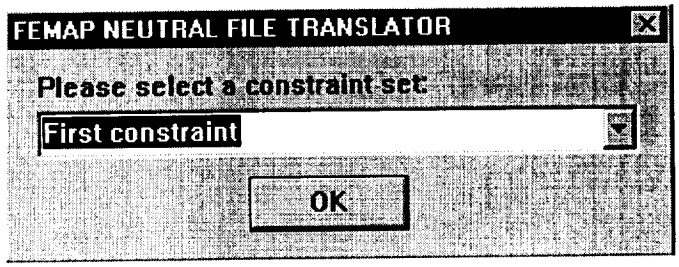


Figure 3. Please select a constraint set.

4. The next window prompts for loading of load information into memory.

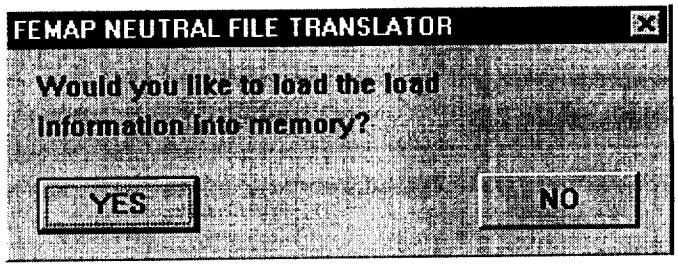


Figure 4. Load the load information into memory.

5. If yes is pressed on the above window another window is displayed prompting for a selection of a load set.

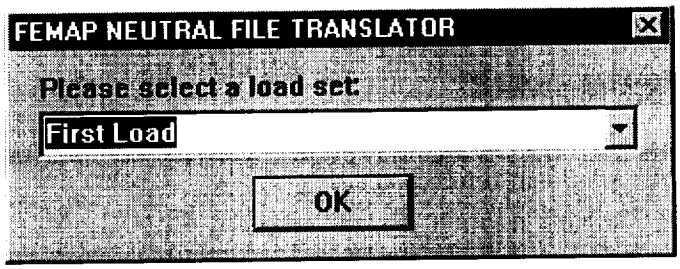


Figure 5. Please select a load set.

6. The next window prompts for a selection of a case number.

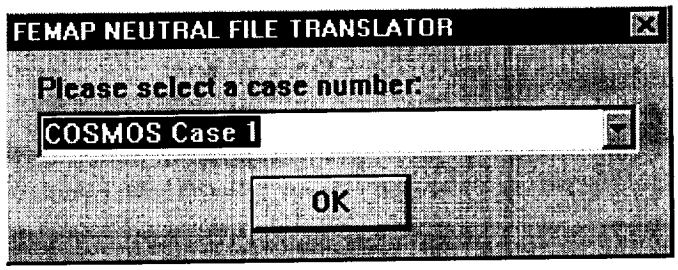


Figure 6. Please select a case number.

7. The next window prompts for a selection of up to 5 sets of elemental output data and 5 sets of nodal output data for visualization.

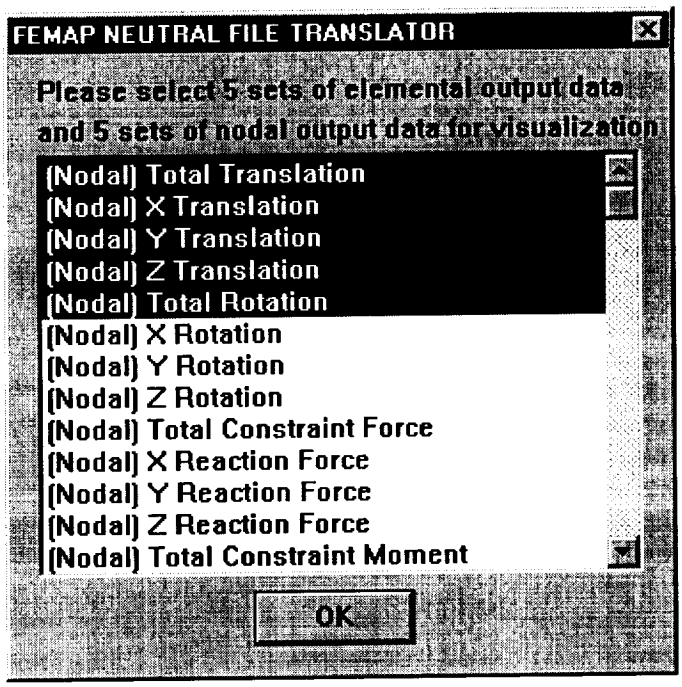


Figure 7. 5 nodal output data sets has been selected.

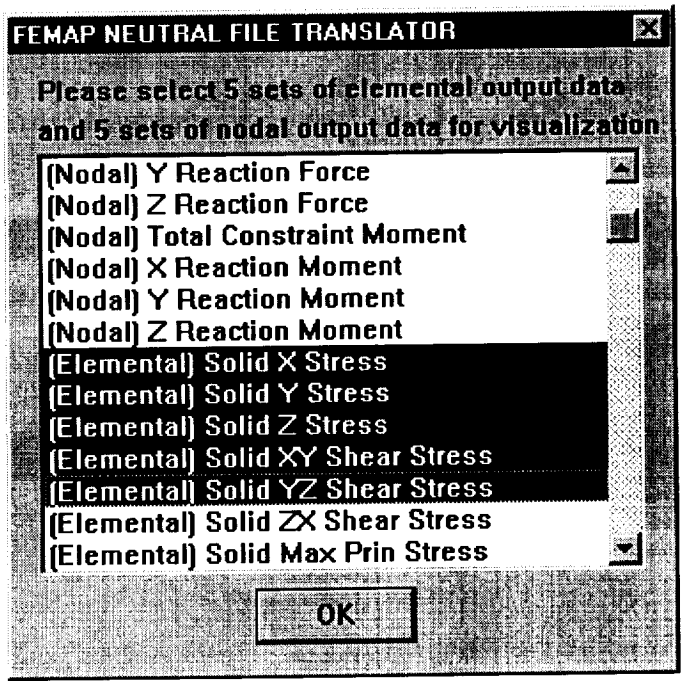


Figure 8. 5 elemental output data sets has been selected.

8. The last graphical user interface window prompt that is displayed is the prompt for filtering the internal surfaces of the model. If there are no brick elements in the model this prompt is not displayed.

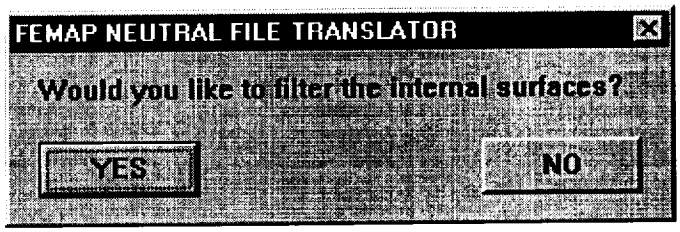


Figure 9. Would you like to filter the internal surfaces?

The DVET system on the SGI platform uses a command line interface that is parallel in functionality to the GUI on the Window NT system.

View Button

The View Button on the DVET system's toolbox is used to manipulate and position the model in select pre-defined view orientations. By selecting the view button an array of pre-defined buttons are displayed where one can select a view of choice. The Next Generation Space Telescope (NGST) is the model being displayed.

1. The view button is the top most button on the upper left hand corner of the display screen. The button remains visible throughout the duration of the analysis.

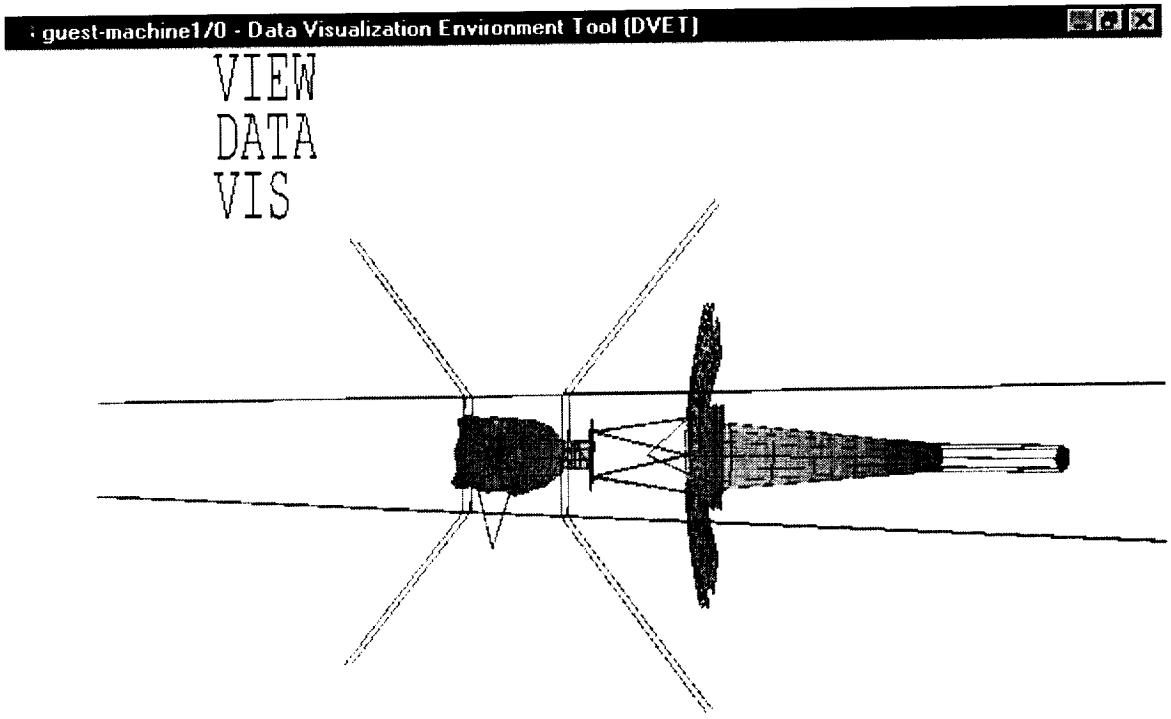


Figure 10. Initial screen

2. Clicking on the view button displays several pre-defined view positions and orientations. A quit button is also displayed for exiting out of the view menu. Another title button indicates what menu screen is being displayed.

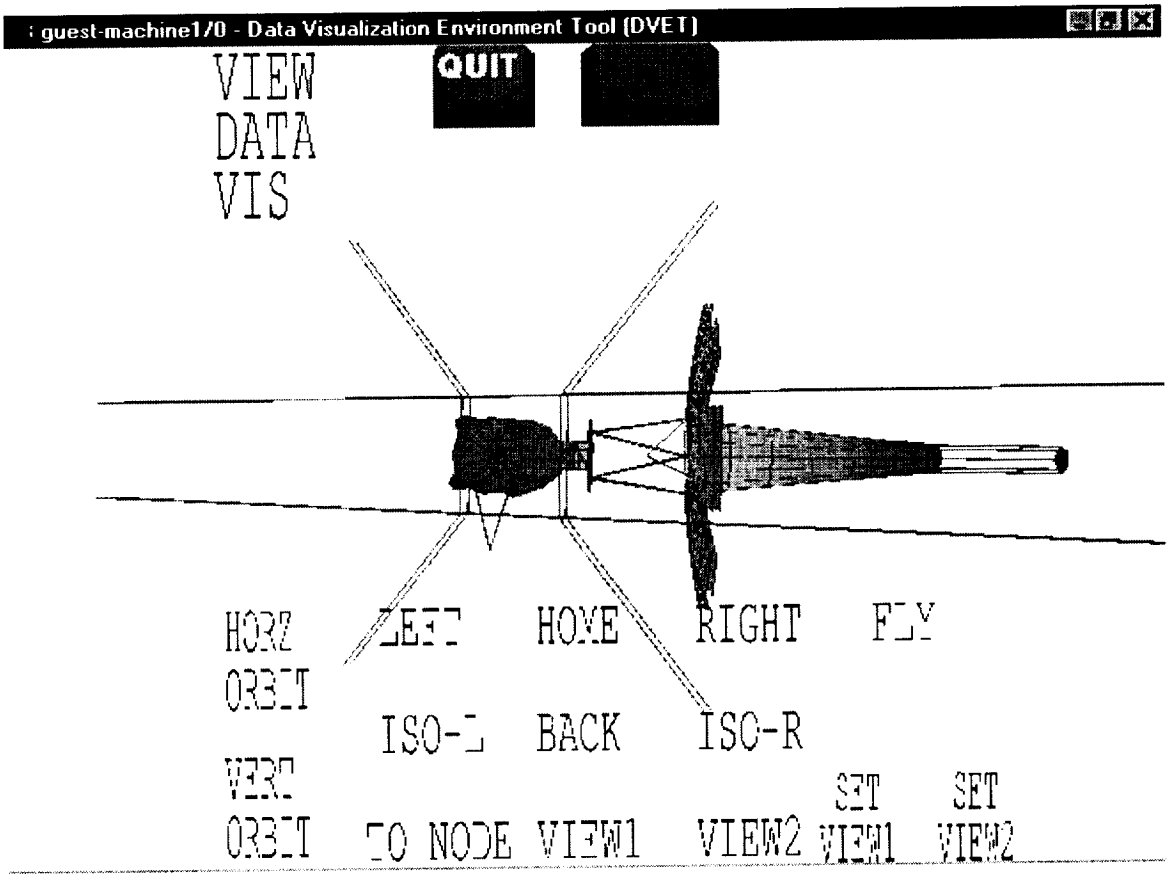


Figure 11. Screen after view button is selected.

3. The figure below indicates the motion of the model when HORZ ORBIT is pressed. The model rotates once around the Y-axis.

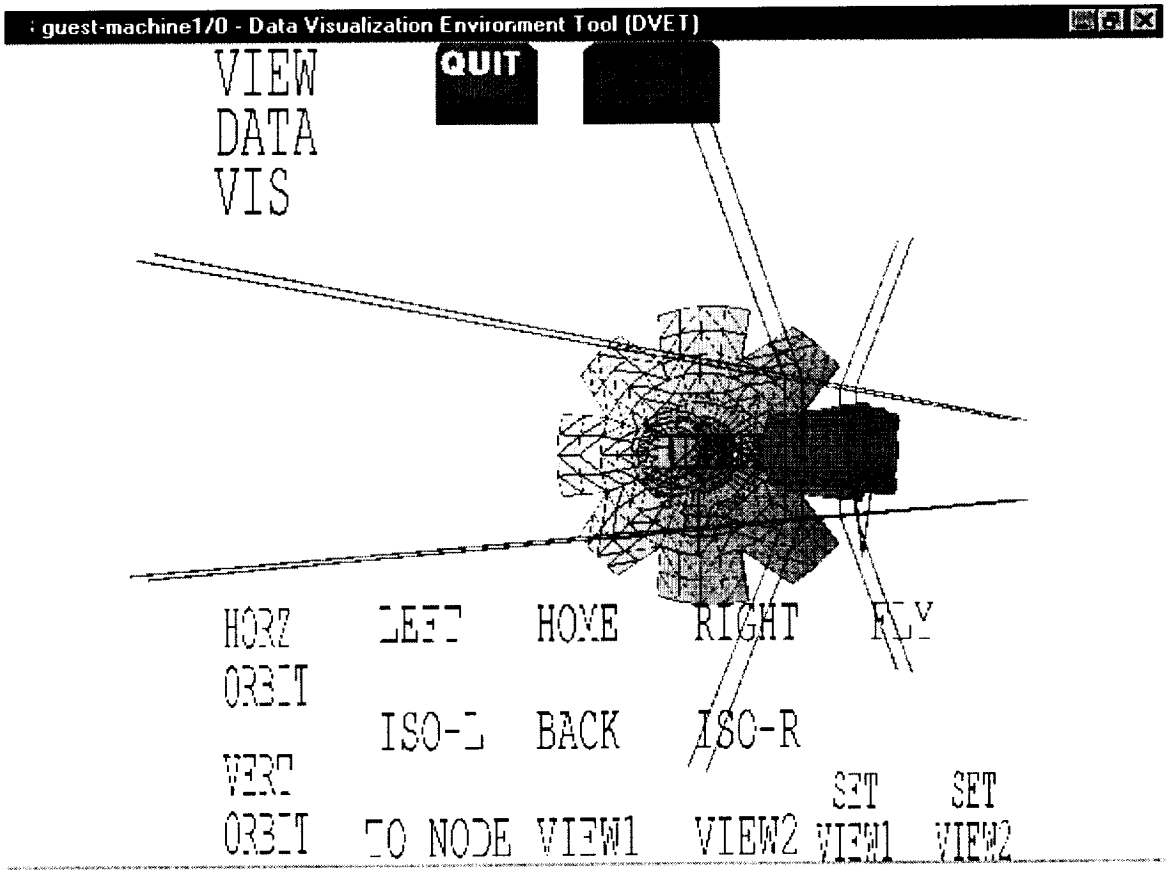


Figure 12. HORZ ORBIT.

4. The model rotates to a left view when the LEFT button is pressed.

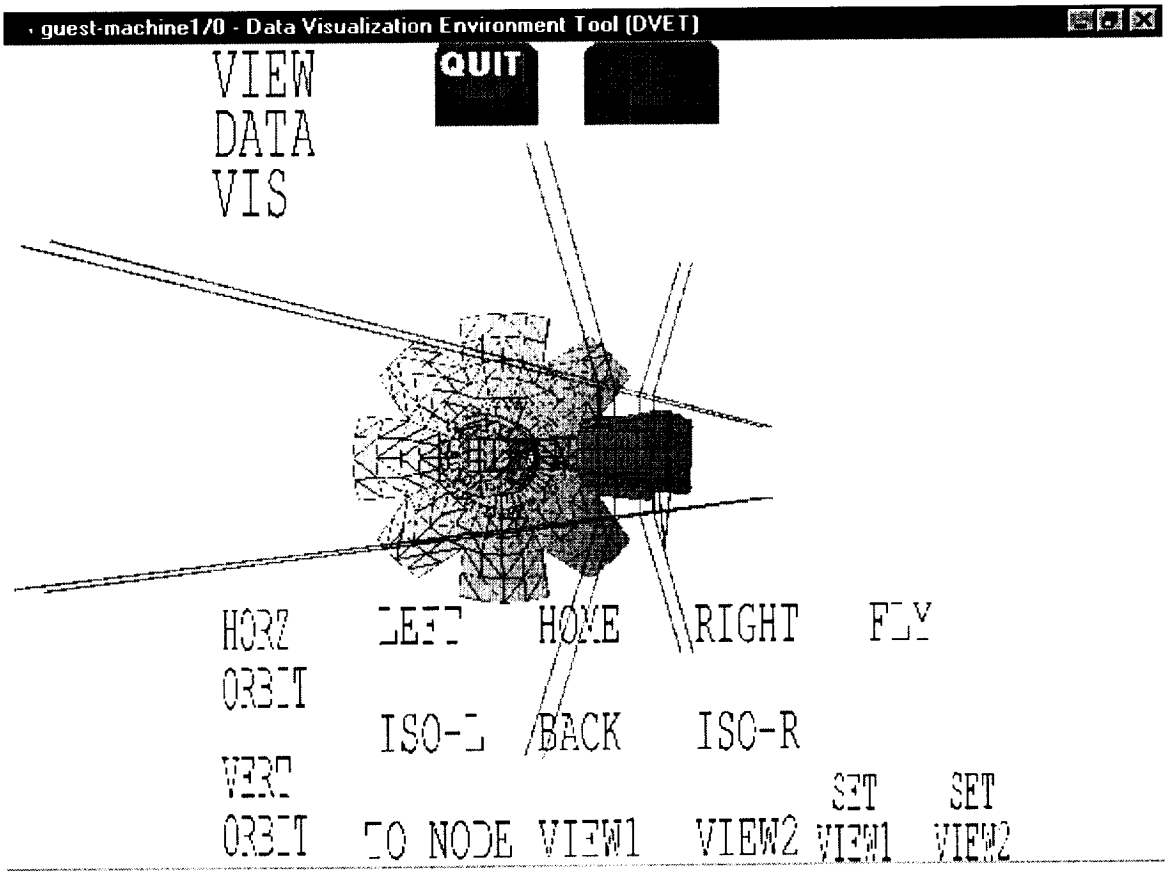


Figure 13. LEFT BUTTON PRESSED.

5. In a particular instance where one looses his or her orientation in space, one can click on HOME to return the model to its initial orientation.

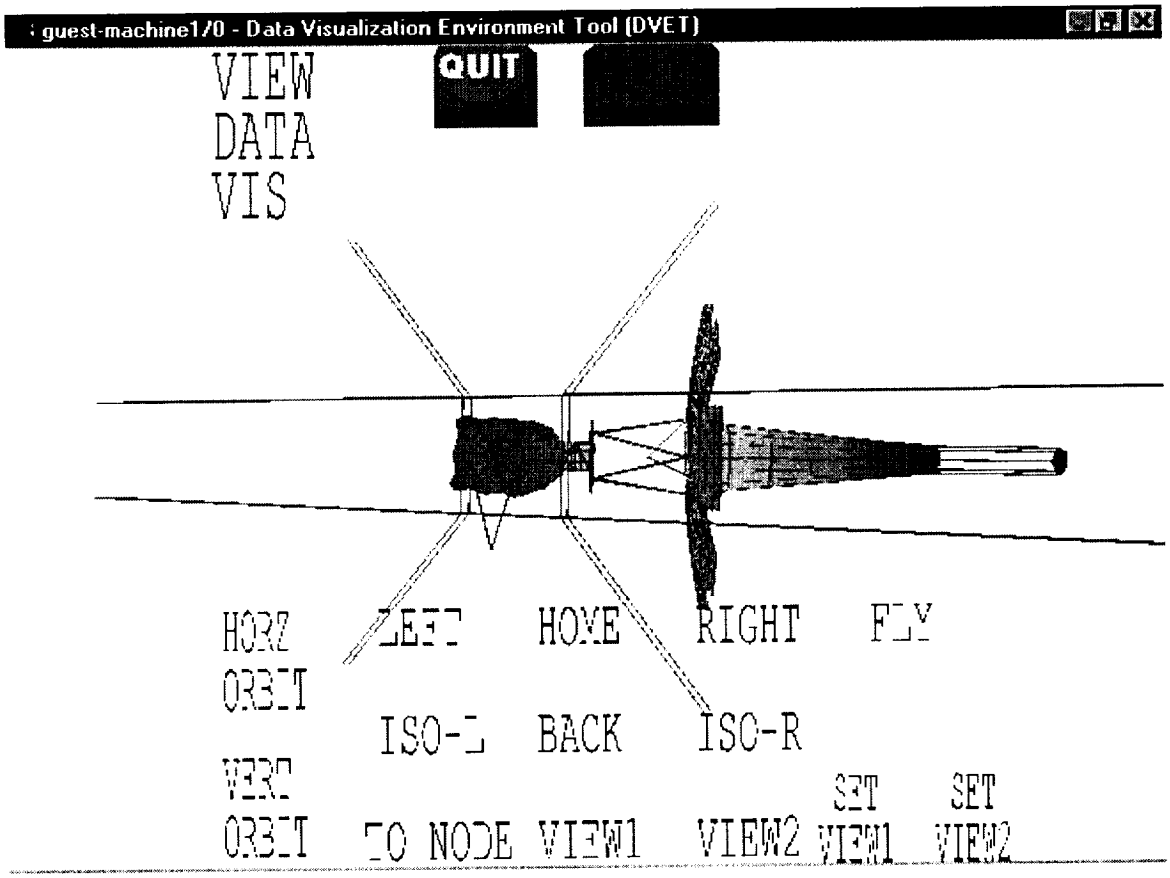


Figure 13. HOME BUTTON PRESSED.

6. The model rotates to a right view when the RIGHT button is pressed.

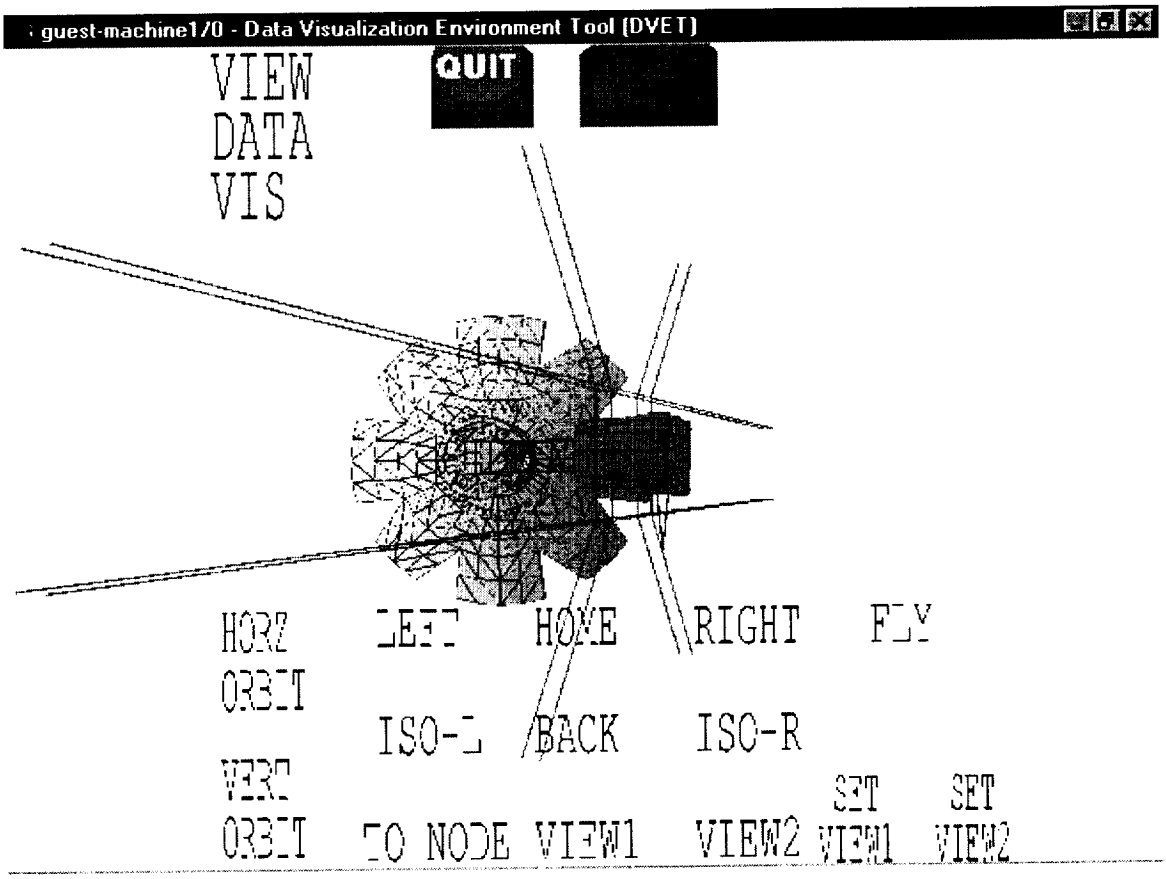


Figure 14. Right View.

7. The FLY button is selected when a smooth change in view is needed.

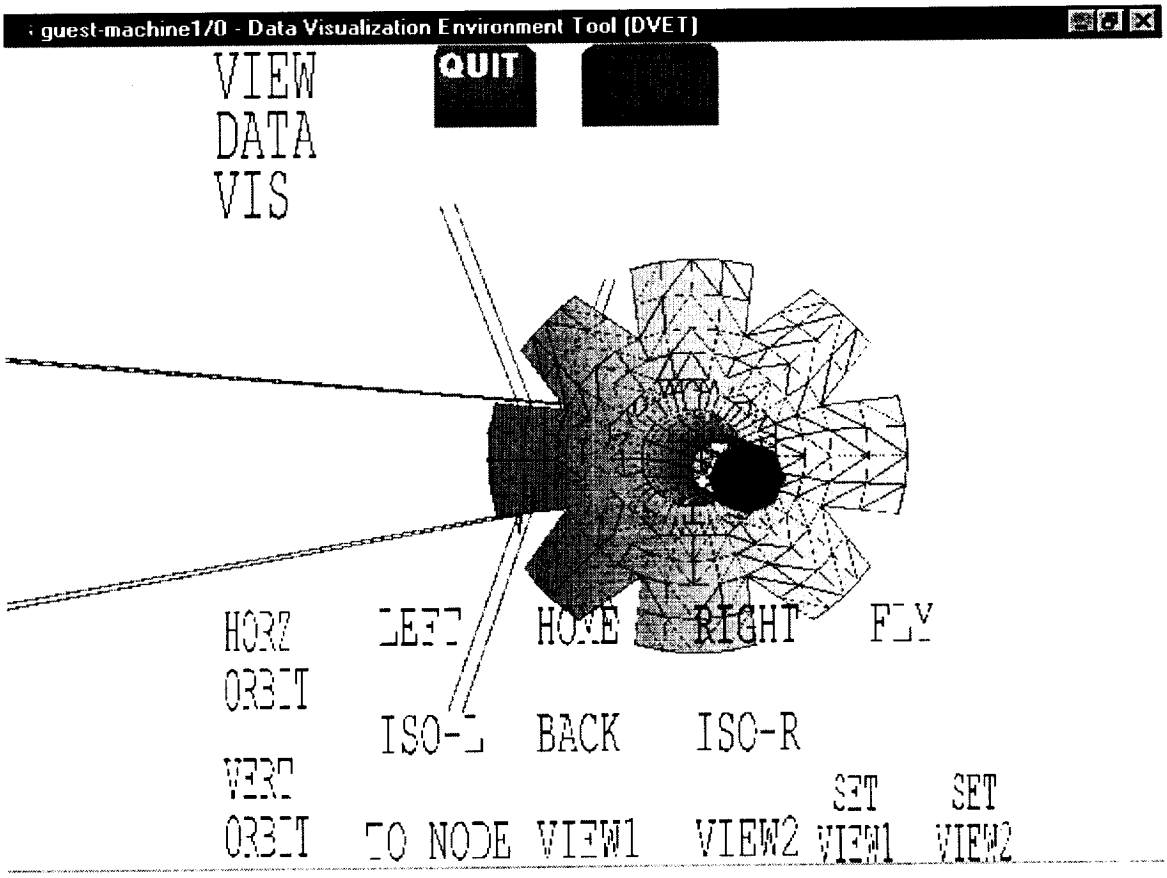


Figure 15. FLY Button.

8. The HYPER button is selected when instantaneous change in views is needed. Normally used when one is more familiar with the DVET system.

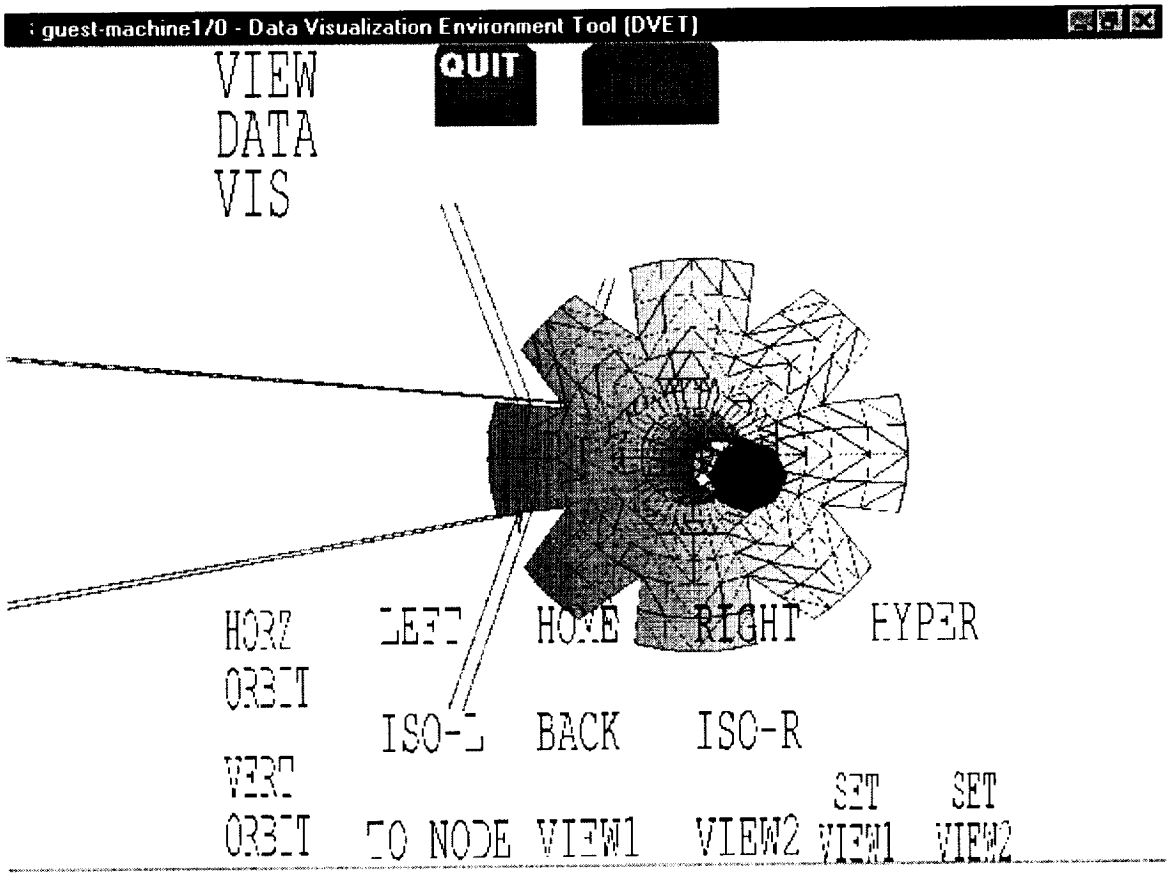


Figure 16. HYPER Button.

9. When VERT ORBIT is pressed the model spins a complete rotation in the Z - axis.

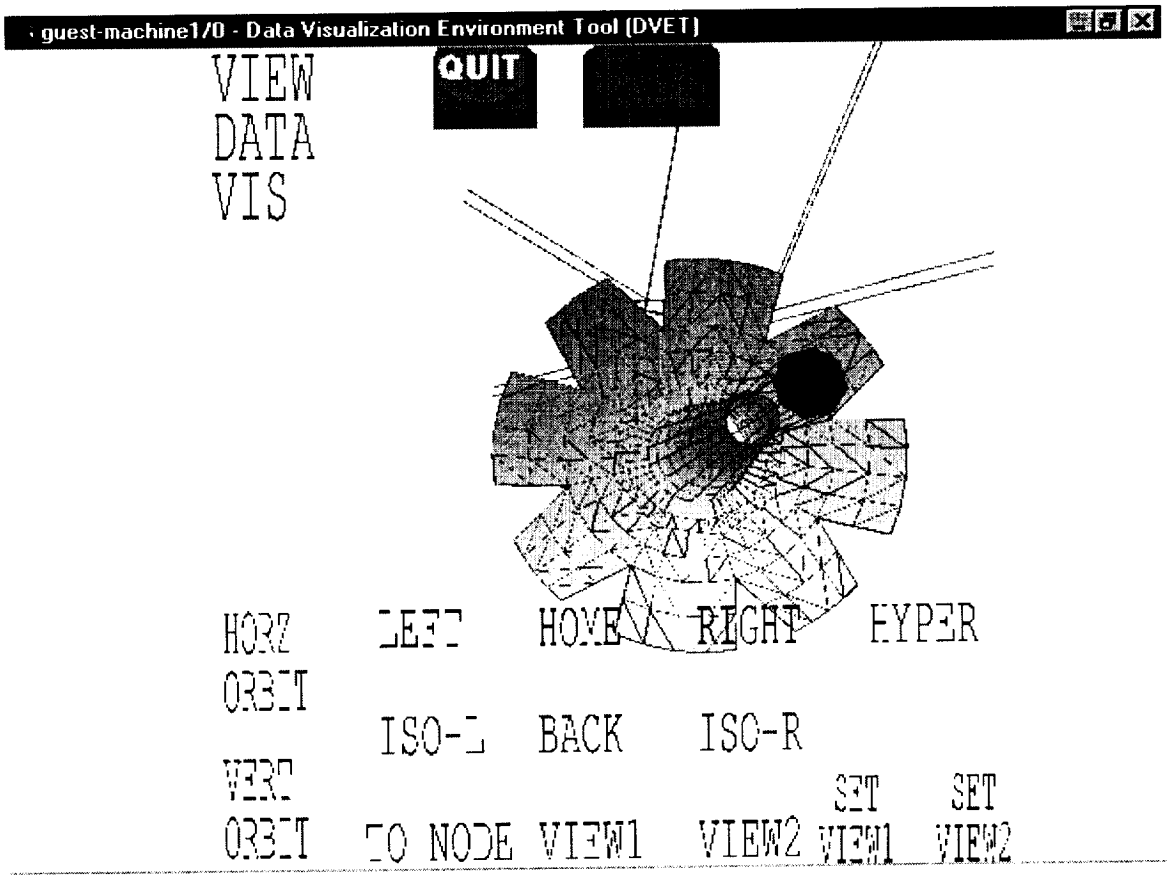


Figure 16. VERT ORBIT.

10. ISO-L is pressed.

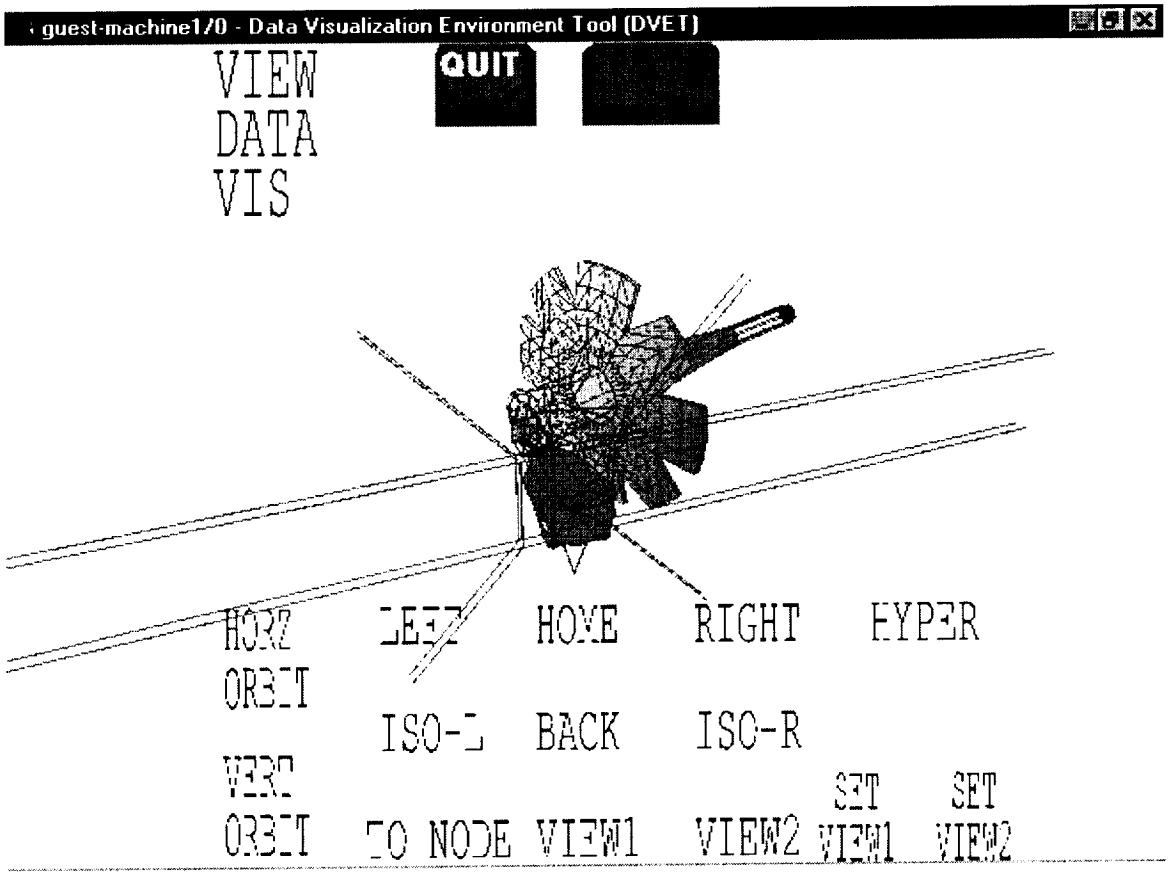


Figure 17. ISO-L.

11. BACK button is pressed.

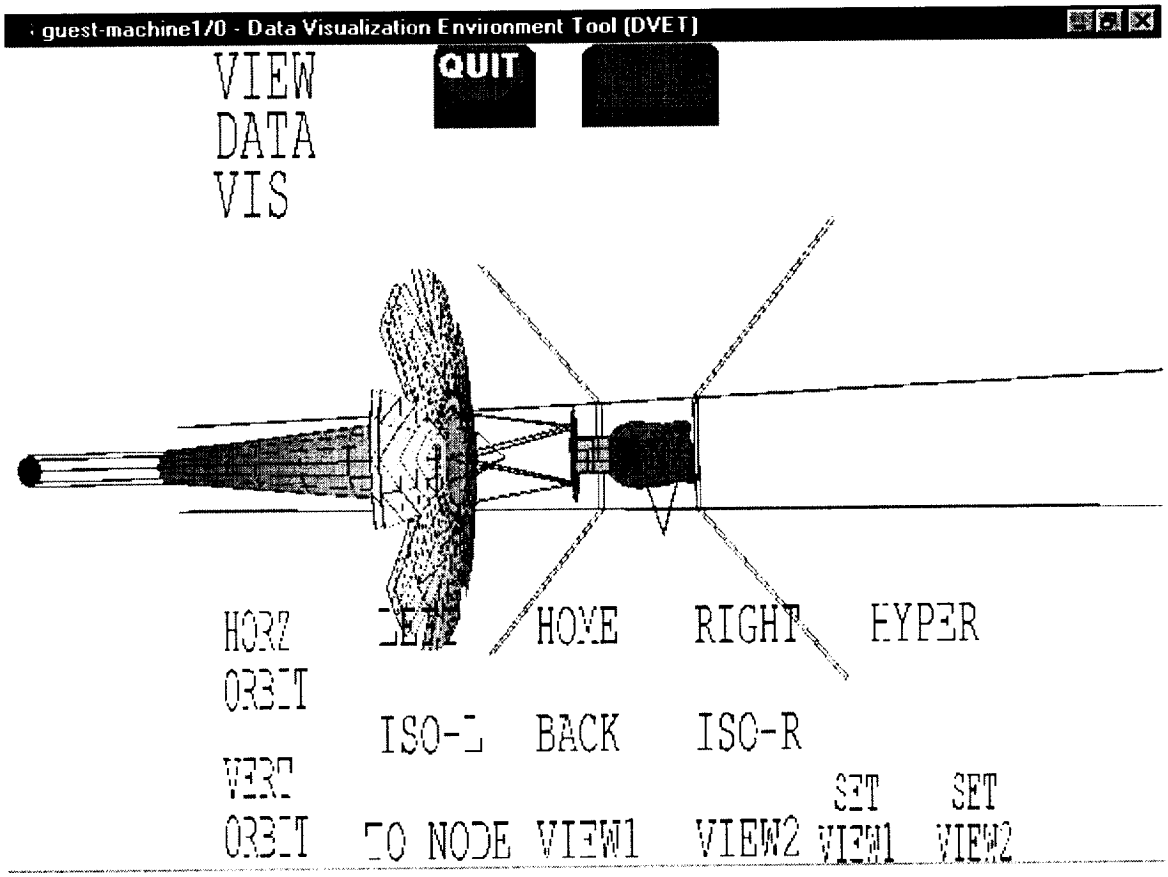


Figure 18. BACK button.

12. ISO-R is pressed.

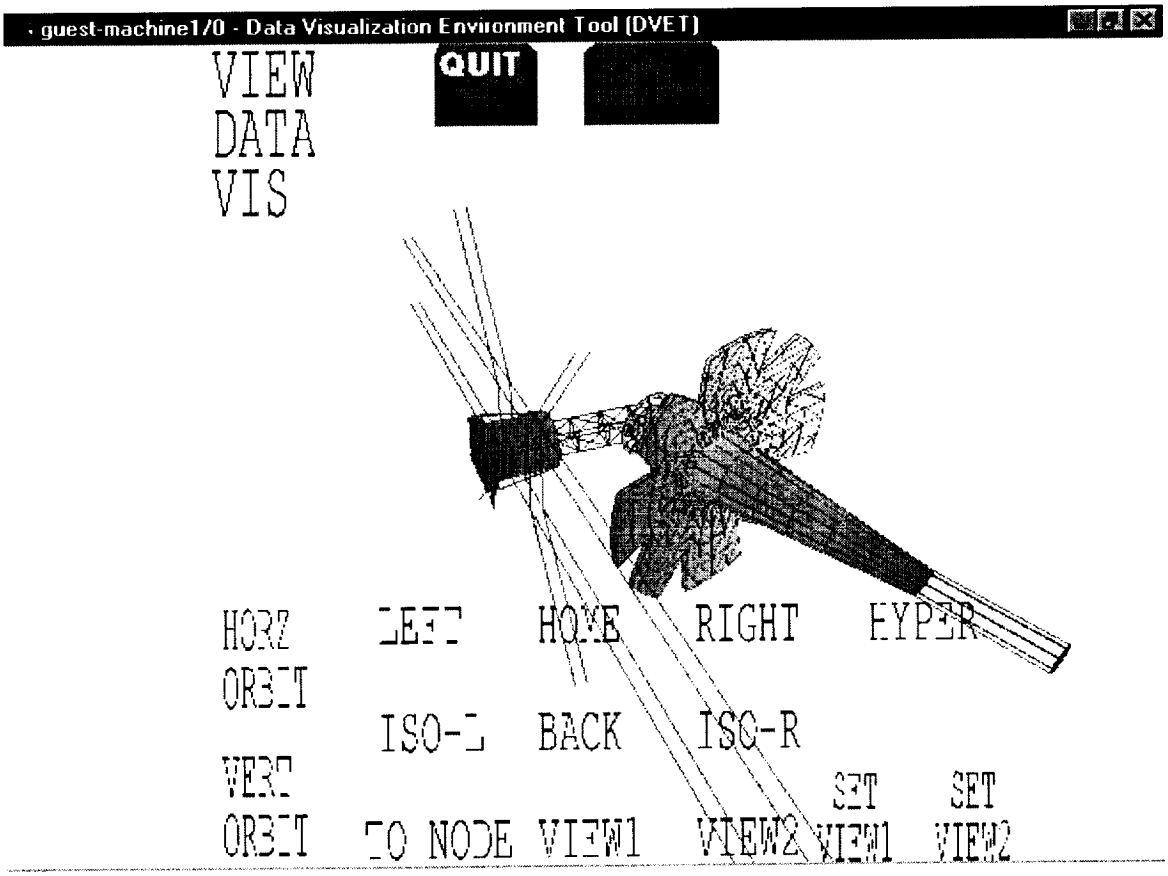


Figure 19. ISO-R button.

13. By selecting a node on the model, one can press the TO NODE button and the view will change to an offset distance away from the selected node.

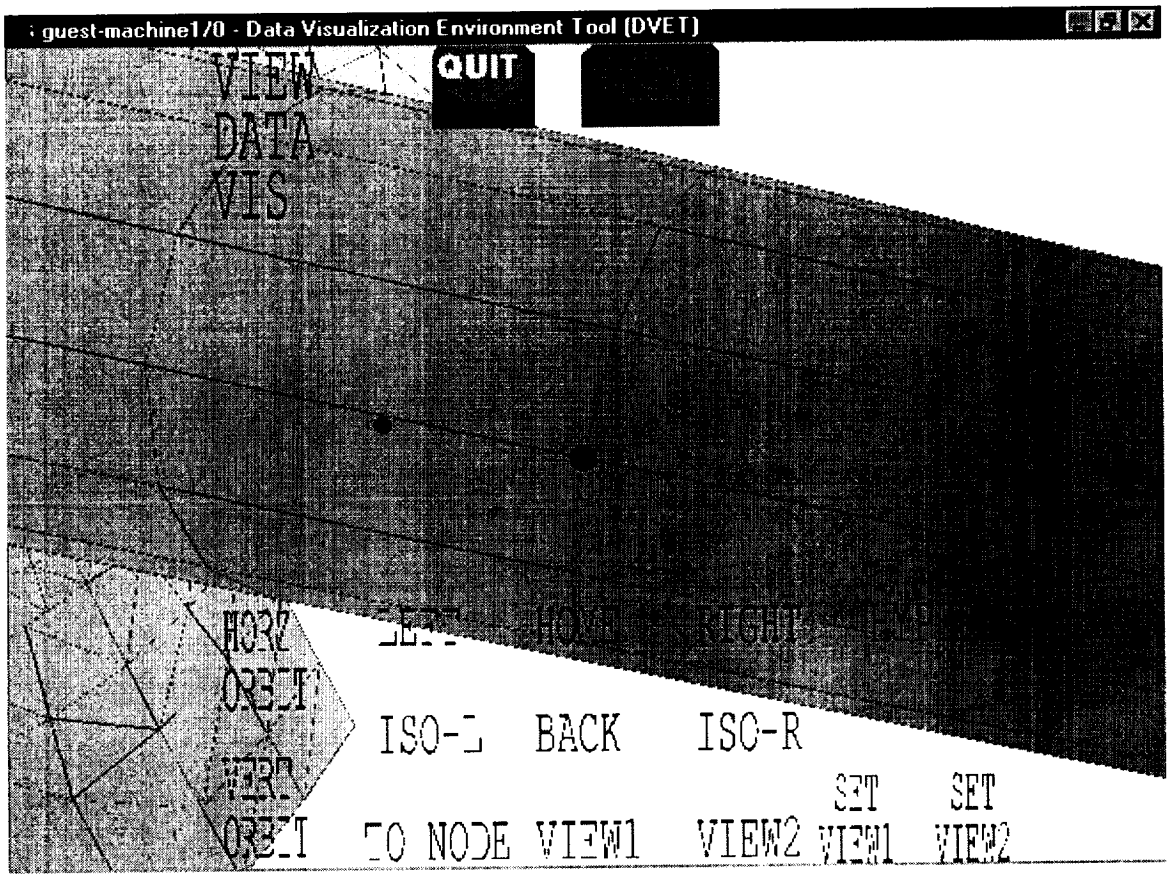


Figure 20. TO NODE button.

14. By selecting the SET VIEW1 or SET VIEW2 button one can define a view of one's choice to be saved and retrieved by pressing whatever view has been saved. The above Figure 20. also indicates the view that is saved by pressing SET VIEW 1. The figure below indicates a second view obtained by selecting another node and pressing TO NODE.

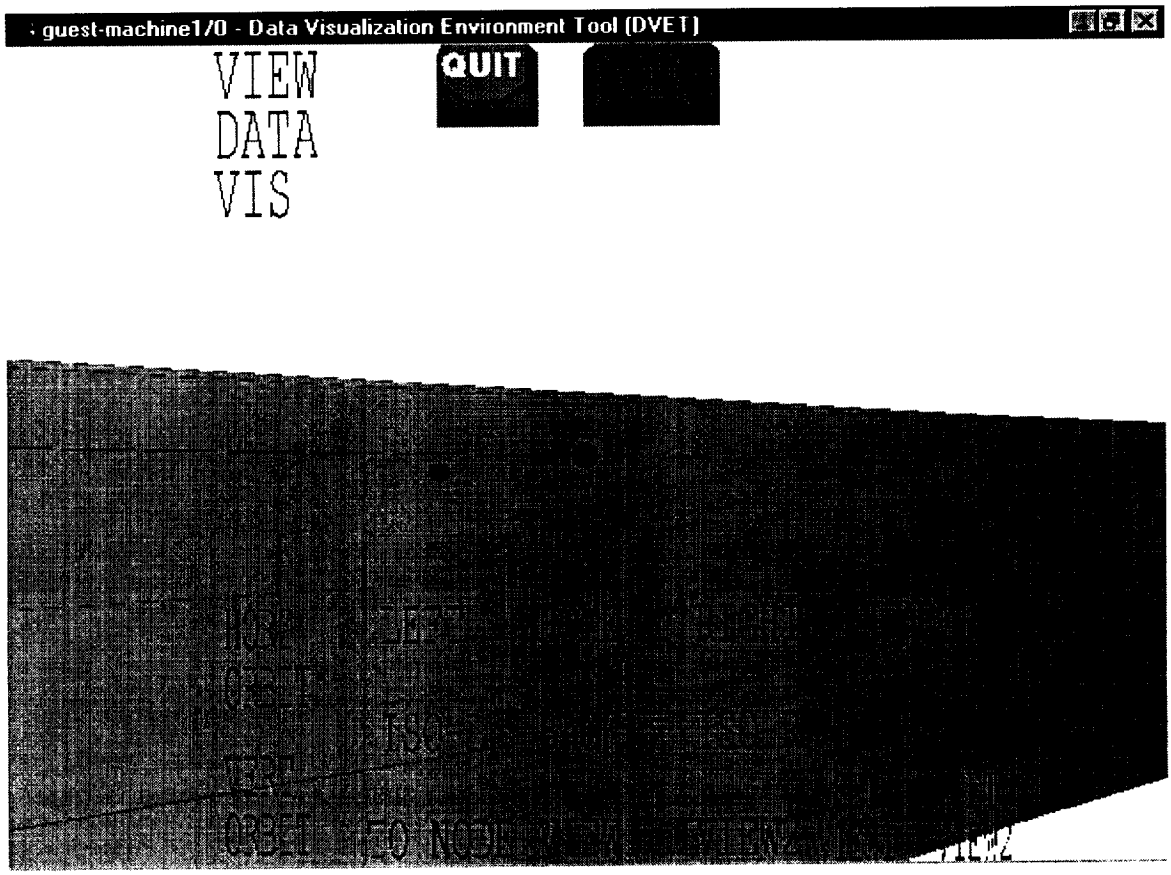


Figure 21. A second TO NODE button press.

15. Figure 21. Indicates the a second view saved using SET VIEW2. The view below shows the view 1 when the button VIEW1 is pressed.

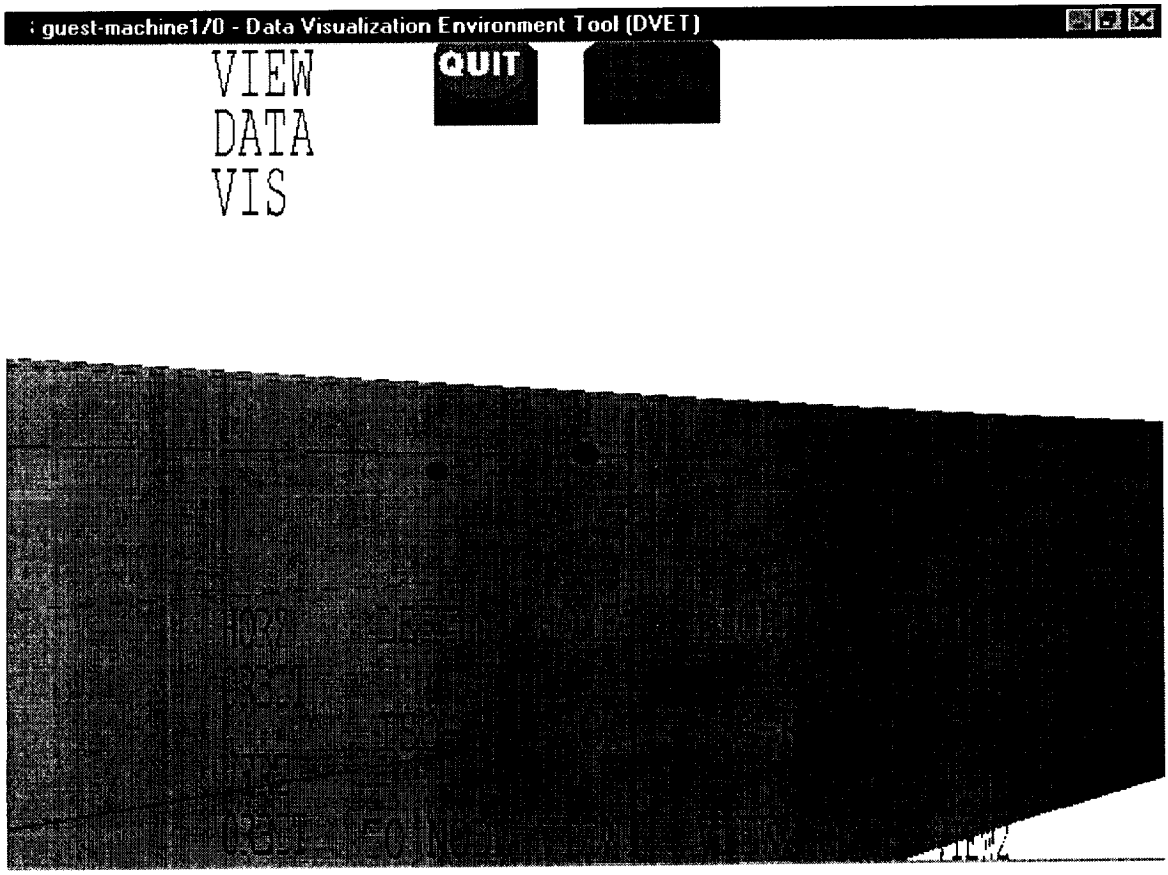


Figure 22. VIEW1 button is pressed displaying a saved view.

16. When VIEW2 Button is pressed the saved view 2 is displayed.

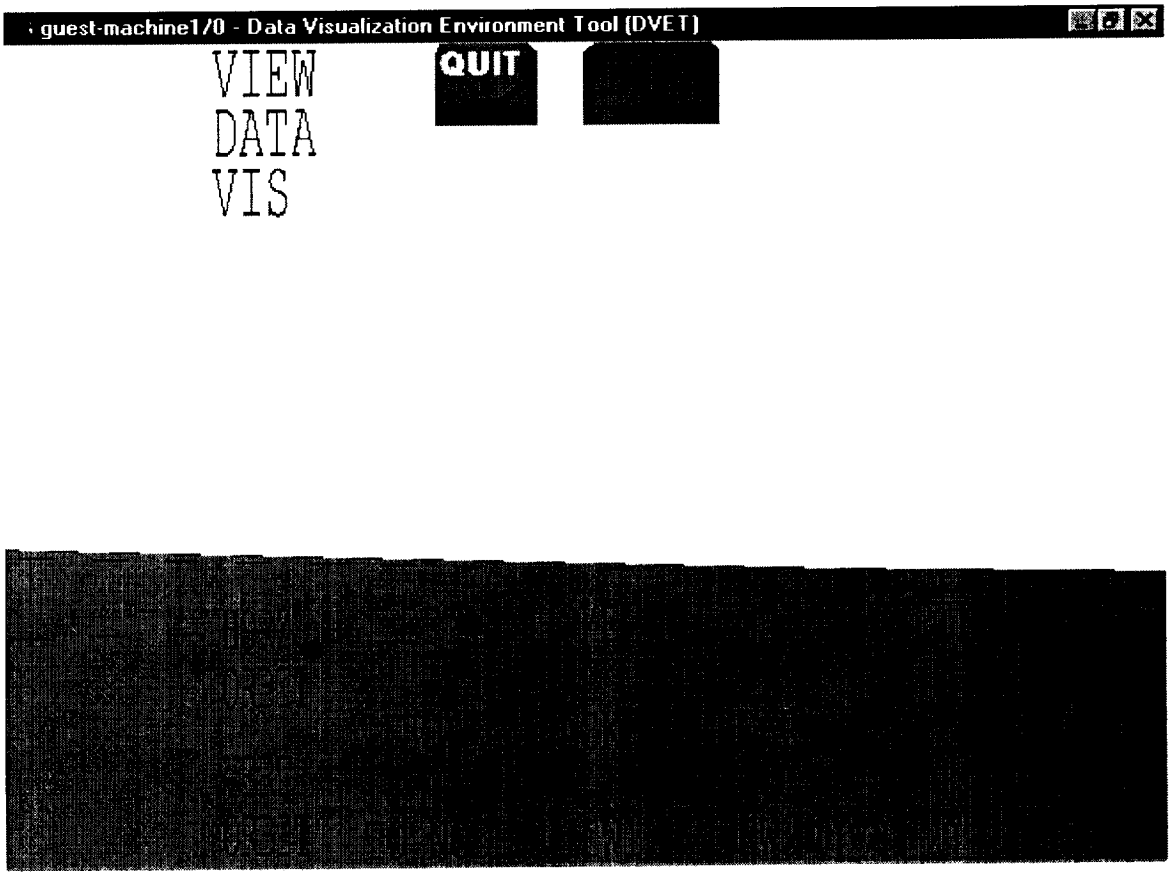


Figure 23. VIEW2 button is pressed displaying a saved view.

Data Button

The Data Button on the DVET system's toolbox is used to display a menu of data output sets, both nodal and elemental. The constraint and load information is also selected from the Data Button menu. A Beam model is used to show the operation of the DATA button menu. Refer to Figure 7 and Figure 8 for the names of the output data sets being displayed.

1. By selecting the DATA button, Figure 24. is displayed. The QUIT button also is displayed in order to exit from the DATA menu. A button with DATA is used to show the screen being displayed.

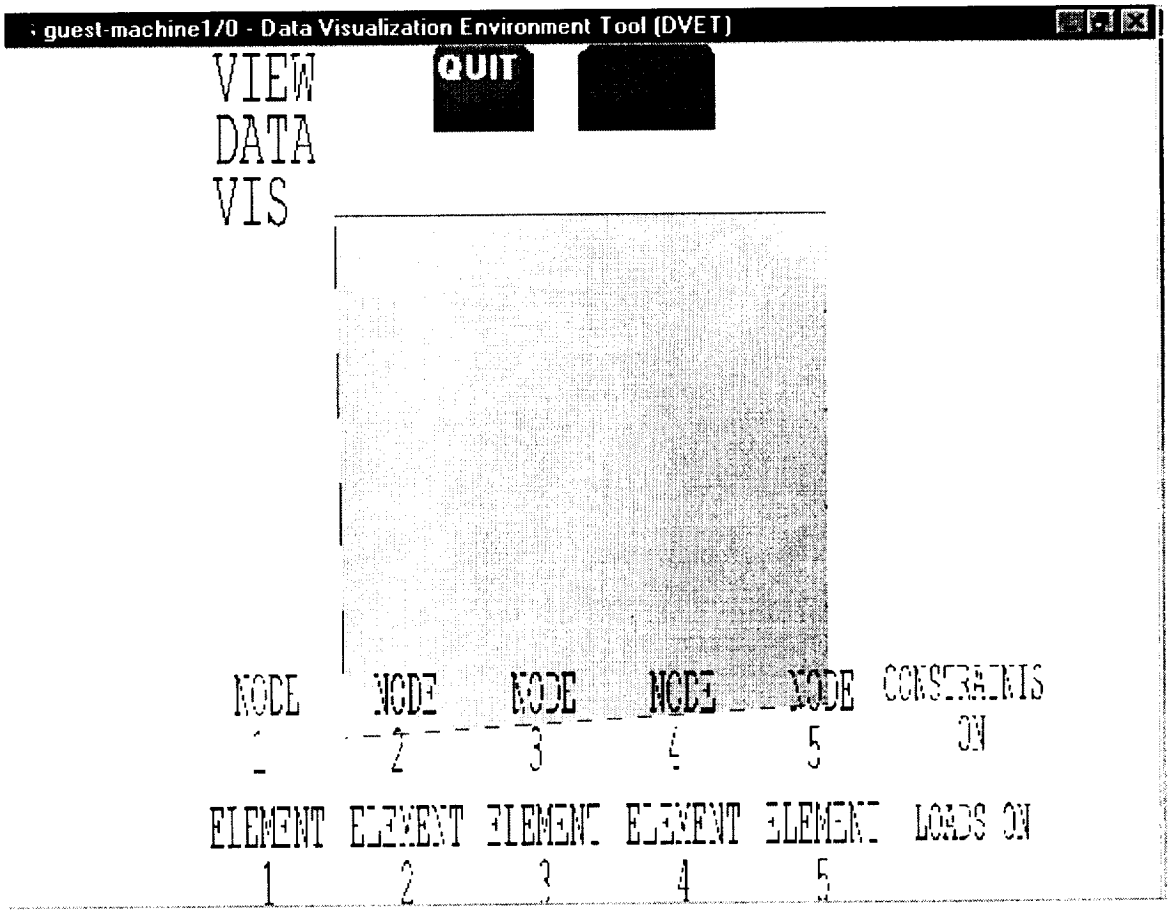


Figure 24. Initial DATA menu screen.

2. In Figure 25. NODE 2 has been pressed notice the difference in output data sets from Figure 24. The default output data set is NODE 1 as seen in Figure 24.

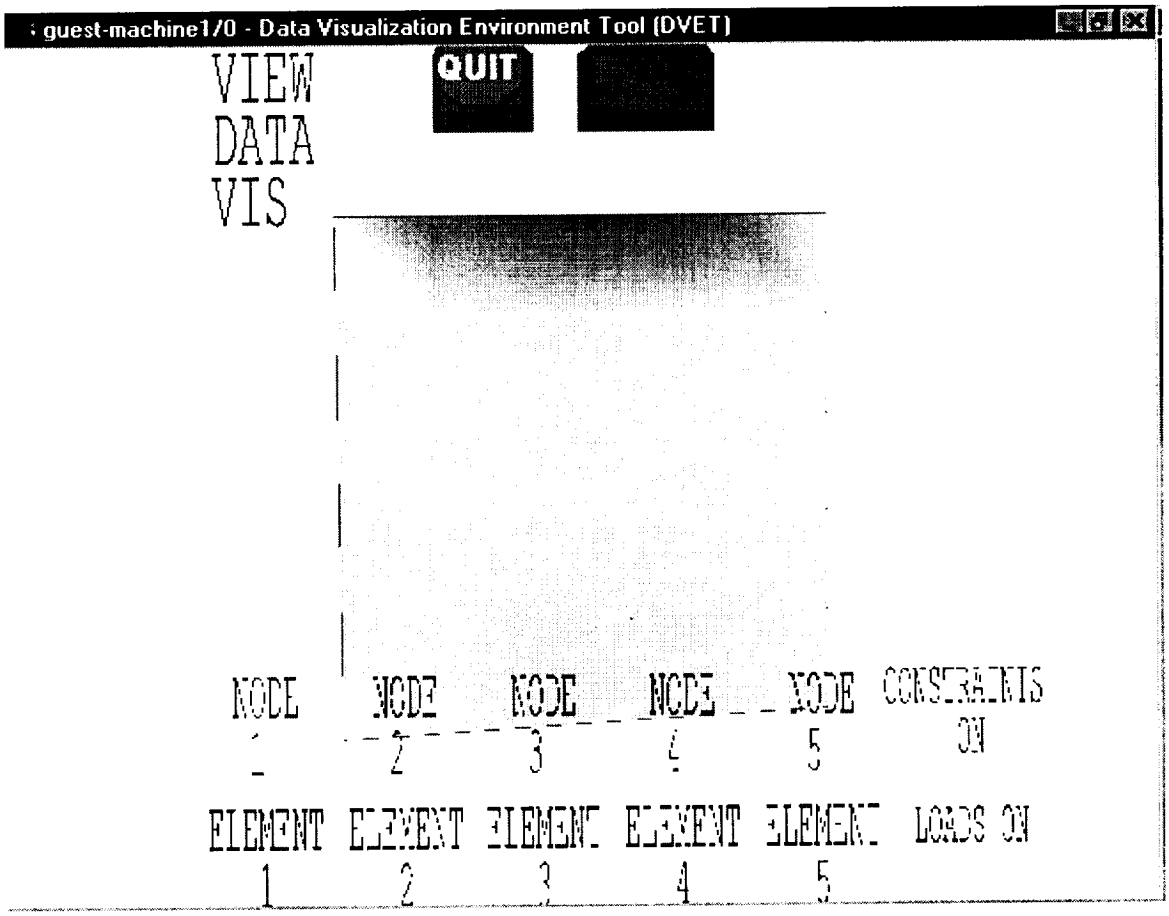


Figure 25. NODE 2.

3. NODE 3 pressed.

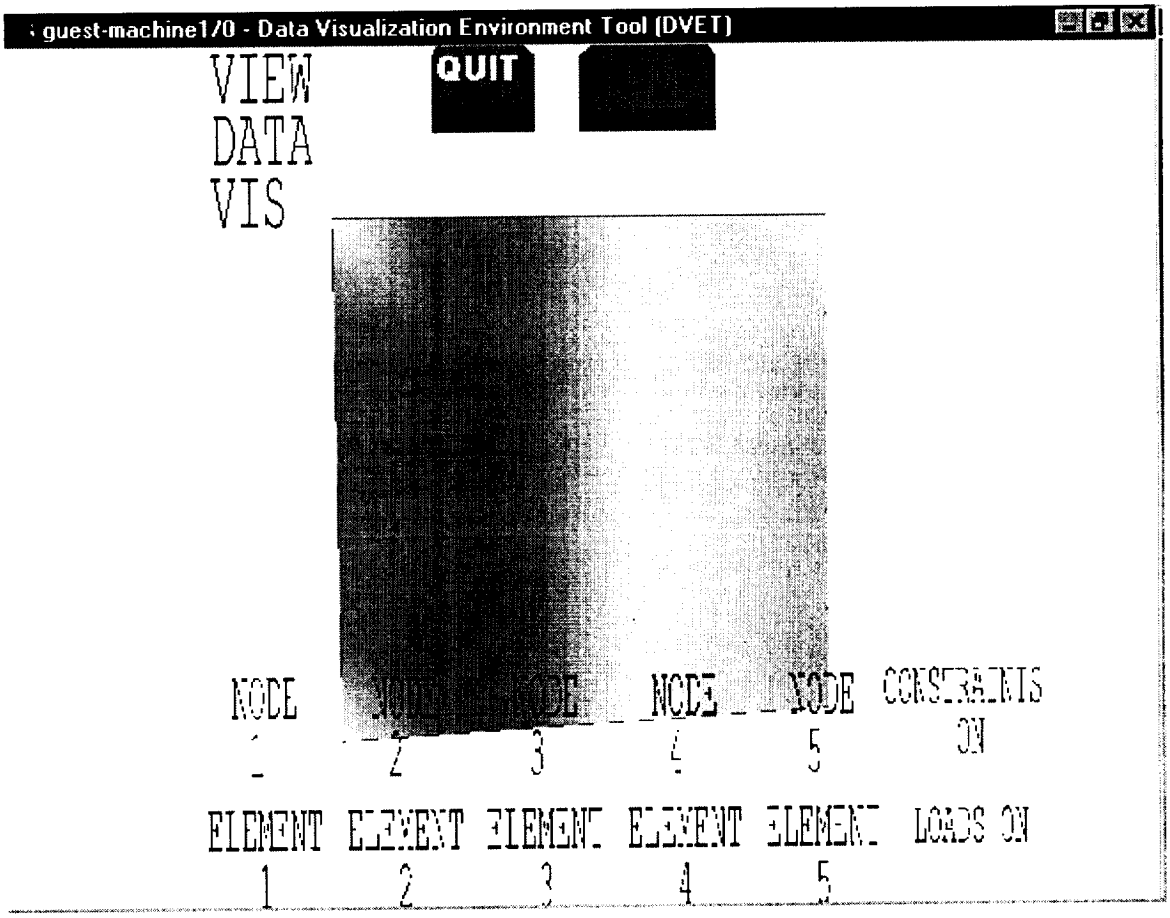


Figure 26. NODE 3.

If the DATA ON button is pressed under the VIS button menu, when an load data set is selected information regarding the load data set is displayed. The information indicates the load data set name the case number it is associated with, etc.

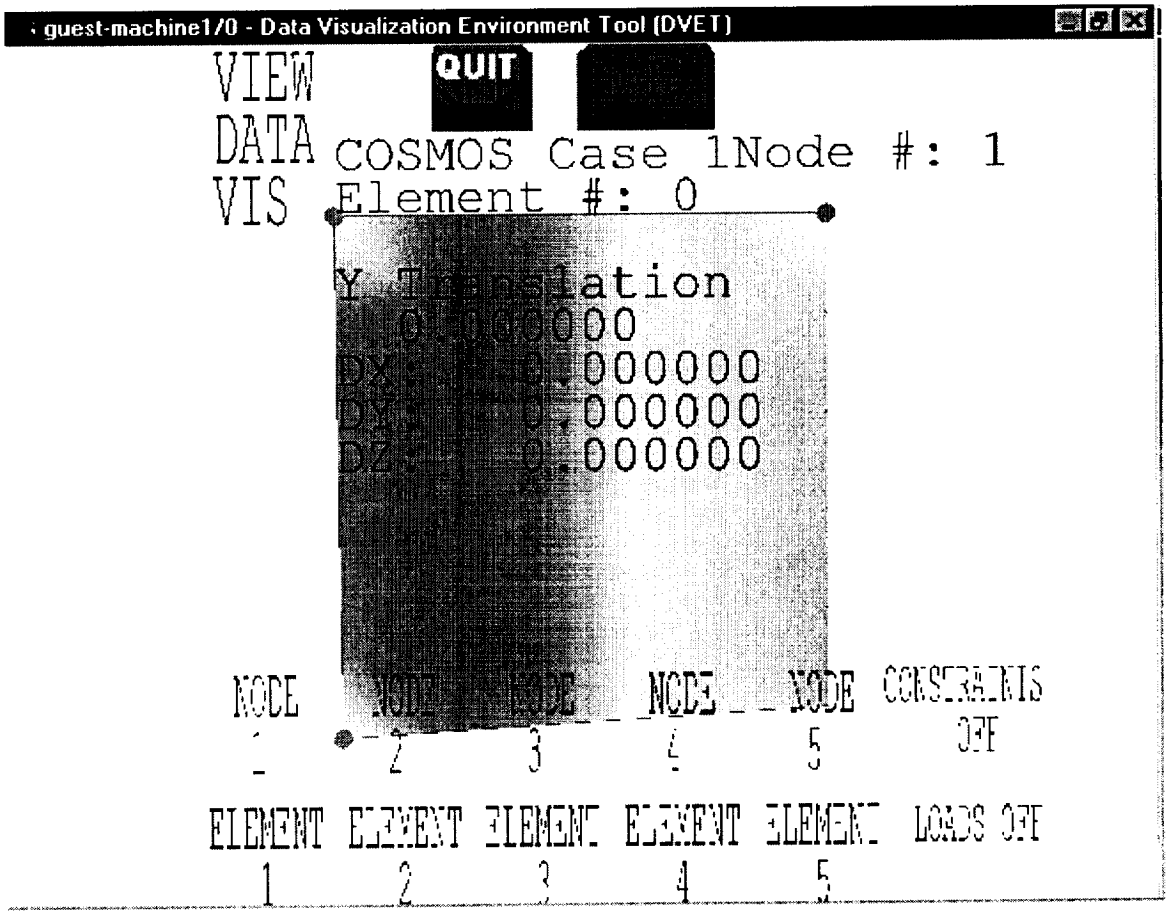


Figure 26a. NODE 3 with associated data information.

4. NODE 4 pressed.

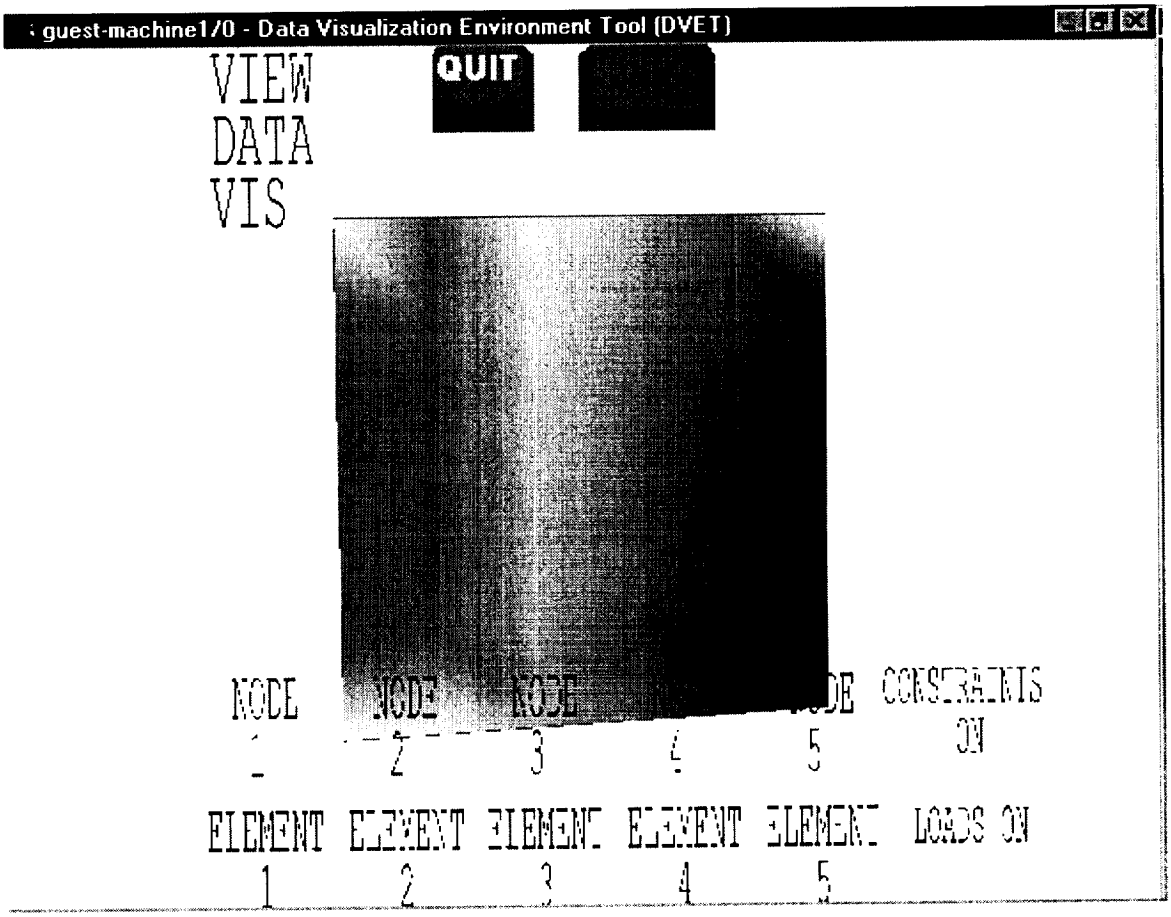


Figure 27. NODE 4.

5. NODE 5 is pressed.

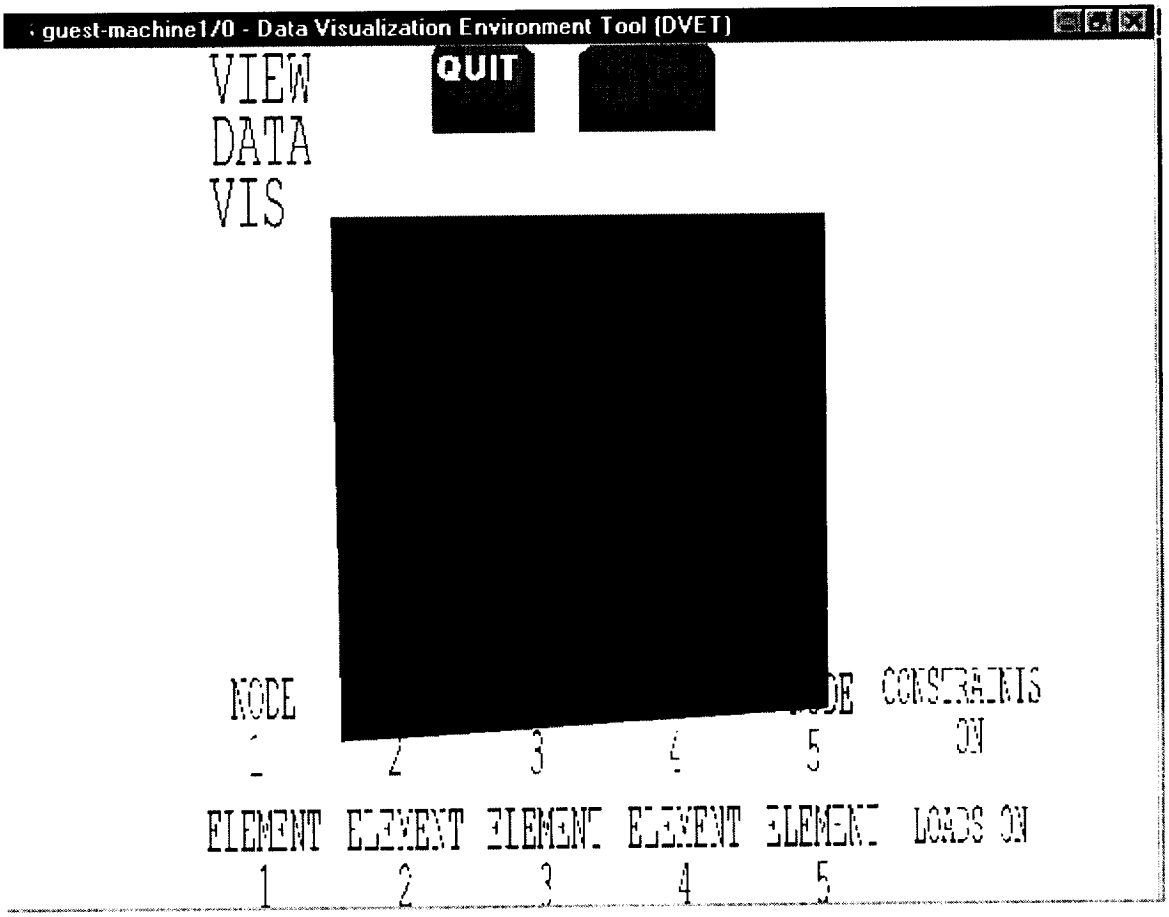


Figure 28. NODE 5.

6. ELEMENT 1 is pressed.

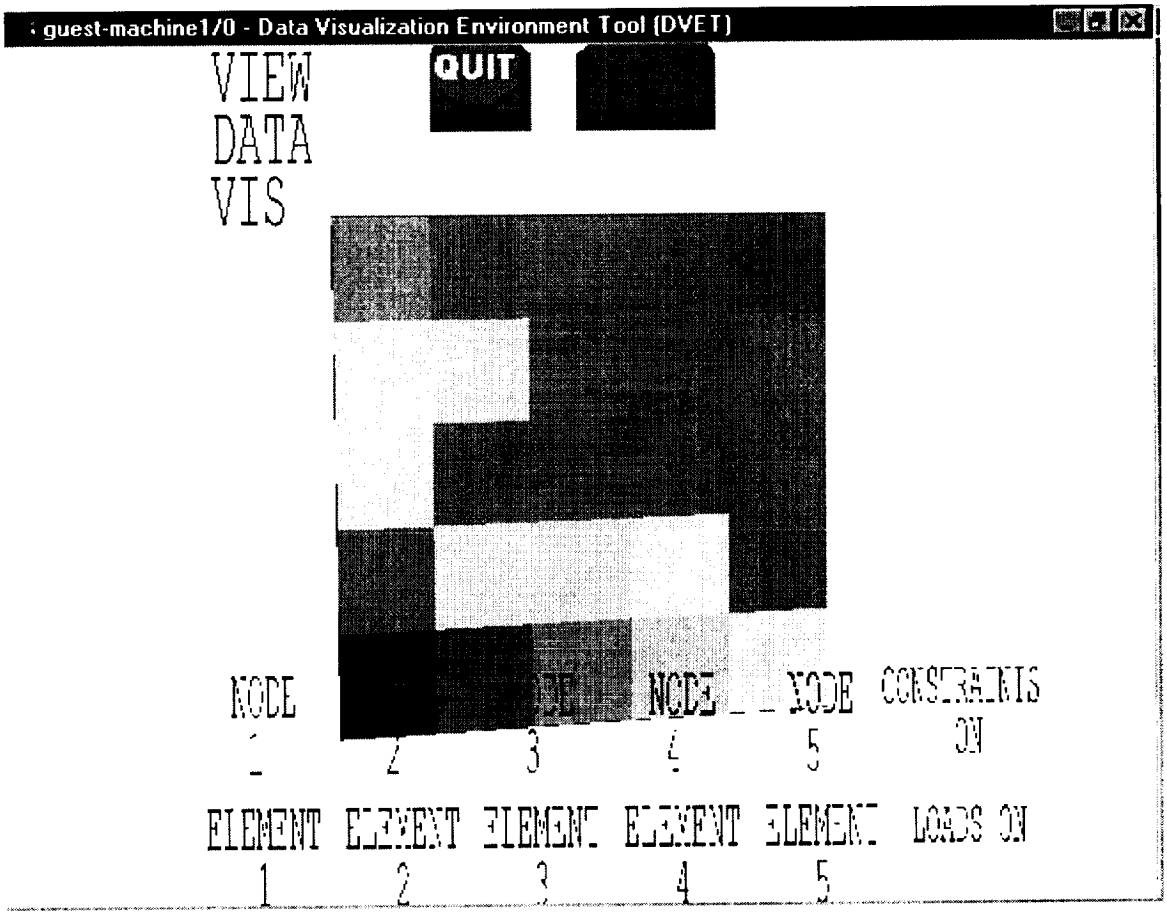


Figure 29. ELEMENT 1.

7. ELEMENT 2 is pressed.

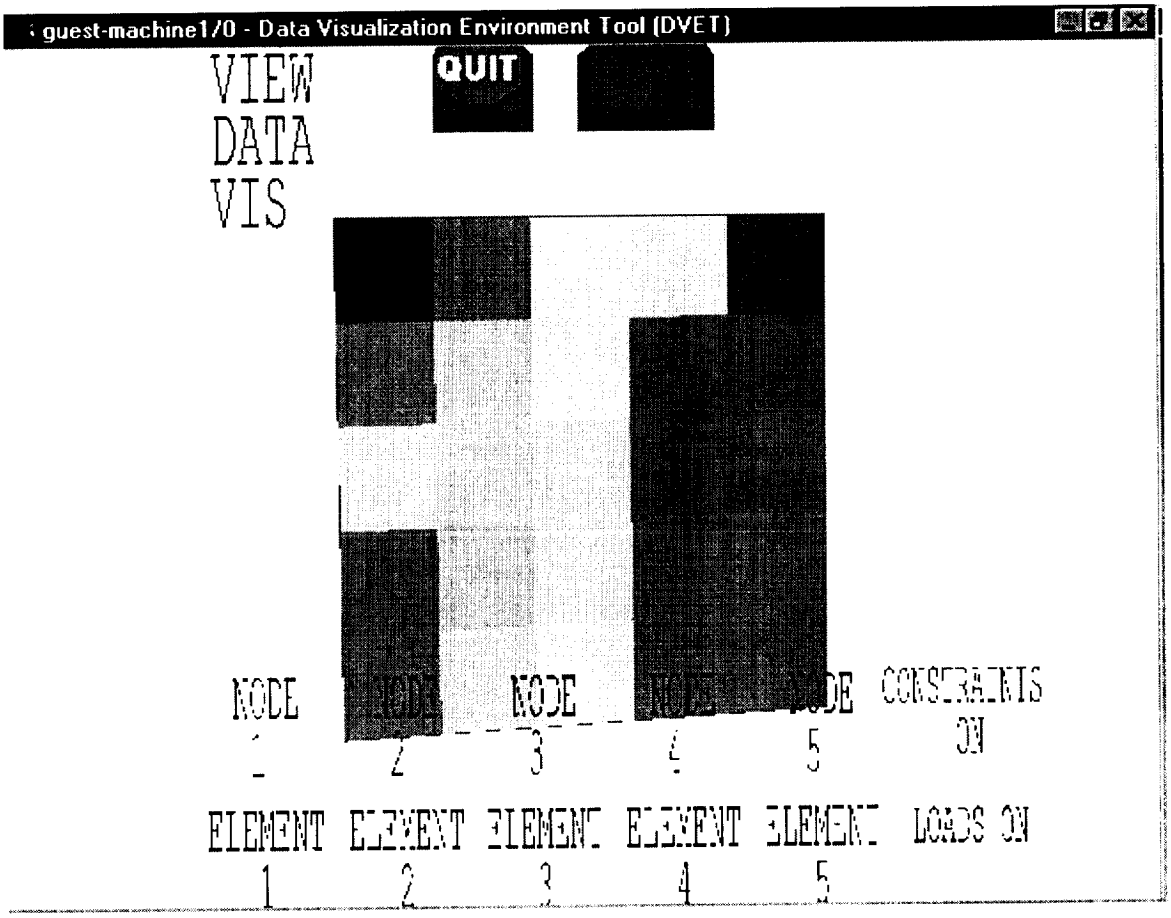


Figure 30. ELEMENT 2.

8. ELEMENT 2 is pressed.

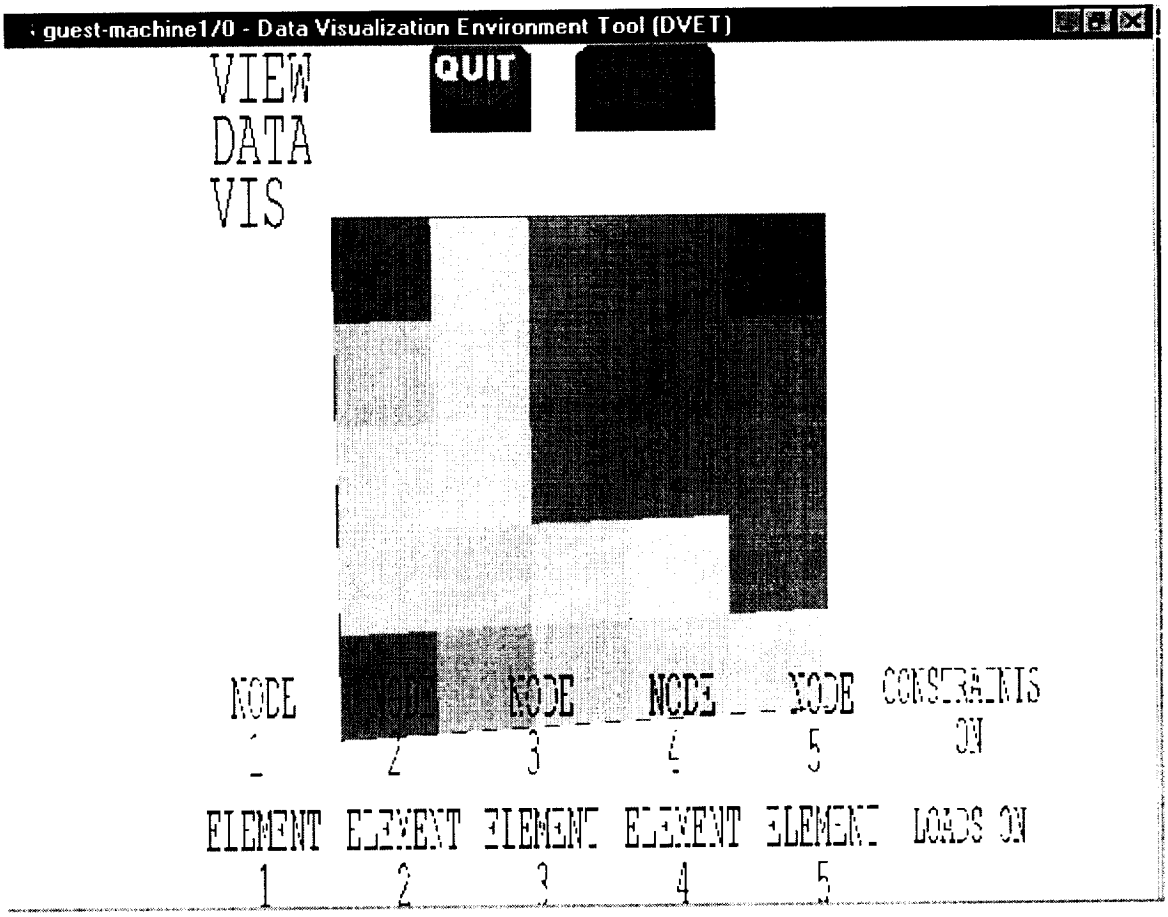


Figure 31. ELEMENT 3.

9. ELEMENT 4 is pressed.

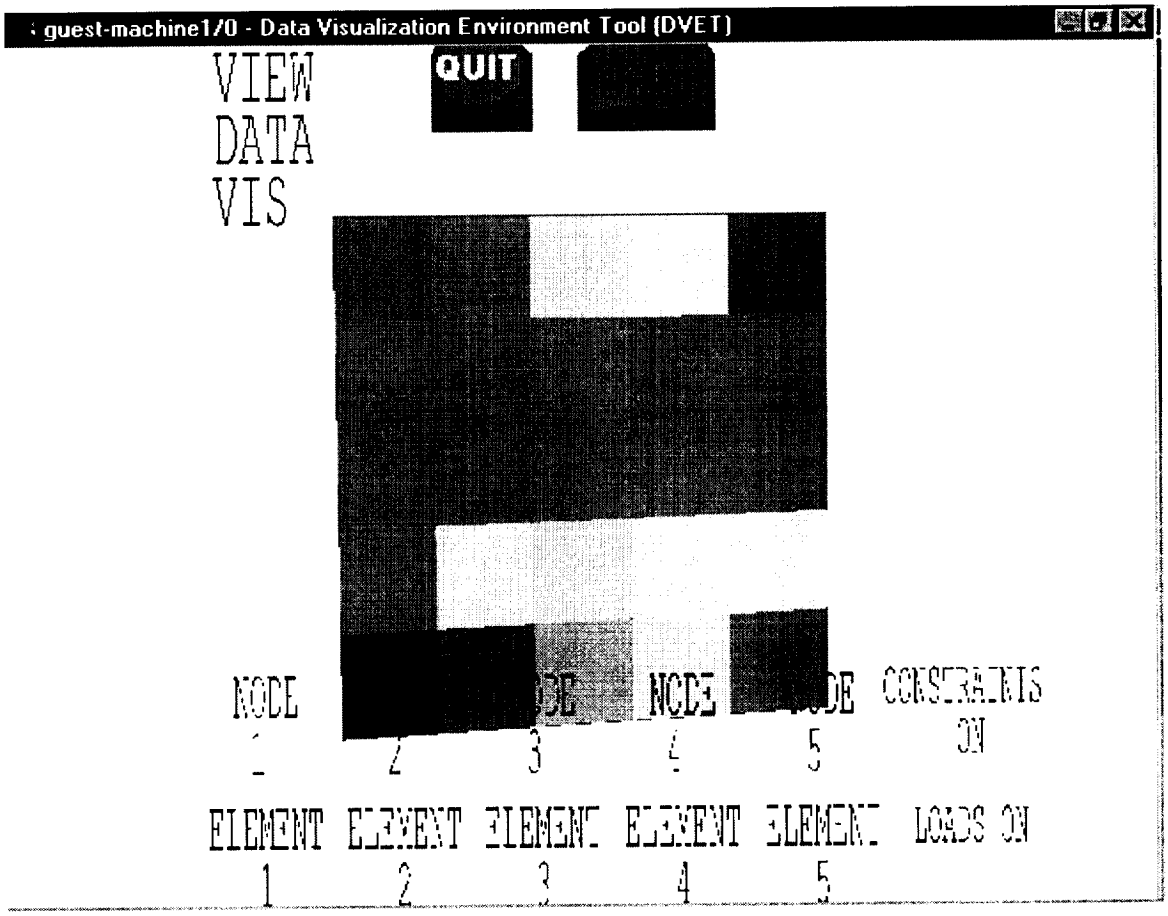


Figure 32. ELEMENT 4.

If the DATA ON button is pressed under the VIS button menu, when an output data set is selected information regarding the output data set is displayed. The information indicates the output data set name the case number it is associated with, etc.

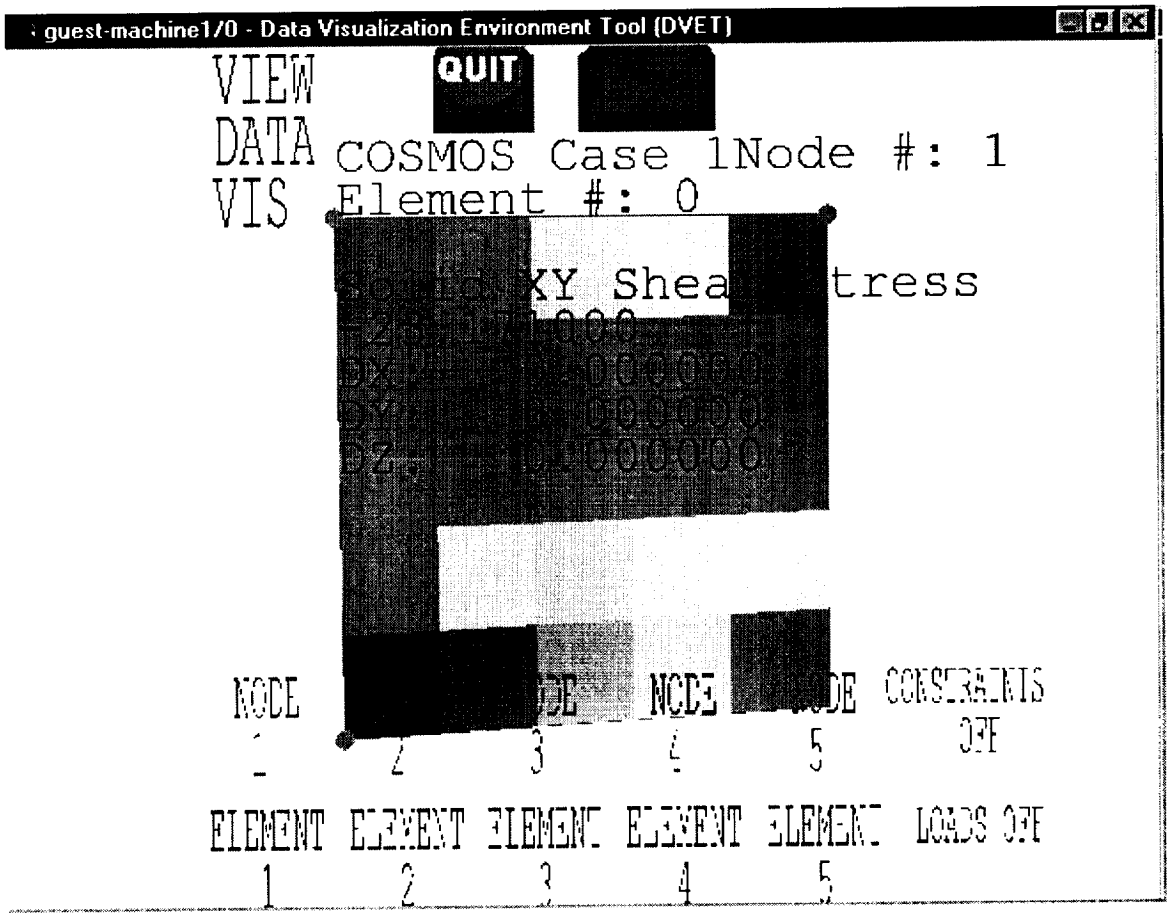


Figure 32a. ELEMENT 4 displayed with associated data information.

10. ELEMENT 5 is pressed.

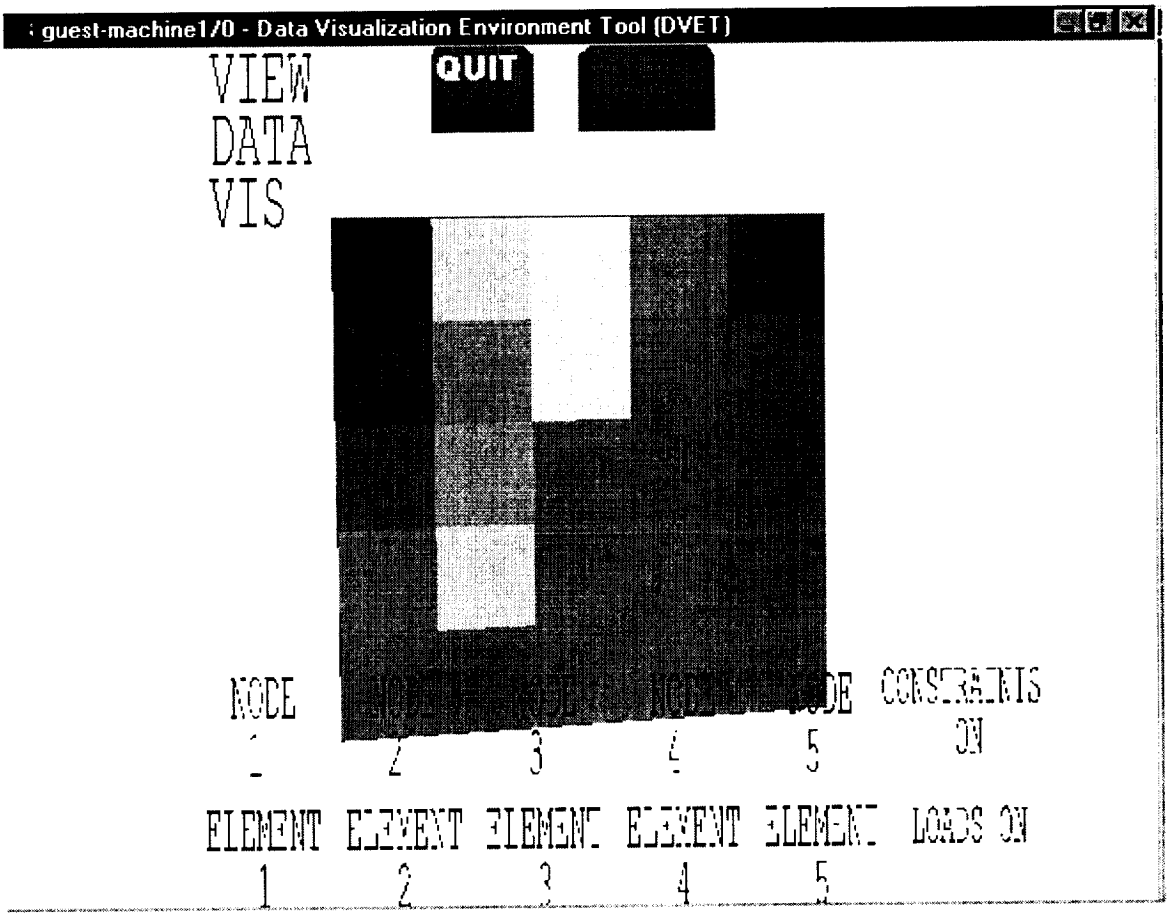


Figure 33. ELEMENT 5.

11. By pressing the CONSTRAINT ON button the constraints are selected. Constraints are visualized using green spheres.

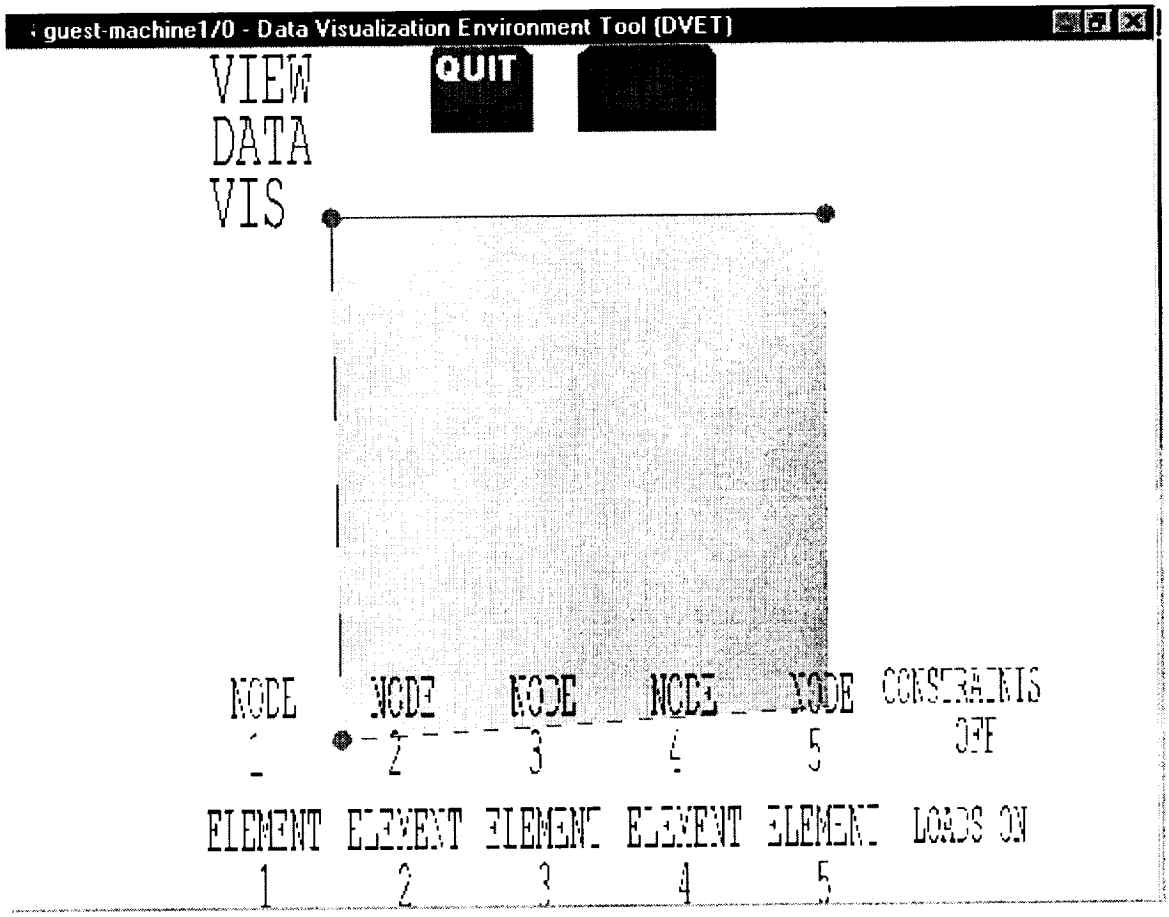


Figure 34. Constraints On.

12. If the DATA ON button is pressed under the VIS button menu, when a green sphere is selected information regarding the constraints is displayed. The information indicates the constraint set name, the node where the constraint is applied and the various constrained degrees of freedom.

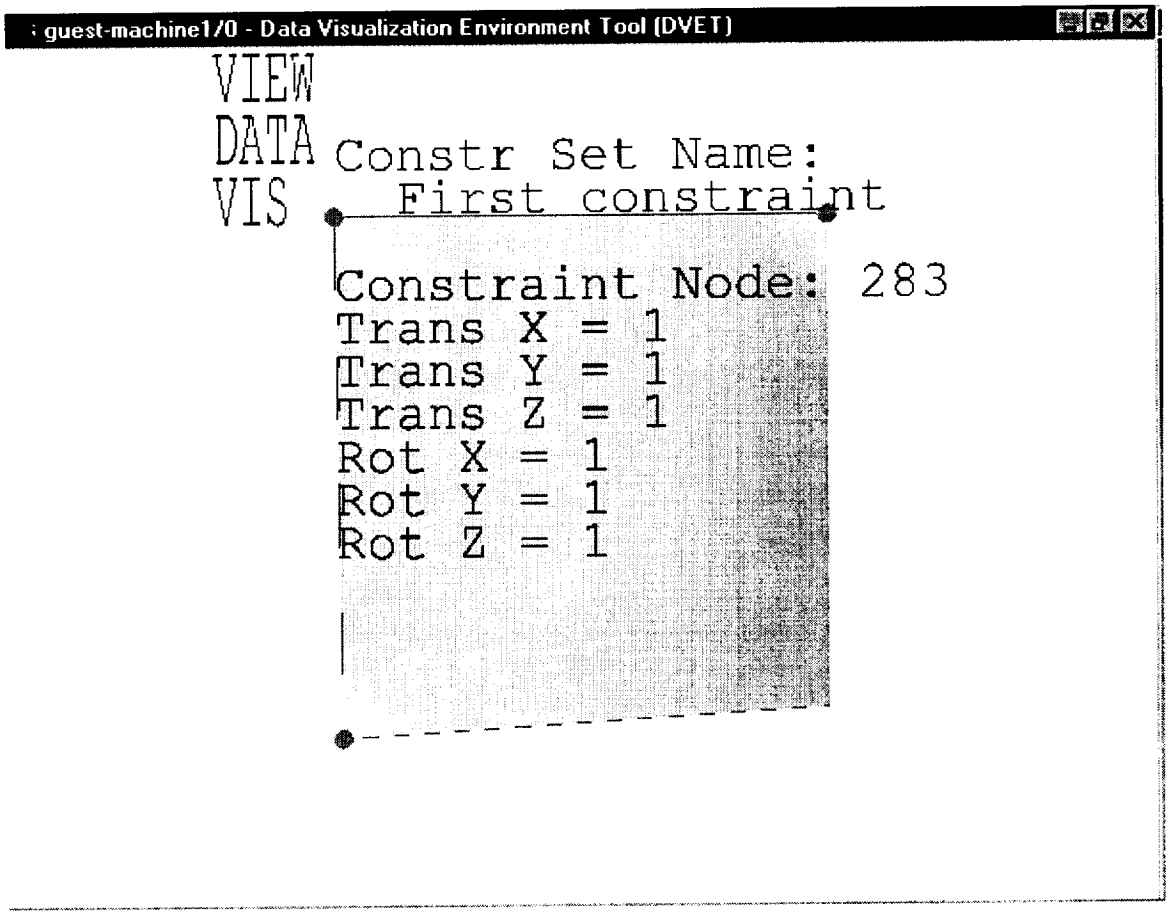


Figure 34a. Constraints On with associated data.

13. By pressing the CONSTRAINT ON button the constraints are selected. Constraints are visualized using green spheres.

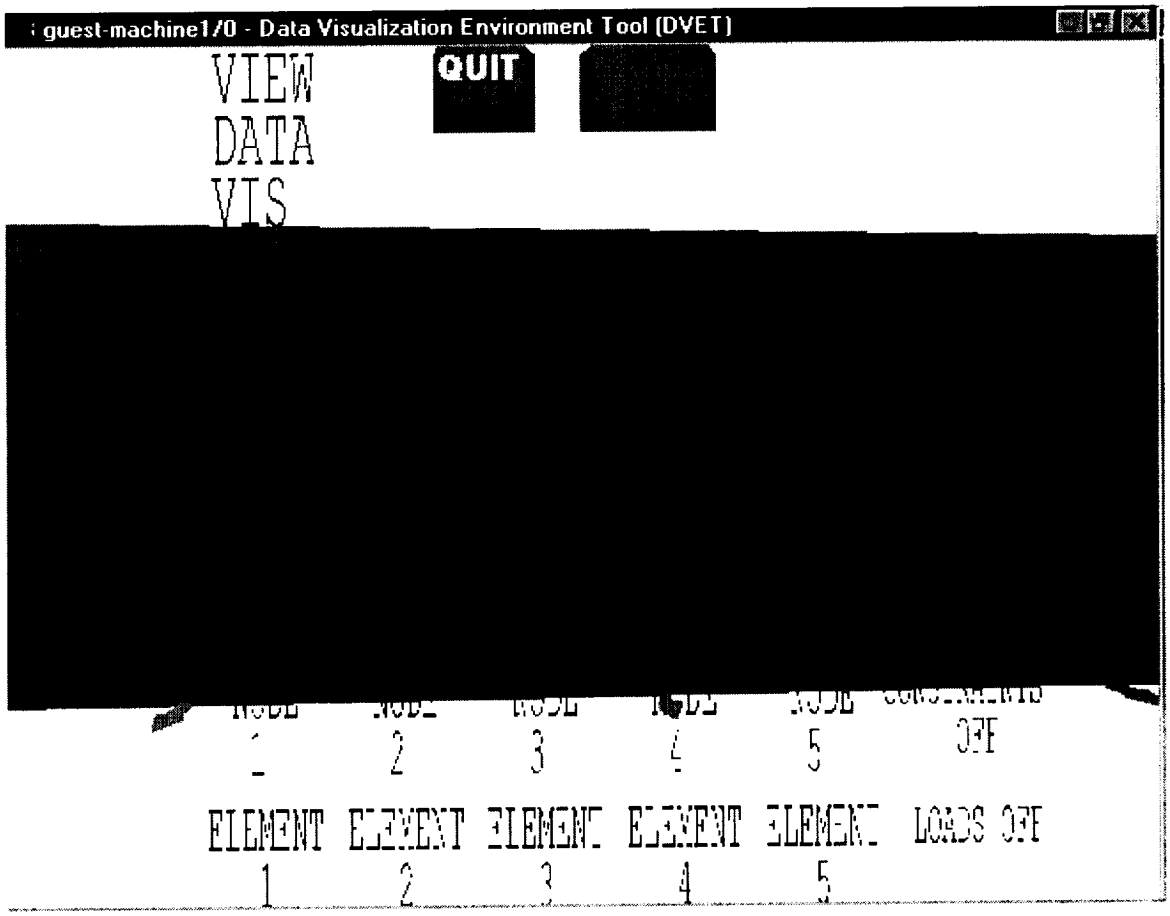


Figure 35. Loads On.

14. If the DATA ON button is pressed under the VIS button menu, when a green arrows is selected information regarding the load is displayed. The information indicates the load set name, the node where the load is applied, the load type and the various load/forces at the 6 degrees of freedom.

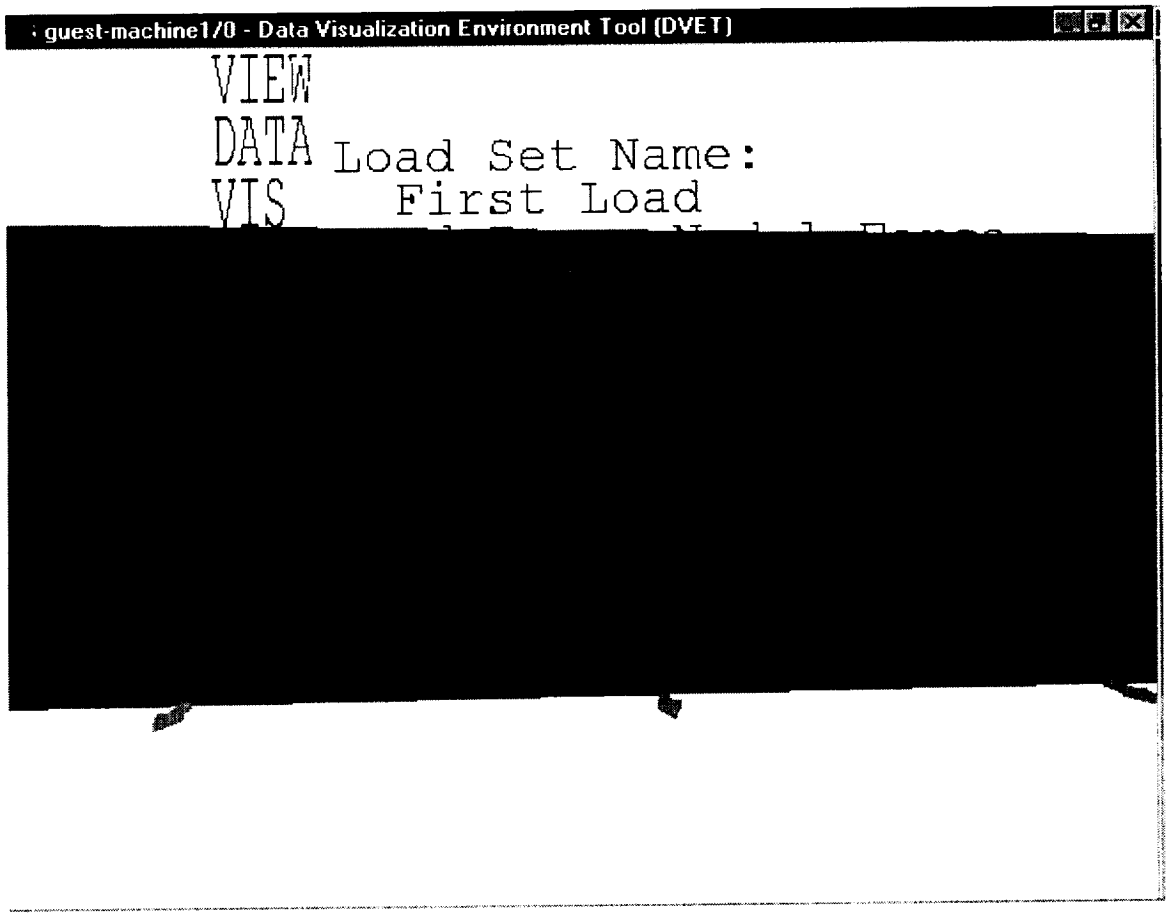


Figure 36. Loads On with associated data.

VIS Button

The VIS Button on the DVET system's toolbox is used to display the visual characteristics of a model. An interactive color scale, load level slider, Animation switch, etc., are all displayed on this screen. The Next Generation Space Telescope (NGST) is the model being displayed.

1. By selecting the VIS button, Figure 37 is displayed. The QUIT button also is displayed in order to exit from the VIS menu. A button with VISUALIZE is used to indicate the screen being displayed.

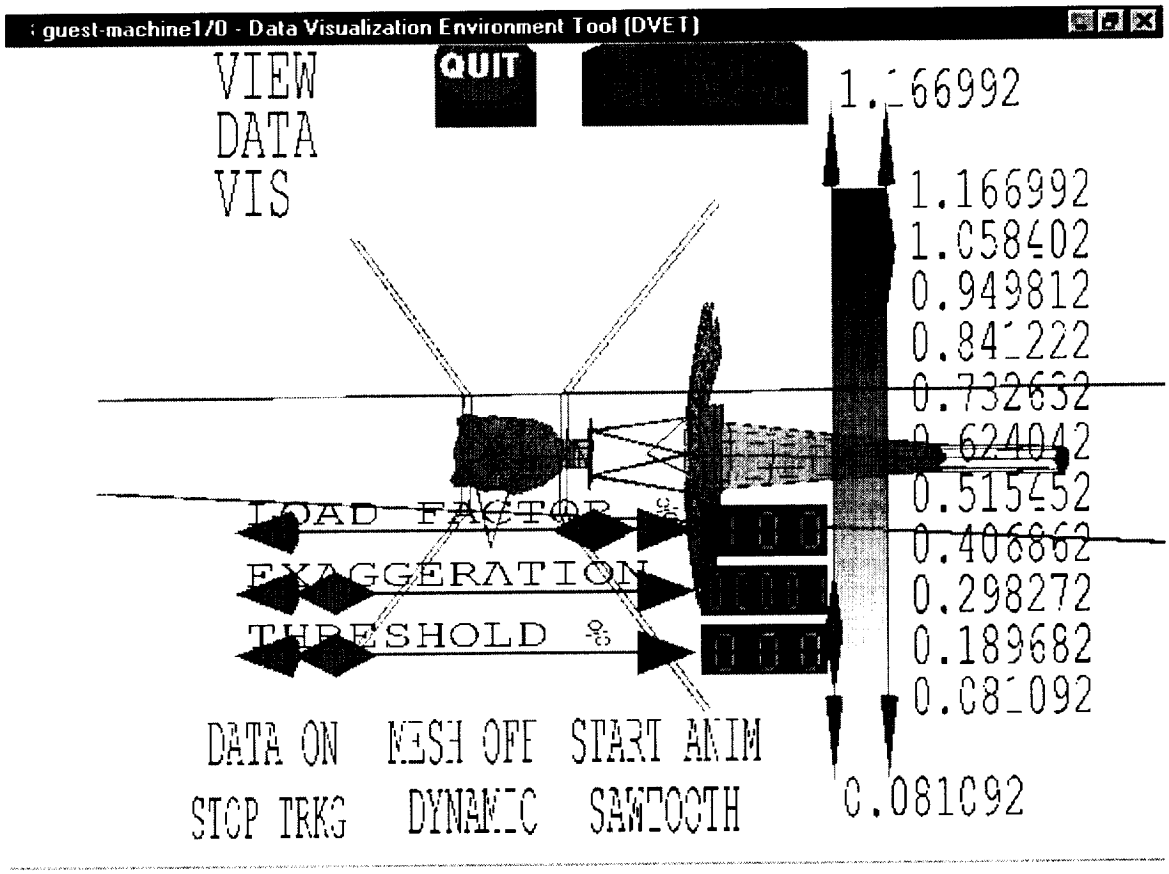


Figure 37. Initial Visualize Screen.

2. When the DATA ON button is pressed, data on various entities is displayed.

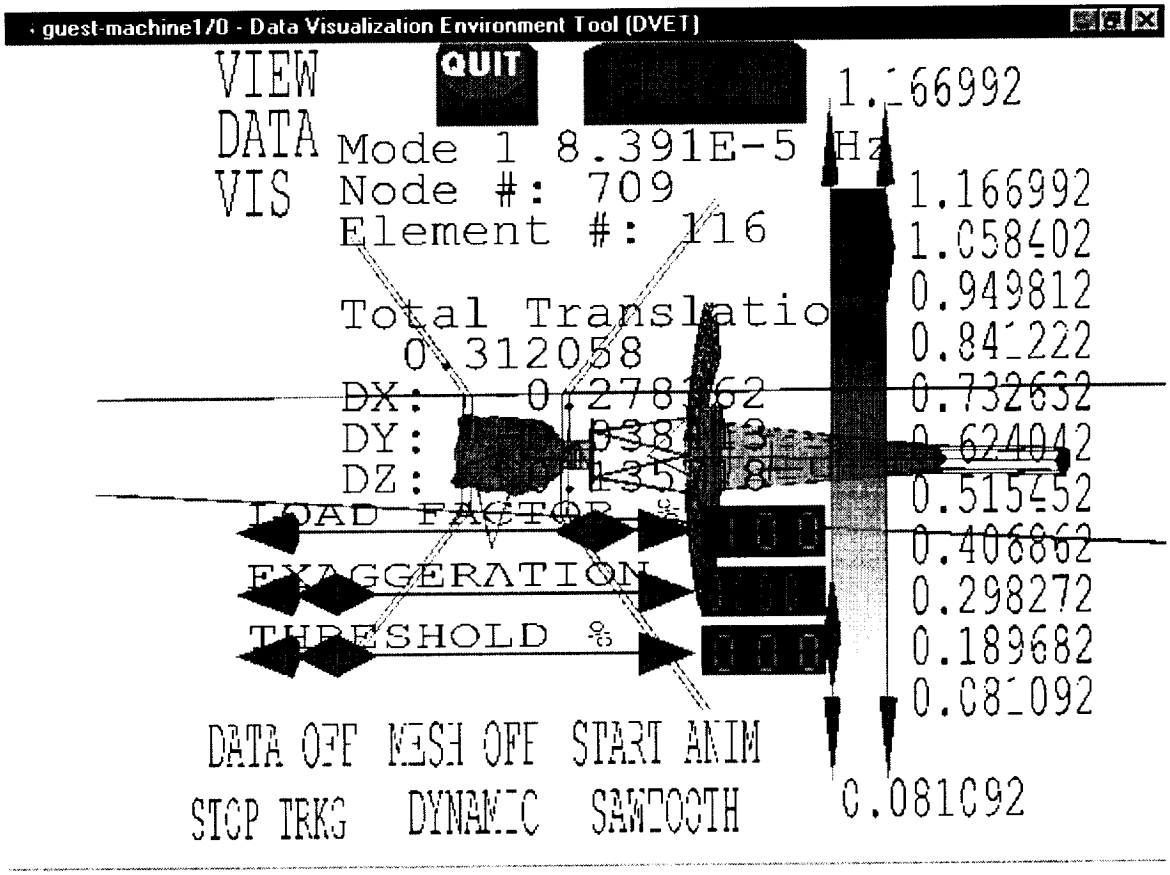


Figure 38. DATA ON.

3. By pressing the MESH OFF button you can eliminate the mesh on the model. This is sometimes useful when trying to view finite sections of the color scale.

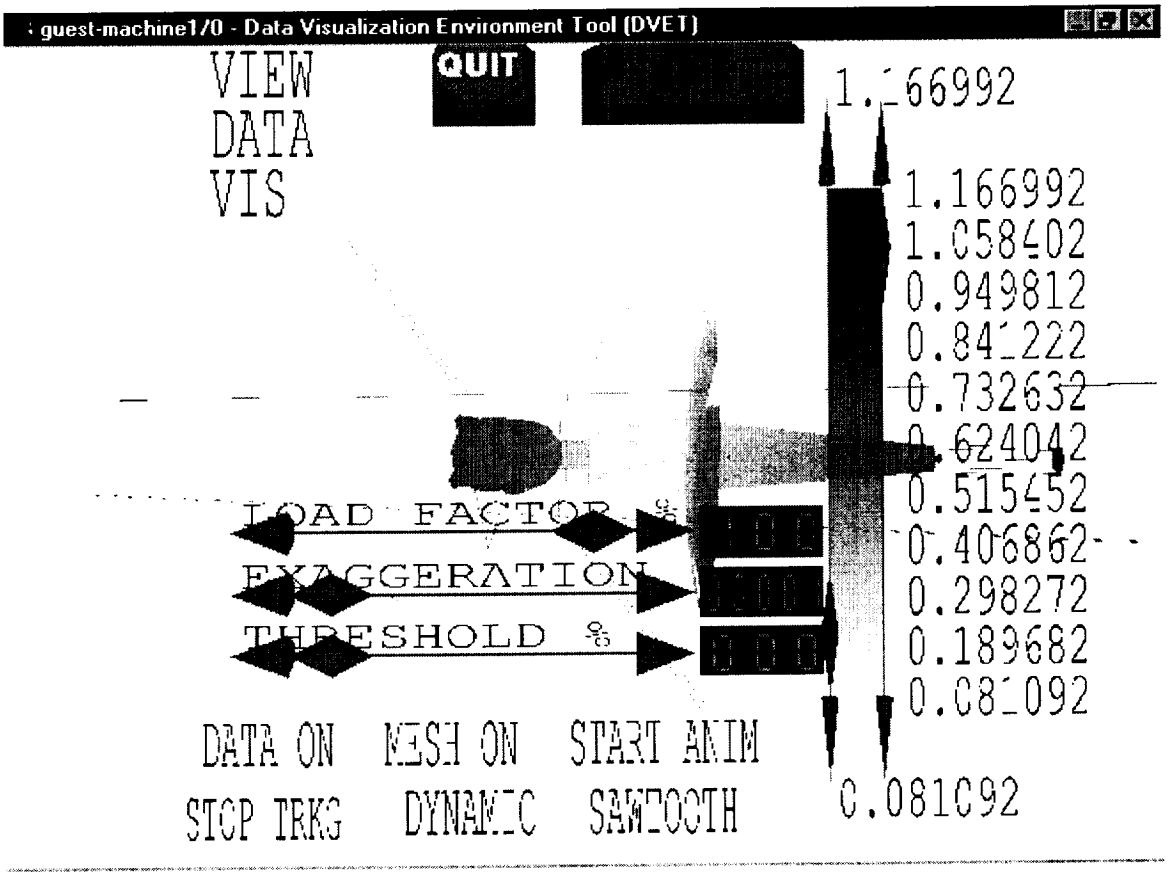


Figure 39. MESH OFF.

4. The START ANIM button is used to start an animation of the model. Geometrical and color changes are indicated in the diagram. A capture has been made as the model is being animated through the entire color scale.

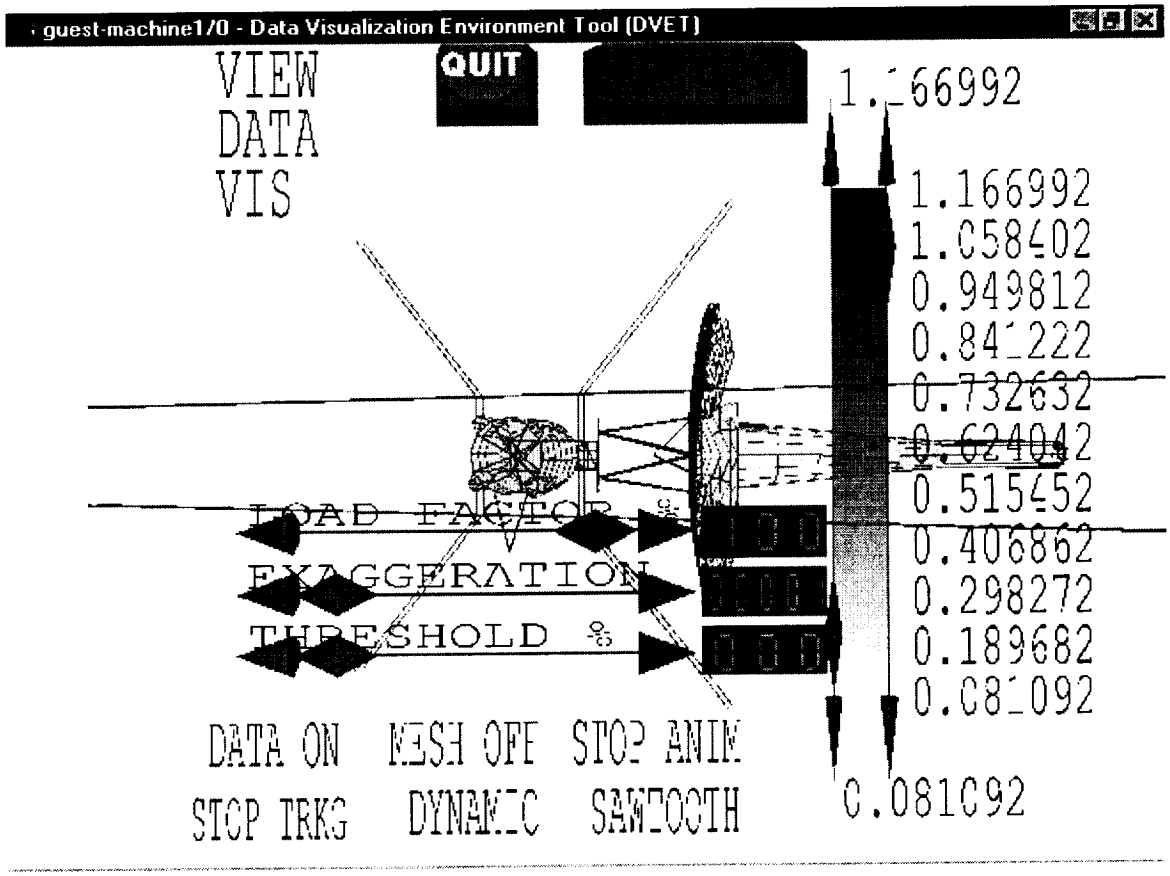


Figure 40. START ANIM.

5. The RAMP mode is an animation type. It allows the steady upward/forward change then it falls back to complete initiation before animation continues.

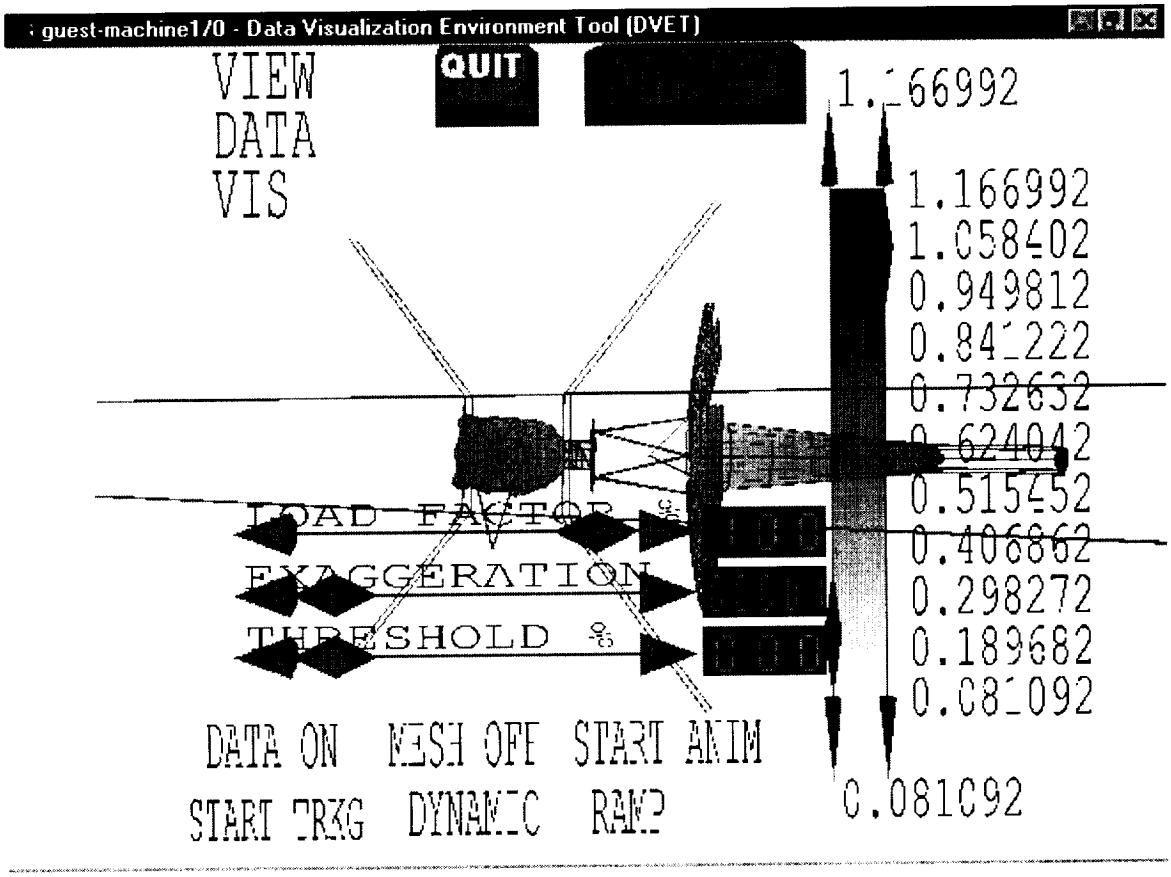


Figure 41. RAMP.

6. The Sawtooth mode is also part of animation. There is a gradual increase and a gradual decrease in the geometry and color of the mode.

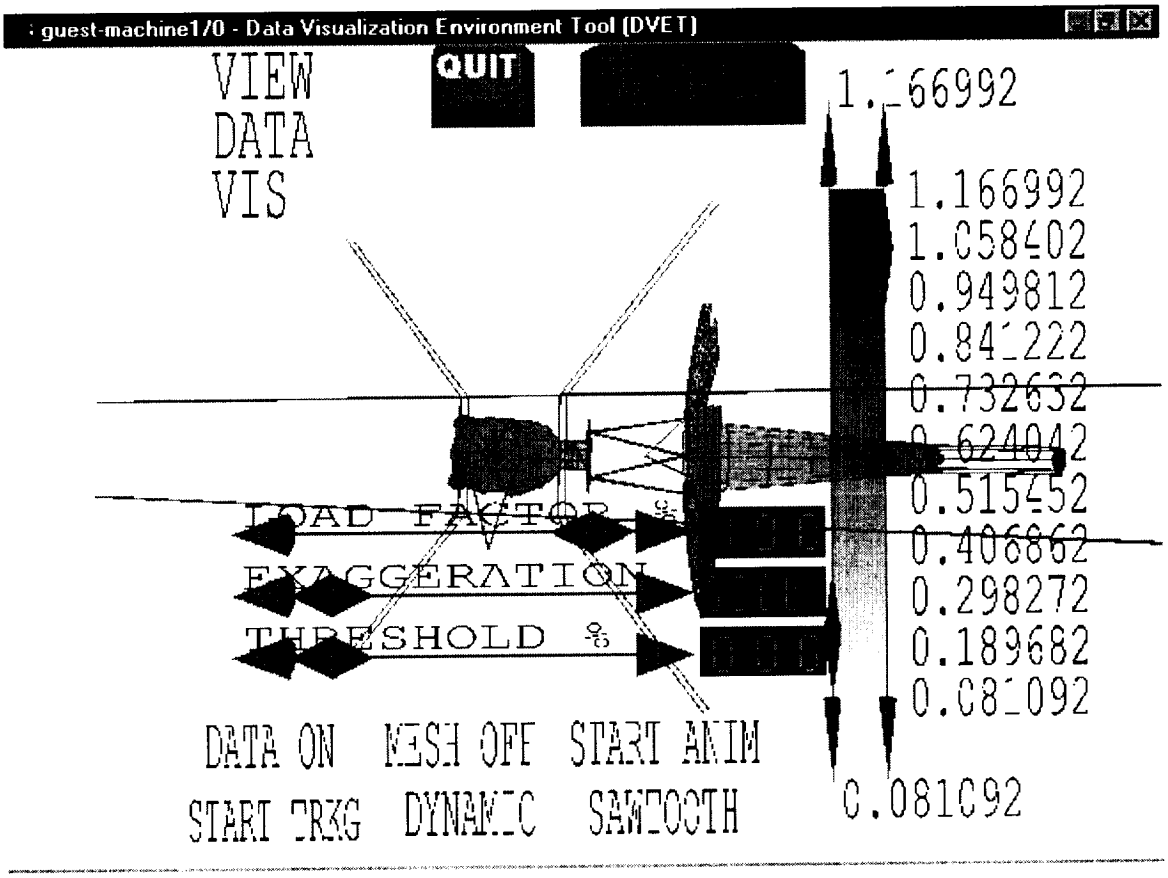


Figure 42. SAWTOOTH.

7. The STOP TRKG button allows you to temporarily prohibit tracking form the HMD. This is particularly beneficial when trying to stable the model in order to make a selection.

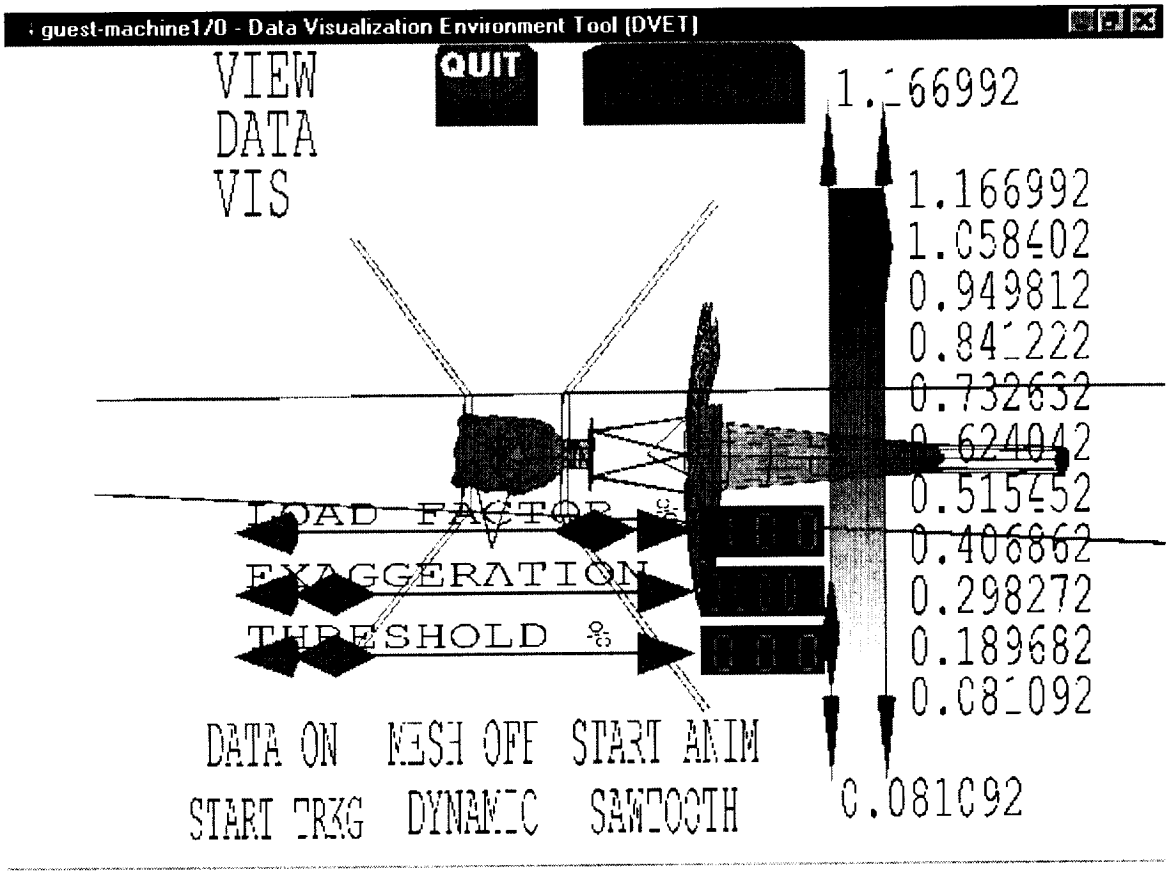


Figure 43. STOP TRKG.

8. The STATIC button allow prevents the model mesh from moving during the animation. This permits us the see the change that is happening and to what amount it is changing over the initial state of the model

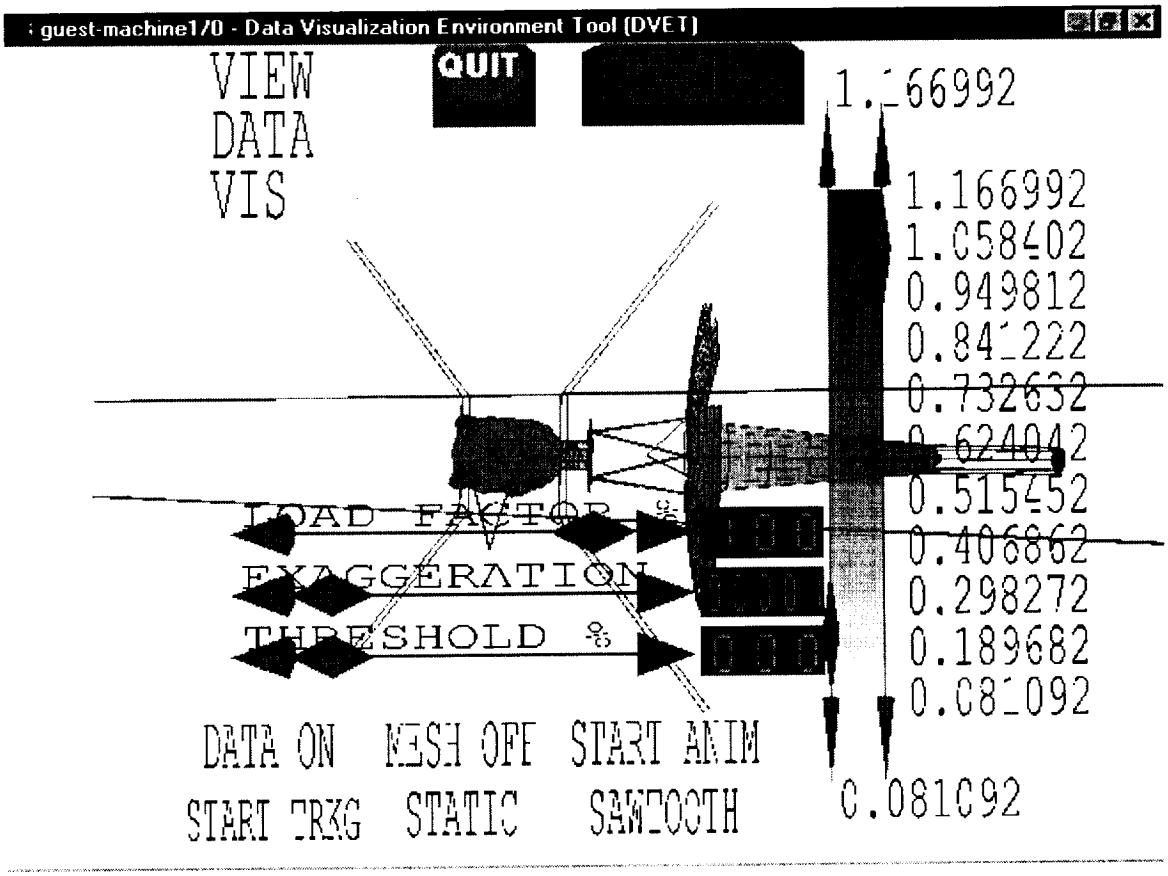


Figure 44. STATIC.

9. The DYNAMIC button permits the mesh to move in accordance to the animation of the model.

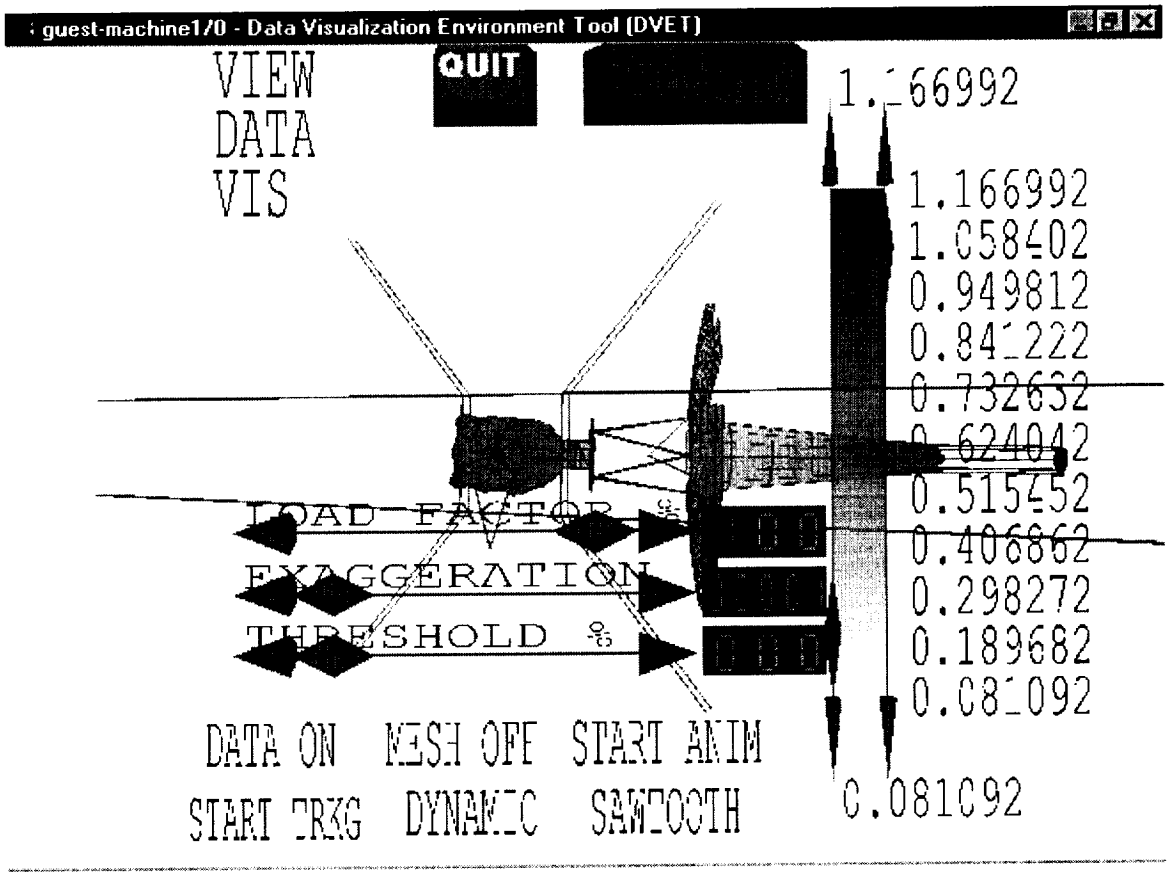


Figure 45. DYNAMIC.

10. Figure 46 shows the highlighted widget (green diamond). It is selected using the left mouse button and moved using the right mouse button.

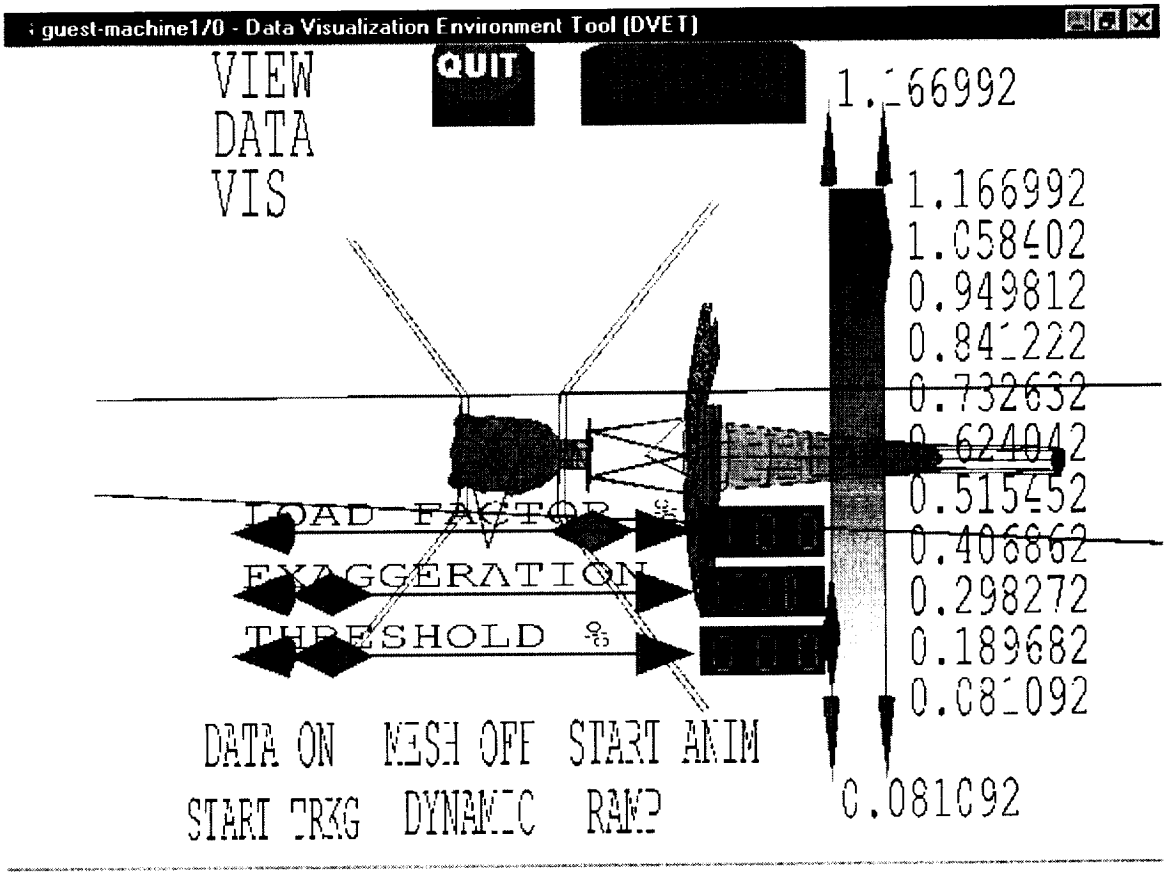


Figure 46. Load Factor 1.

11. Figure 47 indicates the final position of the load widget, observe the change in load to the model.

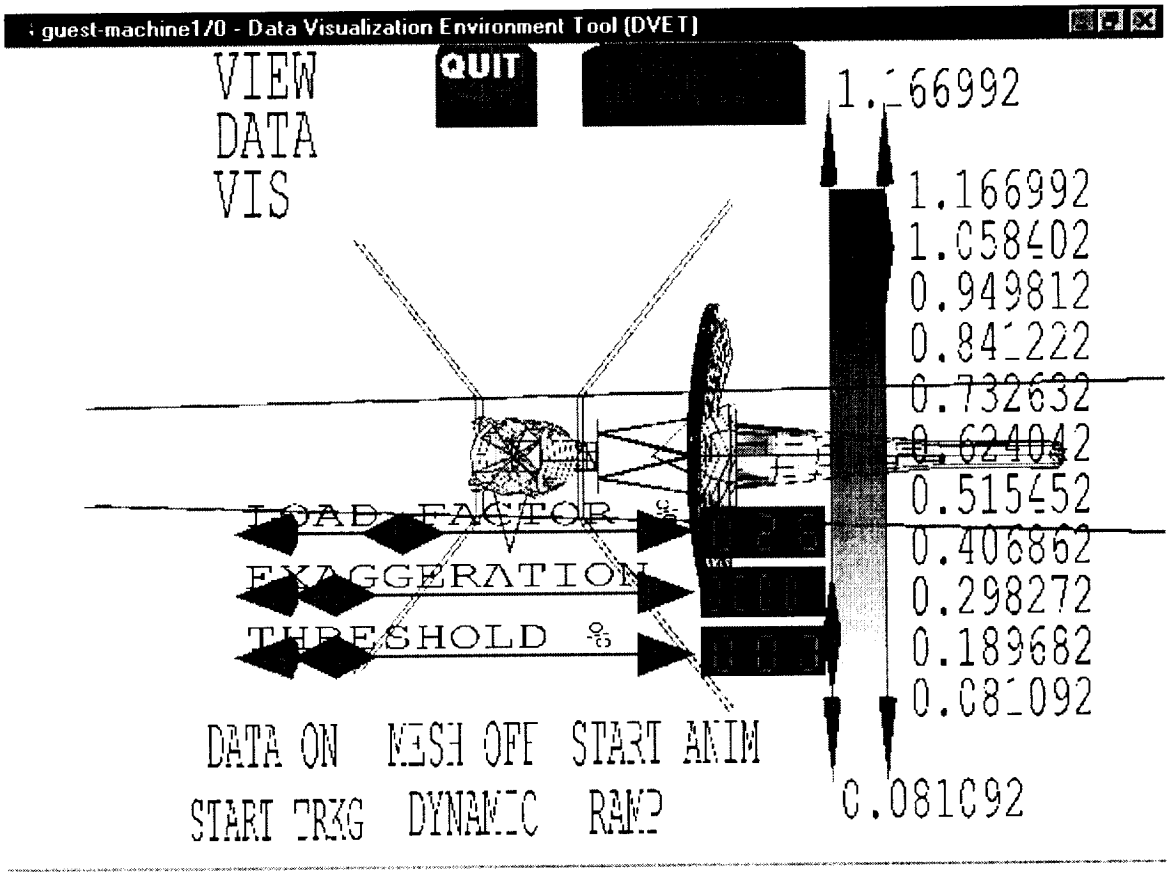


Figure 47. Load Factor 2.

12. Figure 48 shows the highlighted widget (green diamond) on the EXAGGERATION slider. It is selected using the left mouse button and moved using the right mouse button.

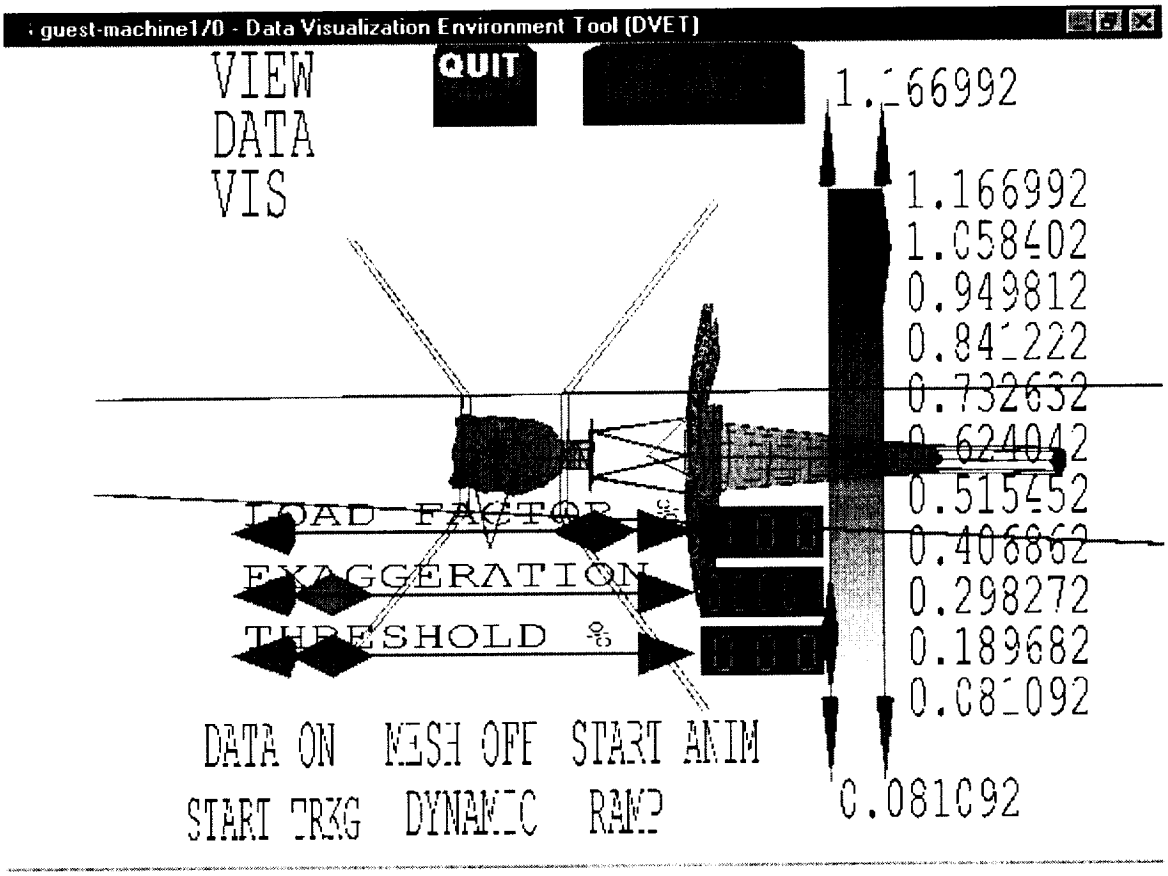


Figure 48. Exaggeration 1.

13. Figure 49 indicates the final position of the exaggeration widget, observe the change in exaggeration to the model.

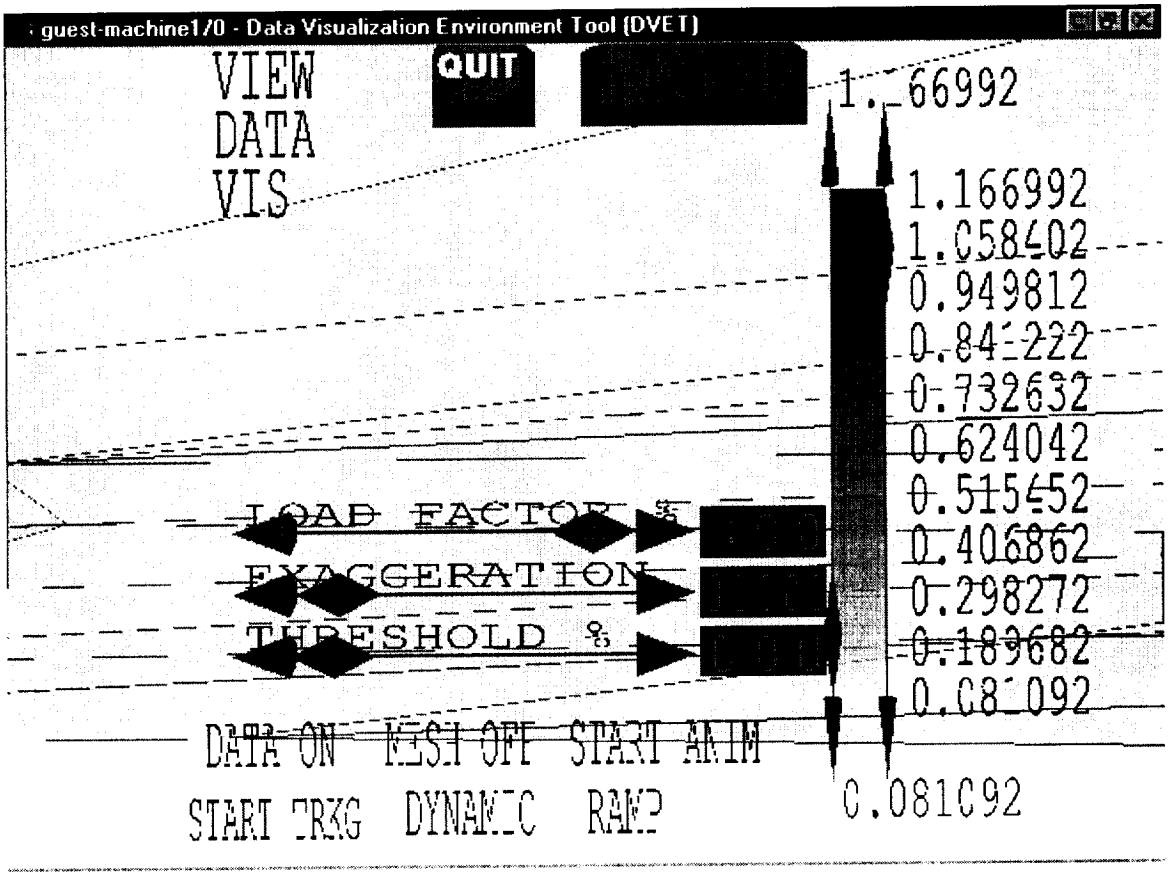


Figure 49. Exaggeration 2.

14. Figure 50 shows the highlighted widget (green diamond) on the THRESHOLD slider. It is selected using the left mouse button and moved using the right mouse button.

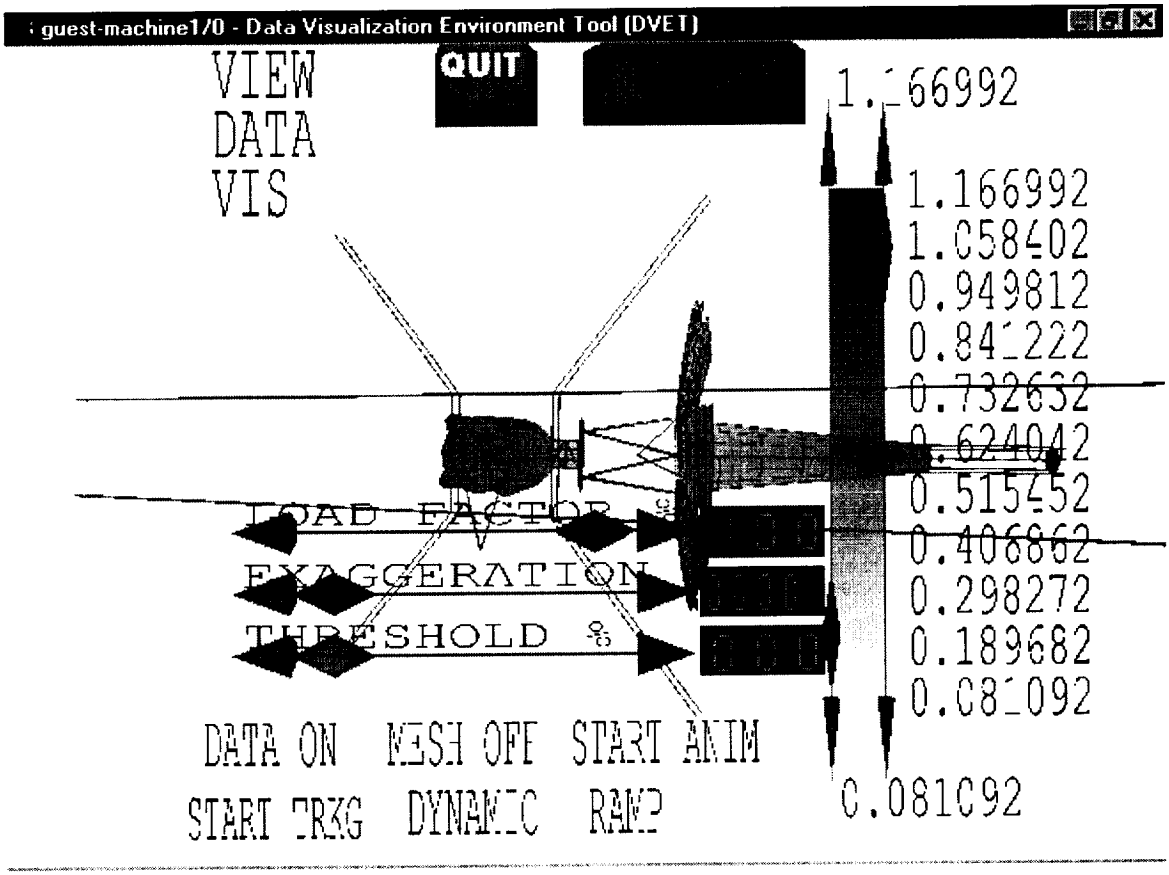


Figure 50. Threshold 1.

15. Figure 51 indicates the final position of the THRESHOLD widget, observe the change in threshold to the model.

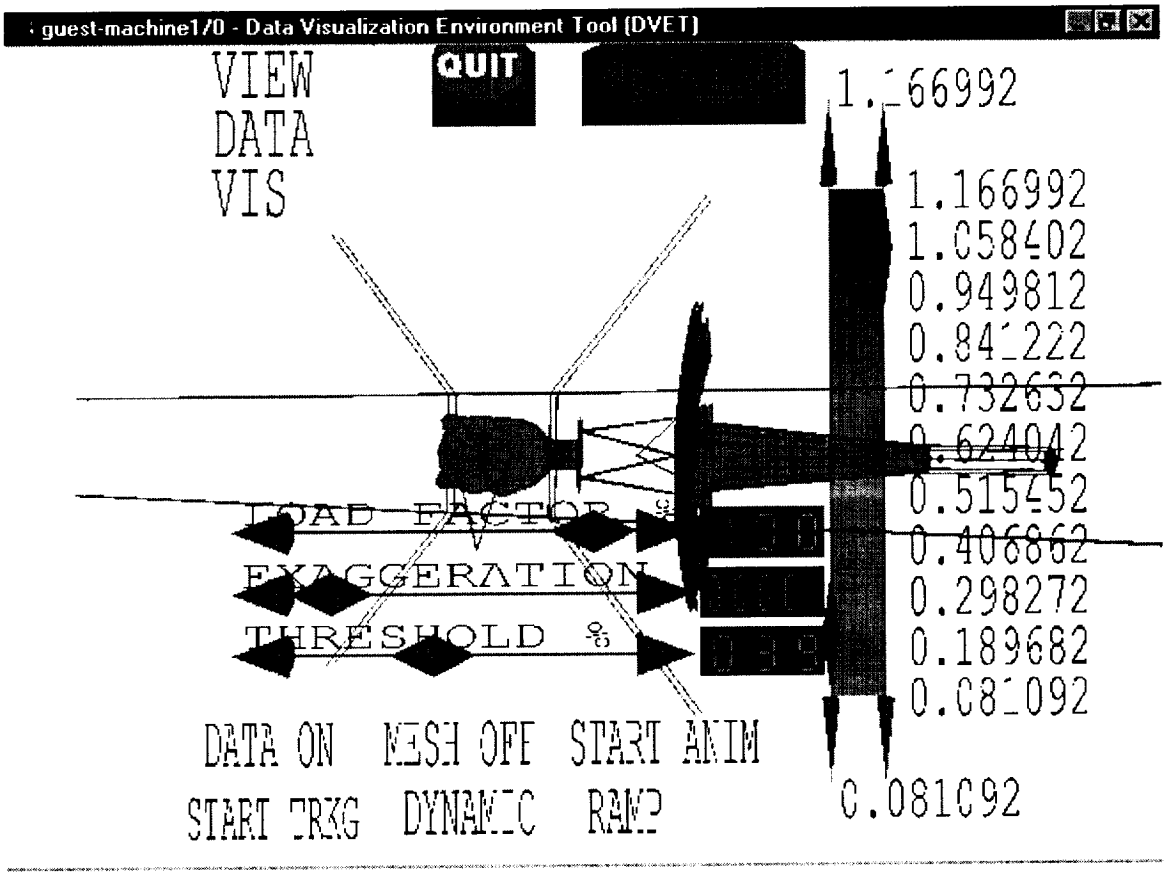


Figure 51. Threshold 2.

16. Figure 52 shows the initial position and color on the color scale. It is selected using the left mouse button and moved using the right mouse button.

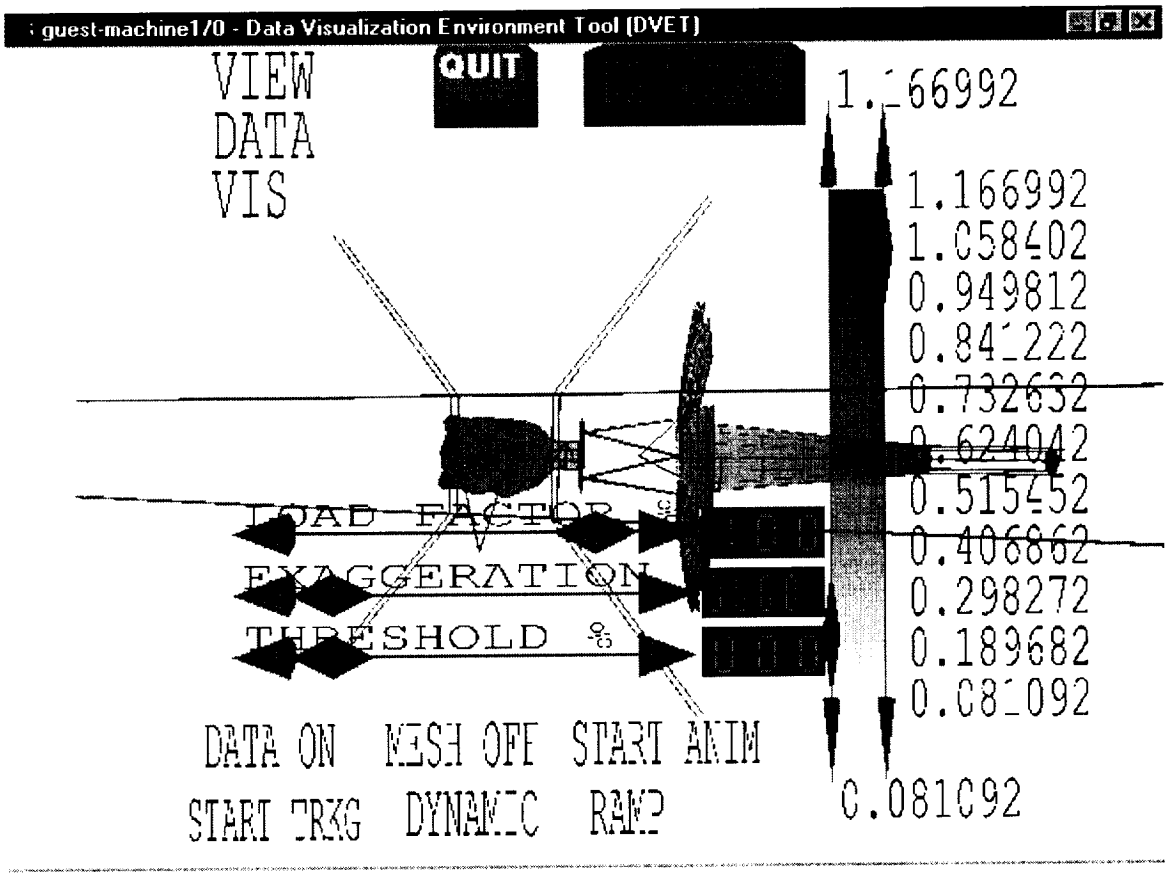


Figure 52. Color Scale 1.

17. Figure 51 indicates the final position of the COLOR SCALE widget, observe the change in color to the model.

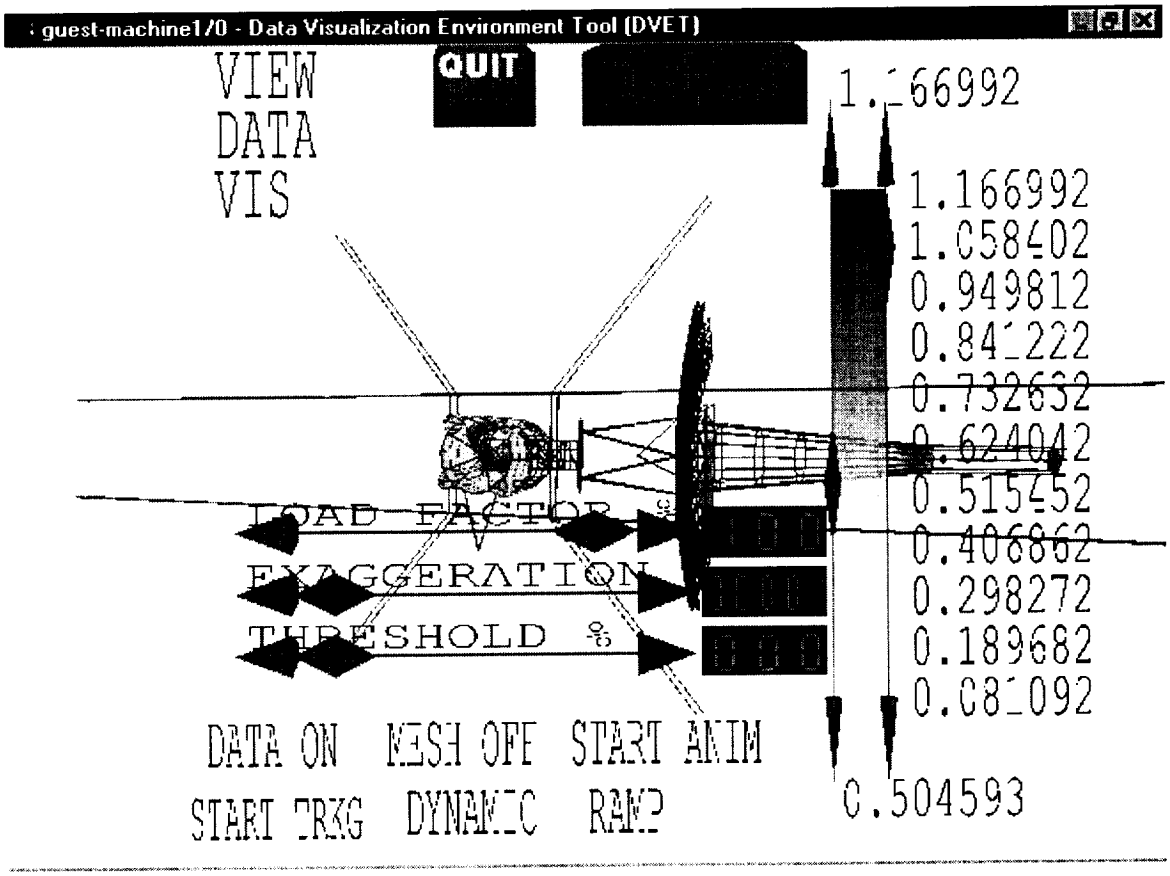


Figure 53. Color Scale 2.

APPENDIX D -

TECHNOLOGY TRANSFER OPPORTUNITY LETTER

National Aeronautics and
Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Reply to Attn of: 721

May 23, 1997

TO: Office of the Secretary of Defense, Operational Test and Evaluation, Live Fire Testing (Attn.: Mr. James F. O'Bryon)

FROM: 721/Systems Analysis Branch

SUBJECT: Technology Transfer Opportunity in Data Visualization

1. I want to bring to your attention work that we are doing in data visualization that, I believe, has application to live fire testing within the Department of Defense. Of particular note is our Integrated Data Visualization and Virtual Reality Tool Project, conducted under a SBIR NASA contract NAS5-33215.

2. This project, currently in the second phase of the SBIR, will allow NASA engineers and scientists to visualize complex multidimensional and multivariate data in a dynamic virtual environment. The objectives are to demonstrate the transfer and manipulation of standard engineering data in a virtual world, demonstrate the effects of design changes using finite element analysis tools and determine the training and engineering design and analysis effectiveness of the visualization system. The virtual reality system will operate on a personal computer workstation.

3. A potential use of this system for live fire testing and training is described as follows:

An observer is positioned in a simulated vehicle (tank), fully immersed in the virtual world through a head-mounted display with full audio and the ability to move within and outside the vehicle. Simulated rounds are fired at the vehicle and, through enhanced Finite Element Analysis (FEA) visualization tools, the observer can witness penetration of the simulated rounds (and fragmentation) and assess the damage to the vehicle. By slowing down the simulated penetration, the observer has visual information including geometry deformations and color-coded data that depicts in three dimension structural damage through physics based models. This visual information is interactively linked to underlying live fire FEA simulation data

bases. Using a joystick, the observer moves within the vehicle to gain different perspectives of incoming round penetrations, move outside the vehicle for an exterior view and transport to a "boundary" view if desired. Design changes to the armor, for example, can be assessed using this system. This data can also be used to determine personnel casualties and ultimately survivability of the vehicle for further use by the training community. The enclosed figure gives an illustration of a sample immersive visualization showing potential live fire effects sustained by an M1 Abrams tank turret. Similar to finite element analysis, colored regions can indicate simulated effects of external conditions (in this case live fire impact) on a

structure, which can be immersively explored and interactively linked to the underlying data.

4. I understand that there is an interest in using modeling and simulation to support testing and training. This NASA-developed technology will provide a unique and cost effective way of visualizing simulated live fire with some modifications. I will be pleased to have our contractor provide you additional information on this project and meet with you at your convenience to transfer this technology.

Timothy M. Carnahan

Enclosures: 1

APPENDIX E -

DVET SOURCE CODE LISTINGS

DVET source code and executable files for Silicon Graphics Workstation and Windows NT Workstation are attached as zipped archives to this draft final report using one standard PC floppy disk electronic media. A directory listing for this electronic media are shown below:

```
 Directory of A:\

03/30/98  10:40a      <DIR>          dvetsrc
          1 File(s)              0 bytes
                       971,776 bytes free

Directory of A:\dvetsrc

03/30/98  10:40a      <DIR>          .
03/30/98  10:40a      <DIR>          ..
03/30/98  10:40a      <DIR>          sgisrc
03/30/98  10:40a      <DIR>          winsrc
          4 File(s)              0 bytes
                       971,776 bytes free

Directory of A:\dvetsrc\sgisrc

03/30/98  10:40a      <DIR>          .
03/30/98  10:40a      <DIR>          ..
02/18/98  02:53p               2,379 fm2vr1120.h
03/03/98  04:36p             150,492 DVETSGI_.C
02/18/98  02:53p              79,795 FM2VRSGI.C
          5 File(s)        232,666 bytes
                       971,776 bytes free

Directory of A:\dvetsrc\winsrc

03/30/98  10:40a      <DIR>          .
03/30/98  10:40a      <DIR>          ..
01/21/98  10:01a               2,384 fm2vr1120.h
02/26/98  01:50p              59,031 FM2VRWIN.C
03/03/98  04:36p             161,273 DVETWIN.C
01/16/98  04:47p              17,474 Prompts.c
10/23/97  11:43a               9,242 inside1120.c
10/23/97  11:17a                 104 inside1120.h
          8 File(s)        249,508 bytes
                       971,776 bytes free
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
9602.024		

4. Title and Subtitle

Integrated Data Visualization and Virtual Reality Tool

5. Report Date
30 March 1998

6. Performing Organization Code
R & T

7. Author(s)

David A. Dryer, Ph.D.

8. Performing Organization Report No.
9602.024

10. Work Unit No.

9. Performing Organization Name and Address

Dual, Incorporated
30 Skyline Drive
Lake Mary, FL 32746

11. Contract or Grant No.
NAS5-33215

13. Type of Report and Period Covered
Final Report
20 February 1996 - 20 March 1998

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

14. Sponsoring Agency Code
721.1

15. Supplementary Notes

Phase II of NASA Small Business Innovation Research Project

16. Abstract

This final report summarizes progress for the 20 February 1996 - 20 March 1998 time period for the Phase II effort of SBIR Topic 05.05 of NASA solicitation 94.1.

17. Key Words (Suggested by Author(s))

Data Visualization
Virtual Reality

18. Distribution Statement
Contracting Officer (212) - 1 copy
Mr. Tim Carnahan (721) - 2 copies
SBIR Program Office (702) - 1 copy

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	94	

NASA FORM 1626 OCT 86

```
/*****************************************************************************
DVET Release 2.2/11/98 for SGI Workstation
dvetsgi.c
11 February 1998
Copyright 1998
Dual Incorporated

        di_add_vertex_color()
        di_animalarm()
        di_animTimer()
        di_create_body_handler()
        di_det_blocks()
        di_FEM_interact()
        di_input_nodes()
        di_input_mods()
        di_intersect_handler()
        di_modify_FEM()
        di_modify_Mesh()
        di_output_mods()
        di_Pmesh_mesh()
        di_Pmesh_obj()
        di_set_range()
        diBodyMoveToFunc()
        diBodyStartupPosFEMFunc()
        diCreateFEMMeshFunc()
        diCreateObjectFunc()
        diCreateTextFunc()
        diImmersDataFunc()
        diNavModeFunc()
        diOutputSetFunc()
        diSetViewFunc()
        diToggleAnimFunc()
        diToggleAnimModeFunc()
        diToggleMeshDynFunc()

AUTHOR:  David A. Dryer, Dual Incorporated

        RegisterScaleToolFunctions()
        ResetSliders_cb()
        SetSliders_cb()
        ToolCreation_cb()
        UpdateSlider_cb()
        UpdateSliderInfo_cb()
        WidgetCreation_cb()

Initial CAU prototype integration of dVS/dVISE widgets: Dr. Sriprakash Sarathy,  Clark Atlanta University 4/29/97
DVET modifications and widget additions: Dr. David Dryer, Dual Incorporated

        fem2vr()
        FindEid()
        FindNid()

Author:  Dr. Baojiu Lin, University of Central Florida
DVET integration and integration modifications: Dr. David Dryer, Dual Incorporated

*****************************************************************************************************/

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#ifdef _UNIX
#include <unistd.h>
#endif /* _UNIX */

#include <dvs/vc.h>
#include <dvise/dvise.h>

//include for FEM2VR translator
```

```
#include "fm2vrsgi.h"

#define min(a,b) ((a)<(b)?(a):(b))
#define max(a,b) ((a)>(b)?(a):(b))

/* PRIVATE STRUCTURES ================================================== */
/*
 * This structure is created by the 'myToolCreation' function and used
 * to store references to the widgets in the interface. These references
 * are filled in by the 'myWidgetCreation' function which is called by
 * each widget when it is created. These references allow the values of the
 * widgets to be set by other functions within the interface since this
 * data structure can be accessed via the Toolbox Used Data.
 */
typedef struct _SliderDataStruct {
  VWidget *LoadFact;
  VWidget *LoadDisp;
  VWidget *ThreshFact;
  VWidget *ThreshDisp;
  VWidget *ExagerFact;
  VWidget *ExagerDisp;
  VWidget *ClrSclTop;
  VWidget *ClrSclTopDisp;
  VWidget *ClrSclBot;
  VWidget *ClrSclBotDisp;
} SliderDataStruct;

typedef struct _intersectArgs
{
   uint32  *event;
   ECObject *object;
} intersectArgs;

typedef struct _MoveInfo {
   VCBody  *body;
   dmPoint  posa;
   dmPoint  posb;
   dmPoint  velocity;
   dmPoint  bodyOffset;
   float32  time, totalTime;
   int32    active;
   ECStateType state;
} MoveInfo;

typedef struct _PmeshInts {
            uint32      noVertices;
            uint32      noVertmesh;
            uint32      noFaces4;
            uint32      noFaces3;
            uint32      rightvert;
   uint32     rightelem;
   uint32     adjindex;
} PmeshInts;

typedef struct _Switches {
            uint32      navstate;
            uint32      navmode;
            uint32      set1;
            uint32      set2;
            uint32      picknode;
            uint32      meshdynmode;
            uint32      outtypenum;//0 is node type output, 1 is element type output
            uint32      outsubnum;//node or element subtype index (0-4) in output array
            uint32      animmode;
            uint32      startanim;
            uint32      loadcasestate;
            uint32      constraintstate;
} Switches;

typedef struct _Points {
```

```
        dmPoint         FEMcenter;
        dmPoint         view1;
        dmPoint         view2;
        dmPoint         rightnodep;
        dmPoint         loadnodep;
} Points;

typedef struct _Floats {
        float32 out_vals[5];
        float32 out_min;
        float32 out_max;
        float32 absmax;
        float32 threshold;
        float32 scale;
        float32 LoadFactor;
        float32 transp;
        float32 exager;
        float32 curout;
        float32 beamdelta;
        float32 xyzmax;
        float32 clrscltop;
        float32 clrsclbot;
        float32   femsclbotl[3];
        float32   femsclbotr[3];
        float32   femscltopr[3];
        float32   femscltopl[3];
        float32 alphainrng;
        float32 alphathresh;
        float32 alphaoutrng;
} Floats;

typedef struct _Chars {
        char    outtxt[200];
        char    scltxt[200];
} Chars;

typedef struct _VCfloats {
        VCColor  vcolour;
        VCColor  posmaxcolor;
        VCColor  posmincolor;
        VCColor  negmaxcolor;
        VCColor  negmincolor;
        VCColor  posthreshcolor;
        VCColor  negthreshcolor;
        VCColor  outofrngcolor;
} VCfloats;

struct NAMES *names;//malloc

typedef struct myEntityList {
        VCEntity *nodeobj;
        VCAttribute *vis;
        dmPoint nodepoint;
//      VCVisual *vis;
        struct myEntityList *next;
} EntityList;


//_amblksiz=16384;


PmeshInts *pmi;//malloc
Points          *points;//malloc
Switches  *switches;//malloc
Floats          *floats;//malloc
Chars           *chars;//malloc
VCfloats  *vcfloats;//malloc
float32         *vertices, *vertmesh;//malloc
float32         *displaceobj, *displacemesh;//malloc
float32         *femsclverts, *femsclgrdverts;//malloc
```

```
uint32              *connections4, *connections3;//malloc
uint32              *femsclconts, *femsclgrdconts;//malloc
uint32              *conmesh4, *conmesh3;//malloc
float32             *outvert;//malloc
uint32              *elearray;//malloc
VCGeometry          *femtextstring;//malloc
VCGeometry          *clrscltextstring;//malloc
VCIntersectionReportData        *intersectionReportData;//malloc
uint32              *loadcoordind = NULL;
uint32              *loaddfind = NULL;
uint32              *loadtrack = NULL;
uint32              *constrcoordind;
uint32              *constrdfind;

static VCTime *syncTime=NULL;

EntityList *LoadList;

EntityList *ConstrList;


//VCColour    white={1,1,1},gray={0.5,0.5,0.5},black={0,0,0},red={1,0,0},yellow={1,1,0},blue={0,0,1},green={0,1,0};


ECObject  *objFEM, *objMesh, *objFEMText,//malloc
                            *objClrScl,*objClrSclGrid,*objClrSclText,
                            *objViewButton, *objViewText,
                            *objDataButton, *objDataText,
                            *objVisButton, *objVisText;

ECObjectReference   *objFEMref, *objMeshref, *objFEMTextref,//malloc
                                            *objClrSclref, *objClrSclGridref,*objClrSclTextref,
                                            *objViewButtonref, *objViewTextref,
                                            *objDataButtonref, *objDataTextref,
                                            *objVisButtonref, *objVisTextref;

//*************************function prototypes*************************//

int
di_create_body_handler(VCBodyCreate_CallbackData *bodyData, void *data);

//*********************************************************************//
// Function:  di_det_blocks
//*********************************************************************//

int di_det_blocks()
{
        int adj=0;
        int i,j,k;
        int         elemindex;
        uint32      *tracknode;
        dmPoint             beam1,beam2;
        dmVector  beamvect;
        float32             beamdist;

        pmi=(PmeshInts *)malloc(sizeof(PmeshInts));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;
        pmi->noVertmesh=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);
```

```
                    dmPointSet (beam2,
                            (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                            (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                            (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                    dmPointSub (beamvect, beam1, beam2);

beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                    if (beamdist > .000000001)
                    {
                            for (i=0; i<2; i++)
                            {
                                    if (i==0)
                                    {
                                            j=0;k=1;
                                    }
                                    else
                                    {
                                            j=3;k=2;
                                    }
                                    if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                    {
                                            tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                    }
                                    else
                                    {
                                            pmi->noVertices++;
                                    }
                                    pmi->noVertmesh++;
                                    pmi->noVertices++;
                            }
                            pmi->noFaces4++;
                    }
            }

//If element type is 4 nodes...
            if((ELEMENT_P+elemindex)->A==4)
            {
                    for (i=0; i<4; i++)
                    {
                            if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                            {
                                    tracknode[((ELEMENT_P+elemindex)->B[i])]=1;
                            }
                            else
                            {
                                    pmi->noVertices++;
                            }
                    }
                    pmi->noFaces4++;
            }

//If element type is 3 nodes...
            if((ELEMENT_P+elemindex)->A==3)
            {
                    for (i=0; i<3; i++)
                    {
                            if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;
                            if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                            {
                                    tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;
                            }
                            else
                            {
                                    pmi->noVertices++;
                            }
                    }
```

```
                                adj=0;
                                pmi->noFaces3++;
                        }
    }
        return 1;
}

//*************************************************************************//
// Function:  di_input_nodes
//*************************************************************************//

int di_input_nodes()
{
        int i,j;
        float32 xmax=0.0;
        float32 ymax=0.0;
        float32 zmax=0.0;
        float32 xmin=10000.0;
        float32 ymin=10000.0;
        float32 zmin=10000.0;

        vertices=malloc((pmi->noVertices*7)*sizeof(float32));
        vertmesh=malloc((pmi->noVertices*3)*sizeof(float32));
        displaceobj=malloc((pmi->noVertices*3)*sizeof(float32));
        displacemesh=malloc((pmi->noVertices*3)*sizeof(float32));
        loadcoordind=malloc((LOADSET_NUM)*sizeof(uint32));
        loaddfind=malloc((LOADSET_NUM)*sizeof(uint32));
        loadtrack=malloc((LOADSET_NUM)*sizeof(uint32));
        constrcoordind=malloc((CONSTRAINTSET_NUM)*sizeof(uint32));
        constrdfind=malloc((CONSTRAINTSET_NUM)*sizeof(uint32));

        for (i=0;i < NODE_NUM;i++)
        {

                //Vertex node coordinates - x,y,z assigned to vertices array elements
                //e.g., vertices[0,1,2...7,8,9...

                vertices[(i*7)+0]  = ((NODE_P+i)->x)*floats->scale;
    vertices[(i*7)+1]= ((NODE_P+i)->y)*floats->scale;
    vertices[(i*7)+2]= ((NODE_P+i)->z)*floats->scale;

                displaceobj[(i*3)+0]=((NODE_P+i)->dx);
    displaceobj[(i*3)+1]=((NODE_P+i)->dy);
    displaceobj[(i*3)+2]=((NODE_P+i)->dz);
    displaceobj[(i*3)+0]*=floats->scale;
    displaceobj[(i*3)+1]*=floats->scale;
    displaceobj[(i*3)+2]*=floats->scale;

                vertmesh[(i*3)+0]  = ((NODE_P+i)->x)*floats->scale;
    vertmesh[(i*3)+1]= ((NODE_P+i)->y)*floats->scale;
    vertmesh[(i*3)+2]= ((NODE_P+i)->z)*floats->scale;

                displacemesh[(i*3)+0]=((NODE_P+i)->dx);
    displacemesh[(i*3)+1]=((NODE_P+i)->dy);
    displacemesh[(i*3)+2]=((NODE_P+i)->dz);
    displacemesh[(i*3)+0]*=floats->scale;
    displacemesh[(i*3)+1]*=floats->scale;
    displacemesh[(i*3)+2]*=floats->scale;

// get min, max x,y,z values
                xmax=max(xmax,((NODE_P+i)->x)*floats->scale);
                xmin=min(xmin,((NODE_P+i)->x)*floats->scale);
                ymax=max(ymax,((NODE_P+i)->y)*floats->scale);
                ymin=min(ymin,((NODE_P+i)->y)*floats->scale);
                zmax=max(zmax,((NODE_P+i)->z)*floats->scale);
                zmin=min(zmin,((NODE_P+i)->z)*floats->scale);

                for (j = 0;j < LOADSET_NUM && j < 100;j++)
                {
                        if (LOAD_SET[LOADSET_PICK].TYPE[j] == 1)
```

```
                        {
                                if (LOAD_SET[LOADSET_PICK].ID[j] == (NODE_P+i)->A)
                                {
                                        loadcoordind[j] = i;
                                        loaddfind[j] = j;
                                }
                        }
                }
                for (j=0;j<CONSTRAINTSET_NUM;j++)
                {
                        if (CONSTRAINT_SET[CONSTRAINTSET_PICK].ID[j] == (NODE_P+i)->A)
                        {
                                constrcoordind[j] = i;
                                constrdfind[j] = j;
                        }
                }

        }

//get FEM center point
        points->FEMcenter[VC_X]=xmin+((xmax-xmin)/2.0);
        points->FEMcenter[VC_Y]=ymin+((ymax-ymin)/2.0);
        points->FEMcenter[VC_Z]=zmin+((zmax-zmin)/2.0);

// get max axis length
        if((xmax >= ymax) && (xmax >= zmax))
        {
                floats->xyzmax=xmax-xmin;
        }
        else if((ymax >= xmax) && (ymax >= zmax))
        {
                floats->xyzmax=ymax-ymin;
        }
        else
        {
                floats->xyzmax=zmax-zmin;
        }
return 1;
}

//*************************************************************************//
// Function:  di_set_range
//*************************************************************************//

int di_set_range()
{
        uint32 outindex,OUT_NUM;

        floats->out_min=100000;
        floats->out_max=-100000;

        if (switches->outtypenum==0)
        {
                OUT_NUM=NODE_NUM;
                for (outindex=0;outindex<OUT_NUM;outindex++)
                {
                        floats->out_min=min(floats->out_min,(NODE_P+outindex)->output_data[switches->outsubnum]);
                        floats->out_max=max(floats->out_max,(NODE_P+outindex)->output_data[switches->outsubnum]);
                }
        }
        else
        {
                OUT_NUM=ELEMENT_NUM;
                for (outindex=0;outindex<OUT_NUM;outindex++)
                {
                        floats->out_min=min(floats->out_min,(ELEMENT_P+outindex)->C[switches->outsubnum]);
                        floats->out_max=max(floats->out_max,(ELEMENT_P+outindex)->C[switches->outsubnum]);
                }
        }
```

```
//initially set threshold here
 floats->out_vals[0]=floats->out_min;
 floats->out_vals[1]=0.0;//threshold
 floats->out_vals[2]=floats->out_max;
 floats->absmax=max(fabs(floats->out_min),fabs(floats->out_max));

 return 1;
}

//*************************************************************************//
// Function:  di_add_vertex_color - sets vertex colours
//*************************************************************************//

int di_add_vertex_color(void)
{

//For each curout value, determines if out is positive and in set color region
   if(floats->curout>=0.0 && floats->curout>=floats->out_vals[0] && floats->curout<=floats->out_vals[2])
          {
//If positive and in color region, determines if out is under threshold level-assign positive threshold color
                        if(floats->curout<floats->threshold*floats->out_max)
                        {
                                vcfloats->vcolour[0]=vcfloats->posthreshcolor[0];
                                vcfloats->vcolour[1]=vcfloats->posthreshcolor[1];
                                vcfloats->vcolour[2]=vcfloats->posthreshcolor[2];
                                floats->transp=floats->alphathresh;
                        }
//If positive, in color region, and not under threshold level-assign color
                        else
                        {
                                vcfloats->vcolour[0]=vcfloats->posmincolor[0]+
                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                        (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
                                vcfloats->vcolour[1]=vcfloats->posmincolor[1]+
                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                        (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
                                vcfloats->vcolour[2]=vcfloats->posmincolor[2]+
                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                        (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
                                floats->transp=floats->alphainmg;
                        }
          }
 //For each curout value, determines if out is negative and in set color region
   else if(floats->curout<0.0 && fabs(floats->curout)>=floats->out_vals[0] && floats->curout>=floats->out_vals[0])
          {
 //If negative and in color region, determines if out is above threshold level-assign negative threshold color
                        if(floats->curout>floats->threshold*floats->out_min)
                        {
                                vcfloats->vcolour[0]=vcfloats->negthreshcolor[0];
                                vcfloats->vcolour[1]=vcfloats->negthreshcolor[1];
                                vcfloats->vcolour[2]=vcfloats->negthreshcolor[2];
                                floats->transp=floats->alphathresh;
                        }
 //If negative, in color region, and not above threshold level-assign color
                        else
                        {
                                vcfloats->vcolour[0]=vcfloats->negmincolor[0]+
                                                        ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
                                                        (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
                                vcfloats->vcolour[1]=vcfloats->negmincolor[1]+
                                                        ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
```

```
                                                                    (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
                        vcfloats->vcolour[2]=vcfloats->negmincolor[2]+
                                                                    ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
                                                                    (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
                        floats->transp=floats->alphainrng;
                }
        }
//For each curout value, determines if curout is out of color scale range - then don't show
        else
        {
                vcfloats->vcolour[0]=vcfloats->outofrngcolor[0];//black
                vcfloats->vcolour[1]=vcfloats->outofrngcolor[1];
                vcfloats->vcolour[2]=vcfloats->outofrngcolor[2];
                floats->transp=floats->alphaoutrng;
        }
        return 1;
}

//*************************************************************************//
// Function: di_input_mods
//*************************************************************************//

int di_input_mods()
{
        int                     elemindex;
        int                     adj=0;
        int                     i,j,k;
        uint32          *tracknode;
        dmPoint         beam1,beam2;
        dmVector beamvect;
        float32         beamdist;
        int                     cused3=0;
        int                     cused4=0;

        connections4=(uint32 *)malloc((pmi->noFaces4*4)*sizeof(uint32));
        connections3=(uint32 *)malloc((pmi->noFaces3*3)*sizeof(uint32));
        conmesh4=(uint32 *)malloc((pmi->noFaces4*4)*sizeof(uint32));
        conmesh3=(uint32 *)malloc((pmi->noFaces3*3)*sizeof(uint32));
        elearray=(uint32 *)malloc(ELEMENT_NUM*5*sizeof(uint32));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;pmi->noVertmesh=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);

                        dmPointSet (beam2,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                        dmPointSub (beamvect, beam1, beam2);

        beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                        if (beamdist > .000000001)
                        {
                                elearray[elemindex*5]=((ELEMENT_P+elemindex)->A)+2;
                                for (i=0; i<2; i++)
                                {
                                        if (i==0)
```

```
{
	j=0;k=1;
}
else
{
	j=3;k=2;
}

if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
{
	connections4[cused4+j]=((ELEMENT_P+elemindex)->B[i]);
	tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

	elearray[(elemindex*5)+(j+1)]=(ELEMENT_P+elemindex)->B[i];
}
else
{
	connections4[cused4+j]=pmi->noVertices;
	vertices[((pmi-
>noVertices)*7)+0]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+0];
	vertices[((pmi-
>noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+1];
	vertices[((pmi-
>noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+2];


	displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
	displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
	displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

	elearray[(elemindex*5)+(j+1)]=pmi->noVertices;

	pmi->noVertices++;
}

conmesh4[cused4+j]=((ELEMENT_P+elemindex)->B[i]);
conmesh4[cused4+k]=pmi->noVertmesh;
vertmesh[((pmi-
>noVertmesh)*3)+0]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+0])+(beamdist/floats->beamdelta);
vertmesh[((pmi-
>noVertmesh)*3)+1]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+1])+(beamdist/floats->beamdelta);
vertmesh[((pmi-
>noVertmesh)*3)+2]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+2])+(beamdist/floats->beamdelta);

displacemesh[(pmi-
>noVertmesh*3)+0]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+0];
displacemesh[(pmi-
>noVertmesh*3)+1]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+1];
displacemesh[(pmi-
>noVertmesh*3)+2]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+2];

connections4[cused4+k]=pmi->noVertices;
vertices[((pmi->noVertices)*7)+0]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+0])+(beamdist/floats->beamdelta);
vertices[((pmi->noVertices)*7)+1]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+1])+(beamdist/floats->beamdelta);
vertices[((pmi->noVertices)*7)+2]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+2])+(beamdist/floats->beamdelta);

displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

elearray[(elemindex*5)+(k+1)]=pmi->noVertices;
```

```
                                        pmi->noVertmesh++;
                                        pmi->noVertices++;
                                }

                                cused4+=4;
                        }
                }

                if((ELEMENT_P+elemindex)->A==4)
                {
                        elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                        for (i=0; i<4; i++)
                        {
                                conmesh4[cused4+i]=((ELEMENT_P+elemindex)->B[i]);

                                if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                {
                                        connections4[cused4+i]=((ELEMENT_P+elemindex)->B[i]);
                                        tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                        elearray[(elemindex*5)+(i+1)]=(ELEMENT_P+elemindex)->B[i];
                                }
                                else
                                {
                                        connections4[cused4+i]=pmi->noVertices;
                                        vertices[(pmi->noVertices)*7]=vertices[((ELEMENT_P+elemindex)-
>B[i])*7];
                                        vertices[((pmi->noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+1];
                                        vertices[((pmi->noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+2];

                                        displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
                                        displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
                                        displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

                                        elearray[(elemindex*5)+(i+1)]=pmi->noVertices;

                                        pmi->noVertices++;
                                }
                        }
                        cused4+=4;
                }

                if((ELEMENT_P+elemindex)->A==3)
                {
                        elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                        for (i=0; i<3; i++)
                        {
                                if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;

                                conmesh3[cused3+i]=((ELEMENT_P+elemindex)->B[i+adj]);

                                if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                                {
                                        connections3[cused3+i]=((ELEMENT_P+elemindex)->B[i+adj]);
                                        tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;

                                        elearray[(elemindex*5)+(i+1)]=(ELEMENT_P+elemindex)->B[i+adj];
                                }
                                else
                                {
                                        connections3[cused3+i]=pmi->noVertices;
```

```
                                        vertices[(pmi->noVertices)*7]=vertices[((ELEMENT_P+elemindex)-
>B[i+adj])*7];
                                        vertices[((pmi->noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+1];
                                        vertices[((pmi->noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+2];

                                        displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+0];
                                        displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+1];
                                        displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+2];

                                        elearray[(elemindex*5)+(i+1)]=pmi->noVertices;

                                        pmi->noVertices++;
                                }
                        }
                        adj=0;
                        cused3+=3;
        }
        return 1;
}

//*************************************************************************//
// Function: di_output_mods
//*************************************************************************//

int di_output_mods()
{
        int                     elemindex;
        int                     adj=0;
        float32         outtmp;
        int                     i,j,k;
        uint32          *tracknode;
        dmPoint         beam1,beam2;
        dmVector beamvect;
        float32         beamdist;

        outvert=(float32 *)malloc(pmi->noVertices*sizeof(float32));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);

                        dmPointSet (beam2,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                        dmPointSub (beamvect, beam1, beam2);

        beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                        if (beamdist > .000000001)
                        {
                                elearray[elemindex*5]=((ELEMENT_P+elemindex)->A)+2;
                                for (i=0; i<2; i++)
                                {
                                        if (i==0)
                                        {
```

```
                                                j=0;k=1;
                                        }
                                        else
                                        {
                                                j=3;k=2;
                                        }

                                        if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                        {
                                                tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                                if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                else outtmp=(ELEMENT_P+elemindex)->C[switches-
>outsubnum];
                                                floats->curout=outtmp;
                                                di_add_vertex_color();

                                                vertices[(((ELEMENT_P+elemindex)->B[i])*7)+3]=vcfloats-
>vcolour[0];
                                                vertices[(((ELEMENT_P+elemindex)->B[i])*7)+4]=vcfloats-
>vcolour[1];
                                                vertices[(((ELEMENT_P+elemindex)->B[i])*7)+5]=vcfloats-
>vcolour[2];
                                                vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+6]=max(0.0,min(1.0,floats->transp));

                                                outvert[((ELEMENT_P+elemindex)->B[i])]=floats->curout;
                                        }
                                        else
                                        {
                                                if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                else outtmp=(ELEMENT_P+elemindex)->C[switches-
>outsubnum];
                                                floats->curout=outtmp;
                                                di_add_vertex_color();

                                                vertices[(pmi->noVertices*7)+3]=vcfloats->vcolour[0];
                                                vertices[(pmi->noVertices*7)+4]=vcfloats->vcolour[1];
                                                vertices[(pmi->noVertices*7)+5]=vcfloats->vcolour[2];
                                                vertices[(pmi->noVertices*7)+6]=max(0.0,min(1.0,floats-
>transp));

                                                outvert[pmi->noVertices]=floats->curout;

                                                pmi->noVertices++;
                                        }

                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                        else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                        floats->curout=outtmp;
                                        di_add_vertex_color();

                                        vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                        vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                        vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                        vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                        outvert[pmi->noVertices]=floats->curout;

                                        pmi->noVertices++;
                                }
                        }
                }

//If element type is 4 nodes,
                if((ELEMENT_P+elemindex)->A==4)
                {
```

```
                        elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                        for (i=0; i<4; i++)
                        {
                                if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                {
                                        tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                        else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                        floats->curout=outtmp;
                                        di_add_vertex_color();

                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+3]=vcfloats->vcolour[0];
                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+4]=vcfloats->vcolour[1];
                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+5]=vcfloats->vcolour[2];
                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+6]=max(0.0,min(1.0,floats-
>transp));

                                        outvert[((ELEMENT_P+elemindex)->B[i])]=floats->curout;
                                }
                                else
                                {
                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                        else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                        floats->curout=outtmp;
                                        di_add_vertex_color();

                                        vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                        vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                        vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                        vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                        outvert[pmi->noVertices]=floats->curout;

                                        pmi->noVertices++;
                                }
                        }
                }

                //If element type is 3 nodes,
                if((ELEMENT_P+elemindex)->A==3)
                {
                        for (i=0; i<3; i++)
                        {
                                if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;

                                if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                                {
                                        tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;

                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i+adj]))->output_data[switches->outsubnum];
                                        else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                        floats->curout=outtmp;
                                        di_add_vertex_color();

                                        vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+3]=vcfloats-
>vcolour[0];
                                        vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+4]=vcfloats-
>vcolour[1];
                                        vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+5]=vcfloats-
>vcolour[2];
                                        vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+6]=max(0.0,min(1.0,floats->transp));

                                        outvert[((ELEMENT_P+elemindex)->B[i+adj])]=floats->curout;
                                }
```

```
                                        else
                                        {
                                                if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i+adj]))->output_data[switches->outsubnum];
                                                else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                floats->curout=outtmp;
                                                di_add_vertex_color();

                                                vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                                vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                                vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                                vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                                outvert[pmi->noVertices]=floats->curout;

                                                pmi->noVertices++;
                                        }
                                }
                                adj=0;
                        }
                }
                return 1;
}

//*************************************************************************//
// Function: di_Pmesh_obj
//*************************************************************************//

int
di_Pmesh_obj(ECObject *object)
{
   dmPoint      reference = { 0.0f, 0.0f, 0.0f };
   VCDynamicVisual *vc_vis;
   VCLod        *vc_lod;
   VCGeogroup     *vc_ggrp;
   VCConnectionData cdata[2];
   VCConnectionList *vc_clist;
   char         *mstr;
   int          len;
   VCMaterial     *material;
   ECVisual       *visual;
   VCAttribute    *attribute;
   ECZone         *zone;
   VCEntity       *entity;


   char         *objectname;
   VCColor              ambient={0.7, 0.5, 0.45};
          VCColor               diffuse={0.7, 0.5, 0.45};
          VCColor               emmisive={0.0,0.0,0.0};
          VCColor               opacity={0.5,0.5,0.5};
          VCSpecular                    specular={0.1, 0.1, 0.0, 0.0};
          VCGeometry  *vc_geom;

   objectname=ECObjectGetName(object);//object and objectname is objFEM)

   mstr=dStringFromOptions(NULL, &len, "objMat", DS_END_OF_OPTIONS);
   material=VCMaterial_Create  (mstr,                                   //
                                                                        Name
          Mode                                                          VC_MATERIAL_ENABLE,     //

          Ambient                                                       ambient,                //

          Diffuse                                                       diffuse,                //

          Specular                                                      specular,               //

          Emmisive                                                      emmisive,               //

          Opacity                                                       opacity,                //
```

```
                                                                    NULL,
        //      Texture
                                                                    NULL,
        //      Ramp
                                                                    NULL);
        //      Env. Map

 if (!material) printf("Failed to create material 'objMat'\n");

 /* Create dynamic visual */
 vc_vis = VCDynamicVisual_Create(objectname, 0);

 /* Create lod */
 vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

 /* Create geogroup */
 vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"objMat","objMat");
  cdata[0].type=VC_CONNECTIONLIST;
   cdata[0].faceCount=4;
        cdata[0].noConnections=pmi->noFaces4;
        cdata[0].data=connections4;

            cdata[1].type=VC_CONNECTIONLIST;
   cdata[1].faceCount=3;
        cdata[1].noConnections=pmi->noFaces3;
        cdata[1].data=connections3;


        vc_geom = VCPmesh_Create(VC_VERTEX_RGBA, pmi->noVertices, vertices, 2, cdata);

        if(vc_geom != NULL)
         VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

entity = ECObjectGetVCEntity(object);

        visual = ECObjectGetVisual(object, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);

//      ECVisualSetVCAttribute(visual,attribute);

//              Dryer switched ECVisualSetVCAttribute(vis,att) to ECVisualToVC (obj, vis) below:
        ECVisualToVC (object, visual);
//              ECVisualToVC flushes the information in the ECVisual structure to the dVS database via the VC Attribute
//              referenced in the data structure. If there is no VC attribute assigned to this ECVisual then a VCAttribute(5) is
//              created and assigned to the VCEntity(5) referenced by the ECObject(5).
//              format: int ECVisualToVC (ECObject *o, ECVisual *o);


        ECObjectToVC(object);

        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateFEMObjectFunc - function creates the model in the 3d
//                                   space.
//*************************************************************************//

int
```

```
diCreateFEMObjectFunc(ECEvent *event, ECEventData data, ECAction *action)
{
 void                **args = action->parameters;
 ECObject            *obj;
 char                                *varNameFactor; /*Modification factor */
 char                                *varFactor; /*Modification factor */
 int i;
 if(ECArgReferenceGetValue(args[2], (void *)&floats->scale, &data.focus) == VC_ERR)
  floats->scale = 1.00;

//BEC moved to main()           ucf_fem2vr();

          di_det_blocks();

          di_input_nodes();

          di_set_range();

          di_input_mods();

          di_output_mods();

          objFEMref = (ECObjectReference *)args[1];

          objFEM = ECReferenceObject(objFEMref, &data.focus);
          if(objFEM == NULL)
          {
                    VC_Error("Could not find object\n");
                    return(ECKeepAction);
          }

          di_Pmesh_obj(objFEM);
          //printf("di_Pmesh_obj() complete\n");

          di_modify_FEM();
          //printf("di_modify_FEM() complete\n");
}

//*************************************************************************//
// Function:  di_modify_FEM - updates the FEM object after changes.
//*************************************************************************//

int di_modify_FEM(void)
{

 ECVisual         *visual;
 VCAttribute      *attr;
 VCEntity         *entity;
 VCDynamicVisual   *dyn_vis;
 VCLod            *dyn_lod;
 VCGeogroup        *dyn_geogrp;
 VCGeometry        *dyn_geom;
 VCVertex_Reference ref;
 int            stat;
 VCDynamicVisual_Traverse traverse1;
 VCLod_Traverse         traverse2;
 VCGeogroup_Traverse     traverse3;
 int                i,index;
 dmPoint              curvertpos;
 char                                *varNameFactor; /*Modification factor */
 char                                *varFactor; /*Modification factor */


 if(objFEM == NULL)
 {
   VC_Error("Could not find object\n");
   return(ECKeepAction);
 }

 entity = ECObjectGetVCEntity(objFEM);
```

```
if(entity == NULL)
{
   VC_Error("Could not find entity\n");
   return(ECKeepAction);
}

visual = ECObjectGetVisual(objFEM, NULL);
if(visual == NULL)
{
   VC_Error("Could not find visual\n");
   return(ECKeepAction);
}
attr = ECVisualGetVCAttribute(visual);

ECObjectToVC(objFEM);

VCVisual_GetDynamicVisual(attr,&dyn_vis);

if(dyn_vis == NULL)
{
   VC_Error("Could not find dynamic visual\n");
   return(ECKeepAction);
}

dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_RGBA,&traverse2);

dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);


i=0;index=0;
stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);
while (stat == VC_OK)
{
 curvertpos[0]=vertices[index]+displaceobj[(i*3)+0]*floats->LoadFactor*floats->exager;
 curvertpos[1]=vertices[index+1]+displaceobj[(i*3)+1]*floats->LoadFactor*floats->exager;
 curvertpos[2]=vertices[index+2]+displaceobj[(i*3)+2]*floats->LoadFactor*floats->exager;

          floats->curout=outvert[i]*floats->LoadFactor;
          di_add_vertex_color();

  ref.data[0]=curvertpos[0];
  ref.data[1]=curvertpos[1];
  ref.data[2]=curvertpos[2];
  ref.data[3]=min(1.0,vcfloats->vcolour[0]);
  ref.data[4]=min(1.0,vcfloats->vcolour[1]);
  ref.data[5]=min(1.0,vcfloats->vcolour[2]);
  ref.data[6]=max(0.0,min(1.0,floats->transp));
  stat = VCGeometry_GetNextVertex(&ref);
  i++;
  index+=7;
 }
 VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function: di_Pmesh_mesh
//*************************************************************************//

int
di_Pmesh_mesh(ECObject *object)
{
   dmPoint        reference = { 0.0f, 0.0f, 0.0f };
   VCDynamicVisual  *vc_vis;
   VCLod          *vc_lod;
   VCGeogroup      *vc_ggrp;
   VCConnectionData cdata[2];
   VCConnectionList *vc_clist;
   char           *mstr;
```

```
int          len;
VCMaterial     *material;
ECVisual       *visual;
VCAttribute    *attribute;
ECZone         *zone;
VCEntity       *entity;
char           *objectname;
        VCColor              emmisive={0.5,0.5,0.5};//gray
        VCColour             white={1,1,1},black={0,0,0};
        VCGeometry  *vc_geom;

objectname=ECObjectGetName(object);//object and objectname is objMesh)

mstr=dStringFromOptions(NULL, &len, "meshMat", DS_END_OF_OPTIONS);
material=VCMaterial_Create  (mstr,                                    //        Name
                                                                              VC_MATERIAL_ENABLE,     //
        Mode
                                                                              black,
        //          Ambient
                                                                              black,                  //
        Diffuse
                                                                              black,                  //
        Specular
                                                                              emmisive,               //
        Emmisive
                                                                              white,                  //
        Opacity
                                                                              NULL,
        //          Texture
                                                                              NULL,
        //          Ramp
                                                                              NULL);
        //          Env. Map

if (!material) printf("Failed to create material 'meshMat\n");

/* Create dynamic visual */
vc_vis = VCDynamicVisual_Create(objectname, 0);

/* Create lod */
vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

/* Create geogroup */
vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_XYZ,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"meshMat", "meshMat");
//  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,
//
        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_WIREFRAME,0,"meshMat", "meshMat");

    cdata[0].type=VC_CONNECTIONLIST;
    cdata[0].faceCount=4;
        cdata[0].noConnections=pmi->noFaces4;
        cdata[0].data=conmesh4;

            cdata[1].type=VC_CONNECTIONLIST;
    cdata[1].faceCount=3;
        cdata[1].noConnections=pmi->noFaces3;
        cdata[1].data=conmesh3;


        vc_geom = VCPmesh_Create(VC_VERTEX_XYZ, pmi->noVertmesh, vertmesh, 2, cdata);

        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

entity = ECObjectGetVCEntity(object);

        visual = ECObjectGetVisual(object, NULL);
```

```
        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);

//      ECVisualSetVCAttribute(visual,attribute);

//              Dryer switched ECVisualSetVCAttribute(vis,att) to ECVisualToVC (obj, vis) below:
        ECVisualToVC (object, visual);
//              ECVisualToVC flushes the information in the ECVisual structure to the dVS database via the VC Attribute
//              referenced in the data structure. If there is no VC attribute assigned to this ECVisual then a VCAttribute(5) is
//              created and assigned to the VCEntity(5) referenced by the ECObject(5).
//              format: int ECVisualToVC (ECObject *o, ECVisual *o);

        ECObjectToVC(object);

        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateFEMMeshFunc
//*************************************************************************//

int
diCreateFEMMeshFunc(ECEvent *event, ECEventData data, ECAction *action)
{
 void              **args = action->parameters;
 ECObject          *obj;
 char                        *varNameFactor;  /*Modification factor */
 char                        *varFactor;  /*Modification factor */
 int i;
 if(ECArgReferenceGetValue(args[2], (void *)&floats->scale, &data.focus) == VC_ERR)
  floats->scale = 1.00;

        objMeshref = (ECObjectReference *)args[1];
        objMesh = ECReferenceObject(objMeshref, &data.focus);
        if(objMesh == NULL)
        {
                VC_Error("Could not find object\n");
                return(ECKeepAction);
        }

        di_Pmesh_mesh(objMesh);
        if (switches->meshdynmode==1) di_modify_Mesh();
}

//*************************************************************************//
// Function:  di_modify_Mesh - updates mesh after a changed has occured.
//*************************************************************************//

int di_modify_Mesh(void)
{

 ECVisual        *visual;
 VCAttribute      *attr;
 VCEntity        *entity;
 VCDynamicVisual   *dyn_vis;
 VCLod           *dyn_lod;
 VCGeogroup       *dyn_geogrp;
 VCGeometry       *dyn_geom;
 VCVertex_Reference ref;
 int             stat;
 VCDynamicVisual_Traverse traverse1;
 VCLod_Traverse         traverse2;
 VCGeogroup_Traverse     traverse3;
```

```
int             i,index;
dmPoint         curvertpos;
char                        *varNameFactor; /*Modification factor */
char                        *varFactor; /*Modification factor */


if(objMesh == NULL)
{
   VC_Error("Could not find object\n");
   return(ECKeepAction);
}

entity = ECObjectGetVCEntity(objMesh);
if(entity == NULL)
{
   VC_Error("Could not find entity\n");
   return(ECKeepAction);
}

visual = ECObjectGetVisual(objMesh, NULL);
if(visual == NULL)
{
   VC_Error("Could not find visual\n");
   return(ECKeepAction);
}
attr = ECVisualGetVCAttribute(visual);

ECObjectToVC(objMesh);

VCVisual_GetDynamicVisual(attr,&dyn_vis);

if(dyn_vis == NULL)
{
   VC_Error("Could not find dynamic visual\n");
   return(ECKeepAction);
}

dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_XYZ,&traverse2);

dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);


i=0;index=0;
stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);
while (stat == VC_OK)
{
 curvertpos[0]=vertmesh[index]+displacemesh[(i*3)+0]*floats->LoadFactor*floats->exager;
 curvertpos[1]=vertmesh[index+1]+displacemesh[(i*3)+1]*floats->LoadFactor*floats->exager;
 curvertpos[2]=vertmesh[index+2]+displacemesh[(i*3)+2]*floats->LoadFactor*floats->exager;

        floats->curout=outvert[i]*floats->LoadFactor;
        di_add_vertex_color();

  ref.data[0]=curvertpos[0];
  ref.data[1]=curvertpos[1];
  ref.data[2]=curvertpos[2];
  stat = VCGeometry_GetNextVertex(&ref);
  i++;
  index+=3;
 }

 VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function: diCreateFEMTextFunc
//*************************************************************************//
```

```
int
diCreateFEMTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECObject       *object;
   ECObjectReference *ref;
   dmPoint        reference = { 0.0f, 0.0f, 0.0f };
   VCDynamicVisual  *vc_vistext;
   VCLod          *vc_lodtext;
   VCGeogroup      *vc_ggrptext;
   int32          text_len=200;
   VCEntity       *text_ent = NULL;
   char           *mstr;
   int            len;
          VCMaterial      *material;
   ECVisual       *visual;
   VCAttribute     *attribute;
   char           *textstring="No Selection";
          VCColour    white={1,1,1},black={0,0,0},blue={0,0,1};
   dmScale        s={0.005, 0.007, 0.007};

          objFEMTextref = (ECObjectReference *)args[1];
          objFEMText = ECReferenceObject(objFEMTextref, &data.focus);

   mstr=dStringFromOptions(NULL, &len, "blue", DS_END_OF_OPTIONS);
   material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, blue,
                                        white, NULL, NULL, NULL);
   if (!material) printf("Text: Failed to create material blue emmissive\n");

                        text_ent=VCEntity_Create(NULL, 0);

                        vc_vistext=VCDynamicVisual_Create("text_ent", 0);

                        vc_lodtext = VCDynamicVisual_AddLod(vc_vistext,"#1", 0.0, -1, reference);

                        vc_ggrptext = VCLod_AddGeogroup(vc_lodtext,VC_VERTEX_XYZ,
                                        0,0,0,0,0,"blue", "blue");

                        femtextstring=VCString_CreateSized(textstring, text_len, 0, NULL, NULL, s);

                        VCGeogroup_AttachString(vc_ggrptext, femtextstring);

                        visual = ECObjectGetVisual(objFEMText, NULL);
                        if (visual == NULL)
                        {
                                VC_Error("visual was NULL\n");
                                return(ECKeepAction);
                        }

                        attribute = ECVisualGetVCAttribute(visual);
                        VCVisual_SetDynamicVisual(attribute,vc_vistext);
                        ECVisualToVC (objFEMText, visual);
                        ECObjectToVC(objFEMText);
                        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateClrSclTextFunc
//*************************************************************************//

int
diCreateClrSclTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECObject       *object;
   ECObjectReference *ref;
   dmPoint        reference = { 0.0f, 0.0f, 0.0f };
   VCDynamicVisual  *vc_vistext;
   VCLod          *vc_lodtext;
   VCGeogroup      *vc_ggrptext;
```

```
  int32          text_len=200;
  VCEntity       *clrscltext_ent = NULL;
  char           *mstr;
  int            len;
          VCMaterial      *material;
  ECVisual       *visual;
  VCAttribute    *attribute;
  char           *clrscltextstr="clrscltextstr";
          VCColour    white={1,1,1},black={0,0,0},blue={0,0,1};
  dmPoint        p={-0.02, 0.34, -0.01};
// dmPoint        p={-0.05, 0.305, 0.01};
          dmScale        s={0.013, 0.030, 0.01};
//        dmScale        s={0.013, 0.029, 0.01};

  objClrSclTextref = (ECObjectReference *)args[1];
  objClrSclText = ECReferenceObject(objClrSclTextref, &data.focus);

  mstr=dStringFromOptions(NULL, &len, "blue", DS_END_OF_OPTIONS);
  material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, blue,
                                        white, NULL, NULL, NULL);
  if (!material) printf("Text: Failed to create material blue emmissive\n");

                clrscltext_ent=VCEntity_Create(NULL, 0);

                vc_vistext=VCDynamicVisual_Create("clrscltext_ent", 0);

                vc_lodtext = VCDynamicVisual_AddLod(vc_vistext,"#1", 0.0, -1, reference);

                vc_ggrptext = VCLod_AddGeogroup(vc_lodtext,VC_VERTEX_XYZ,
                                        0,0,0,0,0,"blue", "blue");

                clrscltextstring=VCString_CreateSized(clrscltextstr, text_len, 0, p, NULL, s);

                di_updateclrscltxt();

                VCGeogroup_AttachString(vc_ggrptext, clrscltextstring);

                visual = ECObjectGetVisual(objClrSclText, NULL);
                if (visual == NULL)
                {
                        VC_Error("visual was NULL\n");
                        return(ECKeepAction);
                }

                attribute = ECVisualGetVCAttribute(visual);
                VCVisual_SetDynamicVisual(attribute,vc_vistext);
                ECVisualToVC (objClrSclText, visual);
                ECObjectToVC(objClrSclText);
                return(ECKeepAction);
}

//*************************************************************************//
// Function: di_updateclrscltxt
//*************************************************************************//

int di_updateclrscltxt()
{
        sprintf(chars->scltxt,"%10.6f\n\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n
%10.6f\n  %10.6f\n  %10.6f\n\n%10.6f",
                                        floats->out_vals[2],
                                        floats->out_min+(1.0*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.9*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.8*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.7*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.6*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.5*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.4*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.3*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.2*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.1*(floats->out_max-floats->out_min)),
```

```
                                        floats->out_min+(0.0*(floats->out_max-floats->out_min)),
                                        floats->out_vals[0]);
        VCString_SetText(clrscltextstring,chars->scltxt);
}

//************************************************************************//
// Function:  diCreateColorSclFunc
//************************************************************************//

int
diCreateColorSclFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECObject        *object;
  ECObjectReference *ref;
  dmPoint         reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual  *vc_vis;
  VCLod           *vc_lod;
  VCGeogroup      *vc_ggrp;
  VCConnectionData cdata[1];
  VCConnectionList *vc_clist;
  char            *mstr;
  int           len;
  VCMaterial      *material;
  ECVisual        *visual;
  VCAttribute     *attribute;
  ECZone          *zone;
  VCEntity        *femscl_ent = NULL;
  VCColor                 ambient={0.7, 0.5, 0.45};
        VCColor                 diffuse={0.7, 0.5, 0.45};
        VCColor                 emmisive={0.0,0.0,0.0};
        VCColor                 opacity={0.5,0.5,0.5};
        VCSpecular                      specular={0.1, 0.1, 0.0, 0.0};
        VCGeometry                      *vc_geom;
        int             i;
        float32                         clevel,dlevel,zerolevel;
        float32                         posmincolormod,negmincolormod;
  dmScale       s={0.85, 1.038, 1.00};

        femsclverts=(float32 *)malloc((24*7)*sizeof(float32));
  femsclconts=(uint32 *)malloc((6*4)*sizeof(uint32));

        objClrSclref = (ECObjectReference *)args[1];
  objClrScl = ECReferenceObject(objClrSclref, &data.focus);

  mstr=dStringFromOptions(NULL, &len, "femsclMat", DS_END_OF_OPTIONS);
  material=VCMaterial_Create  (mstr,                                    //      Name
                                                                        VC_MATERIAL_ENABLE,     //
        Mode                                                            ambient,                //
        Ambient                                                         diffuse,                //
        Diffuse                                                         specular,               //
        Specular                                                        emmisive,               //
        Emmisive                                                        opacity,                //
        Opacity                                                         NULL,
        //          Texture                                             NULL,
        //          Ramp                                                NULL);
        //          Env. Map

  if (!material) printf("Text: Failed to create material 'femsclMat'\n");

        femscl_ent=VCEntity_Create(NULL, 0);
```

```
        /* Create dynamic visual */
vc_vis = VCDynamicVisual_Create("femscl_ent", 0);

// Create lod
vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

// Create geogroup
vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"femsclMat","femsclMat");

        // Set geometry
                femsclverts[(0*7)+0]=floats->femsclbotl[0];
                femsclverts[(0*7)+1]=floats->femsclbotl[1];//level a
                femsclverts[(0*7)+2]=floats->femsclbotl[2];

                femsclverts[(1*7)+0]=floats->femsclbotr[0];
                femsclverts[(1*7)+1]=floats->femsclbotr[1];//level a
                femsclverts[(1*7)+2]=floats->femsclbotr[2];

                femsclverts[(2*7)+0]=floats->femscltopr[0];
                femsclverts[(2*7)+1]=floats->femscltopr[1]*floats->clrsclbot;//level b
                femsclverts[(2*7)+2]=floats->femscltopr[2];

                femsclverts[(3*7)+0]=floats->femscltopl[0];
                femsclverts[(3*7)+1]=floats->femscltopl[1]*floats->clrsclbot;//level b
                femsclverts[(3*7)+2]=floats->femscltopl[2];

                for (i=0;i<4;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->outofrngcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->outofrngcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->outofrngcolor[2];
                        femsclverts[(i*7)+6]=floats->alphaoutrng;
                }

                femsclverts[(4*7)+0]=floats->femscltopl[0];
                femsclverts[(4*7)+1]=floats->femscltopl[1]*floats->clrsclbot;//level b
                femsclverts[(4*7)+2]=floats->femscltopl[2];

                femsclverts[(5*7)+0]=floats->femscltopr[0];
                femsclverts[(5*7)+1]=floats->femscltopr[1]*floats->clrsclbot;//level b
                femsclverts[(5*7)+2]=floats->femscltopr[2];

                for (i=4;i<6;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->negmaxcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->negmaxcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->negmaxcolor[2];
                        femsclverts[(i*7)+6]=floats->alphainrng-.2;
                }

                if (floats->out_vals[2]<=0.0)//case 3
                {
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        clevel=max(min(1.0-floats->threshold,floats->clrscltop),floats->clrsclbot);
//below (commented out) gives a relative threshold percentage of color range
//                      clevel=floats->clrscltop-(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                        zerolevel=floats->clrscltop;
                        dlevel=floats->clrsclbot;
                        negmincolormod=(floats->clrscltop-clevel)/(floats->clrscltop-floats->clrsclbot);
                        posmincolormod=0.0;
                }
                else if (floats->out_vals[0]>=0.0)//case 2
                {
                        clevel=floats->clrsclbot;
                        zerolevel=floats->clrsclbot;
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        dlevel=min(max(floats->threshold,floats->clrsclbot),floats->clrscltop);
//below (commented out) gives a relative threshold percentage of color range
```

```
//                      dlevel=floats->clrsclbot+(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                        negmincolormod=0.0;
                        posmincolormod=(dlevel-floats->clrsclbot)/(floats->clrscltop-floats->clrsclbot);
                }
                else//case 1
                {
                        zerolevel=fabs(floats->out_min)/(floats->out_max-floats->out_min);
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        clevel=min(max((zerolevel-(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrsclbot),zerolevel);
                        dlevel=max(min((zerolevel+(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrscltop),zerolevel);
//below (commented out) gives a relative threshold percentage of color range
//                      clevel=zerolevel-(floats->threshold*(zerolevel-floats->clrsclbot));
//                      dlevel=zerolevel+(floats->threshold*(floats->clrscltop-zerolevel));
                        negmincolormod=(zerolevel-clevel)/(zerolevel-floats->clrsclbot);
                        posmincolormod=(dlevel-zerolevel)/(floats->clrscltop-zerolevel);
                }

                femsclverts[(6*7)+0]=floats->femscltopr[0];
                femsclverts[(6*7)+1]=floats->femscltopr[1]*clevel;//level c
                femsclverts[(6*7)+2]=floats->femscltopr[2];

                femsclverts[(7*7)+0]=floats->femscltopl[0];
                femsclverts[(7*7)+1]=floats->femscltopl[1]*clevel;//level c
                femsclverts[(7*7)+2]=floats->femscltopl[2];

                for (i=6;i<8;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->negmincolor[0]+
                                                                negmincolormod*
                                                                (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
                        femsclverts[(i*7)+4]=vcfloats->negmincolor[1]+
                                                                negmincolormod*
                                                                (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
                        femsclverts[(i*7)+5]=vcfloats->negmincolor[2]+
                                                                negmincolormod*
                                                                (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
                        femsclverts[(i*7)+6]=floats->alphainrng-.2;
                }

                femsclverts[(8*7)+0]=floats->femscltopl[0];
                femsclverts[(8*7)+1]=floats->femscltopl[1]*clevel;//level c
                femsclverts[(8*7)+2]=floats->femscltopl[2];

                femsclverts[(9*7)+0]=floats->femscltopr[0];
                femsclverts[(9*7)+1]=floats->femscltopr[1]*clevel;//level c
                femsclverts[(9*7)+2]=floats->femscltopr[2];

                femsclverts[(10*7)+0]=floats->femscltopr[0];
                femsclverts[(10*7)+1]=floats->femscltopr[1]*zerolevel;//zerolevel
                femsclverts[(10*7)+2]=floats->femscltopr[2];

                femsclverts[(11*7)+0]=floats->femscltopl[0];
                femsclverts[(11*7)+1]=floats->femscltopl[1]*zerolevel;//zerolevel
                femsclverts[(11*7)+2]=floats->femscltopl[2];

                for (i=8;i<12;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->negthreshcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->negthreshcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->negthreshcolor[2];
                        femsclverts[(i*7)+6]=floats->alphathresh;
                }

                femsclverts[(12*7)+0]=floats->femscltopl[0];
                femsclverts[(12*7)+1]=floats->femscltopl[1]*zerolevel;//zerolevel
```

```
femsclverts[(12*7)+2]=floats->femscltopl[2];

femsclverts[(13*7)+0]=floats->femscltopr[0];
femsclverts[(13*7)+1]=floats->femscltopr[1]*zerolevel;//zerolevel
femsclverts[(13*7)+2]=floats->femscltopr[2];

femsclverts[(14*7)+0]=floats->femscltopr[0];
femsclverts[(14*7)+1]=floats->femscltopr[1]*dlevel;//level d
femsclverts[(14*7)+2]=floats->femscltopr[2];

femsclverts[(15*7)+0]=floats->femscltopl[0];
femsclverts[(15*7)+1]=floats->femscltopl[1]*dlevel;//level d
femsclverts[(15*7)+2]=floats->femscltopl[2];

for (i=12;i<16;i++)
{
        femsclverts[(i*7)+3]=vcfloats->posthreshcolor[0];
        femsclverts[(i*7)+4]=vcfloats->posthreshcolor[1];
        femsclverts[(i*7)+5]=vcfloats->posthreshcolor[2];
        femsclverts[(i*7)+6]=floats->alphathresh;
}

femsclverts[(16*7)+0]=floats->femscltopl[0];
femsclverts[(16*7)+1]=floats->femscltopl[1]*dlevel;//level d
femsclverts[(16*7)+2]=floats->femscltopl[2];

femsclverts[(17*7)+0]=floats->femscltopr[0];
femsclverts[(17*7)+1]=floats->femscltopr[1]*dlevel;//level d
femsclverts[(17*7)+2]=floats->femscltopr[2];

for (i=16;i<18;i++)
{
        femsclverts[(i*7)+3]=vcfloats->posmincolor[0]+
                                                posmincolormod*
                                                (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
        femsclverts[(i*7)+4]=vcfloats->posmincolor[1]+
                                                posmincolormod*
                                                (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
        femsclverts[(i*7)+5]=vcfloats->posmincolor[2]+
                                                posmincolormod*
                                                (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
        femsclverts[(i*7)+6]=floats->alphainrng-.2;
}

femsclverts[(18*7)+0]=floats->femscltopr[0];
femsclverts[(18*7)+1]=floats->femscltopr[1]*floats->clrscltop;//level e
femsclverts[(18*7)+2]=floats->femscltopr[2];

femsclverts[(19*7)+0]=floats->femscltopl[0];
femsclverts[(19*7)+1]=floats->femscltopl[1]*floats->clrscltop;//level e
femsclverts[(19*7)+2]=floats->femscltopl[2];

for (i=18;i<20;i++)
{
        femsclverts[(i*7)+3]=vcfloats->posmaxcolor[0];
        femsclverts[(i*7)+4]=vcfloats->posmaxcolor[1];
        femsclverts[(i*7)+5]=vcfloats->posmaxcolor[2];
        femsclverts[(i*7)+6]=floats->alphainrng-.2;
}

femsclverts[(20*7)+0]=floats->femscltopl[0];
femsclverts[(20*7)+1]=floats->femscltopl[1]*floats->clrscltop;//level e
femsclverts[(20*7)+2]=floats->femscltopl[2];

femsclverts[(21*7)+0]=floats->femscltopr[0];
femsclverts[(21*7)+1]=floats->femscltopr[1]*floats->clrscltop;//level e
femsclverts[(21*7)+2]=floats->femscltopr[2];
```

```
                femsclverts[(22*7)+0]=floats->femscltopr[0];
                femsclverts[(22*7)+1]=floats->femscltopr[1];//level f
                femsclverts[(22*7)+2]=floats->femscltopr[2];

                femsclverts[(23*7)+0]=floats->femscltopl[0];
                femsclverts[(23*7)+1]=floats->femscltopl[1];//level f
                femsclverts[(23*7)+2]=floats->femscltopl[2];

                for (i=20;i<24;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->outofrngcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->outofrngcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->outofrngcolor[2];
                        femsclverts[(i*7)+6]=floats->alphaoutrng;
                }


        for (i=0;i<24;i++)
        {
                femsclconts[i] = i;
        }

        cdata[0].type=VC_CONNECTIONLIST;
  cdata[0].faceCount=4;
        cdata[0].noConnections=6;
        cdata[0].data=femsclconts;

        vc_geom = VCPmesh_Create(VC_VERTEX_RGBA, 24, (VCVertex) femsclverts, 1, cdata);

        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

        visual = ECObjectGetVisual(objClrScl, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);
        ECVisualToVC (objClrScl, visual);
        ECObjectToVC(objClrScl);

        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateColorSclGridFunc
//*************************************************************************//

int
diCreateColorSclGridFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECObject      *object;
  ECObjectReference *ref;
  dmPoint       reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual *vc_vis;
  VCLod         *vc_lod;
  VCGeogroup    *vc_ggrp;
  VCConnectionData cdata[1];
  VCConnectionList *vc_clist;
  char          *mstr;
  int           len;
  VCMaterial    *material;
```

```
ECVisual        *visual;
VCAttribute     *attribute;
ECZone          *zone;
VCEntity        *femsclgrd_ent = NULL;
        VCColor                 white={1,1,1},black={0,0,0},grdcolor={1,1,1};
        VCGeometry              *vc_geom;
        int             i;
        dmScale                 s={0.85, 1.038, 1.00};
        float32                 femsclgrdxyz[]=         {
                                                                0,0,0,
                                                                .035,0,0,
                                                                .035,.2833*.1,0,
                                                                0,.2833*.1,0,
                                                                0,.2833*.1,0,
                                                                .035,.2833*.1,0,
                                                                .035,.2833*.2,0,
                                                                0,.2833*.2,0,
                                                                0,.2833*.2,0,
                                                                .035,.2833*.2,0,
                                                                .035,.2833*.3,0,
                                                                0,.2833*.3,0,
                                                                0,.2833*.3,0,
                                                                .035,.2833*.3,0,
                                                                .035,.2833*.4,0,
                                                                0,.2833*.4,0,
                                                                0,.2833*.4,0,
                                                                .035,.2833*.4,0,
                                                                .035,.2833*.5,0,
                                                                0,.2833*.5,0,
                                                                0,.2833*.5,0,
                                                                .035,.2833*.5,0,
                                                                .035,.2833*.6,0,
                                                                0,.2833*.6,0,
                                                                0,.2833*.6,0,
                                                                .035,.2833*.6,0,
                                                                .035,.2833*.7,0,
                                                                0,.2833*.7,0,
                                                                0,.2833*.7,0,
                                                                .035,.2833*.7,0,
                                                                .035,.2833*.8,0,
                                                                0,.2833*.8,0,
                                                                0,.2833*.8,0,
                                                                .035,.2833*.8,0,
                                                                .035,.2833*.9,0,
                                                                0,.2833*.9,0,
                                                                0,.2833*.9,0,
                                                                .035,.2833*.9,0,
                                                                .035,.2833*1.0,0,
                                                                0,.2833*1.0,0,
                                                                };

        femsclgrdverts=(float32 *)malloc((40*3)*sizeof(float32));
femsclgrdconts=(uint32 *)malloc((10*4)*sizeof(uint32));

        objClrSclGridref = (ECObjectReference *)args[1];
objClrSclGrid = ECReferenceObject(objClrSclGridref, &data.focus);

mstr=dStringFromOptions(NULL, &len, "femsclgrdMat", DS_END_OF_OPTIONS);

        material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, grdcolor,
                                        white, NULL, NULL, NULL);

if (!material) printf("Text: Failed to create material 'femsclgrdMat'\n");

        femsclgrd_ent=VCEntity_Create(NULL, 0);

        // Create dynamic visual
vc_vis = VCDynamicVisual_Create("femsclgrd_ent", 0);

// Create lod
```

```
  vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

  // Create geogroup
  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_XYZ,

          0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_WIREFRAME,0,"femsclgrdMat", "femsclMat");

          for (i=0;i<40;i++)
          {
                  femsclgrdverts[(i*3)+0]=femsclgrdxyz[(i*3)+0];
                  femsclgrdverts[(i*3)+1]=femsclgrdxyz[(i*3)+1];
                  femsclgrdverts[(i*3)+2]=femsclgrdxyz[(i*3)+2];
          }

          for (i=0;i<40;i++)
          {
                  femsclgrdconts[i] = i;
          }

          cdata[0].type=VC_CONNECTIONLIST;
  cdata[0].faceCount=4;
          cdata[0].noConnections=10;
          cdata[0].data=femsclgrdconts;

          vc_geom = VCPmesh_Create(VC_VERTEX_XYZ, 40, (VCVertex) femsclgrdverts, 1, cdata);

          if(vc_geom != NULL)
            VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

          visual = ECObjectGetVisual(objClrSclGrid, NULL);

          if (visual == NULL)
           {
            VC_Error("visual was NULL\n");
            return(ECKeepAction);
           }

          attribute = ECVisualGetVCAttribute(visual);

          VCVisual_SetDynamicVisual(attribute,vc_vis);
          ECVisualToVC (objClrSclGrid, visual);
          ECObjectSetPosOrScale(objClrSclGrid,NULL,NULL,s);
          ECObjectToVC(objClrSclGrid);

          return(ECKeepAction);
}

//*************************************************************************//
// Function:  di_modify_ClrScl
//*************************************************************************//

int di_modify_ClrScl(void)
{

          ECVisual        *visual;
          VCAttribute      *attr;
          VCEntity        *entity;
          VCDynamicVisual   *dyn_vis;
          VCGeogroup       *dyn_geogrp;
          VCLod           *dyn_lod;
          VCGeometry       *dyn_geom;
          VCVertex_Reference ref;
          int             stat;
          VCDynamicVisual_Traverse traverse1;
          VCLod_Traverse         traverse2;
          VCGeogroup_Traverse     traverse3;
          int               i,index;
          dmPoint              curvertpos;
           char                                *varNameFactor; /*Modification factor */
          char                                 *varFactor; /*Modification factor */
```

```
        float32                         clevel,dlevel,zerolevel;
        float32                         posmincolormod,negmincolormod;
        dmScale     s={0.85, 1.038, 1.00};//DAD


if(objClrScl == NULL)
 {
   VC_Error("Could not find object\n");
   return(ECKeepAction);
 }

 entity = ECObjectGetVCEntity(objClrScl);
 if(entity == NULL)
 {
   VC_Error("Could not find entity\n");
   return(ECKeepAction);
 }

 visual = ECObjectGetVisual(objClrScl, NULL);
 if(visual == NULL)
 {
   VC_Error("Could not find visual\n");
   return(ECKeepAction);
 }
 attr = ECVisualGetVCAttribute(visual);

// ECObjectSetPosOrScale(objClrScl,NULL,NULL,s);//DAD

 ECObjectToVC(objClrScl);

 VCVisual_GetDynamicVisual(attr,&dyn_vis);

 if(dyn_vis == NULL)
 {
   VC_Error("Could not find dynamic visual\n");
   return(ECKeepAction);
 }

 dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

 dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_RGBA,&traverse2);

 dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);

 stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);//vertex 0
          ref.data[0]=floats->femsclbotl[0];
  ref.data[1]=floats->femsclbotl[1];
  ref.data[2]=floats->femsclbotl[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 1
          ref.data[0]=floats->femsclbotr[0];
  ref.data[1]=floats->femsclbotr[1];
  ref.data[2]=floats->femsclbotr[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 2
          ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*floats->clrsclbot;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;
```

```
stat = VCGeometry_GetNextVertex(&ref);//vertex 3
        ref.data[0]=floats->femscltopl[0];
 ref.data[1]=floats->femscltopl[1]*floats->clrsclbot;
 ref.data[2]=floats->femscltopl[2];
 ref.data[3]=vcfloats->outofrngcolor[0];
 ref.data[4]=vcfloats->outofrngcolor[1];
 ref.data[5]=vcfloats->outofrngcolor[2];
 ref.data[6]=floats->alphaoutrng;

stat = VCGeometry_GetNextVertex(&ref);//vertex 4
        ref.data[0]=floats->femscltopl[0];
 ref.data[1]=floats->femscltopl[1]*floats->clrsclbot;
 ref.data[2]=floats->femscltopl[2];
 ref.data[3]=vcfloats->negmaxcolor[0];
 ref.data[4]=vcfloats->negmaxcolor[1];
 ref.data[5]=vcfloats->negmaxcolor[2];
 ref.data[6]=floats->alphainrng-.2;

stat = VCGeometry_GetNextVertex(&ref);//vertex 5
        ref.data[0]=floats->femscltopr[0];
 ref.data[1]=floats->femscltopr[1]*floats->clrsclbot;
 ref.data[2]=floats->femscltopr[2];
 ref.data[3]=vcfloats->negmaxcolor[0];
 ref.data[4]=vcfloats->negmaxcolor[1];
 ref.data[5]=vcfloats->negmaxcolor[2];
 ref.data[6]=floats->alphainrng-.2;

                if (floats->out_vals[2]<=0.0)//case 3
                {
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        clevel=max(min(1.0-floats->threshold,floats->clrscltop),floats->clrsclbot);
//below (commented out) gives a relative threshold percentage of color range
//                      clevel=floats->clrscltop-(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                        zerolevel=floats->clrscltop;
                        dlevel=floats->clrscltop;
                        negmincolormod=(floats->clrscltop-clevel)/(floats->clrscltop-floats->clrsclbot);
                        posmincolormod=0.0;
                }
                else if (floats->out_vals[0]>=0.0)//case 2
                {
                        clevel=floats->clrsclbot;
                        zerolevel=floats->clrsclbot;
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        dlevel=min(max(floats->threshold,floats->clrsclbot),floats->clrscltop);
//below (commented out) gives a relative threshold percentage of color range
//                      dlevel=floats->clrsclbot+(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                        negmincolormod=0.0;
                        posmincolormod=(dlevel-floats->clrsclbot)/(floats->clrscltop-floats->clrsclbot);
                }
                else//case 1
                {
                        zerolevel=fabs(floats->out_min)/(floats->out_max-floats->out_min);
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        clevel=min(max((zerolevel-(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrsclbot),zerolevel);
                        dlevel=max(min((zerolevel+(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrscltop),zerolevel);
//below (commented out) gives a relative threshold percentage of color range
//                      clevel=zerolevel-(floats->threshold*(zerolevel-floats->clrsclbot));
//                      dlevel=zerolevel+(floats->threshold*(floats->clrscltop-zerolevel));
                        negmincolormod=(zerolevel-clevel)/(zerolevel-floats->clrsclbot);
                        posmincolormod=(dlevel-zerolevel)/(floats->clrscltop-zerolevel);
                }


 stat = VCGeometry_GetNextVertex(&ref);//vertex 6
        ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*clevel;
  ref.data[2]=floats->femscltopr[2];
```

```
   ref.data[3]=vcfloats->negmincolor[0]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
   ref.data[4]=vcfloats->negmincolor[1]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
   ref.data[5]=vcfloats->negmincolor[2]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
   ref.data[6]=floats->alphainrng-.2;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 7
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*clevel;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->negmincolor[0]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
   ref.data[4]=vcfloats->negmincolor[1]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
   ref.data[5]=vcfloats->negmincolor[2]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
   ref.data[6]=floats->alphainrng-.2;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 8
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*clevel;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 9
            ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*clevel;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 10
            ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*zerolevel;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 11
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*zerolevel;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 12
            ref.data[0]=floats->femscltopl[0];
```

```
  ref.data[1]=floats->femscltopl[1]*zerolevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 13
            ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*zerolevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 14
            ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*dlevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 15
            ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*dlevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 16
            ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*dlevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posmincolor[0]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
  ref.data[4]=vcfloats->posmincolor[1]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
  ref.data[5]=vcfloats->posmincolor[2]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
  ref.data[6]=floats->alphainrng-.2;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 17
            ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*dlevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posmincolor[0]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
  ref.data[4]=vcfloats->posmincolor[1]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
  ref.data[5]=vcfloats->posmincolor[2]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
  ref.data[6]=floats->alphainrng-.2;
```

```
stat = VCGeometry_GetNextVertex(&ref);//vertex 18
        ref.data[0]=floats->femscltopr[0];
 ref.data[1]=floats->femscltopr[1]*floats->clrscltop;
 ref.data[2]=floats->femscltopr[2];
 ref.data[3]=vcfloats->posmaxcolor[0];
 ref.data[4]=vcfloats->posmaxcolor[1];
 ref.data[5]=vcfloats->posmaxcolor[2];
 ref.data[6]=floats->alphainrng-.2;

stat = VCGeometry_GetNextVertex(&ref);//vertex 19
        ref.data[0]=floats->femscltopl[0];
 ref.data[1]=floats->femscltopl[1]*floats->clrscltop;
 ref.data[2]=floats->femscltopl[2];
 ref.data[3]=vcfloats->posmaxcolor[0];
 ref.data[4]=vcfloats->posmaxcolor[1];
 ref.data[5]=vcfloats->posmaxcolor[2];
 ref.data[6]=floats->alphainrng-.2;

stat = VCGeometry_GetNextVertex(&ref);//vertex 20
        ref.data[0]=floats->femscltopl[0];
 ref.data[1]=floats->femscltopl[1]*floats->clrscltop;
 ref.data[2]=floats->femscltopl[2];
 ref.data[3]=vcfloats->outofrngcolor[0];
 ref.data[4]=vcfloats->outofrngcolor[1];
 ref.data[5]=vcfloats->outofrngcolor[2];
 ref.data[6]=floats->alphaoutrng;

stat = VCGeometry_GetNextVertex(&ref);//vertex 21
        ref.data[0]=floats->femscltopr[0];
 ref.data[1]=floats->femscltopr[1]*floats->clrscltop;
 ref.data[2]=floats->femscltopr[2];
 ref.data[3]=vcfloats->outofrngcolor[0];
 ref.data[4]=vcfloats->outofrngcolor[1];
 ref.data[5]=vcfloats->outofrngcolor[2];
 ref.data[6]=floats->alphaoutrng;

stat = VCGeometry_GetNextVertex(&ref);//vertex 22
        ref.data[0]=floats->femscltopr[0];
 ref.data[1]=floats->femscltopr[1];
 ref.data[2]=floats->femscltopr[2];
 ref.data[3]=vcfloats->outofrngcolor[0];
 ref.data[4]=vcfloats->outofrngcolor[1];
 ref.data[5]=vcfloats->outofrngcolor[2];
 ref.data[6]=floats->alphaoutrng;

stat = VCGeometry_GetNextVertex(&ref);//vertex 23
        ref.data[0]=floats->femscltopl[0];
 ref.data[1]=floats->femscltopl[1];
 ref.data[2]=floats->femscltopl[2];
 ref.data[3]=vcfloats->outofrngcolor[0];
 ref.data[4]=vcfloats->outofrngcolor[1];
 ref.data[5]=vcfloats->outofrngcolor[2];
 ref.data[6]=floats->alphaoutrng;

        VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function:  di_intersect_handler
//*************************************************************************//

int
di_intersect_handler(VCBodyScreenIntersection_CallbackData *callbackData, void *data)
{
  int           numIntersections;

  if (callbackData == NULL)
        {
    printf("di_intersect_handler : callbackData NULL; exiting handler\n");
    return(ECKeepAction);
```

```
  }

  if(VCIntersection_Get(callbackData->intersection, NULL, &intersectionReportData, &numIntersections, NULL) != VC_OK)
        {
                VC_Error("dvObjectIntersectFunc : VCIntersection_Get returned VC_ERR\n");
                return(ECKeepAction);
        }

        if (intersectionReportData)
        {
                di_FEM_interact();
        }
        return (ECKeepAction);
}

//*************************************************************************//
// Function:  di_FEM_interact - performs operations based on where the
//                              mouse button intersected with the model.
//*************************************************************************//

int di_FEM_interact()
{
        static      VCEntity  *graysphere;
        static      VCEntity  *bluesphere;
        static      VCAttribute          *v=NULL;
        static      VCAttribute          *w=NULL;
        static  VCAttribute *a=NULL;

  VCAttribute               *int_attribute = NULL;
  VCEntity          *entity;
  VCEntity          *parent;
        VCEntity                        *FEMent;
        VCEntity                        *Meshent;
        VCEntity                        *ClrSclent;//temp
  dmPoint           p;
  dmEuler           e;
  dmScale           s;
  dmMatrix          mat,inv_mat,matp,cur_mat;
        dmVector                        intvect1, intvect2;
  VCDynamicVisual     *dvis;
  VCVertex_Reference    ref;
  dmPoint           orgndpt1, orgndpt2, nodep1, nodep2;
  int             numIntersections;
  int             ii = 0, i, j, k, rightindex;
///  int              rightvert, rightelem, adjindex;
        dmEuler                                 o;
        float32                                 badcum, badrec;
        float32                                 anglerec, anglecalc, anglesum, angledif;
        float32                                         intrec, lengthrec, intdist, length;
        dmVector                                sidevect;
        ECVisual                        *visual1,*visual2;
  VCAttribute     *attribute1,*attribute2;
        EntityList              *picked_load = NULL, *picked_constr = NULL;
//      Load                            *p = NULL;
        char                                    *loadtype = "initialization", *name = "initialization", *cname =
"initialization";
        int                                             r, type, node, cnode, face[6], value[6], dof_flag[6];

// Get the attribute of the intersected object
        int_attribute = intersectionReportData->visual;
        if ( (entity = intersectionReportData->entity)==NULL)
                return(ECKeepAction);

        if(ConstrList != NULL)
                {
                        for(picked_constr = ConstrList; picked_constr != NULL; picked_constr = picked_constr->next)
                        {
                                if (entity == picked_constr->nodeobj)
                                {
                                        strcpy(cname, CONSTRAINT_SET[CONSTRAINTSET_PICK].B);
```

```
                                        cnode = CONSTRAINT_SET[CONSTRAINTSET_PICK].ID[ii];

                                        for (r = 0; r < 6; r++)
                                                dof_flag[r] =
CONSTRAINT_SET[CONSTRAINTSET_PICK].INDEX[ii*6+r];

                                        printf("\n");
                                        sprintf(chars->outtxt,"Constr Set Name:\n  %s\nConstraint Node: %d\nTrans
X = %d\nTrans Y = %d\nTrans Z = %d\nRot X = %d\nRot Y = %d\nRot Z = %d\n",
                                                cname, cnode, dof_flag[0], dof_flag[1], dof_flag[2], dof_flag[3],
dof_flag[4], dof_flag[5]);

                                        VCString_SetText(femtextstring, chars->outtxt);

                                        break;
                                }
                                ii++;
                        }
                }

                /*************************************************************************/

                ii = 0;
                if(LoadList != NULL)
                {
                        for(picked_load = LoadList; picked_load != NULL; picked_load = picked_load->next)
                        {
                                if (entity == picked_load->nodeobj)
                                {
                                        strcpy(name, LOAD_SET[LOADSET_PICK].NAME);
                                        type = LOAD_SET[LOADSET_PICK].TYPE[ii];
                                        switch(type)
                                        {
                                                case 1:     strcpy(loadtype, "Nodal Force");
                                                        break;
                                                case 2: strcpy(loadtype, "Nodal Displacement");
                                                        break;
                                                case 3: strcpy(loadtype, "Nodal Accel");
                                                        break;
                                                case 5: strcpy(loadtype, "Nodal Heat Generation");
                                                        break;
                                                case 6: strcpy(loadtype, "Nodal Heat Flux");
                                                        break;
                                                case 7: strcpy(loadtype, "Velocity");
                                                        break;
                                                case 8: strcpy(loadtype, "Nonlinear Transient");
                                                        break;
                                                case 10: strcpy(loadtype, "Distributed Line Load");
                                                        break;
                                                case 11: strcpy(loadtype, "Element Face Pressure");
                                                        break;
                                                case 13: strcpy(loadtype, "Element Heat Generation");
                                                        break;
                                                case 14: strcpy(loadtype, "Element Heat Flux");
                                                        break;
                                                case 15: strcpy(loadtype, "Element Convection");
                                                        break;
                                                case 16: strcpy(loadtype, "Element Radiation");
                                                        break;
                                        }

                                        node = LOAD_SET[LOADSET_PICK].ID[ii];

                                        for (r = 0; r < 6; r++){
                                                face[r] = LOAD_SET[LOADSET_PICK].FACE[ii*6+r];
                                                value[r] = LOAD_SET[LOADSET_PICK].VALUE[ii*8+2+r];
                                        }

                                        sprintf(chars->outtxt,"Load Set Name:\n  %sLoad Type: %s\nLoad Node:
%d\nTrans x value = %d\nTrans y value = %d\nTrans z value = %d\nRot x value = %d\nRot y value = %d\nRot z value = %d\n",
```

```
                                                name, loadtype, node, value[0], value[1], value[2], value[3],
value[4], value[5]);
                                        VCString_SetText(femtextstring,chars->outtxt);
                                        //return;
                                        break;
                                }
                                ii++;
                        }
                }

        FEMent=ECObjectGetVCEntity (objFEM);
        Meshent=ECObjectGetVCEntity (objMesh);
        if ((entity == FEMent)||(entity == Meshent))
                {
                        if (intersectionReportData->point)
                        {
                                VCEntity_GetPositionPointEulerScale(entity,p,e,s);
                                dmMatFromPointEulerScale(mat,p,e,s);

                        if((parent = entity->parent)!=NULL)
                        {
                                VCEntity_GetPositionPointEulerScale(parent,p,e,s);
                                dmMatFromPointEulerScale(matp,p,e,s);
                                dmMatMult(cur_mat,mat,matp);
                        }
                        else dmMatCopy(cur_mat,mat);

                        dmMatInvert(inv_mat,cur_mat);

                        anglerec=360.0;
                        pmi->rightvert=0;

// Element closure angle test
//  to determine correct element intersected by intersection point
// Dryer - 8/97

                        for (i = 0; i < ELEMENT_NUM; i++)
                        {

                                {
                                        anglesum=0.0;

                                        for (j = 0; j < elearray[i * 5]; j++)
                                        {
                                                k=j+1;
                                                if (j==((elearray[i*5])-1)) k=0;

                                                dmPointSet (orgndpt1,
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+0])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+1])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+2])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                                dmPointXformMat(nodep1,orgndpt1,cur_mat);
                                                dmPointSub (intvect1, intersectionReportData->point, nodep1);

                                                dmPointSet (orgndpt2,
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+0])+

        ((displaceobj[((elearray[((i*5)+(k+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+1])+

        ((displaceobj[((elearray[((i*5)+(k+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+2])+
```

```
((displaceobj[((elearray[((i*5)+(k+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                                dmPointXformMat(nodep2,orgndpt2,cur_mat);
                                                dmPointSub (intvect2, intersectionReportData->point, nodep2);

                                                anglecalc=(180.0/3.14159251)*

        acos(((intvect1[0]*intvect2[0])+(intvect1[1]*intvect2[1])+(intvect1[2]*intvect2[2]))/

        ((sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2])))*

        (sqrt((intvect2[0]*intvect2[0])+(intvect2[1]*intvect2[1])+(intvect2[2]*intvect2[2])))));

                                                if (     fabs(1.0-
(((intvect1[0]*intvect2[0])+(intvect1[1]*intvect2[1])+(intvect1[2]*intvect2[2]))/

        ((sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2])))*

        (sqrt((intvect2[0]*intvect2[0])+(intvect2[1]*intvect2[1])+(intvect2[2]*intvect2[2])))))
                                                                        < .000001 )
                                                        anglecalc=0.0;


                                                        anglesum=anglesum+anglecalc;
                                        }
                                        angledif = fabs(360.0-anglesum);
                                        if (angledif<anglerec)
                                        {
                                                anglerec=angledif;
                                                pmi->rightelem=i;
                                        }
                                }
                        }

//Now that correct element is identified,
// Determine nearest vertex to intersection point (pmi->rightvert) in identified element and
// min side length (lengthrec) for sphere marker scaling
//Dryer - 8/97
                        intrec=10000.0;
                        lengthrec=10000.0;

                        for (j=0; j<elearray[pmi->rightelem*5]; j++)
                        {
//Set up variables
                                k=j+1;
                                if (j==((elearray[pmi->rightelem*5])-1)) k=0;

                                dmPointSet (orgndpt1,
                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+0])+
                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+1])+
                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+2])+
                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                dmPointXformMat(nodep1,orgndpt1,cur_mat);

                                dmPointSet (orgndpt2,
                                        (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+0])+
                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                        (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+1])+
                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                        (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+2])+
```

```
                                                        ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                        dmPointXformMat(nodep2,orgndpt2,cur_mat);


//Test for nearest element node
                                        dmPointSub (intvect1, intersectionReportData->point, nodep1);

        intdist=sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2]));
                                        if (intdist<intrec)
                                        {
                                                intrec=intdist;
                                                pmi->rightvert=elearray[(pmi->rightelem*5)+(j+1)];
                                                points->rightnodep[0]=nodep1[0];
                                                points->rightnodep[1]=nodep1[1];
                                                points->rightnodep[2]=nodep1[2];
                                                rightindex = j;
                                        }
// adjust rightindex for beam elements
                                        if (((ELEMENT_P+pmi->rightelem)->A == 2) && ((rightindex == 0) || (rightindex ==
1)))
                                                pmi->adjindex = 0;
                                        else if (((ELEMENT_P+pmi->rightelem)->A == 2) && ((rightindex == 3) || (rightindex
== 2)))
                                                pmi->adjindex = 1;
                                        else
                                                pmi->adjindex = rightindex;


//Calculate max element side length for sphere marker scaling
                                        dmPointSub (sidevect, nodep1, nodep2);

        length=sqrt((sidevect[0]*sidevect[0])+(sidevect[1]*sidevect[1])+(sidevect[2]*sidevect[2]));
                                        if (length<lengthrec) lengthrec=length;

                                }

                                if ((ELEMENT_P+pmi->rightelem)->A == 2)
                                {
                                        lengthrec=lengthrec*(floats->beamdelta/10);
                                }


//Dryer: used to see which vertex is being selected
//                              outvert[pmi->rightvert]=out_max;

        sprintf(chars->outtxt,"%s\nNode #: %i\nElement #: %i\n\n%s%10.6f\nDX: %10.6f\nDY: %10.6f\nDZ: %10.6f\n",
                                        names->actual_case_name,
                                        (NODE_P+((ELEMENT_P+pmi->rightelem)->B[pmi->adjindex]))->A,
                                        (ELEMENT_P+pmi->rightelem)->D,
                                        names->actual_set_name[(switches->outtypenum*5)+switches->outsubnum],
                                        outvert[pmi->rightvert]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+0]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+1]*floats->LoadFactor,
                                        displaceobj[(pmi->rightvert*3)+2]*floats->LoadFactor);

// create graysphere
                                graysphere = VCEntity_Create(NULL, 0);
                                if (VCAttribute_Delete (v) != 0)
                                        VC_Error ("Error cannot destroy attribute\n");

                                v = VCVisual_CreateGeometry ("graysphere");
                                VCEntity_AttachAttribute (graysphere, v);
                                VCEntity_Scale(graysphere, (lengthrec/7.0), (lengthrec/7.0), (lengthrec/7.0));
                                VCEntity_SetPositionPoint(graysphere,intersectionReportData->point);

 // create bluesphere
                                bluesphere = VCEntity_Create(NULL, 0);
                                if (VCAttribute_Delete (w) != 0)
```

```
                                        VC_Error ("Error cannot destroy attribute\n");

                                w = VCVisual_CreateGeometry ("bluesphere");
                                VCEntity_AttachAttribute (bluesphere, w);
                                VCEntity_Scale(bluesphere, (lengthrec/5.0), (lengthrec/5.0), (lengthrec/5.0));
                                VCEntity_SetPositionPoint(bluesphere,points->rightnodep);

                                if (VCAttribute_Delete (a) != 0)
                                        VC_Error ("Error cannot destroy audio attribute\n");
                                a = VCEntity_AddAudioVoice (bluesphere, "explosion");
        if (a == NULL)
                                {
           VC_Error ("Cannot create audio instance\n");
                                }
                                else
                                {
                                        /* Play the audio voice */
         VCAudio_Start (a);

         /* Change the loop count to infinity, set to highest priority
           and play */
         VCAudio_SetLoopCount (a, 1);
         VCAudio_SetPriority (a, VC_AUDIO_PRIORITY_LOCKED);
         VCAudio_Start (a);
                                }

                                di_modify_FEM();
                                if (switches->meshdynmode==1) di_modify_Mesh();
                                di_modify_LoadSet();
                                di_modify_ConstraintSet();
                                switches->picknode=1;//picknode
                                }
                        else
                        {
//                              sprintf(chars->outtxt,"%s","No selection");
                        }
                        VCString_SetText(femtextstring,chars->outtxt);
           }
   return;
}

//*************************************************************************//
// Function: di_create_body_handler
// Comments: Dryer added di_create_body_handler
//*************************************************************************//

int
di_create_body_handler(VCBodyCreate_CallbackData *bodyData, void *data)
{
           VCBody *body = bodyData->body;

   VCBody_AttachScreenIntersectionCallback(body, NULL, di_intersect_handler, NULL);

 return;
}

/*************************************************************************//
// Function: ObjectIntersectedCallback
//*************************************************************************//

int
ObjectIntersectedCallback(VCIntersection_CallbackData *cdata, void *data)
{

   intersectionReportData = VCIntersection_GetFirstIntersectionReport(cdata->intersection, NULL);

   if(intersectionReportData == NULL)
   {
      return;
   }
```

```
        di_FEM_interact();
}

//*************************************************************************//
// Function:  diImmersDataFunc
//*************************************************************************//

int
diImmersDataFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   char          *part = NULL;
   VCAttribute      *vcLimb;
   ECObjectReference *ref;
   VCPositionData    pos;
   dmMatrix          handMat;
   dmPoint       pt;
   dmEuler       ori;
   dmScale       scaledm;
   VC_Traverse       traverseInfo;

   float32           length = 50.0;
   uint32            newMask = 0x10;
   uint32            oldMask;
            intersectArgs intersectData;
            VCEntity    *hitEntity;
            VCAttribute  *intersection;
   VCIntersection *intersectionData;

   // Fix
   pos.mode = 0;

   if(args[1] != NULL)
      part = (char *)args[1];
   else
      part = "hand";

   intersectData.event = (uint32 *)args[2];

   ref = (ECObjectReference *)args[3];
   intersectData.object = ECReferenceObject(ref, &data.focus);

   // Is there a body?
   if(data.body)
   {  // Get limb position
      vcLimb = VCBody_GetBodyPart(data.body, part);
   }
   else
   {  // Get limb position
      vcLimb = VCBody_GetBodyPart(VC_GetFirstBody(&traverseInfo), part);
   }

   if (vcLimb == NULL)
   {
      VC_Error("dvObjectIntersectFunc: Didn't get limb %s.\n", part);
      return(ECKeepAction);
   }

   // Calculate hand matrix from position
   if(VCEntity_GetAbsolutePosition(vcLimb->first, handMat) != VC_OK)
   {
      VC_Error("dvObjectIntersectFunc : VCEntity_GetAbsolutePosition returned VC_ERR\n");
      return(ECKeepAction);
   }
   // First time?
   if(args[0]==NULL)
   {
            int *pInt = (int *)malloc(sizeof(int));
            args[0] = pInt;
```

```
  // Get the hand intersection mask
  if(VCVectorIntersect_GetIntersectMask(vcLimb, &oldMask) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersectMask returned VC_ERR\n");
     return(ECKeepAction);
  }
  // Stop the vector from intersecting the limb named,
  //  by setting its mask value to the same
  if(VCVectorIntersect_ModifyIntersectMask (vcLimb, NULL, oldMask) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCVectorIntersect_ModifyIntersectMask returned VC_ERR\n");
     return(ECKeepAction);
  }
  // Get point, orientation and scale
  dmPointEulerScaleFromMat(pt, ori, scaledm, handMat);
  // Get a position from above
  if(VCPosition_MakePointEulerScale (&pos, pt, ori, scaledm) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCPosition_MakePointEulerScale returned VC_ERR\n");
     return(ECKeepAction);
  }
  // Define the hit entity
  hitEntity = VCEntity_Create(&pos, NULL);
  // Evaluate intersection
  intersection = VCVectorIntersect_Create(VC_VECTORINTERSECT_ENABLE,
                              length, newMask, 1);
  if(intersection == NULL)
  {
     VC_Error("dvObjectIntersectFunc : intersection is NULL\n");
     return(ECKeepAction);
  }
  // Attach vector intersect to an entity
  if(VCEntity_AttachAttribute(hitEntity, intersection) == VC_ERR)
  {
     VC_Error("dvObjectIntersectFunc : could not attach vector intersect to entity\n");
     return(ECKeepAction);
  }

  // Get vector intersection.
  if(VCVectorIntersect_GetIntersection(intersection, &intersectionData) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersection returned VC_ERR\n");
     return(ECKeepAction);
  }

  // Add intersection udpate handler.
  if (VCIntersection_AttachUpdateCallback(intersectionData, ObjectIntersectedCallback,
                              (void *)&intersectData) == NULL)
  {
     VC_Error("dvObjectIntersectFunc : Failed to add intersection update handler.\n");
     return(ECKeepAction);
  }

  // Set back the hand intersect mask
  if(VCVectorIntersect_ModifyIntersectMask (vcLimb, oldMask, NULL) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCVectorIntersect_ModifyIntersectMask returned VC_ERR\n");
     return(ECKeepAction);
  }

}
else
{
  // Get the hand intersect mask
  if(VCVectorIntersect_GetIntersectMask (vcLimb, &oldMask) != VC_OK)
  {
     VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersectMask returned VC_ERR\n");
     return(ECKeepAction);
  }
```

```
    if(VCVectorIntersect_ModifyIntersectMask (vcLimb, NULL, oldMask) != VC_OK)
    {
      VC_Error("dvObjectIntersectFunc : VCVectorIntersect_SetIntersectMask returned VC_ERR\n");
      return(ECKeepAction);
    }
    // Evaluate new hitEntity
    if(VCEntity_SetPositionMatrix (hitEntity, handMat) != VC_OK)
    {
      VC_Error("dvObjectIntersectFunc : VCEntity_SetPositionMatrix returned VC_ERR\n");
      return(ECKeepAction);
    }
    // Evaluate new intersection
    if(VCVectorIntersect_Set(intersection, VC_VECTORINTERSECT_ENABLE, NULL, &length,
                newMask, NULL, NULL) != VC_OK)
    {
      VC_Error("dvObjectIntersectFunc : VCVectorIntersect_Set returned VC_ERR\n");
      return(ECKeepAction);
    }
    // Set back the hand intersect mask
    if(VCVectorIntersect_ModifyIntersectMask (vcLimb, oldMask, NULL) != VC_OK)
    {
      VC_Error("dvObjectIntersectFunc : VCVectorIntersect_SetIntersectMask returned VC_ERR\n");
      return(ECKeepAction);
    }
  }

  return(ECKeepAction);
}

//************************************************************************//
// Function:  di_animTimer
//************************************************************************//

int
di_animTimer(VCTimer_CallbackData *callbackData,  void *data)
{
int i;
        if (switches->startanim==1)
        {
        // FEM animation floats->LoadFactor 0.0 to 1.0
                for (i=0; i<100; i++)//loop for sawtooth animation
                {
                        floats->LoadFactor=i/100.0;

                        di_modify_FEM();
                        if (switches->meshdynmode==1) di_modify_Mesh();
                        di_modify_LoadSet();
                        di_modify_ConstraintSet();
                }
                if (switches->animmode==0)//turn on loop for ramp animation
                {
                        for (i=100; i>0; i--)
                        {
                                floats->LoadFactor=i/100.0;

                                di_modify_FEM();
                                if (switches->meshdynmode==1) di_modify_Mesh();
                                di_modify_LoadSet();
                                di_modify_ConstraintSet();
                        }
                }
        }
}

//************************************************************************//
// Function:  di_animalarm
//************************************************************************//

int
```

```
di_animalarm(VCTimer_CallbackData *cd, void *data)
{
        void *animHandle=data;

        if (switches->startanim==0)
        {
                VCTimer_DetachCallback(animHandle);
        }
        else
        {
                if (animHandle)
                {
                        VCTimer_DetachCallback(animHandle); /* stop the animation */
  /* and re-run this function in one seconds time to restart animation */
                        VCTimer_AttachExpiringCallback(1, di_animalarm, NULL);
                }
                else
                {       /* data is NULL, so restart animation */
                        animHandle = VCTimer_AttachPeriodicCallback(100.0/100.0, di_animTimer, NULL);
                        if (!animHandle)
                                printf("Failed to restart animation\n");
                        else
                                VCTimer_AttachExpiringCallback(1, di_animalarm, animHandle);
                }
        }
}

//************************************************************************//
// Function: diToggleAnimFunc
//************************************************************************//

int diToggleAnimFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->startanim, &data.focus) == VC_ERR)
                switches->startanim = -1;

        if (switches->startanim==1)
        {
                void *animHandle;

                animHandle = VCTimer_AttachPeriodicCallback (100.0/100.0, di_animTimer, NULL);

                VCTimer_AttachExpiringCallback(1, di_animalarm, animHandle);
        }
}


//************************************************************************//
// Function: diBodyStartupPosFEMFunc - Sets the zone startup body position
//************************************************************************//

int
diBodyStartupPosFEMFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  dmPoint  s;
  VCBody     *thisBody;
  VCBody     *body = data.body;
  VC_Traverse  traverseInfo;
  void       **args = action->parameters;
  dmMatrix tempMat; /* Get original body position */
        float32     tempX;
        float32     tempY;
        float32     tempZ;

        /* Is there a body? */
  if (body == NULL)
    body = VC_GetFirstBody(&traverseInfo);

  if(body != NULL)
```

```
  {
                VCBody_GetAbsolutePosition (body, tempMat);
                dmPointFromMat(s, tempMat);

//STARTUP HOME (FRONT) VIEW
                tempX = points->FEMcenter[VC_X];
                tempY = points->FEMcenter[VC_Y];
                tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4.0));

                s[VC_X] = tempX;
                s[VC_Y] = tempY;
                s[VC_Z] = tempZ;

    /* Set the current body startup position */
    VCBody_SetPosition(body, NULL, s, NULL, NULL, NULL, NULL);
  }

  /* Accomodate for NULL values and no body */
        s[VC_X] = tempX;
        s[VC_Y] = tempY;
  s[VC_Z] = tempZ;

        /* Set the Global body position */
  if(body != NULL)
    ECZoneSetBodyStartupPosition(ECBodyGetZone(body), s);
  else
    ECZoneSetBodyStartupPosition(ECTopZoneGet(), s);

  return(ECKeepAction);
}

//************************************************************************//
// Function:  diNavModeFunc
//************************************************************************//

int diNavModeFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->navmode, &data.focus) == VC_ERR)//switches->navmode is
navmode
                switches->navmode = 1;
}

//************************************************************************//
// Function:  diSetViewFunc
//************************************************************************//

int diSetViewFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        dmMatrix  tempMat;
        uint32          viewnum;
        VCBody      *body = data.body;


        if(ECArgReferenceGetValue(args[1], (void *)&viewnum, &data.focus) == VC_ERR)
                viewnum = 1;
                switch(viewnum)
        {
                case 1 :    //Set User View 1
                            switches->set1 = 1;//set1
                            VCBody_GetAbsolutePosition (body, tempMat);
//                          dmPointFromMat(points->view1, tempMat);
                            break;
                case 2 :    //Set User View 2
                            switches->set2 = 1;//set2
                            VCBody_GetAbsolutePosition (body, tempMat);
                            dmPointFromMat(points->view2, tempMat);
                            break;
                default :   //Set User View 1
```

```
                        switches->set1 = 1;//set1
                        VCBody_GetAbsolutePosition (body, tempMat);
                        dmPointFromMat(points->view1, tempMat);
                        break;
        }
}

//*************************************************************************//
// Function:  diBodyMoveToFunc - Navigates (in different modes) the body
//         to a given viewpoint or position while orienting on the
//                              center of the FEM or other designated object center
//*************************************************************************//

int
diBodyMoveToFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        static          float32     lasty=0.0;
  float32       time;
  float32       elapsed;
  int32         done = 0;
  float32       rate = 1;
  float32       len;
  dmPoint       newPos;
        dmPoint         towards;
  MoveInfo      *mi;
  VCBody                *body = data.body;
  void          **args = action->parameters;
  VC_Traverse   traverseInfo;
  dmMatrix      tempMat;
        dmEuler         o;
        float32         xdegree,ydegree,zdegree;
        dmVector orientVect;
        dmVector adjvector;
        float32         tempX;
        float32         tempY;
        float32         tempZ;
        uint32          view=-1;
        float32         standoff=20.0;

  if(ECArgReferenceGetValue(args[1], (void *)&view, &data.focus) == VC_ERR)
                view=1;

        switch(view)
        {
                case 1 :    //TOP VIEW
                            tempX = points->FEMcenter[VC_X];
                            tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/4.0));
                            tempZ = points->FEMcenter[VC_Z];
                            towards[VC_X] = points->FEMcenter[VC_X];
                            towards[VC_Y] = points->FEMcenter[VC_Y];
                            towards[VC_Z] = points->FEMcenter[VC_Z];
                            break;
                case 2 :    //BACK VIEW
                            tempX = points->FEMcenter[VC_X];
                            tempY = points->FEMcenter[VC_Y];
                            tempZ = points->FEMcenter[VC_Z]-(floats->xyzmax-(floats->xyzmax/4.0));
                            towards[VC_X] = points->FEMcenter[VC_X];
                            towards[VC_Y] = points->FEMcenter[VC_Y];
                            towards[VC_Z] = points->FEMcenter[VC_Z];
                            break;
                case 3 :    //LEFT VIEW
                            tempX = points->FEMcenter[VC_X]-(floats->xyzmax-(floats->xyzmax/4.0));
                            tempY = points->FEMcenter[VC_Y];
                            tempZ = points->FEMcenter[VC_Z];
                            towards[VC_X] = points->FEMcenter[VC_X];
                            towards[VC_Y] = points->FEMcenter[VC_Y];
                            towards[VC_Z] = points->FEMcenter[VC_Z];
                            break;
                case 4:             //HOME (FRONT) VIEW
                            tempX = points->FEMcenter[VC_X];
```

```
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 5 :  //RIGHT VIEW
        tempX = points->FEMcenter[VC_X]+(floats->xyzmax-(floats->xyzmax/4.0));
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z];
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 6 :  //ISOFRONTLEFT VIEW
        tempX = points->FEMcenter[VC_X]-(floats->xyzmax-(floats->xyzmax/2.0));
        tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/2.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 7 :  //BOTTOM VIEW
        tempX = points->FEMcenter[VC_X];
        tempY = points->FEMcenter[VC_Y]-(floats->xyzmax-(floats->xyzmax/4.0));
        tempZ = points->FEMcenter[VC_Z];
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 8 :  //ISOFRONTRIGHT VIEW
        tempX = points->FEMcenter[VC_X]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/2.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 9 :  //NODE VIEW
        if (switches->picknode == 1)
        {
                tempX = points->rightnodep[VC_X];
                tempY = points->rightnodep[VC_Y];
                tempZ = points->rightnodep[VC_Z];
                towards[VC_X] = points->rightnodep[VC_X];
                towards[VC_Y] = points->rightnodep[VC_Y];
                towards[VC_Z] = points->rightnodep[VC_Z];
        }
        else
        {  // Return early because no node selection
                args[0] = NULL;
                return(ECKeepAction);
        }
        break;
case 10 : //USER VIEW 1
        if (switches->set1 == 1)//set1
        {
                tempX = points->view1[VC_X];
                tempY = points->view1[VC_Y];
                tempZ = points->view1[VC_Z];
                towards[VC_X] = points->FEMcenter[VC_X];
                towards[VC_Y] = points->FEMcenter[VC_Y];
                towards[VC_Z] = points->FEMcenter[VC_Z];
        }
        else
        {  // Return early because points->view1 not set
                args[0] = NULL;
                return(ECKeepAction);
        }
        break;
```

```
                case 11 :   //USER VIEW 2
                            if (switches->set2 == 1)//set2
                            {
                                        tempX = points->view2[VC_X];
                                        tempY = points->view2[VC_Y];
                                        tempZ = points->view2[VC_Z];
                                        towards[VC_X] = points->FEMcenter[VC_X];
                                        towards[VC_Y] = points->FEMcenter[VC_Y];
                                        towards[VC_Z] = points->FEMcenter[VC_Z];
                            }
                            else
                            {   // Return early because points->view2 not set
                                        args[0] = NULL;
                                        return(ECKeepAction);
                            }
                            break;
                default:    //HOME (FRONT) VIEW
                            tempX = points->FEMcenter[VC_X];
                            tempY = points->FEMcenter[VC_Y];
                            tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4));
                            towards[VC_X] = points->FEMcenter[VC_X];
                            towards[VC_Y] = points->FEMcenter[VC_Y];
                            towards[VC_Z] = points->FEMcenter[VC_Z];
                            break;
        }

                switch(switches->navmode) //navmode
        {
                case 1 :    //fast/hyper mode (orient on FEM center)
                // Is there a body?
                if (body == NULL)
                            body = VC_GetFirstBody(&traverseInfo);

                if ((mi = args[0]) == NULL) // first call
                {
                            args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                            dmPointSet (mi->posa,tempX,tempY,tempZ);

                            rate = 400.0;

                // Setup move information parameters
                            mi->body = body;

                            if (body != NULL)
                            {
                                        VCBody_GetAbsolutePosition (body, tempMat);
                                        dmPointFromMat(mi->bodyOffset, tempMat);
                            }
                            else
                            {
                                        mi->bodyOffset[VC_X] = 0.0;
                                        mi->bodyOffset[VC_Y] = 0.0;
                                        mi->bodyOffset[VC_Z] = 0.0;
                            }

                            if (view == 9)
                            {
                                        dmPointSub (adjvector, mi->bodyOffset, towards);
                                        adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                                    (adjvector[1]*adjvector[1])+
                                                                    (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                        adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                                    (adjvector[1]*adjvector[1])+
                                                                    (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                        adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                                    (adjvector[1]*adjvector[1])+
                                                                    (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
```

```
                        dmPointAddVector (mi->posa, mi->posa, adjvector);
                }

                mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                        mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                        mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                if(len != 0)
                {
                        rate /= len;
                }
                else
                {  // Return early because zero distance to move
                        args[0] = NULL;
                        return(ECKeepAction);
                }
                mi->velocity[VC_X] *= rate;
                mi->velocity[VC_Y] *= rate;
                mi->velocity[VC_Z] *= rate;
                mi->time = -1.f;
                if(rate != 0)
                {
                        mi->totalTime = 1.f / rate;
                }
                else
                {  // Return early because zero speed entered
                        args[0] = NULL;
                        return(ECKeepAction);
                }
                ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
        }
// Added this so that we use the time in the zone
//  where the body is.
        time = ECZoneGetTime(ECBodyGetZone(body));
        if (mi->time== -1.f)
        {
        mi->time=time;
        elapsed=0.f;
        }
        else
        {
                elapsed=time-mi->time;
        }

        if (elapsed < mi->totalTime)
        {
    // Animate body
        newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
        newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
        newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
        //Update orientation to towards (FEM center or node (for node view))
                dmPointSub (orientVect, towards, newPos);

                xdegree=(180.0/3.14159251)*
                                (asin(orientVect[1]/
                                        (sqrt((orientVect[0]*orientVect[0])+
                                (orientVect[1]*orientVect[1])+
                                (orientVect[2]*orientVect[2])))));

                ydegree=-1.0*(90+((180.0/3.14159251)*
                                (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                {

                        ydegree=lasty;
                }
```

```
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                return(ECKeepAction);///added

                }
                else
                {
    // Move body to final position
                newPos[VC_X] = mi->posa[VC_X];
                newPos[VC_Y] = mi->posa[VC_Y];
                newPos[VC_Z] = mi->posa[VC_Z];

//Update final orientation to towards (FEM center or node (node view))
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                (orientVect[1]*orientVect[1])+
                                (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                        done = 1;
                }

                if(mi->body != NULL)
                {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
                }
                else
                {
                        VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
                }

                if(done == 1)
                {
    // Clean up
                        free(mi);
                        args[0] = NULL;
                        return(ECRemoveAction);
                }
                return(ECKeepAction);
                break;


                case 2 :    //straight line fly move (orient on FEM center)
                switches->navstate=0;
                // Is there a body?
                if (body == NULL)
```

```
                body = VC_GetFirstBody(&traverseInfo);

        if ((mi = args[0]) == NULL) // first call
        {
                args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                dmPointSet (mi->posa,tempX,tempY,tempZ);

    // Extract user parameters
                if(ECArgReferenceGetValue(args[2], (void *)&rate, &data.focus) == VC_ERR)
                rate = 4.0;
    // Setup move information parameters
                mi->body = body;

                if (body != NULL)
                {
                        VCBody_GetAbsolutePosition (body, tempMat);
                        dmPointFromMat(mi->bodyOffset, tempMat);
                }
                else
                {
                        mi->bodyOffset[VC_X] = 0.0;
                        mi->bodyOffset[VC_Y] = 0.0;
                        mi->bodyOffset[VC_Z] = 0.0;
                }

                if (view == 9)
                {
                        dmPointSub (adjvector, mi->bodyOffset, towards);
                        adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                        (adjvector[1]*adjvector[1])+
                                                        (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                        adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                        (adjvector[1]*adjvector[1])+
                                                        (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                        adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                        (adjvector[1]*adjvector[1])+
                                                        (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                        dmPointAddVector (mi->posa, mi->posa, adjvector);
                }

                mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                        mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                        mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                if(len != 0)
                {
                        rate /= len;
                }
                else
                {  // Return early because zero distance to move
                        args[0] = NULL;
                        return(ECKeepAction);
                }
                mi->velocity[VC_X] *= rate;
                mi->velocity[VC_Y] *= rate;
                mi->velocity[VC_Z] *= rate;
                mi->time = -1.f;
                if(rate != 0)
                {
                        mi->totalTime = 1.f / rate;
                }
                else
                {  // Return early because zero speed entered
                        args[0] = NULL;
                        return(ECKeepAction);
```

```
                        }
                        ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
                }
// Added this so that we use the time in the zone
//  where the body is.
                time = ECZoneGetTime(ECBodyGetZone(body));
                if (mi->time== -1.f)
                {
                mi->time=time;
                elapsed=0.f;
                }
                else
                {
                        elapsed=time-mi->time;
                }

                if (elapsed < mi->totalTime)
                {
    // Animate body
                newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
                newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
                newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
                //Update orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                }
                else
                {
    // Move body to final position
                newPos[VC_X] = mi->posa[VC_X];
                newPos[VC_Y] = mi->posa[VC_Y];
                newPos[VC_Z] = mi->posa[VC_Z];

 //Update final orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
```

```
                    {
                            ydegree=lasty;
                    }
                    else
                    {
                            lasty = ydegree;
                    }

                    zdegree = 0.0;

                    dmEulerSetD(o,xdegree,ydegree,zdegree);

                    done = 1;
            }

            if(mi->body != NULL)
            {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
            }
            else
            {
                    VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
            }

            if(done == 1)
            {
// Clean up
                    free(mi);
                    args[0] = NULL;
                    switches->navstate=1;
                    return(ECRemoveAction);
            }
            return(ECKeepAction);
            break;

            default:    //straight line fly move (orient on FEM center)
            // Is there a body?
            if (body == NULL)
                    body = VC_GetFirstBody(&traverseInfo);

            if ((mi = args[0]) == NULL) // first call
            {
                    args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                    dmPointSet (mi->posa,tempX,tempY,tempZ);

// Extract user parameters
                    if(ECArgReferenceGetValue(args[2], (void *)&rate, &data.focus) == VC_ERR)
                    rate = 4.0;
// Setup move information parameters
                    mi->body = body;

                    if (body != NULL)
                    {
                            VCBody_GetAbsolutePosition (body, tempMat);
                            dmPointFromMat(mi->bodyOffset, tempMat);
                    }
                    else
                    {
                            mi->bodyOffset[VC_X] = 0.0;
                            mi->bodyOffset[VC_Y] = 0.0;
                            mi->bodyOffset[VC_Z] = 0.0;
                    }

                    if (view == 9)
                    {
                            dmPointSub (adjvector, mi->bodyOffset, towards);
                            adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                            (adjvector[1]*adjvector[1])+
                                                            (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
```

```
                                adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                dmPointAddVector (mi->posa, mi->posa, adjvector);
                        }

                        mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                        mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                        mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                        len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                                mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                                mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                        if(len != 0)
                        {
                                rate /= len;
                        }
                        else
                        {  // Return early because zero distance to move
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        mi->velocity[VC_X] *= rate;
                        mi->velocity[VC_Y] *= rate;
                        mi->velocity[VC_Z] *= rate;
                        mi->time = -1.f;
                        if(rate != 0)
                        {
                                mi->totalTime = 1.f / rate;
                        }
                        else
                        {  // Return early because zero speed entered
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
                }
// Added this so that we use the time in the zone
//  where the body is.
                time = ECZoneGetTime(ECBodyGetZone(body));
                if (mi->time== -1.f)
                {
                mi->time=time;
                elapsed=0.f;
                }
                else
                {
                        elapsed=time-mi->time;
                }

                if (elapsed < mi->totalTime)
                {
    // Animate body
                newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
                newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
                newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
                //Update orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
```

```
                                        (dmSafeAtan2 (orientVect[2], orientVect[0])));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                }
                else
                {
    // Move body to final position
                newPos[VC_X] = mi->posa[VC_X];
                newPos[VC_Y] = mi->posa[VC_Y];
                newPos[VC_Z] = mi->posa[VC_Z];

//Update final orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);

                        done = 1;
                }

                if(mi->body != NULL)
                {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
                }
                else
                {
                        VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
                }

                if(done == 1)
                {
     // Clean up
                        free(mi);
                        args[0] = NULL;
                        return(ECRemoveAction);
                }
                return(ECKeepAction);
```

```
                break;
        }
}

//************************************************************************//
// Function:  diToggleMeshDynFunc
//************************************************************************//

int diToggleMeshDynFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->meshdynmode, &data.focus) == VC_ERR)
                switches->meshdynmode = 1;
}

//************************************************************************//
// Function:  diToggleAnimModeFunc
//************************************************************************//

int diToggleAnimModeFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->animmode, &data.focus) == VC_ERR)
                switches->animmode = 1;
}

//************************************************************************//
// Function:  diOutputSetFunc
//************************************************************************//

int diOutputSetFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->outtypenum, &data.focus) == VC_ERR)
                switches->outtypenum = 0;
        if(ECArgReferenceGetValue(args[2], (void *)&switches->outsubnum, &data.focus) == VC_ERR)
                switches->outsubnum = 0;

        di_set_range();

        di_output_mods();

        di_modify_ClrScl();

        di_modify_FEM();

        sprintf(chars->outtxt,"%sNode #: %i\nElement #: %i\n\n%s%10.6f\nDX: %10.6f\nDY: %10.6f\nDZ: %10.6f\n",
                                names->actual_case_name,
                                (NODE_P+((ELEMENT_P+pmi->rightelem)->B[pmi->adjindex]))->A,
                                (ELEMENT_P+pmi->rightelem)->D,
                                names->actual_set_name[(switches->outtypenum*5)+switches->outsubnum],
                                outvert[pmi->rightvert]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+0]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+1]*floats->LoadFactor,
                                displaceobj[(pmi->rightvert*3)+2]*floats->LoadFactor);
        VCString_SetText(femtextstring,chars->outtxt);
        di_updateclrscltxt();
}

int diToggleNavStateFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->navstate, &data.focus) == VC_ERR)
                switches->navstate = 1;
}

//************************************************************************//
// Function:  diToggleLoadFunc - toggles visibility of loads on model                    //
//************************************************************************//
```

```
int diToggleLoadFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        EntityList *tmp = NULL;
        void            **args = action->parameters;

        tmp = malloc (sizeof (EntityList));
        if(ECArgReferenceGetValue(args[1], (void *)&switches->loadcasestate, &data.focus) == VC_ERR)
                switches->loadcasestate = 1;

        if (switches->loadcasestate == 1)
                for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, VC_VISIBLE, 0);
        else
                for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, 0, VC_VISIBLE);
        free(tmp);
}

//*************************************************************************//
// Function:  diCreateLoadObjectsFunc - creates the loads on the model     //
//*************************************************************************//

int diCreateLoadObjectsFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        int i;
        void            **args = action->parameters;
        EntityList *newItem;
        dmEuler o;
        dmScale s;

        LoadList = malloc (sizeof (EntityList));
        LoadList = NULL;

        for(i = 0; i < LOADSET_NUM && i < 100; i++){
                newItem = malloc (sizeof (EntityList));

                // Initialization
                newItem->nodeobj = newItem->vis = newItem->next = NULL;

                // Populate the new item
                if(loadcoordind != NULL){
                        newItem->nodeobj=VCEntity_Create(NULL,0);
                        newItem->vis=VCVisual_CreateGeometry("greenarw");
                        VCVisual_SetIntersectMask (newItem->vis, 1);
                        VCEntity_AttachAttribute (newItem->nodeobj, newItem->vis);
                        newItem->nodepoint[0] = vertices[(loadcoordind[i]*7)+0];
                        newItem->nodepoint[1] = vertices[(loadcoordind[i]*7)+1];
                        newItem->nodepoint[2] = vertices[(loadcoordind[i]*7)+2];

                // Add the new item to the beginning of the list
                if (LoadList == NULL)
                        LoadList = newItem;
                else{
                        newItem->next = LoadList;
                        LoadList = newItem;
                }
                // Creates the points on the model and sets it invisible

                dmEulerSetD (o, 0, 90, 0);
                s[0]=floats->xyzmax/5;
                s[1]=floats->xyzmax/5;
                s[2]=floats->xyzmax/5;
                VCEntity_SetPositionPointEulerScale (LoadList->nodeobj, LoadList->nodepoint, o, s);
                VCVisual_ModifyMode (LoadList->vis, 0, VC_VISIBLE);
        }
        di_modify_LoadSet();
        di_modify_ConstraintSet();
        }
}
```

```
//*****************************************************************************//
// Function:  di_modify_LoadSet - modifies the loads on the model                //
//*****************************************************************************//

int di_modify_LoadSet(void)
{
        EntityList *tmp = NULL;
        int i=0;

        tmp = malloc (sizeof (EntityList));

        for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
        {
                tmp->nodepoint[0] = vertices[(loadcoordind[i]*7)+0]+displaceobj[(loadcoordind[i]*3)+0]*floats-
>LoadFactor*floats->exager;
    tmp->nodepoint[1] = vertices[(loadcoordind[i]*7)+1]+displaceobj[(loadcoordind[i]*3)+1]*floats->LoadFactor*floats->exager;
    tmp->nodepoint[2] = vertices[(loadcoordind[i]*7)+2]+displaceobj[(loadcoordind[i]*3)+2]*floats->LoadFactor*floats->exager;
                VCEntity_SetPositionPoint (tmp->nodeobj, tmp->nodepoint);
                i++;
        }
        free(tmp);
}

//*****************************************************************************//
// Function:  diToggleConstrFunc - toggles visibility of constraints on model  //
//*****************************************************************************//

int diToggleConstrFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        EntityList *tmp = NULL;
        void            **args = action->parameters;

        tmp = malloc (sizeof (EntityList));
        if(ECArgReferenceGetValue(args[1], (void *)&switches->constraintstate, &data.focus) == VC_ERR)
                switches->constraintstate = 1;

        if (switches->constraintstate == 1)
                for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, VC_VISIBLE, 0);
        else
                for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, 0, VC_VISIBLE);
        free(tmp);
}

//*****************************************************************************//
// Function:  diCreateConstrObjectsFunc - creates the constraints on the model//
//*****************************************************************************//

int diCreateConstrObjectsFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        int i;
        void            **args = action->parameters;
        EntityList *newItem;
        dmEuler o;
        dmScale s;

        ConstrList = malloc (sizeof (EntityList));
        ConstrList = NULL;

        for(i = 0; i < CONSTRAINTSET_NUM; i++){
                newItem = malloc (sizeof (EntityList));

                // Initialization
                newItem->nodeobj = newItem->vis = newItem->next = NULL;

                // Populate the new item
                newItem->nodeobj=VCEntity_Create(NULL,0);
                newItem->vis=VCVisual_CreateGeometry("greensphere");
                VCVisual_SetIntersectMask (newItem->vis, 1);                    // Create Intersect Mask
```

```
                    VCEntity_AttachAttribute (newItem->nodeobj, newItem->vis);
                    newItem->nodepoint[0] = vertices[(constrcoordind[i]*7)+0];
    newItem->nodepoint[1] = vertices[(constrcoordind[i]*7)+1];
    newItem->nodepoint[2] = vertices[(constrcoordind[i]*7)+2];

                    // Add the new item to the beginning of the list
                    if (ConstrList == NULL)
                            ConstrList = newItem;
                    else{
                            newItem->next = ConstrList;
                            ConstrList = newItem;
                    }
                    // Creates the points on the model and sets it invisible
                    s[0] = floats->xyzmax/175;
                    s[1] = floats->xyzmax/175;
                    s[2] = floats->xyzmax/175;
                    VCEntity_SetPositionPointEulerScale (ConstrList->nodeobj, ConstrList->nodepoint, NULL, s);
                    VCVisual_ModifyMode (ConstrList->vis, 0, VC_VISIBLE);
            }
            di_modify_ConstraintSet();
}

//*************************************************************************//
// Function: di_modify_ConstraintSet - modifies the constraints on the model //
//*************************************************************************//

int di_modify_ConstraintSet(void)
{
            EntityList *tmp = NULL;
            int i=0;

            tmp = malloc (sizeof (EntityList));

            for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
            {
                    tmp->nodepoint[0] = vertices[(constrcoordind[i]*7)+0]+displaceobj[(constrcoordind[i]*3)+0]*floats-
>LoadFactor*floats->exager;
    tmp->nodepoint[1] = vertices[(constrcoordind[i]*7)+1]+displaceobj[(constrcoordind[i]*3)+1]*floats->LoadFactor*floats-
>exager;
    tmp->nodepoint[2] = vertices[(constrcoordind[i]*7)+2]+displaceobj[(constrcoordind[i]*3)+2]*floats->LoadFactor*floats-
>exager;
                    VCEntity_SetPositionPoint (tmp->nodeobj, tmp->nodepoint);
                    i++;
            }
            free(tmp);
}

//*************************************************************************//
// Function: diCreateViewButtonFunc
//*************************************************************************//

int
diCreateViewButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;

  ECVisual      *visual;
  VCAttribute   *visattribute;

            objViewButtonref = (ECObjectReference *)args[1];
  objViewButton = ECReferenceObject(objViewButtonref, &data.focus);

            visual = ECObjectGetVisual(objViewButton, NULL);

            if (visual == NULL)
              {
               VC_Error("visual was NULL\n");
               return(ECKeepAction);
              }
            visattribute = ECVisualGetVCAttribute(visual);
```

```
        ECVisualToVC (objViewButton, visual);
        ECObjectToVC(objViewButton);
        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateViewTextFunc
//*************************************************************************//

int
diCreateViewTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

        objViewTextref = (ECObjectReference *)args[1];
  objViewText = ECReferenceObject(objViewTextref, &data.focus);

        visual = ECObjectGetVisual(objViewText, NULL);

        if (visual  == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objViewText, visual);
        ECObjectToVC(objViewText);
        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateDataButtonFunc
//*************************************************************************//

int
diCreateDataButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

        objDataButtonref = (ECObjectReference *)args[1];
  objDataButton = ECReferenceObject(objDataButtonref, &data.focus);

        visual = ECObjectGetVisual(objDataButton, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objDataButton, visual);
        ECObjectToVC(objDataButton);
        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateDataTextFunc
//*************************************************************************//

int
diCreateDataTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;
```

```
        objDataTextref = (ECObjectReference *)args[1];
  objDataText = ECReferenceObject(objDataTextref, &data.focus);

        visual = ECObjectGetVisual(objDataText, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objDataText, visual);
        ECObjectToVC(objDataText);
        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateVisButtonFunc
//*************************************************************************//

int
diCreateVisButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECVisual       *visual;
   VCAttribute    *visattribute;

        objVisButtonref = (ECObjectReference *)args[1];
   objVisButton = ECReferenceObject(objVisButtonref, &data.focus);

        visual = ECObjectGetVisual(objVisButton, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objVisButton, visual);
        ECObjectToVC(objVisButton);
        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateVisTextFunc
//*************************************************************************//

int
diCreateVisTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECVisual       *visual;
   VCAttribute    *visattribute;
   VCEntity       *vistextent = NULL;

        objVisTextref = (ECObjectReference *)args[1];
   objVisText = ECReferenceObject(objVisTextref, &data.focus);

        visual = ECObjectGetVisual(objVisText, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objVisText, visual);
        ECObjectToVC(objVisText);
        return(ECKeepAction);
}
```

```
//*************************************************************************//
// Function:  ToolCreation_cb
//*************************************************************************//

int ToolCreation_cb(TBTool *tool)
{
 SliderDataStruct   *myData;

 printf("Setting up user Data structure...\n");
 myData = (SliderDataStruct *)calloc(2, sizeof(SliderDataStruct));
 TBGenSetUserData(tool, (void *)myData);
}

//*************************************************************************//
// Function:  WidgetCreation_cb
//*************************************************************************//

int WidgetCreation_cb(VWidget *newWidget, TBTool *tool, void *data)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        if ((data != NULL) && (myData != NULL))
  {
                if (!(strcmp((char*)data, "LoadFact")))
                {
                        printf("Got reference to LoadFact = 0x%x\n", newWidget);
                        myData->LoadFact = newWidget;
                        VWScalar_SetValue(myData->LoadFact,100, FALSE);
                }
                if (!(strcmp((char*)data, "LoadDisp")))
                {
                        printf("Got reference to LoadDisp = 0x%x\n", newWidget);
                        myData->LoadDisp = newWidget;
                        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
                }
                if (!(strcmp((char*)data, "ThreshFact")))
                {
                        printf("Got reference to ThreshFact = 0x%x\n", newWidget);
                        myData->ThreshFact = newWidget;
                        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
                }
                if (!(strcmp((char*)data, "ThreshDisp")))
                {
                        printf("Got reference to ThreshDisp = 0x%x\n", newWidget);
                        myData->ThreshDisp = newWidget;
                        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
                }
                if (!(strcmp((char*)data, "ExagerFact")))
                {
                        printf("Got reference to ExagerFact = 0x%x\n", newWidget);
                        myData->ExagerFact = newWidget;
                        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
                }
                if (!(strcmp((char*)data, "ExagerDisp")))
                {
                        printf("Got reference to ExagerDisp = 0x%x\n", newWidget);
                        myData->ExagerDisp = newWidget;
                        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclTop")))
                {
                        printf("Got reference to ClrSclTop = 0x%x\n", newWidget);
                        myData->ClrSclTop = newWidget;
                        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclTopDisp")))
                {
```

```
                printf("Got reference to ClrSclTopDisp = 0x%x\n", newWidget);
                myData->ClrSclTopDisp = newWidget;
                VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
        }
        if (!(strcmp((char*)data, "ClrSclBot")))
        {
                printf("Got reference to ClrSclBot = 0x%x\n", newWidget);
                myData->ClrSclBot = newWidget;
                VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
        }
        if (!(strcmp((char*)data, "ClrSclBotDisp")))
        {
                printf("Got reference to ClrSclBotDisp = 0x%x\n", newWidget);
                myData->ClrSclBotDisp = newWidget;
                VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
        }
  }
}

//*************************************************************************//
// Function:  UpdateSlider_cb
//*************************************************************************//

int UpdateSliderInfo_cb(VWidget *scalarWig, VWEventInfo *info, void *data)
{
        ECObject *obj;
        TBTool *thisTool;
        float32 newValue;
        SliderDataStruct   *myData;
        char            *calldata;

        if (!(thisTool = TBGenGetTool(data)))
  {
                return;
  }

        calldata = (char *)TBGenGetCalldata(data);
        myData = (SliderDataStruct *)TBGenGetUserData(thisTool);

        newValue = VWScalar_GetValue(scalarWig);
        if (!(strcmp((char*)calldata, "LoadFact")))
        {
                VWDigit_SetValue(myData->LoadDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ThreshFact")))
        {
                VWDigit_SetValue(myData->ThreshDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ExagerFact")))
        {
                VWDigit_SetValue(myData->ExagerDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ClrSclTop")))
        {
                VWDigit_SetValue(myData->ClrSclTopDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ClrSclBot")))
        {
                VWDigit_SetValue(myData->ClrSclBotDisp, (int)(newValue), FALSE);
        }
}

//*************************************************************************//
// Function:  UpdateSlider_cb
//*************************************************************************//

int UpdateSlider_cb(VWidget *scalarWig, VWEventInfo *info, void *data)
{
        ECObject *obj;
        uint32 *eventId;
```

```
        TBTool *thisTool;
        float32 newValue;
        SliderDataStruct   *myData;
        char            *calldata;
        float32 delta,out_new;

        if (!(thisTool = TBGenGetTool(data)))
{
                return;
}

        calldata = (char *)TBGenGetCalldata(data);
        myData = (SliderDataStruct *)TBGenGetUserData(thisTool);

        newValue = VWScalar_GetValue(scalarWig);

        if (!(strcmp((char*)calldata, "LoadFact")))
        {
                VWDigit_SetValue(myData->LoadDisp, (int)(newValue), FALSE);
                floats->LoadFactor = (float32)(newValue)/100.0f;
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
                di_modify_LoadSet();
                di_modify_ConstraintSet();
        }
        if (!(strcmp((char*)calldata, "ThreshFact")))
        {
                VWDigit_SetValue(myData->ThreshDisp, (int)(newValue), FALSE);
                floats->threshold = ((float32)(newValue)/100.0f);
                floats->out_vals[1] = floats->absmax*floats->threshold;
                di_modify_ClrScl();
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
        }
        if (!(strcmp((char*)calldata, "ExagerFact")))
        {
                VWDigit_SetValue(myData->ExagerDisp, (int)(newValue), FALSE);
                floats->exager = (float32)(newValue);
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
                di_modify_ConstraintSet();
                di_modify_LoadSet();
        }
        if (!(strcmp((char*)calldata, "ClrSclTop")))
        {
                VWDigit_SetValue(myData->ClrSclTopDisp, (int)(newValue), FALSE);
                floats->clrscltop = (float32)(newValue)/100.0f;
                floats->out_vals[2]=floats->out_min+
                                                        (floats->clrscltop*
                                                        (floats->out_max-floats->out_min));
                di_modify_ClrScl();
                di_updateclrscltxt();
                di_modify_FEM();
        }
        if (!(strcmp((char*)calldata, "ClrSclBot")))
        {
                VWDigit_SetValue(myData->ClrSclBotDisp, (int)(newValue), FALSE);
                floats->clrsclbot = (float32)(newValue)/100.0f;
                floats->out_vals[0]=floats->out_min+
                                                        (floats->clrsclbot*
                                                        (floats->out_max-floats->out_min));
                di_modify_ClrScl();
                di_updateclrscltxt();
                di_modify_FEM();
        }
}

//*************************************************************************//
// Function:  SetSliders_cb
//*************************************************************************//
```

```
int SetSliders_cb(ECObject *obj, VCBody *body, VCAttribute *limb, TBTool *tool)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        VWScalar_SetValue(myData->LoadFact,100, FALSE);
        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
        VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
        VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
        VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
}

//*********************************************************************//
// Function: ResetSliders_cb
//*********************************************************************//

int ResetSliders_cb(ECObject *obj, VCBody *body, VCAttribute *limb, TBTool *tool)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        VWScalar_SetValue(myData->LoadFact,100, FALSE);
        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
        VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
        VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
        VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
}

/* PUBLIC FUNCTION DEFINITIONS ========================================== */

extern void RegisterScaleToolFunctions(void)
{
        TBRegisterToolCreationCallback("myToolCreation",
                ToolCreation_cb);
        TBRegisterGenericWidgetCreationCallback("myWidgetCreation",
                WidgetCreation_cb);
        TBRegisterGenericWidgetCallback("UpdateSliderInfo",
                UpdateSliderInfo_cb);
        TBRegisterGenericWidgetCallback("UpdateSlider",
                UpdateSlider_cb);
        TBRegisterGenericObjectSelectCallback("setSliders",
                SetSliders_cb);
        TBRegisterGenericObjectSelectCallback("resetSliders",
                ResetSliders_cb);
}


//*********************************************************************//
// Function: main
//*********************************************************************//

int
main (int argc, char **argv)
{
        extern void RegisterScaleToolFunctions(void);
```

```
points=(Points *)malloc(sizeof(Points));
switches=(Switches *)malloc(sizeof(Switches));
floats=(Floats *)malloc(sizeof(Floats));
chars=(Chars *)malloc(sizeof(Chars));
vcfloats=(VCfloats *)malloc(sizeof(VCfloats));
intersectionReportData=(VCIntersectionReportData *)malloc(sizeof(VCIntersectionReportData));

objFEM=(ECObject *)malloc(sizeof(ECObject));
objFEMref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objMesh=(ECObject *)malloc(sizeof(ECObject));
objMeshref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objFEMText=(ECObject *)malloc(sizeof(ECObject));
objFEMTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objClrScl=(ECObject *)malloc(sizeof(ECObject));
objClrSclref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objClrSclGrid=(ECObject *)malloc(sizeof(ECObject));
objClrSclGridref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objViewButton=(ECObject *)malloc(sizeof(ECObject));
objViewButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objViewText=(ECObject *)malloc(sizeof(ECObject));
objViewTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objDataButton=(ECObject *)malloc(sizeof(ECObject));
objDataButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objDataText=(ECObject *)malloc(sizeof(ECObject));
objDataTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objVisButton=(ECObject *)malloc(sizeof(ECObject));
objVisButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
objVisText=(ECObject *)malloc(sizeof(ECObject));
objVisTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));

femtextstring=(VCGeometry *)malloc(sizeof(VCGeometry));
clrscltextstring=(VCGeometry *)malloc(sizeof(VCGeometry));

switches->navstate=1;//navmode
switches->navmode=2;//navmode
switches->set1=0;//set1
switches->set2=0;//set2
switches->picknode=0;//picknode
switches->meshdynmode=1;//meshdynmode
switches->outtypenum=0;//0 is node type output, 1 is element type output
switches->outsubnum=0;//node or element subtype index (0-4) in output array
switches->animmode=1;
switches->startanim=-1;//meshdynmode
switches->loadcasestate=0;
switches->constraintstate=0;

floats->LoadFactor=1.0;
floats->exager=1.0;
floats->threshold=0.0;
floats->beamdelta=100;
floats->xyzmax=0.0;
floats->absmax=0.0;
floats->out_min=100000;
floats->out_max=-100000;
floats->clrscltop=1.0;
floats->clrsclbot=0.0;
floats->femsclbotl[0]=          0.0;
floats->femsclbotl[1]=            0.0;
floats->femsclbotl[2]=              0.0;
floats->femsclbotr[0]=          .03;
floats->femsclbotr[1]=            0.0;
floats->femsclbotr[2]=              0.0;
floats->femscltopr[0]=          .03;
floats->femscltopr[1]=            .294;
floats->femscltopr[2]=              0.0;
floats->femscltopl[0]=          0.0;
floats->femscltopl[1]=            .294;
floats->femscltopl[2]=              0.0;
floats->alphainrng=             1.0;
floats->alphathresh=  0.7;
```

```
        floats->alphaoutrng= 0.0;

        vcfloats->posmaxcolor[0]=1.0;//red
        vcfloats->posmaxcolor[1]=  0.0;
        vcfloats->posmaxcolor[2]=     0.0;
        vcfloats->posmincolor[0]=1.0;//yellow
        vcfloats->posmincolor[1]=  1.0;
        vcfloats->posmincolor[2]=     0.0;
        vcfloats->negmincolor[0]=0.0;//greenblue
        vcfloats->negmincolor[1]=  1.0;
        vcfloats->negmincolor[2]=     1.0;
        vcfloats->negmaxcolor[0]=0.0;//green
        vcfloats->negmaxcolor[1]=  1.0;
        vcfloats->negmaxcolor[2]=     0.0;
        vcfloats->posthreshcolor[0]=0.5;//white
        vcfloats->posthreshcolor[1]=  0.5;
        vcfloats->posthreshcolor[2]=     0.5;
        vcfloats->negthreshcolor[0]=1.0;//white
        vcfloats->negthreshcolor[1]=  1.0;
        vcfloats->negthreshcolor[2]=     1.0;
        vcfloats->outofrngcolor[0]=0.0;//black
        vcfloats->outofrngcolor[1]=  0.0;
        vcfloats->outofrngcolor[2]=     0.0;

        chars->outtxt[200]=" ";
        chars->scltxt[200]=" ";

ucf_fem2vr();

        ECUserActionFuncRegister(diCreateFEMObjectFunc,"diCreateFEMObject",
              "Converts FEM output files into objects",
                        ECDataTypeObject, "ObjectName",
              ECDataTypeFloatVar, "ObjectScale",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateFEMMeshFunc,"diCreateFEMMesh",
              "Creates FEM wireframe mesh",
                        ECDataTypeObject, "ObjectName",
              ECDataTypeFloatVar, "ObjectScale",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateFEMTextFunc,"diCreateFEMText",
              "Creates an dynamic text visual in femtext",
                        ECDataTypeObject, "femtext",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateClrSclTextFunc,"diCreateClrSclText",
              "Creates an dynamic color scale number visual in clrscltext",
                        ECDataTypeObject, "clrscltext",
                        ECDataTypeNull);

 ECUserActionFuncRegister(diCreateColorSclFunc,"diCreateColorScl",
              "Creates an color scale visual in femscale",
                        ECDataTypeObject, "colorscale",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateColorSclGridFunc,"diCreateColorSclGrid",
              "Creates an color scale grid visual in femscalegrid",
                        ECDataTypeObject, "colorgrid",
                        ECDataTypeNull);

        ECUserActionFuncRegister(diToggleAnimFunc,"diToggleAnim",
              "Toggles animation of FEM on and off",
              ECDataTypeIntVar, "StartAnim",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diToggleAnimModeFunc,"diToggleAnimMode",
              "Toggles animation mode from Sawtooth to Ramp",
              ECDataTypeIntVar, "AnimMode",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diImmersDataFunc, "diImmersData",
              "Get data at intersection point",
              ECDataTypeString, "bodyPart",
              ECDataTypeEvent, "Event",
              ECDataTypeObject, "FEMObj",
```

```
            ECDataTypeNull);
        ECUserActionFuncRegister(diBodyStartupPosFEMFunc, "diBodyStartupPosFEM",
            "Set StartUp Body Position FEM Viewpoint",
            ECDataTypeNull);
ECUserActionFuncRegister(diBodyMoveToFunc, "diBodyMoveTo",
            "Moves the body to a given viewpoint at a given speed",
            ECDataTypeIntVar, "ViewNumber",
            ECDataTypeFloatVar, "speed(m/s)",
            ECDataTypeNull);
        ECUserActionFuncRegister(diNavModeFunc,"diNavMode",
            "Sets navigation mode parameter",
            ECDataTypeIntVar, "navmode",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diSetViewFunc,"diSetView",
            "Sets user defined viewpoints",
            ECDataTypeIntVar, "viewnum",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diToggleMeshDynFunc,"diToggleMeshDyn",
            "Sets navigation mode parameter",
            ECDataTypeIntVar, "meshdynmode",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diOutputSetFunc,"diOutputSet",
            "Sets FEM output data set",
            ECDataTypeIntVar, "outtypenum",
            ECDataTypeIntVar, "outsubnum",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diToggleNavStateFunc,"diToggleNavState",
            "Toggles navigation mode from No HeadTrack to HeadTrack",
            ECDataTypeIntVar, "NavState",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateLoadObjectsFunc,"diCreateLoadObjects",
                                                        "Creates load case objects",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diToggleLoadFunc,"diToggleLoad",
            "Toggles loadcase visual",
            ECDataTypeIntVar, "LoadCaseState",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateConstrObjectsFunc,"diCreateConstrObjects",
                                                        "Creates constraints objects",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diToggleConstrFunc,"diToggleConstr",
            "Toggles constraints visual",
            ECDataTypeIntVar, "ConstrCaseState",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateViewButtonFunc,"diCreateViewButton",
            "Creates a View button attached to viewpoint",
                        ECDataTypeObject, "viewbutton",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateViewTextFunc,"diCreateViewText",
            "Creates a View text attached to viewpoint",
                        ECDataTypeObject, "viewtext",
                        ECDataTypeNull);
        ECUserActionFuncRegister(diCreateDataButtonFunc,"diCreateDataButton",
            "Creates a Data button attached to viewpoint",
                        ECDataTypeObject, "databutton",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateDataTextFunc,"diCreateDataText",
            "Creates a Data text attached to viewpoint",
                        ECDataTypeObject, "datatext",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateVisButtonFunc,"diCreateVisButton",
            "Creates a Visualize button attached to viewpoint",
                        ECDataTypeObject, "visbutton",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateVisTextFunc,"diCreateVisText",
            "Creates a Visualize text attached to viewpoint",
                        ECDataTypeObject, "vistext",
                        ECDataTypeNull);

        RegisterScaleToolFunctions();
```

```
        //printf("WHY IS THIS HAPPENING\n");
        VC_AttachBodyCreateCallback (di_create_body_handler, NULL);

        dVISE_Initialise(argc,argv);

        VC_MainLoop();
}
```

```
/*****************************************************************************
DVET Release 2.2/11/98 for SGI Workstation
fem2vr1120.h
11 February 1998
Copyright 1998
Dual Incorporated/University of Central Florida


typedef struct NODE_DATA
        {
        long int A;
        double x;
        double y;
        double z;
        double dx;
        double dy;
        double dz;
        double output_data[5];
        //add in Oct., 1997

        int H;

        //add in Oct., 1997
        } NODE_DATA;

typedef struct ELEMENT_REL
        {
        long int A;
        double data[5];
        } ELEMENT_REL;


typedef struct ELEMENT_DATA
        {
        long int A;
        long int B[4];

        double C[5];
        long int D;
//revised on Sept 30, 1997
        //E is the index for internal element (default) and zero for surface element
        int E;
        //F is an index to reference the element property
        int F;
//revised on Sept 30, 1997
        } ELEMENT_DATA;

//revised on Sept 30, 1997
typedef struct ELEMENT_PROPERTY
        {
        //A is the type of element
        int A;
        //revised in Oct., 1997

        int H;//for material id.

                //revised in Oct., 1997
        //B is the element properties, according to the manual of FEMAP neutral file
        double B[100];
        } ELEMENT_PROPERTY;
//above are revised on Sept 30, 1997

typedef struct NAMES
        {
        char actual_case_name[30];
        char actual_set_name[10][30];
        } NAMES;

//revised in Oct. ,1997
```

```
typedef struct MATERIAL
{
int A;
char title[25];
double Young_Modulus[3];
double Shear_Modulus[3];
double Poisson_Ratio[3];
double GMatrix[21];
double alpha[6];
double k[6];
double thermal_cap,density,damping,temperature;
double tension_limit[2];
double comp_limit[2];
double shear_limit;
} MATERIAL;

//
typedef struct CONSTRAINT
{int A;
char B[25];
long int NUM;
fpos_t file_constraint;
long int *ID;
int *INDEX;
} CONSTRAINT;

typedef struct COORDINATE
{
        int A;//id
        int B; //id of
        int C;//type
char D[25];
double E[3];//origin coordiantes
double F[3];//rotation angles
} COORDINATE;

typedef struct LOAD
{
int SET_ID;
char NAME[25];
fpos_t load_file,nt_file,et_file;
long int NUM,NT_NUM,ET_NUM;
long int *ID,*NT_ID,*ET_ID;
int *TYPE,*FACE;
double *VALUE,*NT_VALUE,*ET_VALUE;
} LOAD;//the default limitation for load set number is 100


//revised in Oct., 1997

extern struct NODE_DATA *NODE_P;
extern struct ELEMENT_DATA *ELEMENT_P;
extern struct NAMES *names;
//revised in Oct., 1997

extern struct MATERIAL *MATERIAL_P;
extern struct COORDINATE *COORDINATE_P;
extern struct CONSTRAINT CONSTRAINT_SET[100];
extern struct LOAD LOAD_SET[100];

//revised 16 Jan 98
extern long int NODE_NUM,ELEMENT_NUM,LOADSET_NUM,LOADSET_PICK,
CONSTRAINTSET_NUM,CONSTRAINTSET_PICK;
//revised 16 Jan 98
```

```
/*****************************************************************************
DVET Release 2.2/11/98 for SGI Workstation
fem2vrsgi.c
11 February 1998
Copyright 1998
Dual Incorporated/University of Central Florida

26/12/97   Ola Fakinlede          Added gui prompt for file input
//
//*************************************************************************//

#include "stdio.h"
#include "string.h"
#include "malloc.h"
//#include "process.h"
#include "stdlib.h"

//********************** User defined header file **********************//

#include "fm2vr1120.h"

//*************************************************************************//

struct NODE_DATA *NODE_P;
struct ELEMENT_DATA *ELEMENT_P;
struct NAMES *names;
//added with OCT., 1997

struct MATERIAL *MATERIAL_P;
struct COORDINATE *COORDINATE_P;
struct LOAD LOAD_SET[100];
struct CONSTRAINT CONSTRAINT_SET[100];

//added with OCT., 1997

struct ELEMENT_REL *ELEMENT_TMP;
struct ELEMENT_DATA *ELEMENT_INF;
//revised on Sept 30, 1997
struct ELEMENT_PROPERTY *ELEMENT_PROPERTY_P;
//above revised on Sept 30, 1997

long int
NODE_NUM,ELEMENT_NUM1,ELEMENT_NUM,output_set_num,LOADSET_NUM,LOADSET_PICK,CONSTRAINTSET_NU
M,CONSTRAINTSET_PICK;
long int u;
long int IA,IB,IC,IE;
//revised on Sept 30, 1997
int ELEMENT_PRO_NUM;
//above revised on Sept 30, 1997
// added with OCT., 1997

int MATERIAL_NUM;
int CONSTRAINT_NUM;
int COORDINATE_NUM;
int LOAD_NUM;

// added with OCT., 1997

//Function Prototypes
//revised on Oct. 22,1997
int compare(long int  ELEMENT_i,long int NODE_i);
//revised on Oct. 22, 1997
long int FindNid(long int u);
long int FindEid(long int u);

/*********************************External Declarations ******************/

extern char *file_prompt();                                    // Function that calls file prompt
extern char *bool_prompt(char *);                              // Function that calls bool prompt
extern char *case_prompt(char set_name[3000][30], int);        // Function that calls case prompt
```

```
extern char **output_data_prompt(char temp_name[2000][40], int);                    // Function that calls output data prompt
extern char *loadset_prompt(char loadset_names[100][30], int);                      // Function that calls load prompt
extern char *constraintset_prompt(char constraintset_names[100][40], int);          // Function that calls constraint prompt

//*************************************************************************//
// Function:  main
// Inputs:  none
// Outputs:
// Date revised and comments:
//*************************************************************************//

//void main(void)
void ucf_fem2vr(void)
{
long int CHECKD,NODE_i,ELEMENT_i,CHECKDD,case_set_num[2000], q;
char OUTPUT[30], set_name[3000][30],temp_set_name[2000][30],out_set_name[2000][30], temp_name[2000][40];
char buffer[200];
double X,Y,Z,TIME[2000],MAX_VALUE[2000],MIN_VALUE[2000],AMAX_VALUE[2000];
//add in Oct., 1997

long int NODE_NUM_S, ELEMENT_NUM_S;

//add in Oct., 1997
long int ID[20],Total_num[2000],case_num[2000];
int flag,
        flag_open_file = 0,             // Flag indicating open file
        flag_solid = 0,                 // Flag indicating solid
        LOAD_YES = 0,                   // Flag indicating yes to load loads
        CONSTRAINT_YES = 0;             // Flag indicating yes to load constraints

FILE *NEU_INP;
//
FILE *fp,*fp1,*fp2, *fp_load, *fp_constraint;
//add in Oct., 1997

FILE *tmp1,*tmp2;
int V_NUM = 5,U_NUM = 5;

//add in Oct., 1997
//
//int FLAG,ID_BLOCK,CHECK,TYPE[2000],V[5],U[5],V_NUM,U_NUM,case_n;
int FLAG,ID_BLOCK,CHECK,TYPE[2000],V[5],U[5],case_n;
int i,CHECK1,II,case_i,case_nn, r;
int I1,I2,I3,I4;

fpos_t file_node,file_element,file_output[2000];
//revised on Sept 30, 1997
fpos_t file_pro;
//revised on Sept 30, 1997
//revised in Oct., 1997

fpos_t file_mat,file_coordinate;
char *filename;                         // FEMAP Neutral File
char *bool;                             // YES or NO
char gui_name[3];               // Name id for gui
char *case_name;                // case name
char case_names[3000][30];      // set_name without carriage return
char **output_data = NULL;              // output data
int j = 0, m = 0, k = 0;
char *loadset_name = "initialization";                  // load name
char loadset_names[100][30];    // loadset_name without carriage return
char *constraintset_name = "initialization";            // constraint name
char constraintset_names[100][40];      // constraintset_name without carriage return

//revised in Oct., 1997

//**********************open neutral file*************************//

/*
while(flag_open_file == 0){
```

```
        filename = file_prompt();
        //strncpy(filename, "ngst.neu", 9);
        if(strncmp(filename, "cancel", 6) == 0)
                exit(1);
        if(*filename == '\0')
                continue;
        else if((NEU_INP = fopen(filename, "r+")) != NULL)
                        flag_open_file = 1;
}
*/

sleep(5);
//system("clear");
printf("\n\n\n\nPlease enter the new FEMAP neutral file\n");
scanf("%s",OUTPUT);
//strcpy(OUTPUT,"ngst.neu"); //BEC+

if((NEU_INP=fopen(OUTPUT,"r+"))==NULL)
{
        printf("Error in opening file: %s\n",OUTPUT);
        exit(1);
}

//*************************************************************************//
// Check FEMAP neutral file
//*************************************************************************//

FLAG = 1;
while((!feof(NEU_INP)) && (FLAG == 1))
        {
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);              // Move the file pointer
        if((CHECK==-1)&&(FLAG==1))
                {
                fscanf(NEU_INP,"%d",&ID_BLOCK);
                fgets(buffer,200,NEU_INP);
                //printf("%d\n",ID_BLOCK);
                switch(ID_BLOCK)
                        {
                        case 100:
                                fgets(buffer,200,NEU_INP);
                                fgets(buffer,200,NEU_INP);
                                fscanf(NEU_INP,"%d",&CHECK);
                                fgets(buffer,200,NEU_INP);
                                if(CHECK==-1)
                                        {
                                        FLAG=1;
                                        }
                                break;
                        case 405:
                        //      printf("Process the information about corrdinate systems...\n");
//add in Oct., 1997

                        fgetpos(NEU_INP,&file_coordinate);
                        COORDINATE_NUM=0;

//add in Oct., 1997
                                label405:
                                fscanf(NEU_INP,"%d",&CHECK);
                                if(CHECK!=-1)
                                        {
                                        //add in Oct., 1997

                                        COORDINATE_NUM=COORDINATE_NUM+1;

                                        //add in Oct., 1997
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
```

```
                goto label405;
                }
        FLAG=1;
        break;
case 475:
        //printf("Process the text information...\n");
        label475:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                for(i=0;i<=II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label475;
                }
        FLAG=1;
        break;
case 410:
        //printf("Process the varaible information...\n");
        label410:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label410;
                }
        FLAG=1;
        break;
case 413:
//      printf("Process the layer information...\n");
        label413:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                goto label413;
                }
        FLAG=1;
        break;
case 470:
        //printf("Process the point information...\n");
        label470:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                goto label470;
                }
        FLAG=1;
        break;
case 471:
//      printf("Process the curve information...\n");
        label471:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
```

```
                goto label471;
                }
        FLAG=1;
        break;
case 472:
        //printf("Process the surface information...\n");
        label472:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label472;
                }
        FLAG=1;
        break;
case 473:
        //printf("Process the volume information...\n");
        label473:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label473;
                }
        FLAG=1;
        break;
case 474:
        //printf("Process the boundary information...\n");
        label474:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                goto label474;
                }
        FLAG=1;
        break;
case 401:
        //printf("Process the material information...\n");
//revised in Oct., 1997

        MATERIAL_NUM=0;
        fgetpos(NEU_INP,&file_mat);

//revised in Oct., 1997
        label401:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
                {
//revised in Oct, 1997

        MATERIAL_NUM=MATERIAL_NUM+1;
        //      printf("m=%d\n",MATERIAL_NUM);

//revised in Oct. 1997
                for(i=0;i<32;i++)
                        {
                        fgets(buffer,200,NEU_INP);

                        }
                goto label401;
                }
        FLAG=1;
```

```
            break;
case 402:
            //printf("Process the property information...\n");
//revised on Sept 30, 1997
            ELEMENT_PRO_NUM=0;
            fgetpos(NEU_INP,&file_pro);
//revised on Sept 30, 1997
            label402:
            fscanf(NEU_INP,"%d",&CHECK);
            fgets(buffer,200,NEU_INP);
            if(CHECK!=-1)
                        {
            //revised on Sept 30, 1997
                        ELEMENT_PRO_NUM=ELEMENT_PRO_NUM+1;
            //revised on Sept 30, 1997
                        fgets(buffer,200,NEU_INP);
                        fgets(buffer,200,NEU_INP);
                        fscanf(NEU_INP,"%d",&II);
                        fgets(buffer,200,NEU_INP);
                        for(i=0;i<(float)(II/8)+1.0;i++)
                                    {
                                    fgets(buffer,200,NEU_INP);
                                    }
                        fscanf(NEU_INP,"%d",&II);
                        fgets(buffer,200,NEU_INP);
                        for(i=0;i<(float)(II/5)+1.0;i++)
                                    {
                                    fgets(buffer,200,NEU_INP);
                                    }
                        goto label402;
                        }
            FLAG=1;
            break;
case 403:
            NODE_NUM=0;
            fgetpos(NEU_INP,&file_node);
            //printf("Process the node information...\n");
            label403:
            fscanf(NEU_INP,"%d,",&CHECK);
            if(CHECK!=-1)
                        {
                        fgets(buffer,200,NEU_INP);
                        NODE_NUM=NODE_NUM+1;
            //          printf("n=%ld\n",NODE_NUM);
                        goto label403;
                        }
            FLAG=1;
            break;
case 404:
            fgetpos(NEU_INP,&file_element);
            ELEMENT_NUM=0;
            ELEMENT_NUM1=0;
//          printf("Process the element information...\n");
            label404:
            fscanf(NEU_INP,"%d,",&CHECK);
            if(CHECK!=-1)
                        {
                        fscanf(NEU_INP,"%d,%d,%d,%d,",&I1,&I2,&I3,&I4);
            switch(I4)
                        {
                        case 0:
                                    ELEMENT_NUM=ELEMENT_NUM+1;
                                    break;
                        case 2:
                                    ELEMENT_NUM=ELEMENT_NUM+1;
                                    break;
                        case 3:
                                    ELEMENT_NUM=ELEMENT_NUM+1;
                                    break;
                        case 4:
```

```
                                        ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                case 5:
                                        ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                case 6:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+4;
                                        break;
                                case 7:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+5;
                                        break;
                                case 8:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+6;
                                case 9:
                                        break;
                                case 10:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+4;
                                        break;
                                case 11:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+5;
                                        break;
                                case 12:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+6;
                                        break;
                                case 13:
                                //      ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                }

                        ELEMENT_NUM1=ELEMENT_NUM1+1;
                        for(i=0;i<7;i++)
                                {
                                fgets(buffer,200,NEU_INP);
                                }
                        if(I4==13)
                                {
                                CHECKD=0;
                                while(CHECKD!=-1)
                                        {
                                        fscanf(NEU_INP,"%ld,",&CHECKD);
                                        fgets(buffer,200,NEU_INP);
                                        }
                                }
                        goto label404;
                        }
                        FLAG=1;
                        break;

//*********************     CONSTRAINT INFORMATION          ****************************//

                        case 406:
                                //printf("Process the constraint information...\n");

//Switch block with revised Oct block below
/*                      label406:
                        fscanf(NEU_INP,"%d,",&CHECK);
                        fgets(buffer,200,NEU_INP);
                        if(CHECK!=-1)
                                {
                                fgets(buffer,200,NEU_INP);
                                fscanf(NEU_INP,"%d,",&CHECK1);
                                do{
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
```

```
                                        } while(CHECK1!=-1);
                                fgets(buffer,200,NEU_INP);
                                fscanf(NEU_INP,"%d,",&CHECK1);
                                fgets(buffer,200,NEU_INP);
                                if(CHECK1!=-1)
                                        {
                                        fscanf(NEU_INP,"%d",&II);
                                        fgets(buffer,200,NEU_INP);
                                        for(i=0;i<II;i++)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                }
                                        }
                                goto label406;
                                }
//Switch block with revised Oct block below
*/
//revised in Oct., 1997(Switch with block above)

                        CONSTRAINT_NUM = 0;

                        label406:
                                if(CONSTRAINT_NUM > 100)
                                {
                                        printf("The number of constraint sets exceeds the default value of 100\n");
                                        exit(0);
                                }
                                fscanf(NEU_INP,"%d,",&I1);
                                fgets(buffer,200,NEU_INP);
                                if(I1 != -1)
                                        {
                                        CONSTRAINT_SET[CONSTRAINT_NUM].A = I1;
                                        fgets(CONSTRAINT_SET[CONSTRAINT_NUM].B, 25, NEU_INP);
                                        CONSTRAINT_SET[CONSTRAINT_NUM].NUM = 0;

        fgetpos(NEU_INP,&(CONSTRAINT_SET[CONSTRAINT_NUM].file_constraint));
                                        fscanf(NEU_INP,"%d,",&CHECK1);

                                label4061:
                                        if(CHECK1 != -1)
                                                {
                                                CONSTRAINT_SET[CONSTRAINT_NUM].NUM=
CONSTRAINT_SET[CONSTRAINT_NUM].NUM + 1;
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                goto label4061;
                                                }
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                label4062:
                                                if(CHECK1 != -1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d",&II);
                                                goto label4062;
                                                }

                                                fgets(buffer,200,NEU_INP);
                                                CONSTRAINT_NUM=CONSTRAINT_NUM+1;
                                        goto label406;

                                        }

//revised in Oct., 1997(Switch with block above)
                                FLAG=1;
                                break;

//**********************    LOAD INFORMATION        ******************************//

                        case 407:
```

```
                                        //printf("Prcoess the load information...\n");
//Switch block with revised Oct block below
/*                                      label407:
                                        fscanf(NEU_INP,"%d",&CHECK);
                                        fgets(buffer,200,NEU_INP);
                                        if(CHECK!=-1)
                                                {
                                                for(i=0;i<20;i++)
                                                        {
                                                        fgets(buffer,200,NEU_INP);
                                                        }
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                while(CHECK1!=-1)
                                                        {
                                                        fgets(buffer,200,NEU_INP);
                                                        fgets(buffer,200,NEU_INP);
                                                        fgets(buffer,200,NEU_INP);
                                                        for(i=0;i<9;i++)
                                                                {
                                                        fgets(buffer,200,NEU_INP);
                                                        fscanf(NEU_INP,"%d",&CHECK1);
                                                                }
                                                        }
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                while(CHECK1!=-1)
                                                        {
                                                        fgets(buffer,200,NEU_INP);
                                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                                        }
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                while(CHECK1!=-1)
                                                        {
                                                        fgets(buffer,200,NEU_INP);
                                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                                        }
                                                fgets(buffer,200,NEU_INP);
                                                goto label407;
                                                }
//Switch block with revised Oct block below
*/
//revised in Oct., 1997(changed above block)

                                        LOAD_NUM = 0;
                                        label407:
                                        fscanf(NEU_INP,"%d",&CHECK);                    // check for end of block
                                        fgets(buffer,200,NEU_INP);
                                        if(CHECK != -1)
                                                {
                                                LOAD_SET[LOAD_NUM].SET_ID = CHECK;
                                                fgets(LOAD_SET[LOAD_NUM].NAME,25,NEU_INP);
                                                //puts(LOAD_SET[LOAD_NUM].NAME);
                                                for(i=0;i<19;i++)
                                                        {
                                                        fgets(buffer,200,NEU_INP);

                                                        }
                                                        fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].load_file));
                                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                                        LOAD_SET[LOAD_NUM].NUM = 0;
                                label4071:
                                                        if(CHECK1 != -1)
                                                        {
                                                        LOAD_SET[LOAD_NUM].NUM =
LOAD_SET[LOAD_NUM].NUM + 1;
                                                        for(I1 = 0; I1 < 12; I1++)
                                                                {
                                                                fgets(buffer,200,NEU_INP);
                                                                }
```

```
                                        fscanf(NEU_INP,"%d",&CHECK1);
                                        goto label4071;
                                        }
                                fgets(buffer,200,NEU_INP);
                                LOAD_SET[LOAD_NUM].NT_NUM = 0;
                                fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].nt_file));
                                fscanf(NEU_INP,"%d,",&CHECK1);
                                //printf("%ld\n",LOAD_SET[LOAD_NUM].NUM);
                                //getchar();
                        label4072:
                                if(CHECK1 != -1)
                                        {
                                        LOAD_SET[LOAD_NUM].NT_NUM =
LOAD_SET[LOAD_NUM].NT_NUM + 1;
                                        fgets(buffer, 200, NEU_INP);
                                        fscanf(NEU_INP, "%d,", &CHECK1);
                                        goto label4072;
                                        }
                                fgets(buffer,200,NEU_INP);
                                LOAD_SET[LOAD_NUM].ET_NUM = 0;
                                fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].et_file));
                                fscanf(NEU_INP,"%d,",&CHECK1);
                                        //printf("%ld\n",LOAD_SET[LOAD_NUM].NT_NUM);
                                //getchar();
                        label4073:
                                if(CHECK1!=-1)
                                        {

        LOAD_SET[LOAD_NUM].ET_NUM=LOAD_SET[LOAD_NUM].ET_NUM+1;
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                        goto label4073;
                                        }
                                fgets(buffer,200,NEU_INP);
                                //              printf("%ld\n",LOAD_SET[LOAD_NUM].ET_NUM);
                                //getchar();
                                LOAD_NUM=LOAD_NUM+1;
                                goto label407;
                                }

//revised in Oct. ,1997 (changed block from above)
                                FLAG=1;
                                break;
                        case 408:
                                //printf("Process the group information...\n");
                                label408:
                                fscanf(NEU_INP,"%d",&CHECK);
                                fgets(buffer,200,NEU_INP);
                                if(CHECK!=-1)
                                        {
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                        while(CHECK1!=-1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                }
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                        while(CHECK1!=-1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
                                                }
                                        fgets(buffer,200,NEU_INP);
                                        fscanf(NEU_INP,"%d,",&CHECK1);
                                        while(CHECK1!=-1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                fscanf(NEU_INP,"%d,",&CHECK1);
```

```
                    }
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d,",&CHECK1);
                while(CHECK1!=-1)
                    {
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d,",&CHECK1);
                    }
                    fgets(buffer,200,NEU_INP);
                goto label408;
                }
        FLAG=1;
        break;
case 409:
        //printf("Process the view information...\n");
        label409:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                for(i=0;i<9;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<8;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II-1;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                        fscanf(NEU_INP,"%d",&CHECK1);
                        fgets(buffer,200,NEU_INP);
                while(CHECK1!=-1)
```

```
                        {
                        fscanf(NEU_INP,"%d",&CHECK1);
                        fgets(buffer,200,NEU_INP);
                        }
                goto label409;
                }
        FLAG=1;
        break;
case 411:
        //printf("Process the Report Format information...\n");
        label411:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II-1;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II-1;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label411;
                }
        FLAG=1;
        break;
case 420:
        //printf("Process the function information...\n");
        label420:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%d",&CHECK1);
                if(CHECK1!=-1)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label420;
                }
        FLAG=1;
        break;
case 412:
        //printf("Process the active data information...\n");
        fgets(buffer,200,NEU_INP);
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        FLAG=1;
        break;
case 450:
        case_i=0;
        label450:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
        {
                case_i=case_i+1;
                fgets(set_name[CHECK-1],30,NEU_INP);
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%lg",TIME+CHECK-1);
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%d",&I1);
                fgets(buffer,200,NEU_INP);
```

```
                                        for(i=0;i<I1;i++)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                }
                                        goto label450;
                                }
                                FLAG=1;
                                break;
                        case 451:
                                output_set_num=0;

                label451:
                                fscanf(NEU_INP,"%d",&CHECK);
                                fgets(buffer,200,NEU_INP);
                                if(CHECK!=-1)
                                        {
                                        case_num[output_set_num]=CHECK;
                                        fgets(out_set_name[output_set_num],30,NEU_INP);
                                        ////////////////////printf("%d %s\n", output_set_num,
out_set_name[output_set_num]);
                                        fscanf(NEU_INP,"%lg,%lg,%lg",MIN_VALUE+output_set_num,

        MAX_VALUE+output_set_num,AMAX_VALUE+output_set_num);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);

        fscanf(NEU_INP,"%ld,%ld,%d,%d",&CHECKD,&NODE_i,&I1,TYPE+output_set_num);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgetpos(NEU_INP,file_output+output_set_num);
                                        output_set_num=output_set_num+1;
                                        ELEMENT_i=0;
                                        label4511:
                                        fscanf(NEU_INP,"%ld,",&CHECKD);
                                        if(CHECKD!=-1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                        ELEMENT_i=ELEMENT_i+1;
                                                goto label4511;
                                                }
                                        fgets(buffer,200,NEU_INP);
                                        Total_num[output_set_num-1]=ELEMENT_i;
                                        goto label451;
                                        }
                                FLAG=-1;
                                break;
                        }

                }
        }
//allocation dynamic memory for node data
  NODE_P =(struct NODE_DATA *) malloc(NODE_NUM*(sizeof(NODE_DATA)+1));
  if( NODE_P == NULL )
                {
                printf("Insufficient memory available for node data\n" );
                getchar();
                }
  else{
                //printf("Memory allocation success\n");
        printf("\n");
                }
 //allocation dynamic memory for element data

 //
 ELEMENT_TMP =(struct ELEMENT_REL *) calloc(sizeof(ELEMENT_REL),ELEMENT_NUM1);
  if( ELEMENT_TMP == NULL )
                {
                printf("Insufficient memory available for element relation data\n" );
                getchar();
```

```
                    }
 else{
                    //printf("Memory allocation success\n");
            printf("\n");
                    }

ELEMENT_P =(struct ELEMENT_DATA *) calloc(sizeof(ELEMENT_DATA),ELEMENT_NUM);
 if( ELEMENT_P == NULL )
                    {
                    printf("Insufficient memory available for element data\n" );
                    getchar();
                    }
 else{
                    //printf("Memory allocation success\n");
            printf("\n");
                    }

//revised in Oct., 1997

COORDINATE_P=(struct COORDINATE *)calloc(sizeof(COORDINATE),COORDINATE_NUM);
if(COORDINATE_P==NULL)
          {
          printf("Insufficient memory for coordinate system data\n");
          getchar();
          }

fsetpos(NEU_INP,&file_coordinate);
for(I2=0;I2<COORDINATE_NUM;I2++)
{
fscanf(NEU_INP,"%d",&I1);
          (COORDINATE_P+I2)->A=I1;
          fscanf(NEU_INP,",%d,%d",&((COORDINATE_P+I2)->B),&((COORDINATE_P+I2)->C));
          fgets(buffer,200,NEU_INP);
          fgets((COORDINATE_P+I2)->D,25,NEU_INP);
//        printf("%d\t%s",I2,(COORDINATE_P+I2)->D);
          for(I3=0;I3<3;I3++)
          {
          fscanf(NEU_INP,"%lg,",&((COORDINATE_P+I2)->E[I3]));
          }
          fgets(buffer,200,NEU_INP);
          for(I3=0;I3<3;I3++)
          {
          fscanf(NEU_INP,"%lg,",&((COORDINATE_P+I2)->F[I3]));
          }
          }

if(CONSTRAINT_NUM > 0)
{
printf("Would you like to load the constraint information into memory?\n\n");
FLAG = 1;
while(FLAG == 1)
          {
          printf("\n(Y) Yes; (N) No;\n");
          //gets(OUTPUT);

          scanf("%s",OUTPUT);
  //strcpy(OUTPUT,"Y"); //BEC+

          if((!strcmp(OUTPUT, "Y")) || (!strcmp(OUTPUT, "y")))
          {
                    CONSTRAINT_YES = 1;

for(I2 = 0; I2 < CONSTRAINT_NUM; I2++)
          {
          CONSTRAINT_SET[I2].ID=(long int *)calloc(sizeof(long int),CONSTRAINT_SET[I2].NUM);
          CONSTRAINT_SET[I2].INDEX=(int *)calloc(sizeof(int),(CONSTRAINT_SET[I2].NUM)*6);
          if((CONSTRAINT_SET[I2].ID==NULL)||(CONSTRAINT_SET[I2].INDEX==NULL))
                    {
                    printf("Insufficient memory for data in constraint set #%d\n", I2+1);
```

```
                getchar();
                }

//*************************************************************************//

        fsetpos(NEU_INP,&(CONSTRAINT_SET[I2].file_constraint));

        for(I3 = 0; I3 < CONSTRAINT_SET[I2].NUM; I3++)
        {
                fscanf(NEU_INP,"%ld,%d,%d,%d,%d,%d,%d,%d,%d",(CONSTRAINT_SET[I2].ID)+I3,&I4,&I1,
                CONSTRAINT_SET[I2].INDEX+I3*6,CONSTRAINT_SET[I2].INDEX+(I3*6+1),
                CONSTRAINT_SET[I2].INDEX+(I3*6+2),CONSTRAINT_SET[I2].INDEX+(I3*6+3),
                CONSTRAINT_SET[I2].INDEX+(I3*6+4),CONSTRAINT_SET[I2].INDEX+(I3*6+5));
                fgets(buffer,200,NEU_INP);
        }
        }

        FLAG=0;
        for(i = 0; i < CONSTRAINT_NUM; i++)
                        strcpy(constraintset_names[i], CONSTRAINT_SET[i].B);

        //Ola  constraintset_name = constraintset_prompt(constraintset_names, CONSTRAINT_NUM);  // Function that calls load
set name prompt
        strncpy(constraintset_name, "First constraint", 16);
//printf("%s why oh why\n", constraintset_name);
        for(i = 0; i < CONSTRAINT_NUM; i++)                                     // loop to find
CONSTRAINTSET_PICK
                if(strcmp(constraintset_name, constraintset_names[i]) == 0)
                        CONSTRAINTSET_PICK = i + 1;

        CONSTRAINTSET_NUM = CONSTRAINT_SET[CONSTRAINTSET_PICK].NUM;
        }
        else if((!strcmp(OUTPUT,"N"))||(!strcmp(OUTPUT,"n")))
        {
                FLAG = 0;
        }
}

}

//*************************************************************************//

MATERIAL_P=(struct MATERIAL *)calloc(sizeof(MATERIAL),MATERIAL_NUM);
if(MATERIAL_P==NULL)
        {
        printf("Insufficient memory for material data\n");
        getchar();
        }
fsetpos(NEU_INP,&file_mat);
for (I2=0;I2<MATERIAL_NUM;I2++)
        {
        fscanf(NEU_INP,"%d", &I1);
        fgets(buffer,200,NEU_INP);
                        {
                (MATERIAL_P+I2)->A=I1;
                fgets((MATERIAL_P+I2)->title,25,NEU_INP);
//              puts((MATERIAL_P+I2)->title);
                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Young_Modulus[I3]));
                        }
                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Shear_Modulus[I3]));
                        }
                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Poisson_Ratio[I3]));
                        }
                for(I3=0;I3<21;I3++)
```

```
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->GMatrix[I3]));
                        }
                for(I3=0;I3<6;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->alpha[I3]));
                        }
                for(I3=0;I3<6;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->k[I3]));
                        }
                        fscanf(NEU_INP,"%lg,%lg,%lg,%lg,",&((MATERIAL_P+I2)->thermal_cap),
                        &((MATERIAL_P+I2)->density),&((MATERIAL_P+I2)->damping),&((MATERIAL_P+I2)-
>temperature));
                        fscanf(NEU_INP,"%lg,%lg,%lg,%lg,%lg", &((MATERIAL_P+I2)->tension_limit[0]),
                        &((MATERIAL_P+I2)->tension_limit[1]),&((MATERIAL_P+I2)->comp_limit[0]),
                        &((MATERIAL_P+I2)->comp_limit[1]),&((MATERIAL_P+I2)->shear_limit));
                for(I3=0;I3<17;I3++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                }
}
//
if(LOAD_NUM > 0)
{
        printf("Would you like to load the load information into memory?\n\n");
FLAG=1;
while(FLAG==1)
        {
        printf("\n(Y) Yes; (N) No;\n");
        //gets(OUTPUT);

        scanf("%s",OUTPUT);
  //strcpy(OUTPUT,"Y"); //BEC+

        if((!strcmp(OUTPUT,"Y"))||(!strcmp(OUTPUT,"y")))
        {
                LOAD_YES = 1;
for(I1 = 0; I1 < LOAD_NUM; I1++)
        {
//      printf("I1=%d\t%ld\t%ld\t%ld\n",I1,LOAD_SET[I1].NUM,LOAD_SET[I1].NT_NUM,LOAD_SET[I1].ET_NUM);
//getchar();
        if(LOAD_SET[I1].NUM != 0)
        {
        LOAD_SET[I1].ID = (long int *)calloc(sizeof(long int),LOAD_SET[I1].NUM);
        if(LOAD_SET[I1].ID == NULL)
        {
        printf("Insufficient memory for data in load set #\n",I1+1);
        getchar();
        }
        LOAD_SET[I1].TYPE=(int *)calloc(sizeof(int),LOAD_SET[I1].NUM);
        if(LOAD_SET[I1].TYPE==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }
        LOAD_SET[I1].FACE=(int *)calloc(sizeof(int),(LOAD_SET[I1].NUM)*6);
        if(LOAD_SET[I1].FACE==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }
        LOAD_SET[I1].VALUE=(double *)calloc(sizeof(double),(LOAD_SET[I1].NUM)*8);
        if(LOAD_SET[I1].VALUE==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }
```

```
//*****************************************************************************//
//*****************************************************************************//

        fsetpos(NEU_INP,&(LOAD_SET[I1].load_file));

        for(CHECKD = 0; CHECKD < LOAD_SET[I1].NUM; CHECKD++)
        {
                fscanf(NEU_INP,"%ld,%d",LOAD_SET[I1].ID + CHECKD, LOAD_SET[I1].TYPE + CHECKD);
                //printf("%ld,%d\n",LOAD_SET[I1].ID[CHECKD],LOAD_SET[I1].TYPE[CHECKD]);
                //getchar();
                fgets(buffer,200,NEU_INP);

        fscanf(NEU_INP,"%lg,%lg",LOAD_SET[I1].VALUE+(CHECKD*8),LOAD_SET[I1].VALUE+(CHECKD*8+1));
                //printf("%lg,%lg\n",LOAD_SET[I1].VALUE[CHECKD*8],LOAD_SET[I1].VALUE[CHECKD*8+1]);
                //getchar();
                //printf("%ld\n",CHECKD);
                //getchar();
                fgets(buffer,200,NEU_INP);
                //puts(buffer);
                for (CHECK1 = 0; CHECK1 < 6; CHECK1++)
                {
        //                      printf("%ld\n",CHECK1);


        fscanf(NEU_INP,"%d,%lg",LOAD_SET[I1].FACE+(CHECKD*6+CHECK1),LOAD_SET[I1].VALUE+(CHECKD*8+2
+CHECK1));
        //
        printf("%d,%lg\n",LOAD_SET[I1].FACE[CHECKD*6+CHECK1],LOAD_SET[I1].VALUE[CHECKD*8+2+CHECK1])
;
        //      getchar();
                        fgets(buffer,200,NEU_INP);
                }
                for(I4 = 0;I4 < 4; I4++)
                {
                        fgets(buffer,200,NEU_INP);
                }
        //      getchar();
        }
        //LOADSET_PICK=0;
        //LOADSET_NUM=LOAD_SET[LOADSET_PICK].NUM;
        }

        if(LOAD_SET[I1].NT_NUM != 0)
        {
        LOAD_SET[I1].NT_ID=(long int *)calloc(sizeof(long int),LOAD_SET[I1].NT_NUM);
        if(LOAD_SET[I1].NT_ID==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }
        LOAD_SET[I1].NT_VALUE=(double *)calloc(sizeof(double),LOAD_SET[I1].NT_NUM);
        if(LOAD_SET[I1].NT_VALUE==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }

//*****************************************************************************//

        fsetpos(NEU_INP,&(LOAD_SET[I1].nt_file));
        for(CHECKD = 0; CHECKD < LOAD_SET[I1].NT_NUM; CHECKD++)
        {
        fscanf(NEU_INP,"%ld,%d,%d,%lg",LOAD_SET[I1].NT_ID+CHECKD,&I3,&I4,LOAD_SET[I1].NT_VALUE+CHECK
D);
        //printf("NT=%ld\t%lg",LOAD_SET[I1].NT_ID[CHECKD],LOAD_SET[I1].NT_VALUE[CHECKD]);
        //getchar();
        fgets(buffer,200,NEU_INP);
        }
}
//      printf("&&&&\n");
```

```
        if(LOAD_SET[I1].ET_NUM != 0)
        {
        LOAD_SET[I1].ET_ID=(long int *)calloc(sizeof(long int),LOAD_SET[I1].ET_NUM);
        if(LOAD_SET[I1].ET_ID==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }
        LOAD_SET[I1].ET_VALUE=(double *)calloc(sizeof(double),LOAD_SET[I1].ET_NUM);
        if(LOAD_SET[I1].ET_VALUE==NULL)
        {
        printf("Insufficient memory for data in  load set #\n",I1+1);
        getchar();
        }

//*************************************************************************//

        fsetpos(NEU_INP,&(LOAD_SET[I1].et_file));

        for(CHECKD = 0;CHECKD < LOAD_SET[I1].ET_NUM; CHECKD++)
        {
        fscanf(NEU_INP,"%ld,%d,%d,%lg",LOAD_SET[I1].ET_ID+CHECKD,&I3,&I4,LOAD_SET[I1].ET_VALUE+CHECK
D);
        //printf("ET=%ld\t%lg",LOAD_SET[I1].ET_ID[CHECKD],LOAD_SET[I1].ET_VALUE[CHECKD]);
        //getchar();
        fgets(buffer,200,NEU_INP);
        }
        }
        }
        FLAG=0;
        for(i = 0; i < LOAD_NUM; i++)
                        strcpy(loadset_names[i], LOAD_SET[i].NAME);

//ola     loadset_name = loadset_prompt(loadset_names, LOAD_NUM);  // Function that calls load set name prompt
strcpy(loadset_name, "First Load");
        for(i = 0; i < LOAD_NUM; i++)                                   // loop to find LOADSET_PICK
                if(strcmp(loadset_name, loadset_names[i]) == 0)
                        LOADSET_PICK = i + 1;

        LOADSET_NUM = LOAD_SET[LOADSET_PICK].NUM;
        }
        else if(strncmp(bool, "NO", 2) == 0)
        {
                FLAG=0;
        }
}

}


//*************************************************************************//
//*************************************************************************//

//revised in Oct., 1997
ELEMENT_PROPERTY_P=(struct ELEMENT_PROPERTY *) calloc (sizeof(ELEMENT_PROPERTY),ELEMENT_PRO_NUM);
if(ELEMENT_PROPERTY_P==NULL)
        {
        printf("Insufficient memory for element property data\n");
        getchar();
        }
else{
        printf("\n");
        }
fsetpos(NEU_INP,&file_pro);

for (i=0;i<ELEMENT_PRO_NUM;i++)
        {
        fscanf(NEU_INP,"%d,%d,%d,%d",&CHECKD,&I1,&I2,&I3);
        (ELEMENT_PROPERTY_P+i)->A=I3;
        fgets(buffer,200,NEU_INP);
```

```
            fgets(buffer,200,NEU_INP);
            fgets(buffer,200,NEU_INP);
            fscanf(NEU_INP,"%d",&II);
            fgets(buffer,200,NEU_INP);
            for(I1=0;I1<(float)(II/8)+1.0;I1++)
                        {fgets(buffer,200,NEU_INP);
                        }
            fscanf(NEU_INP,"%d",&II);
            fgets(buffer,200,NEU_INP);
            for (I3=0;I3<II;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((ELEMENT_PROPERTY_P+i)->B[I3]));
                        }
                        fgets(buffer,200,NEU_INP);
            }

  //revised on Sept 30, 1997

  fsetpos(NEU_INP,&file_node);

names =(struct NAMES *) malloc(sizeof(NAMES));//Dryer added 9/20/97


 for (NODE_i=0;NODE_i<NODE_NUM;NODE_i++)
            {
             fscanf(NEU_INP,"%ld,",&CHECKD);
             (NODE_P+NODE_i)->A=CHECKD;
            for(i=0;i<10;i++)
                        {
                        fscanf(NEU_INP,"%d,",&I1);
                        }
            fscanf(NEU_INP,"%lg,%lg,%lg",&X,&Y,&Z);
            (NODE_P+NODE_i)->x=X;
            (NODE_P+NODE_i)->y=Y;
            (NODE_P+NODE_i)->z=Z;
            (NODE_P+NODE_i)->dx=0.0;
            (NODE_P+NODE_i)->dy=0.0;
            (NODE_P+NODE_i)->dz=0.0;

            fgets(buffer,200,NEU_INP);
            for(i=0;i<5;i++)
                        {
                        (NODE_P+NODE_i)->output_data[i]=0.0;
                        }
            }
///
            //processing the output data
 fsetpos(NEU_INP,&file_element);
 //
            for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
                        {
                        fscanf(NEU_INP,"%ld",&CHECKD);
                        (ELEMENT_TMP+ELEMENT_i)->A=CHECKD;
                        for(i=0;i<7;i++)
                                    {
                                    fgets(buffer,200,NEU_INP);
                                    }
                        }

 //*********************************clear screen*******************************//

            for (i=0;i<23;i++)
            {
                        printf("\n");
            }
            //revised on Sept. 30, 1997

 //*****************************************************************************//

 if(case_i!=0)
```

```
{
        //revised on Setp. 30, 1997
        FLAG=0;
while(FLAG==0)
        {
        i=0;
        I1=0;
while(i<case_i)
        {
        if(I1==0)
                {
                printf("Please enter a case number:\n\n");
                }
        printf("(%d) %s",i+1,set_name[i]);

        while((I1==19)||(i==case_i-1))
                {
                I1=0;
                printf("Please select: (N) next; (P) previous; (S) selection\n\n");

                scanf("%s",OUTPUT);
   //strcpy(OUTPUT,"S"); //BEC+

                if((!strncmp(OUTPUT,"S",1))||(!strncmp(OUTPUT,"s",1)))
                        {
                        printf("Please enter the number of case\n");

                        scanf("%d",&I2);
     //I2 = 1; //BEC+

                        getchar();
                        case_n=I2;
                        strncpy(names->actual_case_name,set_name[case_n-1],strlen(set_name[case_n-1]));
                        goto labelnnn;
                        }
                else if((!strncmp(OUTPUT,"N",1))||(!strncmp(OUTPUT,"n",1)))
                        {
                        FLAG=0;
                        break;
                        }
                else if((!strncmp(OUTPUT,"P",1))||(!strncmp(OUTPUT,"p",1)))
                        {
                        i=i-38;
                        if(i<0)
                                {
                                i=-1;
                                }
                        break;
                        }
                }
                i=i+1;
        I1=I1+1;
        }
}
labelnnn:
case_nn = 0;
for(i = 0; i < output_set_num; i++)
        {
        if(case_num[i] == case_n)
                {
                strcpy(temp_set_name[case_nn], out_set_name[i]);
                case_set_num[case_nn] = i;
                case_nn = case_nn + 1;
                }
        }

//*********************************clear screen*******************************//

        for (i=0;i<23;i++)
        {
```

```
                printf("\n");
        }
        //revised on Sept. 30, 1997

//*************************************************************************//

for(i=0;i<case_nn;i++)
        {
        if((!strncmp(temp_set_name[i],"T1 Translation",14))||(!strncmp(temp_set_name[i],"X Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        ELEMENT_i=FindNid(CHECKD);
//                      ELEMENT_i=FindNid(CHECKD,NODE_P,NODE_NUM);
                        fgets(buffer,200,NEU_INP);
                        (NODE_P+ELEMENT_i)->dx=X;
                        }
                }
        if((!strncmp(temp_set_name[i],"T2 Translation",14))||(!strncmp(temp_set_name[i],"Y Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        fgets(buffer,200,NEU_INP);
                        ELEMENT_i=FindNid(CHECKD);
//                      ELEMENT_i=FindNid(CHECKD,NODE_P,NODE_NUM);
                        (NODE_P+ELEMENT_i)->dy=X;
                        }
                }
        if((!strncmp(temp_set_name[i],"T3 Translation",14))||(!strncmp(temp_set_name[i],"Z Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        fgets(buffer,200,NEU_INP);
                        ELEMENT_i=FindNid(CHECKD);
//                      ELEMENT_i=FindNid(CHECKD,NODE_P,NODE_NUM);
                        (NODE_P+ELEMENT_i)->dz=X;
                        }
                }
        }

//*************************************************************************//
/*
        for(i = 0; i < case_nn; i++){                                   // include whether output is nodal or
elemental
                if(TYPE[case_set_num[i]] == 7)
                        sprintf(temp_name[i], "(Nodal) %s", temp_set_name[i]);
                if(TYPE[case_set_num[i]] == 8)
                        sprintf(temp_name[i], "(Elemental) %s", temp_set_name[i]);
        }

        //output_data = output_data_prompt(temp_name, case_nn);         // Function that calls output set name prompt
        strcpy(output_data, "
        j = 0;
        while(output_data[j + 1] != NULL && j < 10){
                for(i = 0; i < case_nn; i++)        // loop to find case_
                {
                        if(strncmp(strchr(output_data[j], ' ') + 1, temp_set_name[i], strlen(strchr(output_data[j], ' ') + 1)) ==
0)
                        {
                                if(strncmp(output_data[j], "(Elemental)", 11) == 0 && k < 5){
                                        V[k++] = i;
                                        V_NUM = V_NUM - 1;
                                }
                                else if(strncmp(output_data[j], "(Nodal)", 7) == 0 && m < 5){
```

```
                                        U[m++] = i;
                                        U_NUM = U_NUM - 1;
                                }
                        }
                }
                j = j + 1;
        }

        goto labelxxx;

        */
V_NUM=5;
U_NUM=5;
I1=0;
labelyyy:
        i=0;
while(i<case_nn)
        {
        if(I1==0)
                {
                printf("\nPlease select %d sets of elemental output data\n",U_NUM);
                printf("and %d sets of nodal output data for visualization\n\n",V_NUM);
                }
        if(TYPE[case_set_num[i]]==7)
                {
                printf("(Nodal) #(%d):  %s",i+1,temp_set_name[i]);
                }
        if(TYPE[case_set_num[i]]==8)
                {
                printf("(Elemental) #(%d):  %s",i+1,temp_set_name[i]);
                }

        if((I1==18)||(i==case_nn-1))
                {
                I1=0;
                FLAG=1;
                while(FLAG==1)
                        {
                        if((U_NUM==0)&&(V_NUM==0))
                                        {
                                        goto labelxxx;
                                        }
                        printf("\n(N) next; (P) previous; (S) select; (D) done;\n");

                        gets(OUTPUT);
    //strcpy(OUTPUT,"S"); //BEC+

                        if((!strcmp(OUTPUT,"N"))||(!strcmp(OUTPUT,"n")))
                                {
                                if(i==case_nn-1)
                                {i=i-18;}
                                if(i<0)
                                        {
                                        i=0;
                                        }
                                I1=0;
                                FLAG=0;
                                break;
                                }
                        else if((!strcmp(OUTPUT,"P"))||(!strcmp(OUTPUT,"p")))
                                {
                                i=i-36;
                                if(i<0)
                                        {
                                        i=0;
                                        }
                                FLAG=0;
                                break;
                                }
                        else if((!strcmp(OUTPUT,"S"))||(!strcmp(OUTPUT,"s")))
```

```
                                        {
                                        printf("Please enter the set #?\n");

                                        scanf("%d",&II);
        //II = 1; //BEC+

                                        getchar();
                                        //revised in Oct., 1997
                                        if(TYPE[case_set_num[II-1]]==7)
                                                {
                                                if(U_NUM-1<0){goto labelxxx;}
                                                U[5-U_NUM]=II-1;
                                                U_NUM=U_NUM-1;
                                                }
                                        if(TYPE[case_set_num[II-1]]==8)
                                                {
                                                if(V_NUM<1){goto labelxxx;}
                                                V[5-V_NUM]=II-1;
                                                V_NUM=V_NUM-1;
                                                }
                                        //revised in Oct., 1997

                                        }
                                else if((!strcmp(OUTPUT,"D"))||(!strcmp(OUTPUT,"d")))
                                        {
                                        goto labelxxx;
                                        }
                                else
                                        {

                                        i=i-18;
                                        if(i<0)
                                                {
                                                i=0;
                                                I1=0;
                                                }
                                        FLAG=0;
                                        break;
                                        }
                                }
                        continue;
                        }
                I1=I1+1;
                i=i+1;
                if(i==case_nn)
                        {
                        I1=0;
                        i=0;
                        goto labelyyy;
                        }
                }
//*****************************************************************************//

labelxxx:
                //printf("****\n");
                if(V_NUM != 5)
                        {
                        //printf("Processing the elemental output data\n");
                        for(i = 0; i < 5 - V_NUM; i++)
                                {//revised in Oct., 1997
//                              printf("i=%d\n",i);
                                //printf("set_num=%d\n",Total_num[case_set_num[V[i]]]);
                                        fsetpos(NEU_INP,file_output+case_set_num[V[i]]);
                                for(ELEMENT_i=0;ELEMENT_i<Total_num[case_set_num[V[i]]];ELEMENT_i++)
                                        {
                                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
//                                      printf("CKD=%ld\n",CHECKD);
                                        (ELEMENT_TMP+FindEid(CHECKD))->data[i]=X;
                                        fgets(buffer,200,NEU_INP);
                                        }
```

```
                        //revised in Oct., 1997
                        }
                }
        else
                {
                for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
                        {
                        for(I1=0;I1<5;I1++)
                        {
                        (ELEMENT_TMP+ELEMENT_i)->data[I1]=0.0;
                        }
                        }
                }
        if(U_NUM != 5)
        {if(U_NUM < 0){U_NUM = 0;}
        //      printf("Processing the nodal output data\n");
        for(i = 0; i < 5 - U_NUM; i++)
                {
                //revised in Oct., 1997
                fsetpos(NEU_INP,file_output+case_set_num[U[i]]);
                for(NODE_i=0;NODE_i<Total_num[case_set_num[U[i]]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);

                        (NODE_P+FindNid(CHECKD))->output_data[i]=X;
                        fgets(buffer,200,NEU_INP);
                        //revised in Oct., 1997

                        }
                }
        }
//revised on Sept. 30 1997
}

//revised on Sept. 30, 1997
 fsetpos(NEU_INP, &file_element);
 NODE_i=0;
 for (ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
        {
         fscanf(NEU_INP,"%ld,%d,%d,%d,%d,",&CHECKDD,&I1,&I2,&I3,&I4);
         fgets(buffer,200,NEU_INP);
        switch(I4)
                        {
                        case 0:
                        //ELEMENT_NUM=ELEMENT_NUM+1;
                                for(i=0;i<20;i++)
                                        {
                                        fscanf(NEU_INP,"%ld,",&CHECKD);
                                        ID[i]=FindNid(CHECKD);
//                                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                                        }
                                (ELEMENT_P+NODE_i)->A=2;
                                (ELEMENT_P+NODE_i)->D=CHECKDD;
                                //for(I1=0;I1<4;I1++)
                                //      {
                                        (ELEMENT_P+NODE_i)->B[0]=ID[0];
                                        (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[2]=-1;
                                        (ELEMENT_P+NODE_i)->B[3]=-1;
                                //revised Sept. 30, 1997
                                        (ELEMENT_P+NODE_i)->F=I3;
                                //revised Sept. 30, 1997
                                //      }
                                for(i=0;i<5;i++)
                                                {
                                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                }
                                NODE_i=NODE_i+1;
                                fgets(buffer,200,NEU_INP);
```

```
                    break;
                case 2:
                    for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                    (ELEMENT_P+NODE_i)->A=3;
                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                    for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                        }
                //revised Sept. 30, 1997
                    (ELEMENT_P+NODE_i)->F=I3;
                //revised Sept. 30, 1997
                    for(i=0;i<5;i++)
                                {
                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                }
                    NODE_i=NODE_i+1;
                    /*(ELEMENT_P+NODE_i)->A=3;
                    for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[3-I1];
                        }
                    for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                    NODE_i=NODE_i+1;*/
                    fgets(buffer,200,NEU_INP);
                    break;
                case 3:
                    for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                        (ELEMENT_P+NODE_i)->A=3;
                        (ELEMENT_P+NODE_i)->D=CHECKDD;

                    for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                        }
                //revised Sept. 30, 1997
                    (ELEMENT_P+NODE_i)->F=I3;
                //revised Sept. 30, 1997
                    for(i=0;i<5;i++)
                                {
                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                }
                    NODE_i=NODE_i+1;
                    /*(ELEMENT_P+NODE_i)->A=3;
                    for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[3-I1];
                        }
                    for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                    NODE_i=NODE_i+1;*/
                    fgets(buffer,200,NEU_INP);
```

```
                break;
        case 4:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                        }
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                /*(ELEMENT_P+NODE_i)->A=4;
                for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[3-I1];
                        }
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;*/
                fgets(buffer,200,NEU_INP);
                break;
        case 5:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                        (ELEMENT_P+NODE_i)->A=4;
                                (ELEMENT_P+NODE_i)->D=CHECKDD;

                for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                        }
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                /*(ELEMENT_P+NODE_i)->A=4;
                for(I1=0;I1<4;I1++)
                        {
                        (ELEMENT_P+NODE_i)->B[I1]=ID[3-I1];
                        }
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;*/
                fgets(buffer,200,NEU_INP);
                break;
        case 6:
```

```
                    for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                    fgets(buffer,200,NEU_INP);
                    (ELEMENT_P+NODE_i)->A=3;
                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                    (ELEMENT_P+NODE_i)->B[0]=ID[0];
                    (ELEMENT_P+NODE_i)->B[1]=ID[1];
                    (ELEMENT_P+NODE_i)->B[2]=ID[2];
                    (ELEMENT_P+NODE_i)->B[3]=ID[3];
//revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                    //revised Oct. 22, 1997
                //revised Sept. 30, 1997
                    (ELEMENT_P+NODE_i)->F=I3;
                //revised Sept. 30, 1997
                    for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                    NODE_i=NODE_i+1;
                    (ELEMENT_P+NODE_i)->A=3;
                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                    (ELEMENT_P+NODE_i)->B[0]=ID[0];
                    (ELEMENT_P+NODE_i)->B[1]=ID[1];
                    (ELEMENT_P+NODE_i)->B[2]=ID[4];
                    (ELEMENT_P+NODE_i)->B[3]=ID[3];
                    //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                    //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                    (ELEMENT_P+NODE_i)->F=I3;
                //revised Sept. 30, 1997
                    for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                    NODE_i=NODE_i+1;
                    (ELEMENT_P+NODE_i)->A=3;
                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                    (ELEMENT_P+NODE_i)->B[0]=ID[0];
                    (ELEMENT_P+NODE_i)->B[1]=ID[2];
                    (ELEMENT_P+NODE_i)->B[2]=ID[4];
                    (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                    //revised Oct. 22, 1997
                //revised Sept. 30, 1997
                    (ELEMENT_P+NODE_i)->F=I3;
                //revised Sept. 30, 1997
                    for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                    NODE_i=NODE_i+1;
                    (ELEMENT_P+NODE_i)->A=3;
                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                    (ELEMENT_P+NODE_i)->B[0]=ID[1];
                    (ELEMENT_P+NODE_i)->B[1]=ID[2];
                    (ELEMENT_P+NODE_i)->B[2]=ID[4];
                    (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
```

```
                //revised Oct. 22, 1997
//revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                /*(ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->B[0]=ID[2];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[0];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[0];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[0];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;*/
                break;
case 7:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                fgets(buffer,200,NEU_INP);
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
//revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
```

```
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[5];
        (ELEMENT_P+NODE_i)->B[2]=ID[6];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[5];
        (ELEMENT_P+NODE_i)->B[3]=ID[4];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[1];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[5];
        (ELEMENT_P+NODE_i)->B[3]=ID[4];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[2];
        (ELEMENT_P+NODE_i)->B[1]=ID[0];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[6];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
```

```
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=l3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        /*(ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->B[0]=ID[2];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[0];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->B[0]=ID[6];
        (ELEMENT_P+NODE_i)->B[1]=ID[5];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[5];
        (ELEMENT_P+NODE_i)->B[2]=ID[1];
        (ELEMENT_P+NODE_i)->B[3]=ID[0];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[5];
        (ELEMENT_P+NODE_i)->B[2]=ID[2];
        (ELEMENT_P+NODE_i)->B[3]=ID[1];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->B[0]=ID[6];
        (ELEMENT_P+NODE_i)->B[1]=ID[4];
        (ELEMENT_P+NODE_i)->B[2]=ID[0];
        (ELEMENT_P+NODE_i)->B[3]=ID[2];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;*/
        break;
case 8:
        for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
//              ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                }
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
```

```
            (ELEMENT_P+NODE_i)->B[1]=ID[3];
            (ELEMENT_P+NODE_i)->B[2]=ID[2];
            (ELEMENT_P+NODE_i)->B[3]=ID[1];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[4];
            (ELEMENT_P+NODE_i)->B[1]=ID[5];
            (ELEMENT_P+NODE_i)->B[2]=ID[6];
            (ELEMENT_P+NODE_i)->B[3]=ID[7];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[0];
            (ELEMENT_P+NODE_i)->B[1]=ID[1];
            (ELEMENT_P+NODE_i)->B[2]=ID[5];
            (ELEMENT_P+NODE_i)->B[3]=ID[4];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[1];
            (ELEMENT_P+NODE_i)->B[1]=ID[2];
            (ELEMENT_P+NODE_i)->B[2]=ID[6];
            (ELEMENT_P+NODE_i)->B[3]=ID[5];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;
```

```
                (ELEMENT_P+NODE_i)->B[0]=ID[2];
                (ELEMENT_P+NODE_i)->B[1]=ID[3];
                (ELEMENT_P+NODE_i)->B[2]=ID[7];
                (ELEMENT_P+NODE_i)->B[3]=ID[6];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[3];
                (ELEMENT_P+NODE_i)->B[1]=ID[0];
                (ELEMENT_P+NODE_i)->B[2]=ID[4];
                (ELEMENT_P+NODE_i)->B[3]=ID[7];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
/*              (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[3];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                (ELEMENT_P+NODE_i)->B[3]=ID[0];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[7];
                (ELEMENT_P+NODE_i)->B[1]=ID[6];
                (ELEMENT_P+NODE_i)->B[2]=ID[5];
                (ELEMENT_P+NODE_i)->B[3]=ID[4];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[5];
                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                (ELEMENT_P+NODE_i)->B[3]=ID[0];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[5];
                (ELEMENT_P+NODE_i)->B[1]=ID[6];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[1];
                for(i=0;i<5;i++)
```

```
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[6];
                (ELEMENT_P+NODE_i)->B[1]=ID[7];
                (ELEMENT_P+NODE_i)->B[2]=ID[3];
                (ELEMENT_P+NODE_i)->B[3]=ID[2];
                for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[7];
                (ELEMENT_P+NODE_i)->B[1]=ID[4];
                (ELEMENT_P+NODE_i)->B[2]=ID[0];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
                NODE_i=NODE_i+1;*/
                fgets(buffer,200,NEU_INP);
                break;
        case 9:
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                break;
        case 10:
                for(i=0;i<20;i++)
                    {
                    fscanf(NEU_INP,"%ld,",&CHECKD);
                    ID[i]=FindNid(CHECKD);
//                  ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                    }
                fgets(buffer,200,NEU_INP);
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                    //revised on Oct. 22,1997
                    (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[4];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                    //revised on Oct. 22,1997
                    (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
```

```
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[1];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                //revised on Oct. 22,1997
                (ELEMENT_P+NODE_i)->E=1000;
        //revised Oct. 22, 1997
//revised Sept. 30, 1997
        (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        /*(ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->B[0]=ID[2];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[0];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[0];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[0];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
```

```
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;*/
                break;
        case 11:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                fgets(buffer,200,NEU_INP);
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[3];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                (ELEMENT_P+NODE_i)->B[3]=ID[0];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[5];
                (ELEMENT_P+NODE_i)->B[2]=ID[6];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[5];
                (ELEMENT_P+NODE_i)->B[3]=ID[4];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
```

```
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[1];
            (ELEMENT_P+NODE_i)->B[1]=ID[2];
            (ELEMENT_P+NODE_i)->B[2]=ID[6];
            (ELEMENT_P+NODE_i)->B[3]=ID[5];
                    //revised on Oct. 22,1997
                    (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[2];
            (ELEMENT_P+NODE_i)->B[1]=ID[0];
            (ELEMENT_P+NODE_i)->B[2]=ID[4];
            (ELEMENT_P+NODE_i)->B[3]=ID[6];
                    //revised on Oct. 22,1997
                    (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            /*(ELEMENT_P+NODE_i)->A=3;
            (ELEMENT_P+NODE_i)->B[0]=ID[2];
            (ELEMENT_P+NODE_i)->B[1]=ID[1];
            (ELEMENT_P+NODE_i)->B[2]=ID[0];
            (ELEMENT_P+NODE_i)->B[3]=ID[3];
            for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=3;
            (ELEMENT_P+NODE_i)->B[0]=ID[6];
            (ELEMENT_P+NODE_i)->B[1]=ID[5];
            (ELEMENT_P+NODE_i)->B[2]=ID[4];
            (ELEMENT_P+NODE_i)->B[3]=ID[3];
            for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->B[0]=ID[4];
            (ELEMENT_P+NODE_i)->B[1]=ID[5];
            (ELEMENT_P+NODE_i)->B[2]=ID[1];
            (ELEMENT_P+NODE_i)->B[3]=ID[0];
            for(i=0;i<5;i++)
                    {
                    (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                    }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
```

```
                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[5];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[1];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->B[0]=ID[6];
                (ELEMENT_P+NODE_i)->B[1]=ID[4];
                (ELEMENT_P+NODE_i)->B[2]=ID[0];
                (ELEMENT_P+NODE_i)->B[3]=ID[2];
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;*/
                break;
        case 12:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
//                      ID[i]=FindNid(CHECKD,NODE_P,NODE_NUM);
                        }
                fgets(buffer,200,NEU_INP);
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[4];
                (ELEMENT_P+NODE_i)->B[1]=ID[5];
                (ELEMENT_P+NODE_i)->B[2]=ID[6];
                (ELEMENT_P+NODE_i)->B[3]=ID[7];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
                //revised Oct. 22, 1997
        //revised Sept. 30, 1997
                (ELEMENT_P+NODE_i)->F=I3;
        //revised Sept. 30, 1997
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=4;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[5];
```

```
            (ELEMENT_P+NODE_i)->B[3]=ID[4];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[1];
            (ELEMENT_P+NODE_i)->B[1]=ID[2];
            (ELEMENT_P+NODE_i)->B[2]=ID[6];
            (ELEMENT_P+NODE_i)->B[3]=ID[5];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[2];
            (ELEMENT_P+NODE_i)->B[1]=ID[3];
            (ELEMENT_P+NODE_i)->B[2]=ID[7];
            (ELEMENT_P+NODE_i)->B[3]=ID[6];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[3];
            (ELEMENT_P+NODE_i)->B[1]=ID[0];
            (ELEMENT_P+NODE_i)->B[2]=ID[4];
            (ELEMENT_P+NODE_i)->B[3]=ID[7];
                        //revised on Oct. 22,1997
                        (ELEMENT_P+NODE_i)->E=1000;
            //revised Oct. 22, 1997
//revised Sept. 30, 1997
            (ELEMENT_P+NODE_i)->F=I3;
//revised Sept. 30, 1997
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            /*(ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->B[0]=ID[3];
            (ELEMENT_P+NODE_i)->B[1]=ID[2];
            (ELEMENT_P+NODE_i)->B[2]=ID[1];
```

```
(ELEMENT_P+NODE_i)->B[3]=ID[0];
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->B[0]=ID[7];
(ELEMENT_P+NODE_i)->B[1]=ID[6];
(ELEMENT_P+NODE_i)->B[2]=ID[5];
(ELEMENT_P+NODE_i)->B[3]=ID[4];
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->B[0]=ID[4];
(ELEMENT_P+NODE_i)->B[1]=ID[5];
(ELEMENT_P+NODE_i)->B[2]=ID[1];
(ELEMENT_P+NODE_i)->B[3]=ID[0];
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->B[0]=ID[5];
(ELEMENT_P+NODE_i)->B[1]=ID[6];
(ELEMENT_P+NODE_i)->B[2]=ID[2];
(ELEMENT_P+NODE_i)->B[3]=ID[1];
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->B[0]=ID[6];
(ELEMENT_P+NODE_i)->B[1]=ID[7];
(ELEMENT_P+NODE_i)->B[2]=ID[3];
(ELEMENT_P+NODE_i)->B[3]=ID[2];
0for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->B[0]=ID[7];
(ELEMENT_P+NODE_i)->B[1]=ID[4];
(ELEMENT_P+NODE_i)->B[2]=ID[0];
(ELEMENT_P+NODE_i)->B[3]=ID[3];
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;*/
break;
case 13:
//ELEMENT_NUM=ELEMENT_NUM+1;
//revised in Oct., 1997

for(i=0;i<20;i++)
        {
        fscanf(NEU_INP,"%ld,",&CHECKD);
        ID[i]=FindNid(CHECKD);
        }
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
```

```
                                        (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[2]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[3]=ID[0];
                                                //revised Sept. 30, 1997
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        //revised Sept. 30, 1997
                                        for(i=0;i<5;i++)
                                                                {
                                                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                                }
                                        NODE_i=NODE_i+1;
                                        fgets(buffer,200,NEU_INP);

//revised in Oct., 1997
//comment out two lines below when switching with above block
//                                      fgets(buffer,200,NEU_INP);
//                                      fgets(buffer,200,NEU_INP);
                                        break;
                                }
        for(i=0;i<4;i++)
                {
                fgets(buffer,200,NEU_INP);
                }
        if(I4==13)
                                                {
                                                CHECKD=0;
                                                while(CHECKD!=-1)
                                                                {
                                                                fscanf(NEU_INP,"%ld,",&CHECKD);
                                                                fgets(buffer,200,NEU_INP);

//!!What's with this code block?-Dryer 11/24/97

//ELEMENT_NUM=ELEMENT_NUM+1;
                                        //for(i=0;i<20;i++)
                                        //              {
                                        //              fscanf(NEU_INP,"%ld,",&CHECKD);
                                                                if(CHECKD!=-1){
                                                        ID[1]=FindNid(CHECKD);
                                        //              }
                                        (ELEMENT_P+NODE_i)->A=4;
                                        (ELEMENT_P+NODE_i)->D=CHECKDD;

                                        (ELEMENT_P+NODE_i)->B[0]=ID[0];
                                        (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[2]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[3]=ID[0];
                                                //revised Sept. 30, 1997
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        //revised Sept. 30, 1997
                                        for(i=0;i<5;i++)
                                                                {
                                                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                                }
                                        NODE_i=NODE_i+1;}
                                        //fgets(buffer,200,NEU_INP);

//!!What's with this code block?-Dryer 11/24/97
                                                                }
                                                }
        }
//revised on Sept, 30, 1997
//to find the surface element

//revised on Oct, 22, 1997
// int flag;

for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM;ELEMENT_i++)
```

```
        {
        flag=0;
        if((ELEMENT_P+ELEMENT_i)->E==1000)
                {
                NODE_i=ELEMENT_i+1;
                while((NODE_i<ELEMENT_NUM)&&(flag==0))
                {
                        if((ELEMENT_P+NODE_i)->E==1000)
                        {
                flag=compare(ELEMENT_i,NODE_i);

                        }
                if(flag==1)
                {
                        (ELEMENT_P+ELEMENT_i)->E=1;
                        (ELEMENT_P+NODE_i)->E=1;
                        }
                else{
                        NODE_i=NODE_i+1;
                        }
                }
                if(flag==0)
                {
(ELEMENT_P+ELEMENT_i)->E=0;
                        (ELEMENT_P+NODE_i)->E=0;
                        }
        }
}
//revised on Oct. 22,1997
//revised on Sept 30, 1997
fclose(NEU_INP);
//add in Oct., 1997

//*************************************************************//

if(flag_solid == 1)
{
printf("Would you like to filter the internal surfaces?\n\n");
FLAG = 1;
while(FLAG == 1)
        {
        printf("\n(Y) Yes; (N) No;\n");

        gets(OUTPUT);
  //strcpy(OUTPUT,"Y"); //BEC+

        if((!strcmp(OUTPUT,"Y")) || (!strcmp(OUTPUT,"y")))
        {
        tmp1 = fopen("test.tmp", "w+");
        tmp2=fopen("ngst.tmp","w+");
        ELEMENT_NUM_S=0;
        for (i = 0; i < ELEMENT_NUM; i++)
                {
                        if((ELEMENT_P + i)->E == 0)
                {
                ELEMENT_NUM_S = ELEMENT_NUM_S + 1;
                fprintf(tmp2, "%ld\t%d\t%ld\t%ld\t%ld\t%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\t%d\n",
                (ELEMENT_P+i)->D,(ELEMENT_P+i)->A,(ELEMENT_P+i)->B[0],(ELEMENT_P+i)->B[1],
                (ELEMENT_P+i)->B[2],(ELEMENT_P+i)->B[3],(ELEMENT_P+i)->C[0],
                (ELEMENT_P+i)->C[1],(ELEMENT_P+i)->C[2],(ELEMENT_P+i)->C[3],
                (ELEMENT_P+i)->C[4],(ELEMENT_P+i)->E,(ELEMENT_P+i)->F);
                (NODE_P + ((ELEMENT_P + i)->B[0]))->H = 1;
                (NODE_P + ((ELEMENT_P + i)->B[1]))->H = 1;
                (NODE_P + ((ELEMENT_P + i)->B[2]))->H = 1;
                (NODE_P + ((ELEMENT_P + i)->B[3]))->H = 1;
                }
                }
                NODE_NUM_S = 0;

        for (i = 0; i < NODE_NUM; i++)
```

```
                    {
                    if((NODE_P + i)->H == 1)
                    {

                    NODE_NUM_S=NODE_NUM_S + 1;
                    fprintf(tmp1,"%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\n",
                    i+1, (NODE_P + i)->x, (NODE_P + i)->y, (NODE_P + i)->z,
                    (NODE_P + i)->dx,(NODE_P + i)->dy, (NODE_P + i)->dz, (NODE_P + i)->output_data[0],
                    (NODE_P + i)->output_data[1], (NODE_P + i)->output_data[2],
                    (NODE_P + i)->output_data[3], (NODE_P + i)->output_data[4],
                    (NODE_P + i)->H);
                    }
                    }

          rewind(tmp2);
          rewind(tmp1);
          ELEMENT_NUM=ELEMENT_NUM_S;
          NODE_NUM=NODE_NUM_S;
          free(ELEMENT_P);
          ELEMENT_P=(struct ELEMENT_DATA *)calloc(sizeof (ELEMENT_DATA),ELEMENT_NUM);
          free(NODE_P);
          NODE_P=(struct NODE_DATA *)calloc (sizeof(NODE_DATA),NODE_NUM);
          //printf("%d\t%d\n",ELEMENT_NUM,NODE_NUM);

          for (i=0;i<NODE_NUM;i++)
                    {
                    //printf("tt=%d\n",i);
                    fscanf(tmp1,"%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\n",
                    &((NODE_P+i)->A),&((NODE_P+i)->x),&((NODE_P+i)->y),
                    &((NODE_P+i)->z),&((NODE_P+i)->dx),&((NODE_P+i)->dy),
                    &((NODE_P+i)->dz),&((NODE_P+i)->output_data[0]),
                    &((NODE_P+i)->output_data[1]),&((NODE_P+i)->output_data[2]),
                    &((NODE_P+i)->output_data[3]),&((NODE_P+i)->output_data[4]),
                    &((NODE_P+i)->H));
                    }
          for (i=0;i<ELEMENT_NUM;i++)
                    {
                    //printf("%d\n",i);
                    fscanf(tmp2,"%ld\t%d\t%ld\t%ld\t%ld\t%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\t%d\n",
                    &((ELEMENT_P+i)->D),&((ELEMENT_P+i)->A),&IA,&IB,&IC,&IE,&((ELEMENT_P+i)->C[0]),
                    &((ELEMENT_P+i)->C[1]),&((ELEMENT_P+i)->C[2]),&((ELEMENT_P+i)->C[3]),
                    &((ELEMENT_P+i)->C[4]),&((ELEMENT_P+i)->E),&((ELEMENT_P+i)->F));
(ELEMENT_P+i)->B[0]=FindNid(IA+1);
(ELEMENT_P+i)->B[1]=FindNid(IB+1);
(ELEMENT_P+i)->B[2]=FindNid(IC+1);
(ELEMENT_P+i)->B[3]=FindNid(IE+1);
                    }
                    fclose(tmp1);
                    fclose(tmp2);
                    FLAG=0;
          }
          else if((!strcmp(OUTPUT,"N"))||(!strcmp(OUTPUT,"n")))
          {FLAG=0;}
}
}
//**********************print to file - node data**************************//

fp = fopen("node.lst", "w+");
for(NODE_i=0;NODE_i<NODE_NUM;NODE_i++)
{
fprintf(fp,"%ld                    %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg
%10.5lg\n",
          (NODE_P+NODE_i)->A,(NODE_P+NODE_i)->x,(NODE_P+NODE_i)->y,(NODE_P+NODE_i)->z,
          (NODE_P+NODE_i)->dx,(NODE_P+NODE_i)->dy,(NODE_P+NODE_i)->dz,
          (NODE_P+NODE_i)->output_data[0],
          (NODE_P+NODE_i)->output_data[1],(NODE_P+NODE_i)->output_data[2],
          (NODE_P+NODE_i)->output_data[3],(NODE_P+NODE_i)->output_data[4]);
}
fclose(fp);
```

```
//*****************************print to file - element data*********************************//

fp1=fopen("element.lst","w+");
for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM;ELEMENT_i++)
{
fprintf(fp1,"%ld      %ld                %ld                %ld                %ld                %ld %10.5lg
%10.5lg %10.5lg %10.5lg %10.5lg %ld %ld\n",
        (ELEMENT_P+ELEMENT_i)->D,
        (ELEMENT_P+ELEMENT_i)->A,
        (ELEMENT_P+ELEMENT_i)->B[0],
        (ELEMENT_P+ELEMENT_i)->B[1],
        (ELEMENT_P+ELEMENT_i)->B[2],
        (ELEMENT_P+ELEMENT_i)->B[3],
        (ELEMENT_P+ELEMENT_i)->C[0],
        (ELEMENT_P+ELEMENT_i)->C[1],
        (ELEMENT_P+ELEMENT_i)->C[2],
        (ELEMENT_P+ELEMENT_i)->C[3],
        (ELEMENT_P+ELEMENT_i)->C[4],
        (ELEMENT_P+ELEMENT_i)->E,
        (ELEMENT_P+ELEMENT_i)->F);

}
fclose(fp1);

//************************print to file - info data*****************************//

fp2=fopen("inf.lst","w+");
fprintf(fp2,"%ld  %ld  %ld\n",NODE_NUM,ELEMENT_NUM,ELEMENT_NUM1);
for(i=0;i<5-U_NUM;i++)
{
        fprintf(fp2,"%ld %10.5lg %10.5lg %10.5lg %s",Total_num[case_set_num[U[i]]-1],MIN_VALUE[case_set_num[U[i]]-
1],MAX_VALUE[case_set_num[U[i]]-1],AMAX_VALUE[case_set_num[U[i]]-1],out_set_name[case_set_num[U[i]]-1]);
        strncpy(names->actual_set_name[i],out_set_name[case_set_num[U[i]]],strlen(out_set_name[case_set_num[U[i]]]));
}
for(i=0;i<5-V_NUM;i++)
{
        fprintf(fp2,"%ld %10.5lg %10.5lg %10.5lg %s",Total_num[case_set_num[V[i]]-1],MIN_VALUE[case_set_num[V[i]]-
1],MAX_VALUE[case_set_num[V[i]]-1],AMAX_VALUE[case_set_num[V[i]]-1],out_set_name[case_set_num[V[i]]-1]);
        strncpy(names->actual_set_name[5+i],out_set_name[case_set_num[V[i]]],strlen(out_set_name[case_set_num[V[i]]]));
}

fclose(fp2);

//*****************************print to file - LOAD data*********************************//

        if(LOAD_YES == 1){
                fp_load = fopen("load.lst", "w+");

                for(i = 0; i < LOAD_NUM; i++){
                        fprintf(fp_load, "%d  %s", LOAD_SET[i].SET_ID, LOAD_SET[i].NAME);

                        for(q = 0; q < LOAD_SET[i].NUM; q++){
                                fprintf(fp_load, "%ld  %d\n", LOAD_SET[i].ID[q], LOAD_SET[i].TYPE[q]); //NODE
OR ELEMENT ID

                                for (r = 0; r < 6; r++)
                                        fprintf(fp_load, "%d  %lg\n", LOAD_SET[i].FACE[q*6+r],
LOAD_SET[i].VALUE[q*8+2+r]);
                                }

                }
                fclose(fp_load);
        }

//*****************************print to file - CONSTRAINT data*********************************//

        if(CONSTRAINT_YES == 1){
                fp_constraint = fopen("constraint.lst", "w+");

                for(I2 = 0; I2 < CONSTRAINT_NUM; I2++){
```

```
                        fprintf(fp_constraint, "%d        %s", CONSTRAINT_SET[I2].A, CONSTRAINT_SET[I2].B);

                        for(I3 = 0; I3 < CONSTRAINT_SET[I2].NUM; I3++)
                        {
                                fprintf(fp_constraint, "%ld        %d        %d        %d        %d        %d
        %d\n", CONSTRAINT_SET[I2].ID[I3],
                                CONSTRAINT_SET[I2].INDEX[I3*6], CONSTRAINT_SET[I2].INDEX[I3*6+1],
                                CONSTRAINT_SET[I2].INDEX[I3*6+2],CONSTRAINT_SET[I2].INDEX[I3*6+3],
                                CONSTRAINT_SET[I2].INDEX[I3*6+4],CONSTRAINT_SET[I2].INDEX[I3*6+5]);
                        }
                }
        }
} // end of main loop

//*************************************************************************//
// Function: FindNid
// Inputs:   Entity ID - ID of node or element for output
// Outputs:  Node ID
// Date revised and comments:
//*************************************************************************//

long int FindNid(long int u)
{
        long int NL,NH,Ntmp;

        if(u == 0)
                return Ntmp = -1;

        if((NODE_P + NODE_NUM)->A == u)
                Ntmp = NODE_NUM;
        else
        {
                NH = NODE_NUM - 1;
                NL = 0;
                while(NL <= NH)
                {
                        Ntmp = (NH + NL) / 2;
                        if(u < (NODE_P + Ntmp)->A)
                                NH = Ntmp - 1;
                        else if((NODE_P + Ntmp)->A < u)
                                        NL = Ntmp + 1;
                        else
                                return Ntmp;
                }
        }
        return Ntmp;
}

//*************************************************************************//
// Function: FindEid
// Inputs:
// Outputs:
// Date revised and comments:
//*************************************************************************//

long int FindEid(long int u)
{
        long int NL, NH, Ntmp;

        if(u == 0)
                return Ntmp = -1;
        if((ELEMENT_TMP + ELEMENT_NUM1)->A == u)
                Ntmp = ELEMENT_NUM1;
        else
        {
                NH = ELEMENT_NUM1 - 1;
                NL = 0;
                while(NL <= NH)
                {
                        Ntmp = (NH + NL) / 2;
```

```
                        if(u < (ELEMENT_TMP + Ntmp)->A)
                                NH = Ntmp - 1;
                        else if((ELEMENT_TMP + Ntmp)->A < u)
                                NL = Ntmp + 1;
                        else
                                return Ntmp;
                }
        }
        return Ntmp;
}

//revised on Oct. 22,1997

//*************************************************************************//
// Function: Compare
// Inputs:   element_i, node_i
// Outputs:  Flag
// Date revised and comments:  Oct. 22,1997
//*************************************************************************//

int compare(long int ELEMENT_i, long int NODE_i)
{
        int FLAG, C[4], i, j;

        for(i = 0; i < 4; i++)
                C[i] = 0;

        for(i = 0; i < 4; i++)
        {
                j = 3;
                FLAG = 0;
                while((j >= 0) && (FLAG == 0))
                {
                        if((ELEMENT_P + ELEMENT_i)->B[i] == (ELEMENT_P + NODE_i)->B[j])
                        {
                                FLAG = 1;
                                C[i] = 1;
                        }
                        j = j-1;
                }
        }

        if((C[0] == 1) && (C[1] == 1) && (C[2] == 1) && (C[3] == 1))
                return FLAG = 1;
        else
                return FLAG = 0;
}

//revised on Oct. 22, 1997
```

```
/*****************************************************************************
DVET Release 2.2/11/98 for WindowsNT Workstation
dvetwin.c
11 February 1998
Copyright 1998
Dual Incorporated

        di_add_vertex_color()
        di_animalarm()
        di_animTimer()
        di_create_body_handler()
        di_det_blocks()
        di_FEM_interact()
        di_input_nodes()
        di_input_mods()
        di_intersect_handler()
        di_modify_FEM()
        di_modify_Mesh()
        di_output_mods()
        di_Pmesh_mesh()
        di_Pmesh_obj()
        di_set_range()
        diBodyMoveToFunc()
        diBodyStartupPosFEMFunc()
        diCreateFEMMeshFunc()
        diCreateObjectFunc()
        diCreateTextFunc()
        diImmersDataFunc()
        diNavModeFunc()
        diOutputSetFunc()
        diSetViewFunc()
        diToggleAnimFunc()
        diToggleAnimModeFunc()
        diToggleMeshDynFunc()

AUTHOR:  David A. Dryer, Dual Incorporated

        RegisterScaleToolFunctions()
        ResetSliders_cb()
        SetSliders_cb()
        ToolCreation_cb()
        UpdateSlider_cb()
        UpdateSliderInfo_cb()
        WidgetCreation_cb()

Initial CAU prototype integration of dVS/dVISE widgets: Dr. Sriprakash Sarathy,  Clark Atlanta University 4/29/97
DVET modifications and widget additions: Dr. David Dryer, Dual Incorporated

        fem2vr()
        FindEid()
        FindNid()

Author:  Dr. Baojiu Lin, University of Central Florida
DVET integration and integration modifications: Dr. David Dryer, Dual Incorporated

*****************************************************************************************************/

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#ifdef _UNIX
#include <unistd.h>
#endif /* _UNIX */

#include <dvs/vc.h>
#include <dvise/dvise.h>

//include for FEM2VR translator
```

```
#include "fm2vr1120.h"
//include for InsideTrak headtracker
#include "inside1120.h"

#define min(a,b) ((a)<(b)?(a):(b))
#define max(a,b) ((a)>(b)?(a):(b))

/* PRIVATE STRUCTURES ================================================== */
/*
 * This structure is created by the 'myToolCreation' function and used
 * to store references to the widgets in the interface. These references
 * are filled in by the 'myWidgetCreation' function which is called by
 * each widget when it is created. These references allow the values of the
 * widgets to be set by other functions within the interface since this
 * data structure can be accessed via the Toolbox Used Data.
 */

typedef struct _SliderDataStruct {
 VWidget *LoadFact;
 VWidget *LoadDisp;
 VWidget *ThreshFact;
 VWidget *ThreshDisp;
 VWidget *ExagerFact;
 VWidget *ExagerDisp;
 VWidget *ClrSclTop;
 VWidget *ClrSclTopDisp;
 VWidget *ClrSclBot;
 VWidget *ClrSclBotDisp;
} SliderDataStruct;

typedef struct _intersectArgs
{
  uint32  *event;
  ECObject *object;
} intersectArgs;

typedef struct _MoveInfo {
  VCBody  *body;
  dmPoint  posa;
  dmPoint  posb;
  dmPoint  velocity;
  dmPoint  bodyOffset;
  float32  time, totalTime;
  int32    active;
  ECStateType state;
} MoveInfo;

typedef struct _PmeshInts {
          uint32     noVertices;
          uint32     noVertmesh;
          uint32     noFaces4;
          uint32     noFaces3;
          uint32     rightvert;
  uint32     rightelem;
  uint32     adjindex;
} PmeshInts;

typedef struct _Switches {
          uint32     navstate;
          uint32     navmode;
          uint32     set1;
          uint32     set2;
          uint32     picknode;
          uint32     meshdynmode;
          uint32     outtypenum;      //0 is node type output, 1 is element type output
          uint32     outsubnum;       //node or element subtype index (0-4) in output array
          uint32     animmode;
          uint32     startanim;
          uint32     loadcasestate;
          uint32     constraintstate;
```

```
} Switches;

typedef struct _Points {
        dmPoint         FEMcenter;
        dmPoint         view1;
        dmPoint         view2;
        dmPoint         rightnodep;
        dmPoint         loadnodep;
} Points;

typedef struct _Floats {
        float32 out_vals[5];
        float32 out_min;
        float32 out_max;
        float32 absmax;
        float32 threshold;
        float32 scale;
        float32 LoadFactor;
        float32 transp;
        float32 exager;
        float32 curout;
        float32 beamdelta;
        float32 xyzmax;
        float32 clrscltop;
        float32 clrsclbot;
        float32   femsclbotl[3];
        float32   femsclbotr[3];
        float32   femscltopr[3];
        float32   femscltopl[3];
        float32 alphainrng;
        float32 alphathresh;
        float32 alphaoutrng;
} Floats;

typedef struct _Chars {
        char    outtxt[200];
        char    scltxt[200];
} Chars;

typedef struct _VCfloats {
        VCColor  vcolour;
        VCColor  posmaxcolor;
        VCColor  posmincolor;
        VCColor  negmaxcolor;
        VCColor  negmincolor;
        VCColor  posthreshcolor;
        VCColor  negthreshcolor;
        VCColor  outofrngcolor;
} VCfloats;

struct NAMES *names;//malloc

typedef struct myEntityList {
        VCEntity *nodeobj;
        VCAttribute *vis;
        dmPoint nodepoint;
//      VCVisual *vis;
        struct myEntityList *next;
} EntityList;


//_amblksiz=16384;


PmeshInts *pmi;//malloc
Points          *points;//malloc
Switches  *switches;//malloc
Floats          *floats;//malloc
Chars           *chars;//malloc
VCfloats  *vcfloats;//malloc
```

```
float32             *vertices, *vertmesh;//malloc
float32             *displaceobj, *displacemesh;//malloc
float32             *femsclverts, *femsclgrdverts;//malloc
uint32              *connections4, *connections3;//malloc
uint32              *femsclconts, *femsclgrdconts;//malloc
uint32              *conmesh4, *conmesh3;//malloc
float32             *outvert;//malloc
uint32              *elearray;//malloc
VCGeometry          *femtextstring;//malloc
VCGeometry          *clrscltextstring;//malloc
VCIntersectionReportData        *intersectionReportData;//malloc
uint32              *loadcoordind = NULL;
uint32              *loaddfind = NULL;
uint32              *loadtrack = NULL;
uint32              *constrcoordind;
uint32              *constrdfind;

static VCTime *syncTime=NULL;

EntityList *LoadList;

EntityList *ConstrList;


//VCColour    white={1,1,1},gray={0.5,0.5,0.5},black={0,0,0},red={1,0,0},yellow={1,1,0},blue={0,0,1},green={0,1,0};


ECObject  *objFEM, *objMesh, *objFEMText,//malloc
                            *objClrScl,*objClrSclGrid,*objClrSclText,
                            *objViewButton, *objViewText,
                            *objDataButton, *objDataText,
                            *objVisButton, *objVisText;

ECObjectReference   *objFEMref, *objMeshref, *objFEMTextref,//malloc
                                                *objClrSclref, *objClrSclGridref,*objClrSclTextref,
                                                *objViewButtonref, *objViewTextref,
                                                *objDataButtonref, *objDataTextref,
                                                *objVisButtonref, *objVisTextref;

//*************************function prototypes*************************//

int
di_create_body_handler(VCBodyCreate_CallbackData *bodyData, void *data);

//*********************************************************************//
// Function:  di_det_blocks
//*********************************************************************//

int di_det_blocks()
{
        int adj=0;
        int i,j,k;
        int         elemindex;
        uint32      *tracknode;
        dmPoint             beam1,beam2;
        dmVector  beamvect;
        float32             beamdist;

        pmi=(PmeshInts *)malloc(sizeof(PmeshInts));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;
        pmi->noVertmesh=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
```

```
                                        (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                        (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                        (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);

                                dmPointSet (beam2,
                                        (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                                        (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                                        (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                                dmPointSub (beamvect, beam1, beam2);

        beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                                if (beamdist > .000000001)
                                {
                                        for (i=0; i<2; i++)
                                        {
                                                if (i==0)
                                                {
                                                        j=0;k=1;
                                                }
                                                else
                                                {
                                                        j=3;k=2;
                                                }
                                                if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                                {
                                                        tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                                }
                                                else
                                                {
                                                        pmi->noVertices++;
                                                }
                                                pmi->noVertmesh++;
                                                pmi->noVertices++;
                                        }
                                        pmi->noFaces4++;
                                }
                        }

//If element type is 4 nodes...
                        if((ELEMENT_P+elemindex)->A==4)
                        {
                                for (i=0; i<4; i++)
                                {
                                        if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                        {
                                                tracknode[((ELEMENT_P+elemindex)->B[i])]=1;
                                        }
                                        else
                                        {
                                                pmi->noVertices++;
                                        }
                                }
                                pmi->noFaces4++;
                        }

//If element type is 3 nodes...
                        if((ELEMENT_P+elemindex)->A==3)
                        {
                                for (i=0; i<3; i++)
                                {
                                        if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;
                                        if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                                        {
                                                tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;
                                        }
                                        else
                                        {
```

```
                                        pmi->noVertices++;
                                }
                        }
                        adj=0;
                        pmi->noFaces3++;
                }
  }
        return 1;
}

//*************************************************************************//
// Function:  di_input_nodes
//*************************************************************************//

int di_input_nodes()
{
        int i,j;
        float32 xmax=0.0;
        float32 ymax=0.0;
        float32 zmax=0.0;
        float32 xmin=10000.0;
        float32 ymin=10000.0;
        float32 zmin=10000.0;

        vertices=malloc((pmi->noVertices*7)*sizeof(float32));
        vertmesh=malloc((pmi->noVertices*3)*sizeof(float32));
        displaceobj=malloc((pmi->noVertices*3)*sizeof(float32));
        displacemesh=malloc((pmi->noVertices*3)*sizeof(float32));
        loadcoordind=malloc((LOADSET_NUM)*sizeof(uint32));
        loaddfind=malloc((LOADSET_NUM)*sizeof(uint32));
        loadtrack=malloc((LOADSET_NUM)*sizeof(uint32));
        constrcoordind=malloc((CONSTRAINTSET_NUM)*sizeof(uint32));
        constrdfind=malloc((CONSTRAINTSET_NUM)*sizeof(uint32));

        for (i=0;i < NODE_NUM;i++)
        {

                //Vertex node coordinates - x,y,z assigned to vertices array elements
                //e.g., vertices[0,1,2...7,8,9...

                vertices[(i*7)+0]  = ((NODE_P+i)->x)*floats->scale;
    vertices[(i*7)+1]= ((NODE_P+i)->y)*floats->scale;
    vertices[(i*7)+2]= ((NODE_P+i)->z)*floats->scale;

                displaceobj[(i*3)+0]=((NODE_P+i)->dx);
    displaceobj[(i*3)+1]=((NODE_P+i)->dy);
    displaceobj[(i*3)+2]=((NODE_P+i)->dz);
    displaceobj[(i*3)+0]*=floats->scale;
    displaceobj[(i*3)+1]*=floats->scale;
    displaceobj[(i*3)+2]*=floats->scale;

                vertmesh[(i*3)+0]  = ((NODE_P+i)->x)*floats->scale;
    vertmesh[(i*3)+1]= ((NODE_P+i)->y)*floats->scale;
    vertmesh[(i*3)+2]= ((NODE_P+i)->z)*floats->scale;

                displacemesh[(i*3)+0]=((NODE_P+i)->dx);
    displacemesh[(i*3)+1]=((NODE_P+i)->dy);
    displacemesh[(i*3)+2]=((NODE_P+i)->dz);
    displacemesh[(i*3)+0]*=floats->scale;
    displacemesh[(i*3)+1]*=floats->scale;
    displacemesh[(i*3)+2]*=floats->scale;

// get min, max x,y,z values
                xmax=max(xmax,((NODE_P+i)->x)*floats->scale);
                xmin=min(xmin,((NODE_P+i)->x)*floats->scale);
                ymax=max(ymax,((NODE_P+i)->y)*floats->scale);
                ymin=min(ymin,((NODE_P+i)->y)*floats->scale);
                zmax=max(zmax,((NODE_P+i)->z)*floats->scale);
                zmin=min(zmin,((NODE_P+i)->z)*floats->scale);
```

```
                for (j = 0;j < LOADSET_NUM && j < 100;j++)
                {
                        if (LOAD_SET[LOADSET_PICK].TYPE[j] == 1)
                        {
                                if (LOAD_SET[LOADSET_PICK].ID[j] == (NODE_P+i)->A)
                                {
                                        loadcoordind[j] = i;
                                        loaddfind[j] = j;
                                }
                        }
                }
                for (j=0;j<CONSTRAINTSET_NUM;j++)
                {
                        if (CONSTRAINT_SET[CONSTRAINTSET_PICK].ID[j] == (NODE_P+i)->A)
                        {
                                constrcoordind[j] = i;
                                constrdfind[j] = j;
                        }
                }

        }

//get FEM center point
        points->FEMcenter[VC_X]=xmin+((xmax-xmin)/2.0);
        points->FEMcenter[VC_Y]=ymin+((ymax-ymin)/2.0);
        points->FEMcenter[VC_Z]=zmin+((zmax-zmin)/2.0);

// get max axis length
        if((xmax >= ymax) && (xmax >= zmax))
        {
                floats->xyzmax=xmax-xmin;
        }
        else if((ymax >= xmax) && (ymax >= zmax))
        {
                floats->xyzmax=ymax-ymin;
        }
        else
        {
                floats->xyzmax=zmax-zmin;
        }
return 1;
}

//*************************************************************************//
// Function:  di_set_range
//*************************************************************************//

int di_set_range()
{
        uint32 outindex,OUT_NUM;

        floats->out_min=100000;
        floats->out_max=-100000;

        if (switches->outtypenum==0)
        {
                OUT_NUM=NODE_NUM;
                for (outindex=0;outindex<OUT_NUM;outindex++)
                {
                        floats->out_min=min(floats->out_min,(NODE_P+outindex)->output_data[switches->outsubnum]);
                        floats->out_max=max(floats->out_max,(NODE_P+outindex)->output_data[switches->outsubnum]);
                }
        }
        else
        {
                OUT_NUM=ELEMENT_NUM;
                for (outindex=0;outindex<OUT_NUM;outindex++)
                {
                        floats->out_min=min(floats->out_min,(ELEMENT_P+outindex)->C[switches->outsubnum]);
                        floats->out_max=max(floats->out_max,(ELEMENT_P+outindex)->C[switches->outsubnum]);
```

```
                }
        }

//initially set threshold here
 floats->out_vals[0]=floats->out_min;
 floats->out_vals[1]=0.0;//threshold
 floats->out_vals[2]=floats->out_max;
 floats->absmax=max(fabs(floats->out_min),fabs(floats->out_max));

 return 1;
}

//*************************************************************************//
// Function:  di_add_vertex_color - sets vertex colours
//*************************************************************************//

int di_add_vertex_color(void)
{

//For each curout value, determines if out is positive and in set color region
   if(floats->curout>=0.0 && floats->curout>=floats->out_vals[0] && floats->curout<=floats->out_vals[2])
        {
//If positive and in color region, determines if out is under threshold level-assign positive threshold color
                if(floats->curout<floats->threshold*floats->out_max)
                {
                        vcfloats->vcolour[0]=vcfloats->posthreshcolor[0];
                        vcfloats->vcolour[1]=vcfloats->posthreshcolor[1];
                        vcfloats->vcolour[2]=vcfloats->posthreshcolor[2];
                        floats->transp=floats->alphathresh;
                }
//If positive, in color region, and not under threshold level-assign color
                else
                {
                        vcfloats->vcolour[0]=vcfloats->posmincolor[0]+
                                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                                        (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
                        vcfloats->vcolour[1]=vcfloats->posmincolor[1]+
                                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                                        (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
                        vcfloats->vcolour[2]=vcfloats->posmincolor[2]+
                                                                        ((floats->curout-max(0.0,floats-
>out_vals[0]))/(floats->out_vals[2]-max(0.0,floats->out_vals[0])))*
                                                                        (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
                        floats->transp=floats->alphainmg;
                }
        }
//For each curout value, determines if out is negative and in set color region
   else if(floats->curout<0.0 && fabs(floats->curout)>=floats->out_vals[0] && floats->curout>=floats->out_vals[0])
        {
//If negative and in color region, determines if out is above threshold level-assign negative threshold color
                if(floats->curout>floats->threshold*floats->out_min)
                {
                        vcfloats->vcolour[0]=vcfloats->negthreshcolor[0];
                        vcfloats->vcolour[1]=vcfloats->negthreshcolor[1];
                        vcfloats->vcolour[2]=vcfloats->negthreshcolor[2];
                        floats->transp=floats->alphathresh;
                }
//If negative, in color region, and not above threshold level-assign color
                else
                {
                        vcfloats->vcolour[0]=vcfloats->negmincolor[0]+
                                                                        ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
                                                                        (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
```

```
                        vcfloats->vcolour[1]=vcfloats->negmincolor[1]+
                                                ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
                                                (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
                        vcfloats->vcolour[2]=vcfloats->negmincolor[2]+
                                                ((min(0.0,floats->out_vals[2])-floats-
>curout)/(min(0.0,floats->out_vals[2])-floats->out_vals[0]))*
                                                (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
                        floats->transp=floats->alphainrng;
                }
        }
//For each curout value, determines if curout is out of color scale range - then don't show
        else
        {
                vcfloats->vcolour[0]=vcfloats->outofrngcolor[0];//black
                vcfloats->vcolour[1]=vcfloats->outofrngcolor[1];
                vcfloats->vcolour[2]=vcfloats->outofrngcolor[2];
                floats->transp=floats->alphaoutrng;
        }
        return 1;
}

//*************************************************************************//
// Function: di_input_mods
//*************************************************************************//

int di_input_mods()
{
        int                     elemindex;
        int                     adj=0;
        int                     i,j,k;
        uint32          *tracknode;
        dmPoint         beam1,beam2;
        dmVector beamvect;
        float32         beamdist;
        int                     cused3=0;
        int                     cused4=0;

        connections4=(uint32 *)malloc((pmi->noFaces4*4)*sizeof(uint32));
        connections3=(uint32 *)malloc((pmi->noFaces3*3)*sizeof(uint32));
        conmesh4=(uint32 *)malloc((pmi->noFaces4*4)*sizeof(uint32));
        conmesh3=(uint32 *)malloc((pmi->noFaces3*3)*sizeof(uint32));
        elearray=(uint32 *)malloc(ELEMENT_NUM*5*sizeof(uint32));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;pmi->noVertmesh=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);

                        dmPointSet (beam2,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                        dmPointSub (beamvect, beam1, beam2);

        beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                        if (beamdist > .000000001)
                        {
                                elearray[elemindex*5]=((ELEMENT_P+elemindex)->A)+2;
```

```
for (i=0; i<2; i++)
{
        if (i==0)
        {
                j=0;k=1;
        }
        else
        {
                j=3;k=2;
        }

        if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
        {
                connections4[cused4+j]=((ELEMENT_P+elemindex)->B[i]);
                tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                elearray[(elemindex*5)+(j+1)]=(ELEMENT_P+elemindex)->B[i];
        }
        else
        {
                connections4[cused4+j]=pmi->noVertices;
                vertices[((pmi-
>noVertices)*7)+0]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+0];
                vertices[((pmi-
>noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+1];
                vertices[((pmi-
>noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)->B[i])*7)+2];


                displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
                displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
                displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

                elearray[(elemindex*5)+(j+1)]=pmi->noVertices;

                pmi->noVertices++;
        }

        conmesh4[cused4+j]=((ELEMENT_P+elemindex)->B[i]);
        conmesh4[cused4+k]=pmi->noVertmesh;
        vertmesh[((pmi-
>noVertmesh)*3)+0]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+0])+(beamdist/floats->beamdelta);
        vertmesh[((pmi-
>noVertmesh)*3)+1]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+1])+(beamdist/floats->beamdelta);
        vertmesh[((pmi-
>noVertmesh)*3)+2]=(vertices[(((ELEMENT_P+elemindex)->B[i])*7)+2])+(beamdist/floats->beamdelta);

        displacemesh[(pmi-
>noVertmesh*3)+0]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+0];
        displacemesh[(pmi-
>noVertmesh*3)+1]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+1];
        displacemesh[(pmi-
>noVertmesh*3)+2]=displacemesh[(((ELEMENT_P+elemindex)->B[i])*3)+2];

        connections4[cused4+k]=pmi->noVertices;
        vertices[((pmi->noVertices)*7)+0]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+0])+(beamdist/floats->beamdelta);
        vertices[((pmi->noVertices)*7)+1]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+1])+(beamdist/floats->beamdelta);
        vertices[((pmi->noVertices)*7)+2]=(vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+2])+(beamdist/floats->beamdelta);

        displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
        displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
```

```
                                        displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

                                        elearray[(elemindex*5)+(k+1)]=pmi->noVertices;

                                        pmi->noVertmesh++;
                                        pmi->noVertices++;
                                }

                                cused4+=4;
                        }
                }

                if((ELEMENT_P+elemindex)->A==4)
                {
                        elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                        for (i=0; i<4; i++)
                        {
                                conmesh4[cused4+i]=((ELEMENT_P+elemindex)->B[i]);

                                if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                {
                                        connections4[cused4+i]=((ELEMENT_P+elemindex)->B[i]);
                                        tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                        elearray[(elemindex*5)+(i+1)]=(ELEMENT_P+elemindex)->B[i];
                                }
                                else
                                {
                                        connections4[cused4+i]=pmi->noVertices;
                                        vertices[(pmi->noVertices)*7]=vertices[((ELEMENT_P+elemindex)-
>B[i])*7];
                                        vertices[((pmi->noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+1];
                                        vertices[((pmi->noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+2];

                                        displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+0];
                                        displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+1];
                                        displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i])*3)+2];

                                        elearray[(elemindex*5)+(i+1)]=pmi->noVertices;

                                        pmi->noVertices++;
                                }
                        }
                        cused4+=4;
                }

                //If element type is 3 nodes, read nodes into connections3
//e.g., connections3[0,1,2...3,4,5...6,7,8...
                if((ELEMENT_P+elemindex)->A==3)
                {
                        elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                        for (i=0; i<3; i++)
                        {
                                if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;

                                conmesh3[cused3+i]=((ELEMENT_P+elemindex)->B[i+adj]);

                                if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                                {
                                        connections3[cused3+i]=((ELEMENT_P+elemindex)->B[i+adj]);
                                        tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;
```

```
                                        elearray[(elemindex*5)+(i+1)]=(ELEMENT_P+elemindex)->B[i+adj];
                                }
                                else
                                {
                                        connections3[cused3+i]=pmi->noVertices;
                                        vertices[(pmi->noVertices)*7]=vertices[((ELEMENT_P+elemindex)-
>B[i+adj])*7];
                                        vertices[((pmi->noVertices)*7)+1]=vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+1];
                                        vertices[((pmi->noVertices)*7)+2]=vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+2];

                                        displaceobj[(pmi-
>noVertices*3)+0]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+0];
                                        displaceobj[(pmi-
>noVertices*3)+1]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+1];
                                        displaceobj[(pmi-
>noVertices*3)+2]=displaceobj[(((ELEMENT_P+elemindex)->B[i+adj])*3)+2];

                                        elearray[(elemindex*5)+(i+1)]=pmi->noVertices;

                                        pmi->noVertices++;
                                }
                        }
                        adj=0;
                        cused3+=3;
                }
                return 1;
}

//*************************************************************************//
// Function: di_output_mods
//*************************************************************************//

int di_output_mods()
{
        int                     elemindex;
        int                     adj=0;
        float32         outtmp;
        int                     i,j,k;
        uint32          *tracknode;
        dmPoint         beam1,beam2;
        dmVector beamvect;
        float32         beamdist;

        outvert=(float32 *)malloc(pmi->noVertices*sizeof(float32));
        tracknode=(uint32 *)calloc(NODE_NUM,sizeof(uint32));

        pmi->noVertices=NODE_NUM;

        for (elemindex=0;elemindex<ELEMENT_NUM;elemindex++)
        {
                if((ELEMENT_P+elemindex)->A==2)
                {
                        dmPointSet (beam1,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[0])->z);

                        dmPointSet (beam2,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->x,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->y,
                                (NODE_P+((ELEMENT_P+elemindex))->B[1])->z);

                        dmPointSub (beamvect, beam1, beam2);

        beamdist=sqrt((beamvect[0]*beamvect[0])+(beamvect[1]*beamvect[1])+(beamvect[2]*beamvect[2]));

                        if (beamdist > .000000001)
                        {
```

```
                                        elearray[elemindex*5]=((ELEMENT_P+elemindex)->A)+2;
                                        for (i=0; i<2; i++)
                                        {
                                                if (i==0)
                                                {
                                                        j=0;k=1;
                                                }
                                                else
                                                {
                                                        j=3;k=2;
                                                }

                                                if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                                {
                                                        tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                        else outtmp=(ELEMENT_P+elemindex)->C[switches-
>outsubnum];
                                                        floats->curout=outtmp;
                                                        di_add_vertex_color();

                                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+3]=vcfloats-
>vcolour[0];
                                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+4]=vcfloats-
>vcolour[1];
                                                        vertices[(((ELEMENT_P+elemindex)->B[i])*7)+5]=vcfloats-
>vcolour[2];
                                                        vertices[(((ELEMENT_P+elemindex)-
>B[i])*7)+6]=max(0.0,min(1.0,floats->transp));

                                                        outvert[((ELEMENT_P+elemindex)->B[i])]=floats->curout;
                                                }
                                                else
                                                {
                                                        if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                        else outtmp=(ELEMENT_P+elemindex)->C[switches-
>outsubnum];
                                                        floats->curout=outtmp;
                                                        di_add_vertex_color();

                                                        vertices[(pmi->noVertices*7)+3]=vcfloats->vcolour[0];
                                                        vertices[(pmi->noVertices*7)+4]=vcfloats->vcolour[1];
                                                        vertices[(pmi->noVertices*7)+5]=vcfloats->vcolour[2];
                                                        vertices[(pmi->noVertices*7)+6]=max(0.0,min(1.0,floats-
>transp));

                                                        outvert[pmi->noVertices]=floats->curout;

                                                        pmi->noVertices++;
                                                }

                                                if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                floats->curout=outtmp;
                                                di_add_vertex_color();

                                                vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                                vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                                vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                                vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                                outvert[pmi->noVertices]=floats->curout;

                                                pmi->noVertices++;
                                        }
                                }
```

```
                    }

//If element type is 4 nodes,
                    if((ELEMENT_P+elemindex)->A==4)
                    {
                              elearray[elemindex*5]=(ELEMENT_P+elemindex)->A;

                              for (i=0; i<4; i++)
                              {
                                        if (tracknode[((ELEMENT_P+elemindex)->B[i])]!=1)
                                        {
                                                  tracknode[((ELEMENT_P+elemindex)->B[i])]=1;

                                                  if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                  else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                  floats->curout=outtmp;
                                                  di_add_vertex_color();

                                                  vertices[(((ELEMENT_P+elemindex)->B[i])*7)+3]=vcfloats->vcolour[0];
                                                  vertices[(((ELEMENT_P+elemindex)->B[i])*7)+4]=vcfloats->vcolour[1];
                                                  vertices[(((ELEMENT_P+elemindex)->B[i])*7)+5]=vcfloats->vcolour[2];
                                                  vertices[(((ELEMENT_P+elemindex)->B[i])*7)+6]=max(0.0,min(1.0,floats-
>transp));

                                                  outvert[((ELEMENT_P+elemindex)->B[i])]=floats->curout;
                                        }
                                        else
                                        {
                                                  if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i]))->output_data[switches->outsubnum];
                                                  else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                  floats->curout=outtmp;
                                                  di_add_vertex_color();

                                                  vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                                  vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                                  vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                                  vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                                  outvert[pmi->noVertices]=floats->curout;

                                                  pmi->noVertices++;
                                        }
                              }
                    }

                    //If element type is 3 nodes,
                    if((ELEMENT_P+elemindex)->A==3)
                    {
                              for (i=0; i<3; i++)
                              {
                                        if ((ELEMENT_P+elemindex)->B[i]==-1) adj=1;

                                        if (tracknode[((ELEMENT_P+elemindex)->B[i+adj])]!=1)
                                        {
                                                  tracknode[((ELEMENT_P+elemindex)->B[i+adj])]=1;

                                                  if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i+adj]))->output_data[switches->outsubnum];
                                                  else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                  floats->curout=outtmp;
                                                  di_add_vertex_color();

                                                  vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+3]=vcfloats-
>vcolour[0];
                                                  vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+4]=vcfloats-
>vcolour[1];
                                                  vertices[(((ELEMENT_P+elemindex)->B[i+adj])*7)+5]=vcfloats-
>vcolour[2];
```

```
                                                vertices[(((ELEMENT_P+elemindex)-
>B[i+adj])*7)+6]=max(0.0,min(1.0,floats->transp));

                                                outvert[((ELEMENT_P+elemindex)->B[i+adj])]=floats->curout;
                                        }
                                        else
                                        {
                                                if(switches->outtypenum==0)
outtmp=(NODE_P+((ELEMENT_P+elemindex)->B[i+adj]))->output_data[switches->outsubnum];
                                                else outtmp=(ELEMENT_P+elemindex)->C[switches->outsubnum];
                                                floats->curout=outtmp;
                                                di_add_vertex_color();

                                                vertices[((pmi->noVertices)*7)+3]=vcfloats->vcolour[0];
                                                vertices[((pmi->noVertices)*7)+4]=vcfloats->vcolour[1];
                                                vertices[((pmi->noVertices)*7)+5]=vcfloats->vcolour[2];
                                                vertices[((pmi->noVertices)*7)+6]=max(0.0,min(1.0,floats->transp));

                                                outvert[pmi->noVertices]=floats->curout;

                                                pmi->noVertices++;
                                        }
                                }
                                adj=0;
                        }
                }
                return 1;
}

//*************************************************************************//
// Function: di_Pmesh_obj
//*************************************************************************//

int
di_Pmesh_obj(ECObject *object)
{
  dmPoint       reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual  *vc_vis;
  VCLod         *vc_lod;
  VCGeogroup      *vc_ggrp;
  VCConnectionData cdata[2];
  VCConnectionList *vc_clist;
  char          *mstr;
  int           len;
  VCMaterial      *material;
  ECVisual        *visual;
  VCAttribute     *attribute;
  ECZone          *zone;
  VCEntity        *entity;


  char          *objectname;
  VCColor               ambient={0.7, 0.5, 0.45};
        VCColor                 diffuse={0.7, 0.5, 0.45};
        VCColor                 emmisive={0.0,0.0,0.0};
        VCColor                 opacity={0.5,0.5,0.5};
        VCSpecular                      specular={0.1, 0.1, 0.0, 0.0};
        VCGeometry  *vc_geom;

  objectname=ECObjectGetName(object);//object and objectname is objFEM)

  mstr=dStringFromOptions(NULL, &len, "objMat", DS_END_OF_OPTIONS);
  material=VCMaterial_Create  (mstr,                                    //      Name
                                                                        VC_MATERIAL_ENABLE,     //
        Mode
                                                                        ambient,                //
        Ambient
                                                                        diffuse,                //
        Diffuse
```

```
                                                                specular,       //
        Specular
                                                                emmisive,       //
        Emmisive
                                                                opacity,        //
        Opacity
                                                                NULL,
        //      Texture
                                                                NULL,
        //      Ramp
                                                                NULL);
        //      Env. Map

  if (!material) printf("Failed to create material 'objMat'\n");

  /* Create dynamic visual */
  vc_vis = VCDynamicVisual_Create(objectname, 0);

  /* Create lod */
  vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

  /* Create geogroup */
  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"objMat","objMat");
//  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,
//
        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_WIREFRAME,0,"objMat", "objMat");

    cdata[0].type=VC_CONNECTIONLIST;
    cdata[0].faceCount=4;
        cdata[0].noConnections=pmi->noFaces4;
        cdata[0].data=connections4;

            cdata[1].type=VC_CONNECTIONLIST;
    cdata[1].faceCount=3;
        cdata[1].noConnections=pmi->noFaces3;
        cdata[1].data=connections3;


        vc_geom = VCPmesh_Create(VC_VERTEX_RGBA, pmi->noVertices, vertices, 2, cdata);

        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

 entity = ECObjectGetVCEntity(object);

        visual = ECObjectGetVisual(object, NULL);

        if (visual == NULL)
          {
           VC_Error("visual was NULL\n");
           return(ECKeepAction);
          }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);

//      ECVisualSetVCAttribute(visual,attribute);

//              Dryer switched ECVisualSetVCAttribute(vis,att) to ECVisualToVC (obj, vis) below:
        ECVisualToVC (object, visual);
//              ECVisualToVC flushes the information in the ECVisual structure to the dVS database via the VC Attribute
//              referenced in the data structure. If there is no VC attribute assigned to this ECVisual then a VCAttribute(5) is
//              created and assigned to the VCEntity(5) referenced by the ECObject(5).
//              format: int ECVisualToVC (ECObject *o, ECVisual *o);


        ECObjectToVC(object);
```

```
        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateFEMObjectFunc - function creates the model in the 3d
//                                    space.
//*************************************************************************//

int
diCreateFEMObjectFunc(ECEvent *event, ECEventData data, ECAction *action)
{
 void                **args = action->parameters;
 ECObject            *obj;
 char                            *varNameFactor; /*Modification factor */
 char                            *varFactor; /*Modification factor */
 int i;
 if(ECArgReferenceGetValue(args[2], (void *)&floats->scale, &data.focus) == VC_ERR)
  floats->scale = 1.00;

        di_det_blocks();

        di_input_nodes();

        di_set_range();

        di_input_mods();

        di_output_mods();

        objFEMref = (ECObjectReference *)args[1];

        objFEM = ECReferenceObject(objFEMref, &data.focus);
        if(objFEM == NULL)
        {
                VC_Error("Could not find object\n");
                return(ECKeepAction);
        }

        di_Pmesh_obj(objFEM);
        //printf("di_Pmesh_obj() complete\n");

        di_modify_FEM();
        //printf("di_modify_FEM() complete\n");
}

//*************************************************************************//
// Function:  di_modify_FEM - updates the FEM object after changes.
//*************************************************************************//

int di_modify_FEM(void)
{

 ECVisual        *visual;
 VCAttribute      *attr;
 VCEntity        *entity;
 VCDynamicVisual   *dyn_vis;
 VCLod           *dyn_lod;
 VCGeogroup       *dyn_geogrp;
 VCGeometry       *dyn_geom;
 VCVertex_Reference ref;
 int           stat;
 VCDynamicVisual_Traverse traverse1;
 VCLod_Traverse         traverse2;
 VCGeogroup_Traverse     traverse3;
 int               i,index;
 dmPoint             curvertpos;
 char                            *varNameFactor; /*Modification factor */
 char                            *varFactor; /*Modification factor */
```

```
if(objFEM == NULL)
 {
   VC_Error("Could not find object\n");
   return(ECKeepAction);
 }

 entity = ECObjectGetVCEntity(objFEM);
 if(entity == NULL)
 {
   VC_Error("Could not find entity\n");
   return(ECKeepAction);
 }

 visual = ECObjectGetVisual(objFEM, NULL);
 if(visual == NULL)
 {
   VC_Error("Could not find visual\n");
   return(ECKeepAction);
 }
 attr = ECVisualGetVCAttribute(visual);

 ECObjectToVC(objFEM);

 VCVisual_GetDynamicVisual(attr,&dyn_vis);

 if(dyn_vis == NULL)
 {
   VC_Error("Could not find dynamic visual\n");
   return(ECKeepAction);
 }

 dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

 dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_RGBA,&traverse2);

 dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);


 i=0;index=0;
 stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);
 while (stat == VC_OK)
 {
  curvertpos[0]=vertices[index]+displaceobj[(i*3)+0]*floats->LoadFactor*floats->exager;
  curvertpos[1]=vertices[index+1]+displaceobj[(i*3)+1]*floats->LoadFactor*floats->exager;
  curvertpos[2]=vertices[index+2]+displaceobj[(i*3)+2]*floats->LoadFactor*floats->exager;

          floats->curout=outvert[i]*floats->LoadFactor;
          di_add_vertex_color();

  ref.data[0]=curvertpos[0];
  ref.data[1]=curvertpos[1];
  ref.data[2]=curvertpos[2];
  ref.data[3]=min(1.0,vcfloats->vcolour[0]);
  ref.data[4]=min(1.0,vcfloats->vcolour[1]);
  ref.data[5]=min(1.0,vcfloats->vcolour[2]);
  ref.data[6]=max(0.0,min(1.0,floats->transp));
  stat = VCGeometry_GetNextVertex(&ref);
  i++;
  index+=7;
 }
 VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function: di_Pmesh_mesh
//*************************************************************************//

int
di_Pmesh_mesh(ECObject *object)
```

```
{
  dmPoint       reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual  *vc_vis;
  VCLod         *vc_lod;
  VCGeogroup     *vc_ggrp;
  VCConnectionData cdata[2];
  VCConnectionList *vc_clist;
  char          *mstr;
  int           len;
  VCMaterial     *material;
  ECVisual       *visual;
  VCAttribute    *attribute;
  ECZone         *zone;
  VCEntity       *entity;
  char          *objectname;
        VCColor              emmisive={0.5,0.5,0.5};//gray
        VCColour             white={1,1,1},black={0,0,0};
        VCGeometry  *vc_geom;

  objectname=ECObjectGetName(object);//object and objectname is objMesh)

  mstr=dStringFromOptions(NULL, &len, "meshMat", DS_END_OF_OPTIONS);
  material=VCMaterial_Create  (mstr,                                    //      Name
                                                                        VC_MATERIAL_ENABLE,     //
        Mode
                                                                        black,
        //          Ambient
                                                                        black,                  //
        Diffuse
                                                                        black,                  //
        Specular
                                                                        emmisive,               //
        Emmisive
                                                                        white,                  //
        Opacity
                                                                        NULL,
        //          Texture
                                                                        NULL,
        //          Ramp
                                                                        NULL);
        //          Env. Map

  if (!material) printf("Failed to create material 'meshMat\n");

  /* Create dynamic visual */
  vc_vis = VCDynamicVisual_Create(objectname, 0);

  /* Create lod */
  vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

  /* Create geogroup */
  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_XYZ,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"meshMat", "meshMat");
//  vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,
//
        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_WIREFRAME,0,"meshMat", "meshMat");

    cdata[0].type=VC_CONNECTIONLIST;
    cdata[0].faceCount=4;
        cdata[0].noConnections=pmi->noFaces4;
        cdata[0].data=conmesh4;

            cdata[1].type=VC_CONNECTIONLIST;
    cdata[1].faceCount=3;
        cdata[1].noConnections=pmi->noFaces3;
        cdata[1].data=conmesh3;


        vc_geom = VCPmesh_Create(VC_VERTEX_XYZ, pmi->noVertmesh, vertmesh, 2, cdata);
```

```
        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

 entity = ECObjectGetVCEntity(object);

        visual = ECObjectGetVisual(object, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);

//      ECVisualSetVCAttribute(visual,attribute);

//              Dryer switched ECVisualSetVCAttribute(vis,att) to ECVisualToVC (obj, vis) below:
        ECVisualToVC (object, visual);
//              ECVisualToVC flushes the information in the ECVisual structure to the dVS database via the VC Attribute
//              referenced in the data structure. If there is no VC attribute assigned to this ECVisual then a VCAttribute(5) is
//              created and assigned to the VCEntity(5) referenced by the ECObject(5).
//              format: int ECVisualToVC (ECObject *o, ECVisual *o);

        ECObjectToVC(object);

        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateFEMMeshFunc
//*************************************************************************//

int
diCreateFEMMeshFunc(ECEvent *event, ECEventData data, ECAction *action)
{
 void              **args = action->parameters;
 ECObject          *obj;
 char                        *varNameFactor;  /*Modification factor */
 char                        *varFactor;  /*Modification factor */
 int i;
 if(ECArgReferenceGetValue(args[2], (void *)&floats->scale, &data.focus) == VC_ERR)
  floats->scale = 1.00;

        objMeshref = (ECObjectReference *)args[1];
        objMesh = ECReferenceObject(objMeshref, &data.focus);
        if(objMesh == NULL)
        {
                VC_Error("Could not find object\n");
                return(ECKeepAction);
        }

        di_Pmesh_mesh(objMesh);
        if (switches->meshdynmode==1) di_modify_Mesh();
}

//*************************************************************************//
// Function:  di_modify_Mesh - updates mesh after a changed has occured.
//*************************************************************************//

int di_modify_Mesh(void)
{

 ECVisual        *visual;
 VCAttribute      *attr;
 VCEntity        *entity;
 VCDynamicVisual   *dyn_vis;
```

```
VCLod           *dyn_lod;
VCGeogroup      *dyn_geogrp;
VCGeometry      *dyn_geom;
VCVertex_Reference ref;
int             stat;
VCDynamicVisual_Traverse traverse1;
VCLod_Traverse          traverse2;
VCGeogroup_Traverse     traverse3;
int             i,index;
dmPoint             curvertpos;
char                            *varNameFactor; /*Modification factor */
char                            *varFactor; /*Modification factor */


if(objMesh == NULL)
{
  VC_Error("Could not find object\n");
  return(ECKeepAction);
}

entity = ECObjectGetVCEntity(objMesh);
if(entity == NULL)
{
  VC_Error("Could not find entity\n");
  return(ECKeepAction);
}

visual = ECObjectGetVisual(objMesh, NULL);
if(visual == NULL)
{
  VC_Error("Could not find visual\n");
  return(ECKeepAction);
}
attr = ECVisualGetVCAttribute(visual);

ECObjectToVC(objMesh);

VCVisual_GetDynamicVisual(attr,&dyn_vis);

if(dyn_vis == NULL)
{
  VC_Error("Could not find dynamic visual\n");
  return(ECKeepAction);
}

dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_XYZ,&traverse2);

dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);


i=0;index=0;
stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);
while (stat == VC_OK)
{
 curvertpos[0]=vertmesh[index]+displacemesh[(i*3)+0]*floats->LoadFactor*floats->exager;
 curvertpos[1]=vertmesh[index+1]+displacemesh[(i*3)+1]*floats->LoadFactor*floats->exager;
 curvertpos[2]=vertmesh[index+2]+displacemesh[(i*3)+2]*floats->LoadFactor*floats->exager;

        floats->curout=outvert[i]*floats->LoadFactor;
        di_add_vertex_color();

 ref.data[0]=curvertpos[0];
 ref.data[1]=curvertpos[1];
 ref.data[2]=curvertpos[2];
 stat = VCGeometry_GetNextVertex(&ref);
 i++;
 index+=3;
}
```

```
 VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function:  diCreateFEMTextFunc
//*************************************************************************//

int
diCreateFEMTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECObject       *object;
   ECObjectReference *ref;
   dmPoint        reference = { 0.0f, 0.0f, 0.0f };
   VCDynamicVisual  *vc_vistext;
   VCLod          *vc_lodtext;
   VCGeogroup      *vc_ggrptext;
   int32          text_len=200;
   VCEntity       *text_ent = NULL;
   char           *mstr;
   int            len;
          VCMaterial      *material;
   ECVisual       *visual;
   VCAttribute     *attribute;
   char           *textstring="No Selection";
          VCColour    white={1,1,1},black={0,0,0},blue={0,0,1};
   dmScale        s={0.005, 0.007, 0.007};

          objFEMTextref = (ECObjectReference *)args[1];
          objFEMText = ECReferenceObject(objFEMTextref, &data.focus);

   mstr=dStringFromOptions(NULL, &len, "blue", DS_END_OF_OPTIONS);
   material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, blue,
                                        white, NULL, NULL, NULL);
   if (!material) printf("Text: Failed to create material blue emmissive\n");

                        text_ent=VCEntity_Create(NULL, 0);

                        vc_vistext=VCDynamicVisual_Create("text_ent", 0);

                        vc_lodtext = VCDynamicVisual_AddLod(vc_vistext,"#1", 0.0, -1, reference);

                        vc_ggrptext = VCLod_AddGeogroup(vc_lodtext,VC_VERTEX_XYZ,
                                        0,0,0,0,0,"blue", "blue");

                        femtextstring=VCString_CreateSized(textstring, text_len, 0, NULL, NULL, s);

                        VCGeogroup_AttachString(vc_ggrptext, femtextstring);

                        visual = ECObjectGetVisual(objFEMText, NULL);
                        if (visual == NULL)
                        {
                                VC_Error("visual was NULL\n");
                                return(ECKeepAction);
                        }

                        attribute = ECVisualGetVCAttribute(visual);
                        VCVisual_SetDynamicVisual(attribute,vc_vistext);
                        ECVisualToVC (objFEMText, visual);
                        ECObjectToVC(objFEMText);
                        return(ECKeepAction);
}

//*************************************************************************//
// Function:  diCreateClrSclTextFunc
//*************************************************************************//

int
diCreateClrSclTextFunc(ECEvent *event, ECEventData data, ECAction *action)
```

```
{
  void          **args = action->parameters;
  ECObject        *object;
  ECObjectReference *ref;
  dmPoint         reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual  *vc_vistext;
  VCLod           *vc_lodtext;
  VCGeogroup       *vc_ggrptext;
  int32           text_len=200;
  VCEntity        *clrscltext_ent = NULL;
  char            *mstr;
  int             len;
          VCMaterial      *material;
  ECVisual        *visual;
  VCAttribute      *attribute;
  char            *clrscltextstr="clrscltextstr";
          VCColour    white={1,1,1},black={0,0,0},blue={0,0,1};
  dmPoint         p={-0.02, 0.345, -0.01};
  dmScale         s={0.013, 0.030, 0.01};
//  dmScale         s={0.013, 0.029, 0.01};

  objClrSclTextref = (ECObjectReference *)args[1];
  objClrSclText = ECReferenceObject(objClrSclTextref, &data.focus);

  mstr=dStringFromOptions(NULL, &len, "blue", DS_END_OF_OPTIONS);
  material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, blue,
                                        white, NULL, NULL, NULL);
  if (!material) printf("Text: Failed to create material blue emmissive\n");

                clrscltext_ent=VCEntity_Create(NULL, 0);

                vc_vistext=VCDynamicVisual_Create("clrscltext_ent", 0);

                vc_lodtext = VCDynamicVisual_AddLod(vc_vistext,"#1", 0.0, -1, reference);

                vc_ggrptext = VCLod_AddGeogroup(vc_lodtext,VC_VERTEX_XYZ,
                                        0,0,0,0,0,"blue", "blue");

                clrscltextstring=VCString_CreateSized(clrscltextstr, text_len, 0, p, NULL, s);

                di_updateclrscltxt();

                VCGeogroup_AttachString(vc_ggrptext, clrscltextstring);

                visual = ECObjectGetVisual(objClrSclText, NULL);
                if (visual == NULL)
                {
                        VC_Error("visual was NULL\n");
                        return(ECKeepAction);
                }

                attribute = ECVisualGetVCAttribute(visual);
                VCVisual_SetDynamicVisual(attribute,vc_vistext);
                ECVisualToVC (objClrSclText, visual);
                ECObjectToVC(objClrSclText);
                return(ECKeepAction);
}

//*************************************************************************//
// Function: di_updateclrscltxt
//*************************************************************************//

int di_updateclrscltxt()
{
        sprintf(chars->scltxt,"%10.6f\n\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n  %10.6f\n
%10.6f\n  %10.6f\n  %10.6f\n\n%10.6f",
                                        floats->out_vals[2],
                                        floats->out_min+(1.0*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.9*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.8*(floats->out_max-floats->out_min)),
```

```
                                        floats->out_min+(0.7*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.6*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.5*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.4*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.3*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.2*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.1*(floats->out_max-floats->out_min)),
                                        floats->out_min+(0.0*(floats->out_max-floats->out_min)),
                                        floats->out_vals[0]);
        VCString_SetText(clrscltextstring,chars->scltxt);
}

//*************************************************************************//
// Function: diCreateColorSclFunc
//*************************************************************************//

int
diCreateColorSclFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECObject      *object;
  ECObjectReference *ref;
  dmPoint       reference = { 0.0f, 0.0f, 0.0f };
  VCDynamicVisual *vc_vis;
  VCLod         *vc_lod;
  VCGeogroup    *vc_ggrp;
  VCConnectionData cdata[1];
  VCConnectionList *vc_clist;
  char          *mstr;
  int           len;
  VCMaterial    *material;
  ECVisual      *visual;
  VCAttribute   *attribute;
  ECZone        *zone;
  VCEntity      *femscl_ent = NULL;
  VCColor               ambient={0.7, 0.5, 0.45};
        VCColor         diffuse={0.7, 0.5, 0.45};
        VCColor         emmisive={0.0,0.0,0.0};
        VCColor         opacity={0.5,0.5,0.5};
        VCSpecular      specular={0.1, 0.1, 0.0, 0.0};
        VCGeometry      *vc_geom;
        int             i;
        float32         clevel,dlevel,zerolevel;
        float32         posmincolormod,negmincolormod;
  dmScale       s={0.85, 1.038, 1.00};

        femsclverts=(float32 *)malloc((24*7)*sizeof(float32));
  femsclconts=(uint32 *)malloc((6*4)*sizeof(uint32));

        objClrSclref = (ECObjectReference *)args[1];
  objClrScl = ECReferenceObject(objClrSclref, &data.focus);

  mstr=dStringFromOptions(NULL, &len, "femsclMat", DS_END_OF_OPTIONS);
  material=VCMaterial_Create  (mstr,                                    //      Name
                                                                        VC_MATERIAL_ENABLE,     //
        Mode                                                            ambient,                //
        Ambient                                                         diffuse,                //
        Diffuse                                                         specular,               //
        Specular                                                        emmisive,               //
        Emmisive                                                        opacity,                //
        Opacity                                                         NULL,
        //      Texture                                                 NULL,
        //      Ramp
```

```
                                                                    NULL);
        //          Env. Map

if (!material) printf("Text: Failed to create material 'femsclMat'\n");

        femscl_ent=VCEntity_Create(NULL, 0);

        /* Create dynamic visual */
vc_vis = VCDynamicVisual_Create("femscl_ent", 0);

// Create lod
vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

// Create geogroup
vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_RGBA,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_SOLID,0,"femsclMat","femsclMat");

        // Set geometry
                femsclverts[(0*7)+0]=floats->femsclbotl[0];
                femsclverts[(0*7)+1]=floats->femsclbotl[1];//level a
                femsclverts[(0*7)+2]=floats->femsclbotl[2];

                femsclverts[(1*7)+0]=floats->femsclbotr[0];
                femsclverts[(1*7)+1]=floats->femsclbotr[1];//level a
                femsclverts[(1*7)+2]=floats->femsclbotr[2];

                femsclverts[(2*7)+0]=floats->femscltopr[0];
                femsclverts[(2*7)+1]=floats->femscltopr[1]*floats->clrsclbot;//level b
                femsclverts[(2*7)+2]=floats->femscltopr[2];

                femsclverts[(3*7)+0]=floats->femscltopl[0];
                femsclverts[(3*7)+1]=floats->femscltopl[1]*floats->clrsclbot;//level b
                femsclverts[(3*7)+2]=floats->femscltopl[2];

                for (i=0;i<4;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->outofrngcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->outofrngcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->outofrngcolor[2];
                        femsclverts[(i*7)+6]=floats->alphaoutrng;
                }

                femsclverts[(4*7)+0]=floats->femscltopl[0];
                femsclverts[(4*7)+1]=floats->femscltopl[1]*floats->clrsclbot;//level b
                femsclverts[(4*7)+2]=floats->femscltopl[2];

                femsclverts[(5*7)+0]=floats->femscltopr[0];
                femsclverts[(5*7)+1]=floats->femscltopr[1]*floats->clrsclbot;//level b
                femsclverts[(5*7)+2]=floats->femscltopr[2];

                for (i=4;i<6;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->negmaxcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->negmaxcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->negmaxcolor[2];
                        femsclverts[(i*7)+6]=floats->alphainrng-.2;
                }

                if (floats->out_vals[2]<=0.0)//case 3
                {
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                        clevel=max(min(1.0-floats->threshold,floats->clrscltop),floats->clrsclbot);
//below (commented out) gives a relative threshold percentage of color range
//                      clevel=floats->clrscltop-(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                        zerolevel=floats->clrscltop;
                        dlevel=floats->clrscltop;
                        negmincolormod=(floats->clrscltop-clevel)/(floats->clrscltop-floats->clrsclbot);
                        posmincolormod=0.0;
                }
```

```
                    else if (floats->out_vals[0]>=0.0)//case 2
                    {
                              clevel=floats->clrsclbot;
                              zerolevel=floats->clrsclbot;
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                              dlevel=min(max(floats->threshold,floats->clrsclbot),floats->clrscltop);
//below (commented out) gives a relative threshold percentage of color range
//                            dlevel=floats->clrsclbot+(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                              negmincolormod=0.0;
                              posmincolormod=(dlevel-floats->clrsclbot)/(floats->clrscltop-floats->clrsclbot);
                    }
                    else//case 1
                    {
                              zerolevel=fabs(floats->out_min)/(floats->out_max-floats->out_min);
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                              clevel=min(max((zerolevel-(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrsclbot),zerolevel);
                              dlevel=max(min((zerolevel+(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrscltop),zerolevel);
//below (commented out) gives a relative threshold percentage of color range
//                            clevel=zerolevel-(floats->threshold*(zerolevel-floats->clrsclbot));
//                            dlevel=zerolevel+(floats->threshold*(floats->clrscltop-zerolevel));
                              negmincolormod=(zerolevel-clevel)/(zerolevel-floats->clrsclbot);
                              posmincolormod=(dlevel-zerolevel)/(floats->clrscltop-zerolevel);
                    }

                    femsclverts[(6*7)+0]=floats->femscltopr[0];
                    femsclverts[(6*7)+1]=floats->femscltopr[1]*clevel;//level c
                    femsclverts[(6*7)+2]=floats->femscltopr[2];

                    femsclverts[(7*7)+0]=floats->femscltopl[0];
                    femsclverts[(7*7)+1]=floats->femscltopl[1]*clevel;//level c
                    femsclverts[(7*7)+2]=floats->femscltopl[2];

                    for (i=6;i<8;i++)
                    {
                              femsclverts[(i*7)+3]=vcfloats->negmincolor[0]+
                                                                  negmincolormod*
                                                                  (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
                              femsclverts[(i*7)+4]=vcfloats->negmincolor[1]+
                                                                  negmincolormod*
                                                                  (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
                              femsclverts[(i*7)+5]=vcfloats->negmincolor[2]+
                                                                  negmincolormod*
                                                                  (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
                              femsclverts[(i*7)+6]=floats->alphainrng-.2;
                    }

                    femsclverts[(8*7)+0]=floats->femscltopl[0];
                    femsclverts[(8*7)+1]=floats->femscltopl[1]*clevel;//level c
                    femsclverts[(8*7)+2]=floats->femscltopl[2];

                    femsclverts[(9*7)+0]=floats->femscltopr[0];
                    femsclverts[(9*7)+1]=floats->femscltopr[1]*clevel;//level c
                    femsclverts[(9*7)+2]=floats->femscltopr[2];

                    femsclverts[(10*7)+0]=floats->femscltopr[0];
                    femsclverts[(10*7)+1]=floats->femscltopr[1]*zerolevel;//zerolevel
                    femsclverts[(10*7)+2]=floats->femscltopr[2];

                    femsclverts[(11*7)+0]=floats->femscltopl[0];
                    femsclverts[(11*7)+1]=floats->femscltopl[1]*zerolevel;//zerolevel
                    femsclverts[(11*7)+2]=floats->femscltopl[2];

                    for (i=8;i<12;i++)
                    {
                              femsclverts[(i*7)+3]=vcfloats->negthreshcolor[0];
```

```
                femsclverts[(i*7)+4]=vcfloats->negthreshcolor[1];
                femsclverts[(i*7)+5]=vcfloats->negthreshcolor[2];
                femsclverts[(i*7)+6]=floats->alphathresh;
        }

        femsclverts[(12*7)+0]=floats->femscltopl[0];
        femsclverts[(12*7)+1]=floats->femscltopl[1]*zerolevel;//zerolevel
        femsclverts[(12*7)+2]=floats->femscltopl[2];

        femsclverts[(13*7)+0]=floats->femscltopr[0];
        femsclverts[(13*7)+1]=floats->femscltopr[1]*zerolevel;//zerolevel
        femsclverts[(13*7)+2]=floats->femscltopr[2];

        femsclverts[(14*7)+0]=floats->femscltopr[0];
        femsclverts[(14*7)+1]=floats->femscltopr[1]*dlevel;//level d
        femsclverts[(14*7)+2]=floats->femscltopr[2];

        femsclverts[(15*7)+0]=floats->femscltopl[0];
        femsclverts[(15*7)+1]=floats->femscltopl[1]*dlevel;//level d
        femsclverts[(15*7)+2]=floats->femscltopl[2];

        for (i=12;i<16;i++)
        {
                femsclverts[(i*7)+3]=vcfloats->posthreshcolor[0];
                femsclverts[(i*7)+4]=vcfloats->posthreshcolor[1];
                femsclverts[(i*7)+5]=vcfloats->posthreshcolor[2];
                femsclverts[(i*7)+6]=floats->alphathresh;
        }

        femsclverts[(16*7)+0]=floats->femscltopl[0];
        femsclverts[(16*7)+1]=floats->femscltopl[1]*dlevel;//level d
        femsclverts[(16*7)+2]=floats->femscltopl[2];

        femsclverts[(17*7)+0]=floats->femscltopr[0];
        femsclverts[(17*7)+1]=floats->femscltopr[1]*dlevel;//level d
        femsclverts[(17*7)+2]=floats->femscltopr[2];

        for (i=16;i<18;i++)
        {
                femsclverts[(i*7)+3]=vcfloats->posmincolor[0]+
                                                        posmincolormod*
                                                        (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
                femsclverts[(i*7)+4]=vcfloats->posmincolor[1]+
                                                        posmincolormod*
                                                        (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
                femsclverts[(i*7)+5]=vcfloats->posmincolor[2]+
                                                        posmincolormod*
                                                        (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
                femsclverts[(i*7)+6]=floats->alphainrng-.2;
        }

        femsclverts[(18*7)+0]=floats->femscltopr[0];
        femsclverts[(18*7)+1]=floats->femscltopr[1]*floats->clrscltop;//level e
        femsclverts[(18*7)+2]=floats->femscltopr[2];

        femsclverts[(19*7)+0]=floats->femscltopl[0];
        femsclverts[(19*7)+1]=floats->femscltopl[1]*floats->clrscltop;//level e
        femsclverts[(19*7)+2]=floats->femscltopl[2];

        for (i=18;i<20;i++)
        {
                femsclverts[(i*7)+3]=vcfloats->posmaxcolor[0];
                femsclverts[(i*7)+4]=vcfloats->posmaxcolor[1];
                femsclverts[(i*7)+5]=vcfloats->posmaxcolor[2];
                femsclverts[(i*7)+6]=floats->alphainrng-.2;
        }
```

```
                femsclverts[(20*7)+0]=floats->femscltopl[0];
                femsclverts[(20*7)+1]=floats->femscltopl[1]*floats->clrscltop;//level e
                femsclverts[(20*7)+2]=floats->femscltopl[2];

                femsclverts[(21*7)+0]=floats->femscltopr[0];
                femsclverts[(21*7)+1]=floats->femscltopr[1]*floats->clrscltop;//level e
                femsclverts[(21*7)+2]=floats->femscltopr[2];


                femsclverts[(22*7)+0]=floats->femscltopr[0];
                femsclverts[(22*7)+1]=floats->femscltopr[1];//level f
                femsclverts[(22*7)+2]=floats->femscltopr[2];

                femsclverts[(23*7)+0]=floats->femscltopl[0];
                femsclverts[(23*7)+1]=floats->femscltopl[1];//level f
                femsclverts[(23*7)+2]=floats->femscltopl[2];

                for (i=20;i<24;i++)
                {
                        femsclverts[(i*7)+3]=vcfloats->outofrngcolor[0];
                        femsclverts[(i*7)+4]=vcfloats->outofrngcolor[1];
                        femsclverts[(i*7)+5]=vcfloats->outofrngcolor[2];
                        femsclverts[(i*7)+6]=floats->alphaoutrng;
                }


        for (i=0;i<24;i++)
        {
                femsclconts[i] = i;
        }

        cdata[0].type=VC_CONNECTIONLIST;
  cdata[0].faceCount=4;
        cdata[0].noConnections=6;
        cdata[0].data=femsclconts;

        vc_geom = VCPmesh_Create(VC_VERTEX_RGBA, 24, (VCVertex) femsclverts, 1, cdata);

        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

        visual = ECObjectGetVisual(objClrScl, NULL);

        if (visual == NULL)
          {
           VC_Error("visual was NULL\n");
           return(ECKeepAction);
          }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);
        ECVisualToVC (objClrScl, visual);
//      ECObjectSetPosOrScale(objClrScl,NULL,NULL,s);//DAD
        ECObjectToVC(objClrScl);

        return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateColorSclGridFunc
//*************************************************************************//

int
diCreateColorSclGridFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   ECObject       *object;
   ECObjectReference *ref;
   dmPoint        reference = { 0.0f, 0.0f, 0.0f };
```

```
VCDynamicVisual *vc_vis;
VCLod           *vc_lod;
VCGeogroup      *vc_ggrp;
VCConnectionData cdata[1];
VCConnectionList *vc_clist;
char            *mstr;
int             len;
VCMaterial      *material;
ECVisual        *visual;
VCAttribute     *attribute;
ECZone          *zone;
VCEntity        *femsclgrd_ent = NULL;
        VCColor                 white={1,1,1},black={0,0,0},grdcolor={1,1,1};
        VCGeometry              *vc_geom;
        int             i;
        dmScale                 s={0.85, 1.038, 1.00};
        float32                 femsclgrdxyz[]=         {
                                                                0,0,0,
                                                                .035,0,0,
                                                                .035,.2833*.1,0,
                                                                0,.2833*.1,0,
                                                                0,.2833*.1,0,
                                                                .035,.2833*.1,0,
                                                                .035,.2833*.2,0,
                                                                0,.2833*.2,0,
                                                                0,.2833*.2,0,
                                                                .035,.2833*.2,0,
                                                                .035,.2833*.3,0,
                                                                0,.2833*.3,0,
                                                                0,.2833*.3,0,
                                                                .035,.2833*.3,0,
                                                                .035,.2833*.4,0,
                                                                0,.2833*.4,0,
                                                                0,.2833*.4,0,
                                                                .035,.2833*.4,0,
                                                                .035,.2833*.5,0,
                                                                0,.2833*.5,0,
                                                                0,.2833*.5,0,
                                                                .035,.2833*.5,0,
                                                                .035,.2833*.6,0,
                                                                0,.2833*.6,0,
                                                                0,.2833*.6,0,
                                                                .035,.2833*.6,0,
                                                                .035,.2833*.7,0,
                                                                0,.2833*.7,0,
                                                                0,.2833*.7,0,
                                                                .035,.2833*.7,0,
                                                                .035,.2833*.8,0,
                                                                0,.2833*.8,0,
                                                                0,.2833*.8,0,
                                                                .035,.2833*.8,0,
                                                                .035,.2833*.9,0,
                                                                0,.2833*.9,0,
                                                                0,.2833*.9,0,
                                                                .035,.2833*.9,0,
                                                                .035,.2833*1.0,0,
                                                                0,.2833*1.0,0,
                                                                };

        femsclgrdverts=(float32 *)malloc((40*3)*sizeof(float32));
femsclgrdconts=(uint32 *)malloc((10*4)*sizeof(uint32));

        objClrSclGridref = (ECObjectReference *)args[1];
objClrSclGrid = ECReferenceObject(objClrSclGridref, &data.focus);

mstr=dStringFromOptions(NULL, &len, "femsclgrdMat", DS_END_OF_OPTIONS);

        material=VCMaterial_Create(mstr, VC_MATERIAL_ENABLE, black, black, black, grdcolor,
                                        white, NULL, NULL, NULL);
```

```
 if (!material) printf("Text: Failed to create material 'femsclgrdMat\n");

        femsclgrd_ent=VCEntity_Create(NULL, 0);

        // Create dynamic visual
 vc_vis = VCDynamicVisual_Create("femsclgrd_ent", 0);

 // Create lod
 vc_lod = VCDynamicVisual_AddLod(vc_vis,"#1", 0.0, -1, reference);

 // Create geogroup
 vc_ggrp = VCLod_AddGeogroup(vc_lod, VC_VERTEX_XYZ,

        0,0,VC_GEOGROUP_LOCK_OFF,VC_GEOGROUP_DRAWMODE_WIREFRAME,0,"femsclgrdMat", "femsclMat");

        for (i=0;i<40;i++)
        {
                femsclgrdverts[(i*3)+0]=femsclgrdxyz[(i*3)+0];
                femsclgrdverts[(i*3)+1]=femsclgrdxyz[(i*3)+1];
                femsclgrdverts[(i*3)+2]=femsclgrdxyz[(i*3)+2];
        }

        for (i=0;i<40;i++)
        {
                femsclgrdconts[i] = i;
        }

        cdata[0].type=VC_CONNECTIONLIST;
 cdata[0].faceCount=4;
        cdata[0].noConnections=10;
        cdata[0].data=femsclgrdconts;

        vc_geom = VCPmesh_Create(VC_VERTEX_XYZ, 40, (VCVertex) femsclgrdverts, 1, cdata);

        if(vc_geom != NULL)
          VCGeogroup_AttachGeometry(vc_ggrp,vc_geom);

        visual = ECObjectGetVisual(objClrSclGrid, NULL);

        if (visual == NULL)
         {
          VC_Error("visual was NULL\n");
          return(ECKeepAction);
         }

        attribute = ECVisualGetVCAttribute(visual);

        VCVisual_SetDynamicVisual(attribute,vc_vis);
        ECVisualToVC (objClrSclGrid, visual);
        ECObjectSetPosOrScale(objClrSclGrid,NULL,NULL,s);
        ECObjectToVC(objClrSclGrid);

        return(ECKeepAction);
}

//*************************************************************************//
// Function:  di_modify_ClrScl
//*************************************************************************//

int di_modify_ClrScl(void)
{

        ECVisual        *visual;
        VCAttribute      *attr;
        VCEntity        *entity;
        VCDynamicVisual   *dyn_vis;
        VCGeogroup       *dyn_geogrp;
        VCLod           *dyn_lod;
        VCGeometry       *dyn_geom;
        VCVertex_Reference ref;
```

```
        int             stat;
        VCDynamicVisual_Traverse traverse1;
        VCLod_Traverse          traverse2;
        VCGeogroup_Traverse     traverse3;
        int                 i,index;
        dmPoint             curvertpos;
         char                                *varNameFactor; /*Modification factor */
        char                                 *varFactor; /*Modification factor */
        float32                             clevel,dlevel,zerolevel;
        float32                             posmincolormod,negmincolormod;
        dmScale       s={0.85, 1.038, 1.00};//DAD


if(objClrScl == NULL)
 {
   VC_Error("Could not find object\n");
   return(ECKeepAction);
 }

 entity = ECObjectGetVCEntity(objClrScl);
 if(entity == NULL)
 {
   VC_Error("Could not find entity\n");
   return(ECKeepAction);
 }

 visual = ECObjectGetVisual(objClrScl, NULL);
 if(visual == NULL)
 {
   VC_Error("Could not find visual\n");
   return(ECKeepAction);
 }
 attr = ECVisualGetVCAttribute(visual);

// ECObjectSetPosOrScale(objClrScl,NULL,NULL,s);//DAD

 ECObjectToVC(objClrScl);

 VCVisual_GetDynamicVisual(attr,&dyn_vis);

 if(dyn_vis == NULL)
 {
   VC_Error("Could not find dynamic visual\n");
   return(ECKeepAction);
 }

 dyn_lod = VCDynamicVisual_GetFirstLod(dyn_vis, &traverse1);

 dyn_geogrp = VCLod_GetFirstGeogroup(dyn_lod,VC_VERTEX_RGBA,&traverse2);

 dyn_geom = VCGeogroup_GetFirstGeometry(dyn_geogrp,VC_PMESH,&traverse3);

 stat = VCGeometry_GetFirstVertex(dyn_geom,&ref);//vertex 0
           ref.data[0]=floats->femsclbotl[0];
  ref.data[1]=floats->femsclbotl[1];
  ref.data[2]=floats->femsclbotl[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 1
           ref.data[0]=floats->femsclbotr[0];
  ref.data[1]=floats->femsclbotr[1];
  ref.data[2]=floats->femsclbotr[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;
```

```
stat = VCGeometry_GetNextVertex(&ref);//vertex 2
        ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*floats->clrsclbot;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 3
        ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*floats->clrsclbot;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->outofrngcolor[0];
  ref.data[4]=vcfloats->outofrngcolor[1];
  ref.data[5]=vcfloats->outofrngcolor[2];
  ref.data[6]=floats->alphaoutrng;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 4
        ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*floats->clrsclbot;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->negmaxcolor[0];
  ref.data[4]=vcfloats->negmaxcolor[1];
  ref.data[5]=vcfloats->negmaxcolor[2];
  ref.data[6]=floats->alphainrng-.2;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 5
        ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*floats->clrsclbot;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->negmaxcolor[0];
  ref.data[4]=vcfloats->negmaxcolor[1];
  ref.data[5]=vcfloats->negmaxcolor[2];
  ref.data[6]=floats->alphainrng-.2;

                        if (floats->out_vals[2]<=0.0)//case 3
                        {
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                                clevel=max(min(1.0-floats->threshold,floats->clrscltop),floats->clrsclbot);
//below (commented out) gives a relative threshold percentage of color range
//                              clevel=floats->clrscltop-(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                                zerolevel=floats->clrscltop;
                                dlevel=floats->clrscltop;
                                negmincolormod=(floats->clrscltop-clevel)/(floats->clrscltop-floats->clrsclbot);
                                posmincolormod=0.0;
                        }
                        else if (floats->out_vals[0]>=0.0)//case 2
                        {
                                clevel=floats->clrsclbot;
                                zerolevel=floats->clrsclbot;
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                                dlevel=min(max(floats->threshold,floats->clrsclbot),floats->clrscltop);
//below (commented out) gives a relative threshold percentage of color range
//                              dlevel=floats->clrsclbot+(floats->threshold*(floats->clrscltop-floats->clrsclbot));
                                negmincolormod=0.0;
                                posmincolormod=(dlevel-floats->clrsclbot)/(floats->clrscltop-floats->clrsclbot);
                        }
                        else//case 1
                        {
                                zerolevel=fabs(floats->out_min)/(floats->out_max-floats->out_min);
//below shows absolute threshold value on color scale, which cannot exceed color range limits
                                clevel=min(max((zerolevel-(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrsclbot),zerolevel);
                                dlevel=max(min((zerolevel+(floats->threshold*max(zerolevel,1.0-zerolevel))),floats-
>clrscltop),zerolevel);
//below (commented out) gives a relative threshold percentage of color range
//                              clevel=zerolevel-(floats->threshold*(zerolevel-floats->clrsclbot));
//                              dlevel=zerolevel+(floats->threshold*(floats->clrscltop-zerolevel));
                                negmincolormod=(zerolevel-clevel)/(zerolevel-floats->clrsclbot);
```

```
                posmincolormod=(dlevel-zerolevel)/(floats->clrscltop-zerolevel);
            }


 stat = VCGeometry_GetNextVertex(&ref);//vertex 6
        ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*clevel;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->negmincolor[0]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
   ref.data[4]=vcfloats->negmincolor[1]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
   ref.data[5]=vcfloats->negmincolor[2]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
   ref.data[6]=floats->alphainrng-.2;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 7
        ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*clevel;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->negmincolor[0]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[0]-vcfloats-
>negmincolor[0]);
   ref.data[4]=vcfloats->negmincolor[1]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[1]-vcfloats-
>negmincolor[1]);
   ref.data[5]=vcfloats->negmincolor[2]+
                                                    negmincolormod*
                                                    (vcfloats->negmaxcolor[2]-vcfloats-
>negmincolor[2]);
   ref.data[6]=floats->alphainrng-.2;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 8
        ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*clevel;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 9
        ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*clevel;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 10
        ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*zerolevel;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->negthreshcolor[0];
   ref.data[4]=vcfloats->negthreshcolor[1];
   ref.data[5]=vcfloats->negthreshcolor[2];
   ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 11
        ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*zerolevel;
```

```
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->negthreshcolor[0];
  ref.data[4]=vcfloats->negthreshcolor[1];
  ref.data[5]=vcfloats->negthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 12
          ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*zerolevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 13
          ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*zerolevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 14
          ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*dlevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 15
          ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*dlevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posthreshcolor[0];
  ref.data[4]=vcfloats->posthreshcolor[1];
  ref.data[5]=vcfloats->posthreshcolor[2];
  ref.data[6]=floats->alphathresh;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 16
          ref.data[0]=floats->femscltopl[0];
  ref.data[1]=floats->femscltopl[1]*dlevel;
  ref.data[2]=floats->femscltopl[2];
  ref.data[3]=vcfloats->posmincolor[0]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
  ref.data[4]=vcfloats->posmincolor[1]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
  ref.data[5]=vcfloats->posmincolor[2]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
  ref.data[6]=floats->alphainmg-.2;

 stat = VCGeometry_GetNextVertex(&ref);//vertex 17
          ref.data[0]=floats->femscltopr[0];
  ref.data[1]=floats->femscltopr[1]*dlevel;
  ref.data[2]=floats->femscltopr[2];
  ref.data[3]=vcfloats->posmincolor[0]+
                                                    posmincolormod*
                                                    (vcfloats->posmaxcolor[0]-vcfloats-
>posmincolor[0]);
  ref.data[4]=vcfloats->posmincolor[1]+
                                                    posmincolormod*
```

```
                                                        (vcfloats->posmaxcolor[1]-vcfloats-
>posmincolor[1]);
   ref.data[5]=vcfloats->posmincolor[2]+
                                                        posmincolormod*
                                                        (vcfloats->posmaxcolor[2]-vcfloats-
>posmincolor[2]);
   ref.data[6]=floats->alphainrng-.2;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 18
            ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*floats->clrscltop;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->posmaxcolor[0];
   ref.data[4]=vcfloats->posmaxcolor[1];
   ref.data[5]=vcfloats->posmaxcolor[2];
   ref.data[6]=floats->alphainrng-.2;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 19
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*floats->clrscltop;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->posmaxcolor[0];
   ref.data[4]=vcfloats->posmaxcolor[1];
   ref.data[5]=vcfloats->posmaxcolor[2];
   ref.data[6]=floats->alphainrng-.2;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 20
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1]*floats->clrscltop;
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->outofrngcolor[0];
   ref.data[4]=vcfloats->outofrngcolor[1];
   ref.data[5]=vcfloats->outofrngcolor[2];
   ref.data[6]=floats->alphaoutrng;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 21
            ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1]*floats->clrscltop;
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->outofrngcolor[0];
   ref.data[4]=vcfloats->outofrngcolor[1];
   ref.data[5]=vcfloats->outofrngcolor[2];
   ref.data[6]=floats->alphaoutrng;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 22
            ref.data[0]=floats->femscltopr[0];
   ref.data[1]=floats->femscltopr[1];
   ref.data[2]=floats->femscltopr[2];
   ref.data[3]=vcfloats->outofrngcolor[0];
   ref.data[4]=vcfloats->outofrngcolor[1];
   ref.data[5]=vcfloats->outofrngcolor[2];
   ref.data[6]=floats->alphaoutrng;

  stat = VCGeometry_GetNextVertex(&ref);//vertex 23
            ref.data[0]=floats->femscltopl[0];
   ref.data[1]=floats->femscltopl[1];
   ref.data[2]=floats->femscltopl[2];
   ref.data[3]=vcfloats->outofrngcolor[0];
   ref.data[4]=vcfloats->outofrngcolor[1];
   ref.data[5]=vcfloats->outofrngcolor[2];
   ref.data[6]=floats->alphaoutrng;

            VCGeometry_Flush(dyn_geom);
}

//*************************************************************************//
// Function: di_intersect_handler
//*************************************************************************//

int
```

```
di_intersect_handler(VCBodyScreenIntersection_CallbackData *callbackData, void *data)
{
  int             numIntersections;

  if (callbackData == NULL)
        {
    printf("di_intersect_handler : callbackData NULL; exiting handler\n");
    return(ECKeepAction);
  }

  if(VCIntersection_Get(callbackData->intersection, NULL, &intersectionReportData, &numIntersections, NULL) != VC_OK)
        {
                VC_Error("dvObjectIntersectFunc : VCIntersection_Get returned VC_ERR\n");
                return(ECKeepAction);
        }

        if (intersectionReportData)
        {
                di_FEM_interact();
        }
        return (ECKeepAction);
}

//*************************************************************************//
// Function:  di_FEM_interact - performs operations based on where the
//                              mouse button intersected with the model.
//*************************************************************************//

int di_FEM_interact()
{
        static      VCEntity  *graysphere;
        static      VCEntity  *bluesphere;
        static      VCAttribute         *v=NULL;
        static      VCAttribute         *w=NULL;
        static  VCAttribute *a=NULL;

  VCAttribute           *int_attribute = NULL;
  VCEntity          *entity;
  VCEntity          *parent;
        VCEntity                        *FEMent;
        VCEntity                        *Meshent;
        VCEntity                        *ClrSclent;//temp
  dmPoint           p;
  dmEuler           e;
  dmScale           s;
  dmMatrix          mat,inv_mat,matp,cur_mat;
        dmVector                        intvect1, intvect2;
  VCDynamicVisual     *dvis;
  VCVertex_Reference    ref;
  dmPoint           orgndpt1, orgndpt2, nodep1, nodep2;
  int             numIntersections;
  int             ii = 0, i, j, k, rightindex;
///  int              rightvert, rightelem, adjindex;
        dmEuler                                 o;
        float32                                 badcum, badrec;
        float32                                 anglerec, anglecalc, anglesum, angledif;
        float32                                         intrec, lengthrec, intdist, length;
        dmVector                                sidevect;
        ECVisual                        *visual1,*visual2;
  VCAttribute     *attribute1,*attribute2;
        EntityList              *picked_load = NULL, *picked_constr = NULL;
//      Load                            *p = NULL;
        char                                    *loadtype = "initialization", *name = "initialization", *cname =
"initialization";
        int                                             r, type, node, cnode, face[6], value[6], dof_flag[6];

// Get the attribute of the intersected object
        int_attribute = intersectionReportData->visual;
        if ( (entity = intersectionReportData->entity)==NULL) return(ECKeepAction);
```

```
        if (intersectionReportData->point)
        {
                VCEntity_GetPositionPointEulerScale(entity,p,e,s);
                dmMatFromPointEulerScale(mat,p,e,s);

//              visual1 = ECObjectGetVisual(LoadList->nodeobj, NULL);
//              attribute1 = ECVisualGetVCAttribute(visual1);

        if(ConstrList != NULL)
                {
                        for(picked_constr = ConstrList; picked_constr != NULL; picked_constr = picked_constr->next)
                        {
                                if (entity == picked_constr->nodeobj)
                                {
                                        strcpy(cname, CONSTRAINT_SET[CONSTRAINTSET_PICK].B);
                                        cnode = CONSTRAINT_SET[CONSTRAINTSET_PICK].ID[ii];

                                        for (r = 0; r < 6; r++)
                                                dof_flag[r] =
CONSTRAINT_SET[CONSTRAINTSET_PICK].INDEX[ii*6+r];

                                        printf("\n");
                                        sprintf(chars->outtxt,"Constr Set Name:\n  %s\nConstraint Node: %d\nTrans
X = %d\nTrans Y = %d\nTrans Z = %d\nRot X = %d\nRot Y = %d\nRot Z = %d\n",
                                                cname, cnode, dof_flag[0], dof_flag[1], dof_flag[2], dof_flag[3],
dof_flag[4], dof_flag[5]);

                                        VCString_SetText(femtextstring, chars->outtxt);

                                        break;
                                }
                                ii++;
                        }
                }

                /*************************************************************************/

                ii = 0;
                if(LoadList != NULL)
                {
                        for(picked_load = LoadList; picked_load != NULL; picked_load = picked_load->next)
                        {
                                if (entity == picked_load->nodeobj)
                                {
                                        strcpy(name, LOAD_SET[LOADSET_PICK].NAME);
                                        type = LOAD_SET[LOADSET_PICK].TYPE[ii];
                                        switch(type)
                                        {
                                                case 1:     strcpy(loadtype, "Nodal Force");
                                                        break;
                                                case 2: strcpy(loadtype, "Nodal Displacement");
                                                        break;
                                                case 3: strcpy(loadtype, "Nodal Accel");
                                                        break;
                                                case 5: strcpy(loadtype, "Nodal Heat Generation");
                                                        break;
                                                case 6: strcpy(loadtype, "Nodal Heat Flux");
                                                        break;
                                                case 7: strcpy(loadtype, "Velocity");
                                                        break;
                                                case 8: strcpy(loadtype, "Nonlinear Transient");
                                                        break;
                                                case 10: strcpy(loadtype, "Distributed Line Load");
                                                        break;
                                                case 11: strcpy(loadtype, "Element Face Pressure");
                                                        break;
                                                case 13: strcpy(loadtype, "Element Heat Generation");
                                                        break;
                                                case 14: strcpy(loadtype, "Element Heat Flux");
                                                        break;
```

```
                                                case 15: strcpy(loadtype, "Element Convection");
                                                        break;
                                                case 16: strcpy(loadtype, "Element Radiation");
                                                        break;
                                        }

                                        node = LOAD_SET[LOADSET_PICK].ID[ii];

                                        for (r = 0; r < 6; r++){
                                                face[r] = LOAD_SET[LOADSET_PICK].FACE[ii*6+r];
                                                value[r] = LOAD_SET[LOADSET_PICK].VALUE[ii*8+2+r];
                                        }

                                        sprintf(chars->outtxt,"Load Set Name:\n  %sLoad Type: %s\nLoad Node:
%d\nTrans x value = %d\nTrans y value = %d\nTrans z value = %d\nRot x value = %d\nRot y value = %d\nRot z value = %d\n",
                                                name, loadtype, node, value[0], value[1], value[2], value[3],
value[4], value[5]);
                                        VCString_SetText(femtextstring,chars->outtxt);
                                        //return;
                                        break;
                                }
                                ii++;
                        }
                }

        FEMent=ECObjectGetVCEntity (objFEM);
        Meshent=ECObjectGetVCEntity (objMesh);
        if ((entity == FEMent)||(entity == Meshent))
                {

                        if((parent = entity->parent)!=NULL)
                        {
                                VCEntity_GetPositionPointEulerScale(parent,p,e,s);
                                dmMatFromPointEulerScale(matp,p,e,s);
                                dmMatMult(cur_mat,mat,matp);
                        }
                        else dmMatCopy(cur_mat,mat);

                        dmMatInvert(inv_mat,cur_mat);

                        anglerec=360.0;
                        pmi->rightvert=0;

// Element closure angle test
//  to determine correct element intersected by intersection point
// Dryer - 8/97

                        for (i = 0; i < ELEMENT_NUM; i++)
                        {

                                {
                                        anglesum=0.0;

                                        for (j = 0; j < elearray[i * 5]; j++)
                                        {
                                                k=j+1;
                                                if (j==((elearray[i*5])-1)) k=0;

                                                dmPointSet (orgndpt1,
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+0])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+1])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(j+1))])*7)+2])+

        ((displaceobj[((elearray[((i*5)+(j+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                                dmPointXformMat(nodep1,orgndpt1,cur_mat);
```

```
                                                dmPointSub (intvect1, intersectionReportData->point, nodep1);

                                                dmPointSet (orgndpt2,
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+0])+

        ((displaceobj[((elearray[((i*5)+(k+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+1])+

        ((displaceobj[((elearray[((i*5)+(k+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((i*5)+(k+1))])*7)+2])+

        ((displaceobj[((elearray[((i*5)+(k+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                                dmPointXformMat(nodep2,orgndpt2,cur_mat);
                                                dmPointSub (intvect2, intersectionReportData->point, nodep2);

                                                anglecalc=(180.0/3.14159251)*

        acos(((intvect1[0]*intvect2[0])+(intvect1[1]*intvect2[1])+(intvect1[2]*intvect2[2]))/

        ((sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2])))*

        (sqrt((intvect2[0]*intvect2[0])+(intvect2[1]*intvect2[1])+(intvect2[2]*intvect2[2]))));

                                                if (     fabs(1.0-
(((intvect1[0]*intvect2[0])+(intvect1[1]*intvect2[1])+(intvect1[2]*intvect2[2]))/

        ((sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2])))*

        (sqrt((intvect2[0]*intvect2[0])+(intvect2[1]*intvect2[1])+(intvect2[2]*intvect2[2])))))
                                                                < .000001 )
                                                        anglecalc=0.0;


                                                        anglesum=anglesum+anglecalc;
                                                }
                                                angledif = fabs(360.0-anglesum);
                                                if (angledif<anglerec)
                                                {
                                                        anglerec=angledif;
                                                        pmi->rightelem=i;
                                                }
                                        }
                                }

//Now that correct element is identified,
// Determine nearest vertex to intersection point (pmi->rightvert) in identified element and
// min side length (lengthrec) for sphere marker scaling
//Dryer - 8/97
                                intrec=10000.0;
                                lengthrec=10000.0;

                                for (j=0; j<elearray[pmi->rightelem*5]; j++)
                                {
//Set up variables
                                        k=j+1;
                                        if (j==((elearray[pmi->rightelem*5])-1)) k=0;

                                        dmPointSet (orgndpt1,
                                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+0])+
                                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+1])+
                                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                        (vertices[((elearray[((pmi->rightelem*5)+(j+1))])*7)+2])+
                                                                ((displaceobj[((elearray[((pmi-
>rightelem*5)+(j+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                        dmPointXformMat(nodep1,orgndpt1,cur_mat);
```

```
                                        dmPointSet (orgndpt2,
                                                (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+0])+
                                                        ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+0])*floats->LoadFactor*floats->exager),
                                                (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+1])+
                                                        ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+1])*floats->LoadFactor*floats->exager),
                                                (vertices[((elearray[((pmi->rightelem*5)+(k+1))])*7)+2])+
                                                        ((displaceobj[((elearray[((pmi-
>rightelem*5)+(k+1))])*3)+2])*floats->LoadFactor*floats->exager));

                                        dmPointXformMat(nodep2,orgndpt2,cur_mat);


//Test for nearest element node
                                        dmPointSub (intvect1, intersectionReportData->point, nodep1);

        intdist=sqrt((intvect1[0]*intvect1[0])+(intvect1[1]*intvect1[1])+(intvect1[2]*intvect1[2]));
                                        if (intdist<intrec)
                                        {
                                                intrec=intdist;
                                                pmi->rightvert=elearray[(pmi->rightelem*5)+(j+1)];
                                                points->rightnodep[0]=nodep1[0];
                                                points->rightnodep[1]=nodep1[1];
                                                points->rightnodep[2]=nodep1[2];
                                                rightindex = j;
                                        }
// adjust rightindex for beam elements
                                        if (((ELEMENT_P+pmi->rightelem)->A == 2) && ((rightindex == 0) || (rightindex ==
1)))
                                                pmi->adjindex = 0;
                                        else if (((ELEMENT_P+pmi->rightelem)->A == 2) && ((rightindex == 3) || (rightindex
== 2)))
                                                pmi->adjindex = 1;
                                        else
                                                pmi->adjindex = rightindex;


//Calculate max element side length for sphere marker scaling
                                        dmPointSub (sidevect, nodep1, nodep2);

        length=sqrt((sidevect[0]*sidevect[0])+(sidevect[1]*sidevect[1])+(sidevect[2]*sidevect[2]));
                                        if (length<lengthrec) lengthrec=length;

                                }

                                if ((ELEMENT_P+pmi->rightelem)->A == 2)
                                {
                                        lengthrec=lengthrec*(floats->beamdelta/10);
                                }


//Dryer: used to see which vertex is being selected
//                              outvert[pmi->rightvert]=out_max;

        sprintf(chars->outtxt,"%s\nNode #: %i\nElement #: %i\n\n%s%10.6f\nDX: %10.6f\nDY: %10.6f\nDZ: %10.6f\n",
                                        names->actual_case_name,
                                        (NODE_P+((ELEMENT_P+pmi->rightelem)->B[pmi->adjindex]))->A,
                                        (ELEMENT_P+pmi->rightelem)->D,
                                        names->actual_set_name[(switches->outtypenum*5)+switches->outsubnum],
                                        outvert[pmi->rightvert]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+0]*floats->LoadFactor,
                displaceobj[(pmi->rightvert*3)+1]*floats->LoadFactor,
                                        displaceobj[(pmi->rightvert*3)+2]*floats->LoadFactor);

// create graysphere
                                graysphere = VCEntity_Create(NULL, 0);
                                if (VCAttribute_Delete (v) != 0)
                                        VC_Error ("Error cannot destroy attribute\n");
```

```
                                v = VCVisual_CreateGeometry ("graysphere");
                                VCEntity_AttachAttribute (graysphere, v);
                                VCEntity_Scale(graysphere, (lengthrec/7.0), (lengthrec/7.0), (lengthrec/7.0));
                                VCEntity_SetPositionPoint(graysphere,intersectionReportData->point);

// create bluesphere
                                bluesphere = VCEntity_Create(NULL, 0);
                                if (VCAttribute_Delete (w) != 0)
                                        VC_Error ("Error cannot destroy attribute\n");

                                w = VCVisual_CreateGeometry ("bluesphere");
                                VCEntity_AttachAttribute (bluesphere, w);
                                VCEntity_Scale(bluesphere, (lengthrec/5.0), (lengthrec/5.0), (lengthrec/5.0));
                                VCEntity_SetPositionPoint(bluesphere,points->rightnodep);

                                if (VCAttribute_Delete (a) != 0)
                                        VC_Error ("Error cannot destroy audio attribute\n");
                                a = VCEntity_AddAudioVoice (bluesphere, "explosion");
        if (a == NULL)
                                {
            VC_Error ("Cannot create audio instance\n");
                                }
                                else
                                {
                                        /* Play the audio voice */
         VCAudio_Start (a);

         /* Change the loop count to infinity, set to highest priority
           and play */
         VCAudio_SetLoopCount (a, 1);
         VCAudio_SetPriority (a, VC_AUDIO_PRIORITY_LOCKED);
         VCAudio_Start (a);
                                }

                                di_modify_FEM();
                                if (switches->meshdynmode==1) di_modify_Mesh();
                                di_modify_LoadSet();
                                di_modify_ConstraintSet();
                                switches->picknode=1;//picknode
                                }
                        else
                        {
//                              sprintf(chars->outtxt,"%s","No selection");
                        }
                        VCString_SetText(femtextstring,chars->outtxt);
            }
   return;
}

//*************************************************************************//
// Function: di_create_body_handler
// Comments: Dryer added di_create_body_handler
//*************************************************************************//

int
di_create_body_handler(VCBodyCreate_CallbackData *bodyData, void *data)
{
            VCBody *body = bodyData->body;

   VCBody_AttachScreenIntersectionCallback(body, NULL, di_intersect_handler, NULL);

 return;
}

//*************************************************************************//
// Function: ObjectIntersectedCallback
//*************************************************************************//

int
```

```
ObjectIntersectedCallback(VCIntersection_CallbackData *cdata, void *data)
{

   intersectionReportData = VCIntersection_GetFirstIntersectionReport(cdata->intersection, NULL);

   if(intersectionReportData == NULL)
   {
      return;
   }

            di_FEM_interact();
}

//*************************************************************************//
// Function:  diImmersDataFunc
//*************************************************************************//

int
diImmersDataFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void          **args = action->parameters;
   char          *part = NULL;
   VCAttribute      *vcLimb;
   ECObjectReference *ref;
   VCPositionData    pos;
   dmMatrix          handMat;
   dmPoint      pt;
   dmEuler      ori;
   dmScale      scaledm;
   VC_Traverse       traverseInfo;

   float32           length = 50.0;
   uint32            newMask = 0x10;
   uint32            oldMask;
            intersectArgs intersectData;
            VCEntity    *hitEntity;
            VCAttribute  *intersection;
   VCIntersection *intersectionData;

   // Fix
   pos.mode = 0;

   if(args[1] != NULL)
      part = (char *)args[1];
   else
      part = "hand";

   intersectData.event = (uint32 *)args[2];

   ref = (ECObjectReference *)args[3];
   intersectData.object = ECReferenceObject(ref, &data.focus);

   // Is there a body?
   if(data.body)
   {  // Get limb position
      vcLimb = VCBody_GetBodyPart(data.body, part);
   }
   else
   {  // Get limb position
      vcLimb = VCBody_GetBodyPart(VC_GetFirstBody(&traverseInfo), part);
   }

   if (vcLimb == NULL)
   {
      VC_Error("dvObjectIntersectFunc: Didn't get limb %s.\n", part);
      return(ECKeepAction);
   }

   // Calculate hand matrix from position
   if(VCEntity_GetAbsolutePosition(vcLimb->first, handMat) != VC_OK)
```

```
{
  VC_Error("dvObjectIntersectFunc : VCEntity_GetAbsolutePosition returned VC_ERR\n");
  return(ECKeepAction);
}
// First time?
if(args[0]==NULL)
{
        int *pInt = (int *)malloc(sizeof(int));
        args[0] = pInt;

  // Get the hand intersection mask
  if(VCVectorIntersect_GetIntersectMask(vcLimb, &oldMask) != VC_OK)
  {
    VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersectMask returned VC_ERR\n");
    return(ECKeepAction);
  }
  // Stop the vector from intersecting the limb named,
  //  by setting its mask value to the same
  if(VCVectorIntersect_ModifyIntersectMask (vcLimb, NULL, oldMask) != VC_OK)
  {
    VC_Error("dvObjectIntersectFunc : VCVectorIntersect_ModifyIntersectMask returned VC_ERR\n");
    return(ECKeepAction);
  }
  // Get point, orientation and scale
  dmPointEulerScaleFromMat(pt, ori, scaledm, handMat);
  // Get a position from above
  if(VCPosition_MakePointEulerScale (&pos, pt, ori, scaledm) != VC_OK)
  {
    VC_Error("dvObjectIntersectFunc : VCPosition_MakePointEulerScale returned VC_ERR\n");
    return(ECKeepAction);
  }
  // Define the hit entity
  hitEntity = VCEntity_Create(&pos, NULL);
  // Evaluate intersection
  intersection = VCVectorIntersect_Create(VC_VECTORINTERSECT_ENABLE,
                        length, newMask, 1);
  if(intersection == NULL)
  {
    VC_Error("dvObjectIntersectFunc : intersection is NULL\n");
    return(ECKeepAction);
  }
  // Attach vector intersect to an entity
  if(VCEntity_AttachAttribute(hitEntity, intersection) == VC_ERR)
  {
    VC_Error("dvObjectIntersectFunc : could not attach vector intersect to entity\n");
    return(ECKeepAction);
  }

  // Get vector intersection.
  if(VCVectorIntersect_GetIntersection(intersection, &intersectionData) != VC_OK)
  {
    VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersection returned VC_ERR\n");
    return(ECKeepAction);
  }

  // Add intersection udpate handler.
  if (VCIntersection_AttachUpdateCallback(intersectionData, ObjectIntersectedCallback,
                        (void *)&intersectData) == NULL)
  {
    VC_Error("dvObjectIntersectFunc : Failed to add intersection update handler.\n");
    return(ECKeepAction);
  }

  // Set back the hand intersect mask
  if(VCVectorIntersect_ModifyIntersectMask (vcLimb, oldMask, NULL) != VC_OK)
  {
    VC_Error("dvObjectIntersectFunc : VCVectorIntersect_ModifyIntersectMask returned VC_ERR\n");
    return(ECKeepAction);
  }
```

```
    }
    else
    {
      // Get the hand intersect mask
      if(VCVectorIntersect_GetIntersectMask (vcLimb, &oldMask) != VC_OK)
      {
        VC_Error("dvObjectIntersectFunc : VCVectorIntersect_GetIntersectMask returned VC_ERR\n");
        return(ECKeepAction);
      }

      if(VCVectorIntersect_ModifyIntersectMask (vcLimb, NULL, oldMask) != VC_OK)
      {
        VC_Error("dvObjectIntersectFunc : VCVectorIntersect_SetIntersectMask returned VC_ERR\n");
        return(ECKeepAction);
      }
      // Evaluate new hitEntity
      if(VCEntity_SetPositionMatrix (hitEntity, handMat) != VC_OK)
      {
        VC_Error("dvObjectIntersectFunc : VCEntity_SetPositionMatrix returned VC_ERR\n");
        return(ECKeepAction);
      }
      // Evaluate new intersection
      if(VCVectorIntersect_Set(intersection, VC_VECTORINTERSECT_ENABLE, NULL, &length,
                  newMask, NULL, NULL) != VC_OK)
      {
        VC_Error("dvObjectIntersectFunc : VCVectorIntersect_Set returned VC_ERR\n");
        return(ECKeepAction);
      }
      // Set back the hand intersect mask
      if(VCVectorIntersect_ModifyIntersectMask (vcLimb, oldMask, NULL) != VC_OK)
      {
        VC_Error("dvObjectIntersectFunc : VCVectorIntersect_SetIntersectMask returned VC_ERR\n");
        return(ECKeepAction);
      }
    }

    return(ECKeepAction);
}

//*************************************************************************//
// Function:  di_animTimer
//*************************************************************************//

int
di_animTimer(VCTimer_CallbackData *callbackData,  void *data)
{
int i;
        if (switches->startanim==1)
        {
        // FEM animation floats->LoadFactor 0.0 to 1.0
                for (i=0; i<100; i++)//loop for sawtooth animation
                {
                        floats->LoadFactor=i/100.0;

                        di_modify_FEM();
                        if (switches->meshdynmode==1) di_modify_Mesh();
                        di_modify_LoadSet();
                        di_modify_ConstraintSet();
                }
                if (switches->animmode==0)//turn on loop for ramp animation
                {
                        for (i=100; i>0; i--)
                        {
                                floats->LoadFactor=i/100.0;

                                di_modify_FEM();
                                if (switches->meshdynmode==1) di_modify_Mesh();
                                di_modify_LoadSet();
                                di_modify_ConstraintSet();
                        }
```

```
                }
        }
}

//************************************************************************//
// Function: di_animalarm
//************************************************************************//

int
di_animalarm(VCTimer_CallbackData *cd, void *data)
{
        void *animHandle=data;

        if (switches->startanim==0)
        {
                VCTimer_DetachCallback(animHandle);
        }
        else
        {
                if (animHandle)
                {
                        VCTimer_DetachCallback(animHandle); /* stop the animation */
  /* and re-run this function in one seconds time to restart animation */
                        VCTimer_AttachExpiringCallback(1, di_animalarm, NULL);
                }
                else
                {       /* data is NULL, so restart animation */
                        animHandle = VCTimer_AttachPeriodicCallback(100.0/100.0, di_animTimer, NULL);
                        if (!animHandle)
                                printf("Failed to restart animation\n");
                        else
                                VCTimer_AttachExpiringCallback(1, di_animalarm, animHandle);
                }
        }
}

//************************************************************************//
// Function: diToggleAnimFunc
//************************************************************************//

int diToggleAnimFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->startanim, &data.focus) == VC_ERR)
                switches->startanim = -1;

        if (switches->startanim==1)
        {
                void *animHandle;

                animHandle = VCTimer_AttachPeriodicCallback (100.0/100.0, di_animTimer, NULL);

                VCTimer_AttachExpiringCallback(1, di_animalarm, animHandle);
        }
}

//************************************************************************//
// Function: diBodyStartupPosFEMFunc - Sets the zone startup body position
//************************************************************************//

int
diBodyStartupPosFEMFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  dmPoint  s;
  VCBody     *thisBody;
  VCBody     *body = data.body;
  VC_Traverse  traverseInfo;
  void      **args = action->parameters;
  dmMatrix tempMat; /* Get original body position */
        float32     tempX;
```

```
        float32     tempY;
        float32     tempZ;

        /* Is there a body? */
  if (body == NULL)
    body = VC_GetFirstBody(&traverseInfo);

  if(body != NULL)
  {
                VCBody_GetAbsolutePosition (body, tempMat);
                dmPointFromMat(s, tempMat);

//STARTUP HOME (FRONT) VIEW
                tempX = points->FEMcenter[VC_X];
                tempY = points->FEMcenter[VC_Y];
                tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4.0));

                s[VC_X] = tempX;
                s[VC_Y] = tempY;
                s[VC_Z] = tempZ;

   /* Set the current body startup position */
   VCBody_SetPosition(body, NULL, s, NULL, NULL, NULL, NULL);
  }

  /* Accomodate for NULL values and no body */
        s[VC_X] = tempX;
        s[VC_Y] = tempY;
  s[VC_Z] = tempZ;

        /* Set the Global body position */
  if(body != NULL)
    ECZoneSetBodyStartupPosition(ECBodyGetZone(body), s);
  else
    ECZoneSetBodyStartupPosition(ECTopZoneGet(), s);

  return(ECKeepAction);
}

//*************************************************************************//
// Function: diNavModeFunc
//*************************************************************************//

int diNavModeFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void                **args = action->parameters;
        if(ECArgReferenceGetValue(args[1], (void *)&switches->navmode, &data.focus) == VC_ERR)//switches->navmode is
navmode
                switches->navmode = 1;
}

//*************************************************************************//
// Function: diSetViewFunc
//*************************************************************************//

int diSetViewFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void                **args = action->parameters;
        dmMatrix  tempMat;
        uint32              viewnum;
        VCBody      *body = data.body;


        if(ECArgReferenceGetValue(args[1], (void *)&viewnum, &data.focus) == VC_ERR)
                viewnum = 1;
                switch(viewnum)
        {
                case 1 :    //Set User View 1
                            switches->set1 = 1;//set1
                            VCBody_GetAbsolutePosition (body, tempMat);
```

```
                        dmPointFromMat(points->view1, tempMat);
                        break;
                case 2 :    //Set User View 2
                        switches->set2 = 1;//set2
                        VCBody_GetAbsolutePosition (body, tempMat);
                        dmPointFromMat(points->view2, tempMat);
                        break;
                default :   //Set User View 1
                        switches->set1 = 1;//set1
                        VCBody_GetAbsolutePosition (body, tempMat);
                        dmPointFromMat(points->view1, tempMat);
                        break;
        }
}

//*************************************************************************//
// Function:  diBodyMoveToFunc - Navigates (in different modes) the body
//          to a given viewpoint or position while orienting on the
//                              center of the FEM or other designated object center
//*************************************************************************//

int
diBodyMoveToFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        static          float32     lasty=0.0;
  float32       time;
  float32       elapsed;
  int32         done = 0;
  float32       rate = 1;
  float32       len;
  dmPoint       newPos;
        dmPoint         towards;
  MoveInfo      *mi;
  VCBody                *body = data.body;
  void          **args = action->parameters;
  VC_Traverse   traverseInfo;
  dmMatrix      tempMat;
        dmEuler         o;
        float32         xdegree,ydegree,zdegree;
        dmVector        orientVect;
        dmVector        adjvector;
        float32         tempX;
        float32         tempY;
        float32         tempZ;
        uint32          view=-1;
        float32         standoff=20.0;

  if(ECArgReferenceGetValue(args[1], (void *)&view, &data.focus) == VC_ERR)
                view=1;

        switch(view)
        {
                case 1 :    //TOP VIEW
                        tempX = points->FEMcenter[VC_X];
                        tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/4.0));
                        tempZ = points->FEMcenter[VC_Z];
                        towards[VC_X] = points->FEMcenter[VC_X];
                        towards[VC_Y] = points->FEMcenter[VC_Y];
                        towards[VC_Z] = points->FEMcenter[VC_Z];
                        break;
                case 2 :    //BACK VIEW
                        tempX = points->FEMcenter[VC_X];
                        tempY = points->FEMcenter[VC_Y];
                        tempZ = points->FEMcenter[VC_Z]-(floats->xyzmax-(floats->xyzmax/4.0));
                        towards[VC_X] = points->FEMcenter[VC_X];
                        towards[VC_Y] = points->FEMcenter[VC_Y];
                        towards[VC_Z] = points->FEMcenter[VC_Z];
                        break;
                case 3 :    //LEFT VIEW
                        tempX = points->FEMcenter[VC_X]-(floats->xyzmax-(floats->xyzmax/4.0));
```

```
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z];
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 4:                 //HOME (FRONT) VIEW
        tempX = points->FEMcenter[VC_X];
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 5 : //RIGHT VIEW
        tempX = points->FEMcenter[VC_X]+(floats->xyzmax-(floats->xyzmax/4.0));
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z];
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 6 : //ISOFRONTLEFT VIEW
        tempX = points->FEMcenter[VC_X]-(floats->xyzmax-(floats->xyzmax/2.0));
        tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/2.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 7 : //BOTTOM VIEW
        tempX = points->FEMcenter[VC_X];
        tempY = points->FEMcenter[VC_Y]-(floats->xyzmax-(floats->xyzmax/4.0));
        tempZ = points->FEMcenter[VC_Z];
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 8 : //ISOFRONTRIGHT VIEW
        tempX = points->FEMcenter[VC_X]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempY = points->FEMcenter[VC_Y]+(floats->xyzmax-(floats->xyzmax/2.0));
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/2.0));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
case 9 : //NODE VIEW
        if (switches->picknode == 1)
        {
                tempX = points->rightnodep[VC_X];
                tempY = points->rightnodep[VC_Y];
                tempZ = points->rightnodep[VC_Z];
                towards[VC_X] = points->rightnodep[VC_X];
                towards[VC_Y] = points->rightnodep[VC_Y];
                towards[VC_Z] = points->rightnodep[VC_Z];
        }
        else
        {  // Return early because no node selection
                args[0] = NULL;
                return(ECKeepAction);
        }
        break;
case 10 : //USER VIEW 1
        if (switches->set1 == 1)//set1
        {
                tempX = points->view1[VC_X];
                tempY = points->view1[VC_Y];
                tempZ = points->view1[VC_Z];
                towards[VC_X] = points->FEMcenter[VC_X];
                towards[VC_Y] = points->FEMcenter[VC_Y];
```

```
                towards[VC_Z] = points->FEMcenter[VC_Z];
        }
        else
        {  // Return early because points->view1 not set
                args[0] = NULL;
                return(ECKeepAction);
        }
        break;
case 11 :  //USER VIEW 2
        if (switches->set2 == 1)//set2
        {
                tempX = points->view2[VC_X];
                tempY = points->view2[VC_Y];
                tempZ = points->view2[VC_Z];
                towards[VC_X] = points->FEMcenter[VC_X];
                towards[VC_Y] = points->FEMcenter[VC_Y];
                towards[VC_Z] = points->FEMcenter[VC_Z];
        }
        else
        {  // Return early because points->view2 not set
                args[0] = NULL;
                return(ECKeepAction);
        }
        break;
default:   //HOME (FRONT) VIEW
        tempX = points->FEMcenter[VC_X];
        tempY = points->FEMcenter[VC_Y];
        tempZ = points->FEMcenter[VC_Z]+(floats->xyzmax-(floats->xyzmax/4));
        towards[VC_X] = points->FEMcenter[VC_X];
        towards[VC_Y] = points->FEMcenter[VC_Y];
        towards[VC_Z] = points->FEMcenter[VC_Z];
        break;
}

        switch(switches->navmode) //navmode
{
        case 1 :   //fast/hyper mode (orient on FEM center)
        // Is there a body?
        if (body == NULL)
                body = VC_GetFirstBody(&traverseInfo);

        if ((mi = args[0]) == NULL) // first call
        {
                args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                dmPointSet (mi->posa,tempX,tempY,tempZ);

                rate = 400.0;

        // Setup move information parameters
                mi->body = body;

                if (body != NULL)
                {
                        VCBody_GetAbsolutePosition (body, tempMat);
                        dmPointFromMat(mi->bodyOffset, tempMat);
                }
                else
                {
                        mi->bodyOffset[VC_X] = 0.0;
                        mi->bodyOffset[VC_Y] = 0.0;
                        mi->bodyOffset[VC_Z] = 0.0;
                }

                if (view == 9)
                {
                        dmPointSub (adjvector, mi->bodyOffset, towards);
                        adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                        (adjvector[1]*adjvector[1])+
                                                        (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
```

```
                                adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                dmPointAddVector (mi->posa, mi->posa, adjvector);
                        }

                        mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                        mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                        mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                        len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                                mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                                mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                        if(len != 0)
                        {
                                rate /= len;
                        }
                        else
                        {  // Return early because zero distance to move
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        mi->velocity[VC_X] *= rate;
                        mi->velocity[VC_Y] *= rate;
                        mi->velocity[VC_Z] *= rate;
                        mi->time = -1.f;
                        if(rate != 0)
                        {
                                mi->totalTime = 1.f / rate;
                        }
                        else
                        {  // Return early because zero speed entered
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
                }
// Added this so that we use the time in the zone
//  where the body is.
                time = ECZoneGetTime(ECBodyGetZone(body));
                if (mi->time== -1.f)
                {
                mi->time=time;
                elapsed=0.f;
                }
                else
                {
                        elapsed=time-mi->time;
                }

                if (elapsed < mi->totalTime)
                {
    // Animate body
                newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
                newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
                newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
                //Update orientation to towards (FEM center or node (for node view))
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
```

```
                                        (dmSafeAtan2 (orientVect[2], orientVect[0])));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                return(ECKeepAction);///added

                }
                else
                {
    // Move body to final position
                newPos[VC_X] = mi->posa[VC_X];
                newPos[VC_Y] = mi->posa[VC_Y];
                newPos[VC_Z] = mi->posa[VC_Z];

//Update final orientation to towards (FEM center or node (node view))
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                (orientVect[1]*orientVect[1])+
                                (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                        done = 1;
                }

                if(mi->body != NULL)
                {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
                }
                else
                {
                        VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
                }

                if(done == 1)
                {
    // Clean up
                        free(mi);
                        args[0] = NULL;
                        return(ECRemoveAction);
                }
```

```
                return(ECKeepAction);
                break;


                case 2 :    //straight line fly move (orient on FEM center)
                switches->navstate=0;
                // Is there a body?
                if (body == NULL)
                        body = VC_GetFirstBody(&traverseInfo);

                if ((mi = args[0]) == NULL) // first call
                {
                        args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                        dmPointSet (mi->posa,tempX,tempY,tempZ);

    // Extract user parameters
                        if(ECArgReferenceGetValue(args[2], (void *)&rate, &data.focus) == VC_ERR)
                        rate = 4.0;
    // Setup move information parameters
                        mi->body = body;

                        if (body != NULL)
                        {
                                VCBody_GetAbsolutePosition (body, tempMat);
                                dmPointFromMat(mi->bodyOffset, tempMat);
                        }
                        else
                        {
                                mi->bodyOffset[VC_X] = 0.0;
                                mi->bodyOffset[VC_Y] = 0.0;
                                mi->bodyOffset[VC_Z] = 0.0;
                        }

                        if (view == 9)
                        {
                                dmPointSub (adjvector, mi->bodyOffset, towards);
                                adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                dmPointAddVector (mi->posa, mi->posa, adjvector);
                        }

                        mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                        mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                        mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                        len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                                mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                                mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                        if(len != 0)
                        {
                                rate /= len;
                        }
                        else
                        {  // Return early because zero distance to move
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        mi->velocity[VC_X] *= rate;
                        mi->velocity[VC_Y] *= rate;
                        mi->velocity[VC_Z] *= rate;
                        mi->time = -1.f;
```

```
                        if(rate != 0)
                        {
                                mi->totalTime = 1.f / rate;
                        }
                        else
                        {  // Return early because zero speed entered
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
                }
// Added this so that we use the time in the zone
//  where the body is.
                time = ECZoneGetTime(ECBodyGetZone(body));
                if (mi->time== -1.f)
                {
                mi->time=time;
                elapsed=0.f;
                }
                else
                {
                        elapsed=time-mi->time;
                }

                if (elapsed < mi->totalTime)
                {
    // Animate body
                newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
                newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
                newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
                //Update orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                }
                else
                {
     // Move body to final position
                 newPos[VC_X] = mi->posa[VC_X];
                 newPos[VC_Y] = mi->posa[VC_Y];
                 newPos[VC_Z] = mi->posa[VC_Z];

//Update final orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                        (asin(orientVect[1]/
                                                (sqrt((orientVect[0]*orientVect[0])+
```

```
                                        (orientVect[1]*orientVect[1])+
                                        (orientVect[2]*orientVect[2])));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                        (dmSafeAtan2 (orientVect[2], orientVect[0])));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);

                        done = 1;
                }

                if(mi->body != NULL)
                {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
                }
                else
                {
                        VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
                }

                if(done == 1)
                {
    // Clean up
                        free(mi);
                        args[0] = NULL;
                        switches->navstate=1;
                        return(ECRemoveAction);
                }
                return(ECKeepAction);
                break;

                default:    //straight line fly move (orient on FEM center)
                // Is there a body?
                if (body == NULL)
                        body = VC_GetFirstBody(&traverseInfo);

                if ((mi = args[0]) == NULL) // first call
                {
                        args[0]= mi=(MoveInfo *)malloc(sizeof(MoveInfo));
                        dmPointSet (mi->posa,tempX,tempY,tempZ);

    // Extract user parameters
                        if(ECArgReferenceGetValue(args[2], (void *)&rate, &data.focus) == VC_ERR)
                        rate = 4.0;
    // Setup move information parameters
                        mi->body = body;

                        if (body != NULL)
                        {
                                VCBody_GetAbsolutePosition (body, tempMat);
                                dmPointFromMat(mi->bodyOffset, tempMat);
                        }
                        else
                        {
                                mi->bodyOffset[VC_X] = 0.0;
                                mi->bodyOffset[VC_Y] = 0.0;
                                mi->bodyOffset[VC_Z] = 0.0;
                        }
```

```
                        if (view == 9)
                        {
                                dmPointSub (adjvector, mi->bodyOffset, towards);
                                adjvector[0]=(adjvector[0]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[1]=(adjvector[1]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                adjvector[2]=(adjvector[2]/sqrt((adjvector[0]*adjvector[0])+
                                                                (adjvector[1]*adjvector[1])+
                                                                (adjvector[2]*adjvector[2])))*(floats-
>xyzmax/standoff);
                                dmPointAddVector (mi->posa, mi->posa, adjvector);
                        }

                        mi->velocity[VC_X] = mi->posa[VC_X] - mi->bodyOffset[VC_X];
                        mi->velocity[VC_Y] = mi->posa[VC_Y] - mi->bodyOffset[VC_Y];
                        mi->velocity[VC_Z] = mi->posa[VC_Z] - mi->bodyOffset[VC_Z];
                        len=sqrt(mi->velocity[VC_X] * mi->velocity[VC_X]+
                                mi->velocity[VC_Y] * mi->velocity[VC_Y]+
                                mi->velocity[VC_Z] * mi->velocity[VC_Z]);
                        if(len != 0)
                        {
                                rate /= len;
                        }
                        else
                        {  // Return early because zero distance to move
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        mi->velocity[VC_X] *= rate;
                        mi->velocity[VC_Y] *= rate;
                        mi->velocity[VC_Z] *= rate;
                        mi->time = -1.f;
                        if(rate != 0)
                        {
                                mi->totalTime = 1.f / rate;
                        }
                        else
                        {  // Return early because zero speed entered
                                args[0] = NULL;
                                return(ECKeepAction);
                        }
                        ECZoneAddAnimateAction(ECBodyGetZone(body), event, action);
                }
// Added this so that we use the time in the zone
//  where the body is.
                time = ECZoneGetTime(ECBodyGetZone(body));
                if (mi->time== -1.f)
                {
                mi->time=time;
                elapsed=0.f;
                }
                else
                {
                        elapsed=time-mi->time;
                }

                if (elapsed < mi->totalTime)
                {
    // Animate body
                newPos[VC_X] = mi->bodyOffset[VC_X] + elapsed * mi->velocity[VC_X];
                newPos[VC_Y] = mi->bodyOffset[VC_Y] + elapsed * mi->velocity[VC_Y];
                newPos[VC_Z] = mi->bodyOffset[VC_Z] + elapsed * mi->velocity[VC_Z];
                //Update orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);
```

```
                        xdegree=(180.0/3.14159251)*
                                (asin(orientVect[1]/
                                        (sqrt((orientVect[0]*orientVect[0])+
                                (orientVect[1]*orientVect[1])+
                                (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);
                }
                else
                {
    // Move body to final position
                newPos[VC_X] = mi->posa[VC_X];
                newPos[VC_Y] = mi->posa[VC_Y];
                newPos[VC_Z] = mi->posa[VC_Z];

//Update final orientation towards FEM center
                        dmPointSub (orientVect, towards, newPos);

                        xdegree=(180.0/3.14159251)*
                                (asin(orientVect[1]/
                                        (sqrt((orientVect[0]*orientVect[0])+
                                (orientVect[1]*orientVect[1])+
                                (orientVect[2]*orientVect[2])))));

                        ydegree=-1.0*(90+((180.0/3.14159251)*
                                (dmSafeAtan2 (orientVect[2], orientVect[0]))));

                        if ((orientVect[2] < .00001 && orientVect[2] > -.00001)&&(orientVect[0] < .00001 &&
orientVect[0] > -.00001))
                        {

                                ydegree=lasty;
                        }
                        else
                        {
                                lasty = ydegree;
                        }

                        zdegree = 0.0;

                        dmEulerSetD(o,xdegree,ydegree,zdegree);

                        done = 1;
                }

                if(mi->body != NULL)
                {       VCBody_SetPosition(mi->body, NULL, newPos, o, NULL, NULL, NULL);
                }
                else
                {
                        VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, newPos, o, NULL, NULL, NULL);
                }
```

```
                    if(done == 1)
                    {
    // Clean up
                            free(mi);
                            args[0] = NULL;
                            return(ECRemoveAction);
                    }
                    return(ECKeepAction);
                    break;
            }
}

//************************************************************************//
// Function: diToggleMeshDynFunc
//************************************************************************//

int diToggleMeshDynFunc(ECEvent *event, ECEventData data, ECAction *action)
{
            void                **args = action->parameters;
            if(ECArgReferenceGetValue(args[1], (void *)&switches->meshdynmode, &data.focus) == VC_ERR)
                    switches->meshdynmode = 1;
}

//************************************************************************//
// Function: diToggleAnimModeFunc
//************************************************************************//

int diToggleAnimModeFunc(ECEvent *event, ECEventData data, ECAction *action)
{
            void                **args = action->parameters;
            if(ECArgReferenceGetValue(args[1], (void *)&switches->animmode, &data.focus) == VC_ERR)
                    switches->animmode = 1;
}

//************************************************************************//
// Function: diOutputSetFunc
//************************************************************************//

int diOutputSetFunc(ECEvent *event, ECEventData data, ECAction *action)
{
            void                **args = action->parameters;
            if(ECArgReferenceGetValue(args[1], (void *)&switches->outtypenum, &data.focus) == VC_ERR)
                    switches->outtypenum = 0;
            if(ECArgReferenceGetValue(args[2], (void *)&switches->outsubnum, &data.focus) == VC_ERR)
                    switches->outsubnum = 0;

            di_set_range();

            di_output_mods();

            di_modify_ClrScl();

            di_modify_FEM();

            sprintf(chars->outtxt,"%sNode #: %i\nElement #: %i\n\n%s%10.6f\nDX: %10.6f\nDY: %10.6f\nDZ: %10.6f\n",
                                    names->actual_case_name,
                                    (NODE_P+((ELEMENT_P+pmi->rightelem)->B[pmi->adjindex]))->A,
                                    (ELEMENT_P+pmi->rightelem)->D,
                                    names->actual_set_name[(switches->outtypenum*5)+switches->outsubnum],
                                    outvert[pmi->rightvert]*floats->LoadFactor,
                    displaceobj[(pmi->rightvert*3)+0]*floats->LoadFactor,
                    displaceobj[(pmi->rightvert*3)+1]*floats->LoadFactor,
                                    displaceobj[(pmi->rightvert*3)+2]*floats->LoadFactor);
            VCString_SetText(femtextstring,chars->outtxt);
            di_updateclrsclxtxt();
}

//************************************************************************//
// Function: headtrackTimerHandler
//************************************************************************//
```

```
static void
headtrackTimerHandler (VCTimer_CallbackData *callbackData,
                                                        void *data)
{
  VCBody      *body = data;
        VC_Traverse         traverseInfo;
  dmMatrix          tempMat;
        dmEuler             o;
        dmQuaternion q,q1,q2;
        float32             xrad,yrad,zrad;
        float32             xquat,yquat,zquat,wquat;

  /* Timer gone off - read the events */
        VCBody_GetAbsolutePosition (body, tempMat);
//      dmEulerFromMat (o, tempMat);
//      printf("eulers are %f %f %f\n",o[0],o[1],o[2]);

        dmQuatFromMat (q1,tempMat);

//      xrad=o[0];
//      yrad=o[1];
//      zrad=o[2];
//      dmEulerSet(o,xrad,yrad,zrad);

        xquat=0.0;
        yquat=0.0;
        zquat=0.0+.005;
        wquat=1.0;
        dmQuatSet (q2,xquat,yquat,zquat,wquat);
        dmQuatMult (q, q1, q2);
//      printf("quats are %f %f %f %f\n\n",q[0],q[1],q[2],q[3]);
        dmEulerFromQuat (o, q);

        if(body != NULL)
        {
                VCBody_SetPosition(body, NULL, NULL, o, NULL, NULL, NULL);
        }
        else
        {
                VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), NULL, NULL, o, NULL, NULL, NULL);
        }
}

//*************************************************************************//
// Function:  diInsideTrakInitFunc
//*************************************************************************//

int diInsideTrakInitFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        void            **args = action->parameters;
//  VCBody      *body = data.body;
//      if (body == NULL)
//              body = VC_GetFirstBody(&traverseInfo);
//      VCBody_SetFlyMode (body, VC_BODY_FLY_VERTICAL );

        insideinit();
}

//*************************************************************************//
// Function:  headtrackSyncHandler
//*************************************************************************//

static void
headtrackSyncHandler(VCSync_CallbackData *callbackData, void *data)
{
  VCBody      *body = data;
  VCAttribute       *vcLimb;
  char                *part = NULL;
        static dmEuler              oprev={0,0,0};
```

```
        dmEuler             o,otemp;
        dmEuler             e;
        VC_Traverse     traverseInfo;
  dmMatrix          tempMat;
static  dmQuaternion qprev={0,0,0,1};
        dmQuaternion q,q1,q2,qtemp;
/*      VCSync  *sync = callbackData->sync;;

        if (!syncTime)
        {
                syncTime=(VCTime *)malloc(sizeof(VCTime));
                VCSync_GetTime(sync, syncTime);
                syncTime->secs=syncTime->secs+10;
                syncTime->uSecs=syncTime->uSecs+0;
//              VCSync_SetTime(sync,syncTime);
                printf("syncTime is %d %d\n",syncTime->secs,syncTime->uSecs);
        }

        else
        {
                VCSync_SetTime(sync,syncTime);
//              VCSync_GetTime(sync, syncTime);
                syncTime->secs=syncTime->secs+10;
                syncTime->uSecs=syncTime->uSecs+0;
//              VCSync_SetTime(sync,syncTime);
                printf("syncTime is %d %d\n",syncTime->secs,syncTime->uSecs);
        }
*/
  /* Find part if defined, else find head */
  if(part != NULL)
    vcLimb = VCBody_GetBodyPart (body, part);
  else
    vcLimb = VCBody_GetBodyPart (body, "head");
  /* Check that the limb part was found */
  if(vcLimb == NULL)
  {
    VC_Error("dvBodyPartAttachFunc : Limb part %s was not found\n", part);
    return(ECKeepAction);
  }


        VCBody_GetAbsolutePosition ((void *) body, tempMat);
        dmEulerFromMat (e, tempMat);
//      printf("curs x y z euler is %f %f %f\n",e[0],e[1],e[2]);
        dmQuatFromMat (q1,tempMat);
///     printf("quats are %f %f %f %f\n\n",q1[0],q1[1],q1[2],q1[3]);

        if (switches->navstate==1)
        {
                insidetick();
                dmEulerSetD (o,yeulfloat,-xeulfloat,-zeulfloat);
//              printf("o x y z euler is %f %f %f\n",yeulfloat,-xeulfloat,-zeulfloat);
///             dmQuatFromEuler (q2,o);
//              printf("quats are %f %f %f %f\n\n",q2[0],q2[1],q2[2],q2[3]);
//              printf("o x y z euler is %f %f %f\n",o[0],o[1],o[2]);

//              otemp[0]=o[0];
                otemp[1]=o[1];
//              otemp[2]=o[2];

//              qtemp[0]=q2[0];
//              qtemp[1]=q2[1];
//              qtemp[2]=q2[2];
//              qtemp[3]=q2[3];
//              o[0]=o[0]-oprev[0];
                o[1]=o[1]-oprev[1];
///             o[2]=o[2]-oprev[2];

//      if (fabs(q1[0])<.25 && fabs(q1[2])<.25)
//      {
```

```
                    e[0]=o[0];
                    e[1]=e[1]+o[1];
                    e[2]=o[2];
//                  printf("curs x y z euler is %f %f %f\n",e[0],e[1],e[2]);

//          }
//          else
//          {
//                  e[0]=e[0]+o[1];
//                  e[1]=e[1]+o[0];
//                  e[2]=e[2]+o[1];
//          }

//                  q2[0]=q2[0]-qprev[0];
//                  q2[1]=q2[1]-qprev[1];
//                  q2[2]=q2[2]-qprev[2];
//                  q2[3]=q2[3]-qprev[3];
//                  printf("quats are %f %f %f %f\n\n",q2[0],q2[1],q2[2],q2[3]);

//                  dmQuatMult (q, q1, q2);
//                  printf("1quats are %f %f %f %f\n\n",q[0],q[1],q[2],q[3]);
//                  dmEulerFromQuat (o, q);
                    if(body != NULL)
                    {
                            VCBody_SetPosition(body, vcLimb, NULL, e, NULL, NULL, NULL);
                    }
                    else
                    {
                            VCBody_SetPosition(VC_GetFirstBody(&traverseInfo), vcLimb, NULL, e, NULL, NULL, NULL);
                    }

//                  oprev[0]=otemp[0];
                    oprev[1]=otemp[1];
//                  oprev[2]=otemp[2];

//                  qprev[0]=qtemp[0];
//                  qprev[1]=qtemp[1];
//                  qprev[2]=qtemp[2];
//                  qprev[3]=qtemp[3];
            }
}

//************************************************************************//
// Function:  diSynchPartTrackFunc
//************************************************************************//

int
diSynchPartTrackFunc(ECEvent *event, ECEventData data, ECAction *action)
{
   void         **args = action->parameters;
   VCBody       *body = data.body;
//          VCSync  *sync;
//          VCTime  syncTime;

//          sync = callbackData->sync;
//          syncTime.secs  = 0;
//  syncTime.uSecs = 1000;
//          VCSync_Create ("local", "visual", "syncTime");

            // Changed to sync update callback for new 2D interface
            VC_AttachSyncCallbacks("local", "visual", headtrackSyncHandler, NULL, (void *)body);
   return(ECKeepAction);
}

int diToggleNavStateFunc(ECEvent *event, ECEventData data, ECAction *action)
{
            void                    **args = action->parameters;
            if(ECArgReferenceGetValue(args[1], (void *)&switches->navstate, &data.focus) == VC_ERR)
                    switches->navstate = 1;
}
```

```
//*****************************************************************************//
// Function:  diToggleLoadFunc - toggles visibility of loads on model                //
//*****************************************************************************//

int diToggleLoadFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        EntityList *tmp = NULL;
        void            **args = action->parameters;

        tmp = malloc (sizeof (EntityList));
        if(ECArgReferenceGetValue(args[1], (void *)&switches->loadcasestate, &data.focus) == VC_ERR)
                switches->loadcasestate = 1;

        if (switches->loadcasestate == 1)
                for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, VC_VISIBLE, 0);
        else
                for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, 0, VC_VISIBLE);
        free(tmp);
}

//*****************************************************************************//
// Function:  diCreateLoadObjectsFunc - creates the loads on the model               //
//*****************************************************************************//

int diCreateLoadObjectsFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        int i;
        void            **args = action->parameters;
        EntityList *newItem;
        dmEuler o;
        dmScale s;

        LoadList = malloc (sizeof (EntityList));
        LoadList = NULL;

        for(i = 0; i < LOADSET_NUM && i < 100; i++){
                newItem = malloc (sizeof (EntityList));

                // Initialization
                newItem->nodeobj = newItem->vis = newItem->next = NULL;

                // Populate the new item
                if(loadcoordind != NULL){
                        newItem->nodeobj=VCEntity_Create(NULL,0);
                        newItem->vis=VCVisual_CreateGeometry("greenarw");
                        VCVisual_SetIntersectMask (newItem->vis, 1);
                        VCEntity_AttachAttribute (newItem->nodeobj, newItem->vis);
                        newItem->nodepoint[0] = vertices[(loadcoordind[i]*7)+0];
                        newItem->nodepoint[1] = vertices[(loadcoordind[i]*7)+1];
                        newItem->nodepoint[2] = vertices[(loadcoordind[i]*7)+2];

                // Add the new item to the beginning of the list
                if (LoadList == NULL)
                        LoadList = newItem;
                else{
                        newItem->next = LoadList;
                        LoadList = newItem;
                }
                // Creates the points on the model and sets it invisible

                dmEulerSetD (o, 0, 90, 0);
                s[0]=floats->xyzmax/5;
                s[1]=floats->xyzmax/5;
                s[2]=floats->xyzmax/5;
                VCEntity_SetPositionPointEulerScale (LoadList->nodeobj, LoadList->nodepoint, o, s);
                VCVisual_ModifyMode (LoadList->vis, 0, VC_VISIBLE);
        }
```

```
        di_modify_LoadSet();
        di_modify_ConstraintSet();
        }
}

//*****************************************************************************//
// Function: di_modify_LoadSet - modifies the loads on the model                //
//*****************************************************************************//

int di_modify_LoadSet(void)
{
        EntityList *tmp = NULL;
        int i=0;

        tmp = malloc (sizeof (EntityList));

        for(tmp = LoadList; tmp != NULL; tmp = tmp->next)
        {
                tmp->nodepoint[0] = vertices[(loadcoordind[i]*7)+0]+displaceobj[(loadcoordind[i]*3)+0]*floats-
>LoadFactor*floats->exager;
    tmp->nodepoint[1] = vertices[(loadcoordind[i]*7)+1]+displaceobj[(loadcoordind[i]*3)+1]*floats->LoadFactor*floats->exager;
    tmp->nodepoint[2] = vertices[(loadcoordind[i]*7)+2]+displaceobj[(loadcoordind[i]*3)+2]*floats->LoadFactor*floats->exager;
                VCEntity_SetPositionPoint (tmp->nodeobj, tmp->nodepoint);
                i++;
        }
        free(tmp);
}

//*****************************************************************************//
// Function: diToggleConstrFunc - toggles visibility of constraints on model   //
//*****************************************************************************//

int diToggleConstrFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        EntityList *tmp = NULL;
        void            **args = action->parameters;

        tmp = malloc (sizeof (EntityList));
        if(ECArgReferenceGetValue(args[1], (void *)&switches->constraintstate, &data.focus) == VC_ERR)
                switches->constraintstate = 1;

        if (switches->constraintstate == 1)
                for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, VC_VISIBLE, 0);
        else
                for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
                        VCVisual_ModifyMode (tmp->vis, 0, VC_VISIBLE);
        free(tmp);
}

//*****************************************************************************//
// Function: diCreateConstrObjectsFunc - creates the constraints on the model//
//*****************************************************************************//

int diCreateConstrObjectsFunc(ECEvent *event, ECEventData data, ECAction *action)
{
        int i;
        void            **args = action->parameters;
        EntityList *newItem;
        dmEuler o;
        dmScale s;

        ConstrList = malloc (sizeof (EntityList));
        ConstrList = NULL;

        for(i = 0; i < CONSTRAINTSET_NUM; i++){
                newItem = malloc (sizeof (EntityList));

                // Initialization
                newItem->nodeobj = newItem->vis = newItem->next = NULL;
```

```
                    // Populate the new item
                    newItem->nodeobj=VCEntity_Create(NULL,0);
                    newItem->vis=VCVisual_CreateGeometry("greensphere");
                    VCVisual_SetIntersectMask (newItem->vis, 1);                    // Create Intersect Mask
                    VCEntity_AttachAttribute (newItem->nodeobj, newItem->vis);
                    newItem->nodepoint[0] = vertices[(constrcoordind[i]*7)+0];
    newItem->nodepoint[1] = vertices[(constrcoordind[i]*7)+1];
    newItem->nodepoint[2] = vertices[(constrcoordind[i]*7)+2];

                    // Add the new item to the beginning of the list
                    if (ConstrList == NULL)
                            ConstrList = newItem;
                    else{
                            newItem->next = ConstrList;
                            ConstrList = newItem;
                    }
                    // Creates the points on the model and sets it invisible
                    s[0] = floats->xyzmax/175;
                    s[1] = floats->xyzmax/175;
                    s[2] = floats->xyzmax/175;
                    VCEntity_SetPositionPointEulerScale (ConstrList->nodeobj, ConstrList->nodepoint, NULL, s);
                    VCVisual_ModifyMode (ConstrList->vis, 0, VC_VISIBLE);
            }
            di_modify_ConstraintSet();
}

//*************************************************************************//
// Function: di_modify_ConstraintSet - modifies the constraints on the model //
//*************************************************************************//

int di_modify_ConstraintSet(void)
{
            EntityList *tmp = NULL;
            int i=0;

            tmp = malloc (sizeof (EntityList));

            for(tmp = ConstrList; tmp != NULL; tmp = tmp->next)
            {
                    tmp->nodepoint[0] = vertices[(constrcoordind[i]*7)+0]+displaceobj[(constrcoordind[i]*3)+0]*floats-
>LoadFactor*floats->exager;
    tmp->nodepoint[1] = vertices[(constrcoordind[i]*7)+1]+displaceobj[(constrcoordind[i]*3)+1]*floats->LoadFactor*floats-
>exager;
    tmp->nodepoint[2] = vertices[(constrcoordind[i]*7)+2]+displaceobj[(constrcoordind[i]*3)+2]*floats->LoadFactor*floats-
>exager;
                    VCEntity_SetPositionPoint (tmp->nodeobj, tmp->nodepoint);
                    i++;
            }
            free(tmp);
}

//*************************************************************************//
// Function: diCreateViewButtonFunc
//*************************************************************************//

int
diCreateViewButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;

  ECVisual      *visual;
  VCAttribute   *visattribute;

          objViewButtonref = (ECObjectReference *)args[1];
  objViewButton = ECReferenceObject(objViewButtonref, &data.focus);

          visual = ECObjectGetVisual(objViewButton, NULL);

          if (visual == NULL)
```

```
            {
              VC_Error("visual was NULL\n");
              return(ECKeepAction);
            }
            visattribute = ECVisualGetVCAttribute(visual);
            ECVisualToVC (objViewButton, visual);
            ECObjectToVC(objViewButton);
            return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateViewTextFunc
//*************************************************************************//

int
diCreateViewTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

            objViewTextref = (ECObjectReference *)args[1];
  objViewText = ECReferenceObject(objViewTextref, &data.focus);

            visual = ECObjectGetVisual(objViewText, NULL);

            if (visual  == NULL)
              {
               VC_Error("visual was NULL\n");
               return(ECKeepAction);
              }
            visattribute = ECVisualGetVCAttribute(visual);
            ECVisualToVC (objViewText, visual);
            ECObjectToVC(objViewText);
            return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateDataButtonFunc
//*************************************************************************//

int
diCreateDataButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

            objDataButtonref = (ECObjectReference *)args[1];
  objDataButton = ECReferenceObject(objDataButtonref, &data.focus);

            visual = ECObjectGetVisual(objDataButton, NULL);

            if (visual == NULL)
              {
               VC_Error("visual was NULL\n");
               return(ECKeepAction);
              }
            visattribute = ECVisualGetVCAttribute(visual);
            ECVisualToVC (objDataButton, visual);
            ECObjectToVC(objDataButton);
            return(ECKeepAction);
}

//*************************************************************************//
// Function: diCreateDataTextFunc
//*************************************************************************//

int
diCreateDataTextFunc(ECEvent *event, ECEventData data, ECAction *action)
```

```
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

          objDataTextref = (ECObjectReference *)args[1];
  objDataText = ECReferenceObject(objDataTextref, &data.focus);

          visual = ECObjectGetVisual(objDataText, NULL);

          if (visual == NULL)
            {
             VC_Error("visual was NULL\n");
             return(ECKeepAction);
            }
          visattribute = ECVisualGetVCAttribute(visual);
          ECVisualToVC (objDataText, visual);
          ECObjectToVC(objDataText);
          return(ECKeepAction);
}

//************************************************************************//
// Function:  diCreateVisButtonFunc
//************************************************************************//

int
diCreateVisButtonFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;

          objVisButtonref = (ECObjectReference *)args[1];
  objVisButton = ECReferenceObject(objVisButtonref, &data.focus);

          visual = ECObjectGetVisual(objVisButton, NULL);

          if (visual == NULL)
            {
             VC_Error("visual was NULL\n");
             return(ECKeepAction);
            }
          visattribute = ECVisualGetVCAttribute(visual);
          ECVisualToVC (objVisButton, visual);
          ECObjectToVC(objVisButton);
          return(ECKeepAction);
}

//************************************************************************//
// Function:  diCreateVisTextFunc
//************************************************************************//

int
diCreateVisTextFunc(ECEvent *event, ECEventData data, ECAction *action)
{
  void          **args = action->parameters;
  ECVisual       *visual;
  VCAttribute    *visattribute;
  VCEntity       *vistextent = NULL;

          objVisTextref = (ECObjectReference *)args[1];
  objVisText = ECReferenceObject(objVisTextref, &data.focus);

          visual = ECObjectGetVisual(objVisText, NULL);

          if (visual == NULL)
            {
             VC_Error("visual was NULL\n");
             return(ECKeepAction);
            }
```

```
        visattribute = ECVisualGetVCAttribute(visual);
        ECVisualToVC (objVisText, visual);
        ECObjectToVC(objVisText);
        return(ECKeepAction);
}

//************************************************************************//
// Function:  ToolCreation_cb
//************************************************************************//

int ToolCreation_cb(TBTool *tool)
{
 SliderDataStruct   *myData;

 printf("Setting up user Data structure...\n");
 myData = (SliderDataStruct *)calloc(2, sizeof(SliderDataStruct));
 TBGenSetUserData(tool, (void *)myData);
}

//************************************************************************//
// Function:  WidgetCreation_cb
//************************************************************************//

int WidgetCreation_cb(VWidget *newWidget, TBTool *tool, void *data)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        if ((data != NULL) && (myData != NULL))
  {
                if (!(strcmp((char*)data, "LoadFact")))
                {
                        printf("Got reference to LoadFact = 0x%x\n", newWidget);
                        myData->LoadFact = newWidget;
                        VWScalar_SetValue(myData->LoadFact,100, FALSE);
                }
                if (!(strcmp((char*)data, "LoadDisp")))
                {
                        printf("Got reference to LoadDisp = 0x%x\n", newWidget);
                        myData->LoadDisp = newWidget;
                        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
                }
                if (!(strcmp((char*)data, "ThreshFact")))
                {
                        printf("Got reference to ThreshFact = 0x%x\n", newWidget);
                        myData->ThreshFact = newWidget;
                        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
                }
                if (!(strcmp((char*)data, "ThreshDisp")))
                {
                        printf("Got reference to ThreshDisp = 0x%x\n", newWidget);
                        myData->ThreshDisp = newWidget;
                        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
                }
                if (!(strcmp((char*)data, "ExagerFact")))
                {
                        printf("Got reference to ExagerFact = 0x%x\n", newWidget);
                        myData->ExagerFact = newWidget;
                        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
                }
                if (!(strcmp((char*)data, "ExagerDisp")))
                {
                        printf("Got reference to ExagerDisp = 0x%x\n", newWidget);
                        myData->ExagerDisp = newWidget;
                        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclTop")))
                {
                        printf("Got reference to ClrSclTop = 0x%x\n", newWidget);
```

```
                        myData->ClrSclTop = newWidget;
                        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclTopDisp")))
                {
                        printf("Got reference to ClrSclTopDisp = 0x%x\n", newWidget);
                        myData->ClrSclTopDisp = newWidget;
                        VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclBot")))
                {
                        printf("Got reference to ClrSclBot = 0x%x\n", newWidget);
                        myData->ClrSclBot = newWidget;
                        VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
                }
                if (!(strcmp((char*)data, "ClrSclBotDisp")))
                {
                        printf("Got reference to ClrSclBotDisp = 0x%x\n", newWidget);
                        myData->ClrSclBotDisp = newWidget;
                        VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
                }
  }
}

//************************************************************************//
// Function:  UpdateSliderInfo_cb
//************************************************************************//

int UpdateSliderInfo_cb(VWidget *scalarWig, VWEventInfo *info, void *data)
{
        ECObject *obj;
        TBTool *thisTool;
        float32 newValue;
        SliderDataStruct    *myData;
        char            *calldata;

        if (!(thisTool = TBGenGetTool(data)))
  {
                return;
  }

        calldata = (char *)TBGenGetCalldata(data);
        myData = (SliderDataStruct *)TBGenGetUserData(thisTool);

        newValue = VWScalar_GetValue(scalarWig);
        if (!(strcmp((char*)calldata, "LoadFact")))
        {
                VWDigit_SetValue(myData->LoadDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ThreshFact")))
        {
                VWDigit_SetValue(myData->ThreshDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ExagerFact")))
        {
                VWDigit_SetValue(myData->ExagerDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ClrSclTop")))
        {
                VWDigit_SetValue(myData->ClrSclTopDisp, (int)(newValue), FALSE);
        }
        if (!(strcmp((char*)calldata, "ClrSclBot")))
        {
                VWDigit_SetValue(myData->ClrSclBotDisp, (int)(newValue), FALSE);
        }
}

//************************************************************************//
// Function:  UpdateSlider_cb
//************************************************************************//
```

```
int UpdateSlider_cb(VWidget *scalarWig, VWEventInfo *info, void *data)
{
        ECObject *obj;
        uint32 *eventId;
        TBTool *thisTool;
        float32 newValue;
        SliderDataStruct   *myData;
        char            *calldata;
        float32 delta,out_new;

        if (!(thisTool = TBGenGetTool(data)))
  {
                return;
  }

        calldata = (char *)TBGenGetCalldata(data);
        myData = (SliderDataStruct *)TBGenGetUserData(thisTool);

        newValue = VWScalar_GetValue(scalarWig);

        if (!(strcmp((char*)calldata, "LoadFact")))
        {
                VWDigit_SetValue(myData->LoadDisp, (int)(newValue), FALSE);
                floats->LoadFactor = (float32)(newValue)/100.0f;
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
                di_modify_LoadSet();
                di_modify_ConstraintSet();
        }
        if (!(strcmp((char*)calldata, "ThreshFact")))
        {
                VWDigit_SetValue(myData->ThreshDisp, (int)(newValue), FALSE);
                floats->threshold = ((float32)(newValue)/100.0f);
                floats->out_vals[1] = floats->absmax*floats->threshold;
                di_modify_ClrScl();
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
        }
        if (!(strcmp((char*)calldata, "ExagerFact")))
        {
                VWDigit_SetValue(myData->ExagerDisp, (int)(newValue), FALSE);
                floats->exager = (float32)(newValue);
                di_modify_FEM();
                if (switches->meshdynmode==1) di_modify_Mesh();
                di_modify_ConstraintSet();
                di_modify_LoadSet();
        }
        if (!(strcmp((char*)calldata, "ClrSclTop")))
        {
                VWDigit_SetValue(myData->ClrSclTopDisp, (int)(newValue), FALSE);
                floats->clrscltop = (float32)(newValue)/100.0f;
                floats->out_vals[2]=floats->out_min+
                                                        (floats->clrscltop*
                                                        (floats->out_max-floats->out_min));
                di_modify_ClrScl();
                di_updateclrscltxt();
                di_modify_FEM();
        }
        if (!(strcmp((char*)calldata, "ClrSclBot")))
        {
                VWDigit_SetValue(myData->ClrSclBotDisp, (int)(newValue), FALSE);
                floats->clrsclbot = (float32)(newValue)/100.0f;
                floats->out_vals[0]=floats->out_min+
                                                        (floats->clrsclbot*
                                                        (floats->out_max-floats->out_min));
                di_modify_ClrScl();
                di_updateclrscltxt();
                di_modify_FEM();
        }
```

```
}

//*************************************************************************//
// Function:  SetSliders_cb
//*************************************************************************//

int SetSliders_cb(ECObject *obj, VCBody *body, VCAttribute *limb, TBTool *tool)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        VWScalar_SetValue(myData->LoadFact,100, FALSE);
        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
        VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
        VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
        VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
}

//*************************************************************************//
// Function:  ResetSliders_cb
//*************************************************************************//

int ResetSliders_cb(ECObject *obj, VCBody *body, VCAttribute *limb, TBTool *tool)
{
        SliderDataStruct   *myData;

        myData = (SliderDataStruct *)TBGenGetUserData(tool);

        VWScalar_SetValue(myData->LoadFact,100, FALSE);
        VWDigit_SetValue(myData->LoadDisp, 100, FALSE);
        VWScalar_SetValue(myData->ThreshFact,0, FALSE);
        VWDigit_SetValue(myData->ThreshDisp, 0, FALSE);
        VWScalar_SetValue(myData->ExagerFact,1, FALSE);
        VWDigit_SetValue(myData->ExagerDisp, 1, FALSE);
        VWScalar_SetValue(myData->ClrSclTop,100, FALSE);
        VWDigit_SetValue(myData->ClrSclTopDisp, 100, FALSE);
        VWScalar_SetValue(myData->ClrSclBot,0, FALSE);
        VWDigit_SetValue(myData->ClrSclBotDisp, 0, FALSE);
}

/* PUBLIC FUNCTION DEFINITIONS ========================================== */

extern void RegisterScaleToolFunctions(void)
{
        TBRegisterToolCreationCallback("myToolCreation",
                ToolCreation_cb);
        TBRegisterGenericWidgetCreationCallback("myWidgetCreation",
                WidgetCreation_cb);
        TBRegisterGenericWidgetCallback("UpdateSliderInfo",
                UpdateSliderInfo_cb);
        TBRegisterGenericWidgetCallback("UpdateSlider",
                UpdateSlider_cb);
        TBRegisterGenericObjectSelectCallback("setSliders",
                SetSliders_cb);
        TBRegisterGenericObjectSelectCallback("resetSliders",
                ResetSliders_cb);
}

//*************************************************************************//
// Function:  main
//*************************************************************************//
```

```
int
main (int argc, char **argv)
{

        extern void RegisterScaleToolFunctions(void);

        points=(Points *)malloc(sizeof(Points));
        switches=(Switches *)malloc(sizeof(Switches));
        floats=(Floats *)malloc(sizeof(Floats));
        chars=(Chars *)malloc(sizeof(Chars));
        vcfloats=(VCfloats *)malloc(sizeof(VCfloats));
        intersectionReportData=(VCIntersectionReportData *)malloc(sizeof(VCIntersectionReportData));

        objFEM=(ECObject *)malloc(sizeof(ECObject));
        objFEMref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objMesh=(ECObject *)malloc(sizeof(ECObject));
        objMeshref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objFEMText=(ECObject *)malloc(sizeof(ECObject));
        objFEMTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objClrScl=(ECObject *)malloc(sizeof(ECObject));
        objClrSclref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objClrSclGrid=(ECObject *)malloc(sizeof(ECObject));
        objClrSclGridref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objViewButton=(ECObject *)malloc(sizeof(ECObject));
        objViewButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objViewText=(ECObject *)malloc(sizeof(ECObject));
        objViewTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objDataButton=(ECObject *)malloc(sizeof(ECObject));
        objDataButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objDataText=(ECObject *)malloc(sizeof(ECObject));
        objDataTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objVisButton=(ECObject *)malloc(sizeof(ECObject));
        objVisButtonref=(ECObjectReference *)malloc(sizeof(ECObjectReference));
        objVisText=(ECObject *)malloc(sizeof(ECObject));
        objVisTextref=(ECObjectReference *)malloc(sizeof(ECObjectReference));

        femtextstring=(VCGeometry *)malloc(sizeof(VCGeometry));
        clrscltextstring=(VCGeometry *)malloc(sizeof(VCGeometry));

        switches->navstate=1;//navmode
        switches->navmode=2;//navmode
        switches->set1=0;//set1
        switches->set2=0;//set2
        switches->picknode=0;//picknode
        switches->meshdynmode=1;//meshdynmode
        switches->outtypenum=0;//0 is node type output, 1 is element type output
        switches->outsubnum=0;//node or element subtype index (0-4) in output array
        switches->animmode=1;
        switches->startanim=-1;//meshdynmode
        switches->loadcasestate=0;
        switches->constraintstate=0;

        floats->LoadFactor=1.0;
        floats->exager=1.0;
        floats->threshold=0.0;
        floats->beamdelta=100;
        floats->xyzmax=0.0;
        floats->absmax=0.0;
        floats->out_min=100000;
        floats->out_max=-100000;
        floats->clrscltop=1.0;
        floats->clrsclbot=0.0;
        floats->femsclbotl[0]=          0.0;
        floats->femsclbotl[1]=            0.0;
        floats->femsclbotl[2]=              0.0;
        floats->femsclbotr[0]=          .03;
        floats->femsclbotr[1]=            0.0;
        floats->femsclbotr[2]=              0.0;
        floats->femscltopr[0]=          .03;
        floats->femscltopr[1]=            .294;
```

```
floats->femscltopr[2]=                  0.0;
floats->femscltopl[0]=              0.0;
floats->femscltopl[1]=                 .294;
floats->femscltopl[2]=                  0.0;
floats->alphainrng=                 1.0;
floats->alphathresh=  0.7;
floats->alphaoutrng=  0.0;

vcfloats->posmaxcolor[0]=1.0;//red
vcfloats->posmaxcolor[1]=   0.0;
vcfloats->posmaxcolor[2]=      0.0;
vcfloats->posmincolor[0]=1.0;//yellow
vcfloats->posmincolor[1]=   1.0;
vcfloats->posmincolor[2]=      0.0;
vcfloats->negmincolor[0]=0.0;//greenblue
vcfloats->negmincolor[1]=   1.0;
vcfloats->negmincolor[2]=      1.0;
vcfloats->negmaxcolor[0]=0.0;//green
vcfloats->negmaxcolor[1]=   1.0;
vcfloats->negmaxcolor[2]=      0.0;
vcfloats->posthreshcolor[0]=0.5;//white
vcfloats->posthreshcolor[1]=   0.5;
vcfloats->posthreshcolor[2]=      0.5;
vcfloats->negthreshcolor[0]=1.0;//white
vcfloats->negthreshcolor[1]=   1.0;
vcfloats->negthreshcolor[2]=      1.0;
vcfloats->outofrngcolor[0]=0.0;//black
vcfloats->outofrngcolor[1]=   0.0;
vcfloats->outofrngcolor[2]=      0.0;

chars->outtxt[200]=" ";
chars->scltxt[200]=" ";

ucf_fem2vr();

ECUserActionFuncRegister(diCreateFEMObjectFunc,"diCreateFEMObject",
      "Converts FEM output files into objects",
                        ECDataTypeObject, "ObjectName",
      ECDataTypeFloatVar, "ObjectScale",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateFEMMeshFunc,"diCreateFEMMesh",
      "Creates FEM wireframe mesh",
                        ECDataTypeObject, "ObjectName",
      ECDataTypeFloatVar, "ObjectScale",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateFEMTextFunc,"diCreateFEMText",
      "Creates an dynamic text visual in femtext",
                        ECDataTypeObject, "femtext",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateClrSclTextFunc,"diCreateClrSclText",
      "Creates an dynamic color scale number visual in clrscltext",
                        ECDataTypeObject, "clrscltext",
                        ECDataTypeNull);

ECUserActionFuncRegister(diCreateColorSclFunc,"diCreateColorScl",
      "Creates an color scale visual in femscale",
                        ECDataTypeObject, "colorscale",
                        ECDataTypeNull);
ECUserActionFuncRegister(diCreateColorSclGridFunc,"diCreateColorSclGrid",
      "Creates an color scale grid visual in femscalegrid",
                        ECDataTypeObject, "colorgrid",
                        ECDataTypeNull);

ECUserActionFuncRegister(diToggleAnimFunc,"diToggleAnim",
      "Toggles animation of FEM on and off",
      ECDataTypeIntVar, "StartAnim",
                        ECDataTypeNull);
ECUserActionFuncRegister(diToggleAnimModeFunc,"diToggleAnimMode",
      "Toggles animation mode from Sawtooth to Ramp",
      ECDataTypeIntVar, "AnimMode",
```

```
                                ECDataTypeNull);
        ECUserActionFuncRegister(diImmersDataFunc, "diImmersData",
                "Get data at intersection point",
                ECDataTypeString, "bodyPart",
                ECDataTypeEvent, "Event",
                ECDataTypeObject, "FEMObj",
                ECDataTypeNull);
        ECUserActionFuncRegister(diBodyStartupPosFEMFunc, "diBodyStartupPosFEM",
                "Set StartUp Body Position FEM Viewpoint",
                ECDataTypeNull);
ECUserActionFuncRegister(diBodyMoveToFunc, "diBodyMoveTo",
                "Moves the body to a given viewpoint at a given speed",
                ECDataTypeIntVar, "ViewNumber",
                ECDataTypeFloatVar, "speed(m/s)",
                ECDataTypeNull);
        ECUserActionFuncRegister(diNavModeFunc,"diNavMode",
                "Sets navigation mode parameter",
                ECDataTypeIntVar, "navmode",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diSetViewFunc,"diSetView",
                "Sets user defined viewpoints",
                ECDataTypeIntVar, "viewnum",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diToggleMeshDynFunc,"diToggleMeshDyn",
                "Sets navigation mode parameter",
                ECDataTypeIntVar, "meshdynmode",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diOutputSetFunc,"diOutputSet",
                "Sets FEM output data set",
                ECDataTypeIntVar, "outtypenum",
                ECDataTypeIntVar, "outsubnum",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diInsideTrakInitFunc,"diInsideTrakInit",
                "Initializes Polhemus InsideTrak with MetaTrak Driver",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diSynchPartTrackFunc,"diSynchPartTrack",
                "Enables Synch Tracking of Named Body Part",
                ECDataTypeString, "BodyPart",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diToggleNavStateFunc,"diToggleNavState",
                "Toggles navigation mode from No HeadTrack to HeadTrack",
                ECDataTypeIntVar, "NavState",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diCreateLoadObjectsFunc,"diCreateLoadObjects",
                                                                "Creates load case objects",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diToggleLoadFunc,"diToggleLoad",
                "Toggles loadcase visual",
                ECDataTypeIntVar, "LoadCaseState",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diCreateConstrObjectsFunc,"diCreateConstrObjects",
                                                                "Creates constraints objects",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diToggleConstrFunc,"diToggleConstr",
                "Toggles constraints visual",
                ECDataTypeIntVar, "ConstrCaseState",
                                ECDataTypeNull);
ECUserActionFuncRegister(diCreateViewButtonFunc,"diCreateViewButton",
                "Creates a View button attached to viewpoint",
                                ECDataTypeObject, "viewbutton",
                                ECDataTypeNull);
ECUserActionFuncRegister(diCreateViewTextFunc,"diCreateViewText",
                "Creates a View text attached to viewpoint",
                                ECDataTypeObject, "viewtext",
                                ECDataTypeNull);
        ECUserActionFuncRegister(diCreateDataButtonFunc,"diCreateDataButton",
                "Creates a Data button attached to viewpoint",
                                ECDataTypeObject, "databutton",
                                ECDataTypeNull);
ECUserActionFuncRegister(diCreateDataTextFunc,"diCreateDataText",
```

```
                "Creates a Data text attached to viewpoint",
                                ECDataTypeObject, "datatext",
                                ECDataTypeNull);
    ECUserActionFuncRegister(diCreateVisButtonFunc,"diCreateVisButton",
                "Creates a Visualize button attached to viewpoint",
                                ECDataTypeObject, "visbutton",
                                ECDataTypeNull);
    ECUserActionFuncRegister(diCreateVisTextFunc,"diCreateVisText",
                "Creates a Visualize text attached to viewpoint",
                                ECDataTypeObject, "vistext",
                                ECDataTypeNull);

        RegisterScaleToolFunctions();

        VC_AttachBodyCreateCallback (di_create_body_handler, NULL);

        dVISE_Initialise(argc,argv);

        VC_MainLoop();
}
```

```
//*************************************************************************//
DVET Release 2.2/11/98 for WindowsNT Workstation
fm2vr1120.h
11 February 1998
Copyright 1998
Dual Incorporated/University of Central Florida


typedef struct NODE_DATA
        {
        long int A;
        double x;
        double y;
        double z;
        double dx;
        double dy;
        double dz;
        double output_data[5];
        //add in Oct., 1997

        int H;

        //add in Oct., 1997
        } NODE_DATA;

typedef struct ELEMENT_REL
        {
        long int A;
        double data[5];
        } ELEMENT_REL;


typedef struct ELEMENT_DATA
        {
        long int A;
        long int B[4];

        double C[5];
        long int D;
//revised on Sept 30, 1997
        //E is the index for internal element (default) and zero for surface element
        int E;
        //F is an index to reference the element property
        int F;
//revised on Sept 30, 1997
        } ELEMENT_DATA;

//revised on Sept 30, 1997
typedef struct ELEMENT_PROPERTY
        {
        //A is the type of element
        int A;
        //revised in Oct., 1997

        int H;//for material id.

                //revised in Oct., 1997
        //B is the element properties, according to the manual of FEMAP neutral file
        double B[100];
        } ELEMENT_PROPERTY;
//above are revised on Sept 30, 1997

typedef struct NAMES
        {
        char actual_case_name[30];
        char actual_set_name[10][30];
        } NAMES;

//revised in Oct. ,1997
```

```
typedef struct MATERIAL
{
int A;
char title[25];
double Young_Modulus[3];
double Shear_Modulus[3];
double Poisson_Ratio[3];
double GMatrix[21];
double alpha[6];
double k[6];
double thermal_cap,density,damping,temperature;
double tension_limit[2];
double comp_limit[2];
double shear_limit;
} MATERIAL;

//
typedef struct CONSTRAINT
{int A;
char B[25];
long int NUM;
fpos_t file_constraint;
long int *ID;
int *INDEX;
} CONSTRAINT;

typedef struct COORDINATE
{
        int A;//id
        int B; //id of
        int C;//type
char D[25];
double E[3];//origin coordiantes
double F[3];//rotation angles
} COORDINATE;

typedef struct LOAD
{
int SET_ID;
char NAME[25];
fpos_t load_file,nt_file,et_file;
long int NUM,NT_NUM,ET_NUM;
long int *ID,*NT_ID,*ET_ID;
int *TYPE,*FACE;
double *VALUE,*NT_VALUE,*ET_VALUE;
} LOAD;//the default limitation for load set number is 100


//revised in Oct., 1997

extern struct NODE_DATA *NODE_P;
extern struct ELEMENT_DATA *ELEMENT_P;
extern struct NAMES *names;
//revised in Oct., 1997

extern struct MATERIAL *MATERIAL_P;
extern struct COORDINATE *COORDINATE_P;
extern struct CONSTRAINT CONSTRAINT_SET[100];
extern struct LOAD LOAD_SET[100];

//revised 16 Jan 98
extern long int NODE_NUM,ELEMENT_NUM,LOADSET_NUM,LOADSET_PICK,
CONSTRAINTSET_NUM,CONSTRAINTSET_PICK;
//revised 16 Jan 98
```

```
//*************************************************************************//
DVET Release 2.2/11/98 for WindowsNT Workstation
fm2vrwin.c
11 February 1998
Copyright 1998
Dual Incorporated/University of Central Florida
26/12/97   Ola Fakinlede          Added gui prompt for file input
//
//*************************************************************************//

#include "stdio.h"
#include "string.h"
#include "malloc.h"
#include "process.h"
#include "stdlib.h"

//********************** User defined header file **********************//

#include "fm2vr1120.h"

//*************************************************************************//

struct NODE_DATA *NODE_P;
struct ELEMENT_DATA *ELEMENT_P;
struct NAMES *names;
struct MATERIAL *MATERIAL_P;
struct COORDINATE *COORDINATE_P;
struct LOAD LOAD_SET[100];
struct CONSTRAINT CONSTRAINT_SET[100];
struct ELEMENT_REL *ELEMENT_TMP;
struct ELEMENT_DATA *ELEMENT_INF;
struct ELEMENT_PROPERTY *ELEMENT_PROPERTY_P;

long int
NODE_NUM,ELEMENT_NUM1,ELEMENT_NUM,output_set_num,LOADSET_NUM,LOADSET_PICK,CONSTRAINTSET_NU
M,CONSTRAINTSET_PICK;
long int u;
long int IA,IB,IC,IE;
int ELEMENT_PRO_NUM;
int MATERIAL_NUM;
int CONSTRAINT_NUM;
int COORDINATE_NUM;
int LOAD_NUM;

//*****************************Function Prototypes************************/

int compare(long int  ELEMENT_i,long int NODE_i);
long int FindNid(long int u);
long int FindEid(long int u);

/*********************************External Declarations ******************/

extern char *file_prompt();                                          // Function that calls file prompt
extern char *bool_prompt(char *);                                    // Function that calls bool prompt
extern char *case_prompt(char set_name[3000][30], int);              // Function that calls case prompt
extern char **output_data_prompt(char temp_name[2000][40], int);     // Function that calls output data prompt
extern char *loadset_prompt(char loadset_names[100][30], int);       // Function that calls load prompt
extern char *constraintset_prompt(char constraintset_names[100][40], int);  // Function that calls constraint prompt

//*************************************************************************//
// Function:  ucf_fem2vr
// Purpose:  The function main reads the FEMAP file and stores information
//                                  into various data blocks.
//*************************************************************************//

void ucf_fem2vr(void)
{
        long int CHECKD,NODE_i,ELEMENT_i,CHECKDD,case_set_num[2000], q;
        char set_name[3000][30],temp_set_name[2000][30],out_set_name[2000][30], temp_name[2000][40];
        char buffer[200];
```

```
        double X,Y,Z,TIME[2000],MAX_VALUE[2000],MIN_VALUE[2000],AMAX_VALUE[2000];

        long int NODE_NUM_S, ELEMENT_NUM_S;
        long int ID[20],Total_num[2000],case_num[2000];
        int flag,
                flag_open_file = 0,             // Flag indicating open file
                flag_solid = 0,                 // Flag indicating solid
                LOAD_YES = 0,                   // Flag indicating yes to load loads
                CONSTRAINT_YES = 0;             // Flag indicating yes to load constraints

        FILE *NEU_INP;
        FILE *fp,*fp1,*fp2, *fp_load, *fp_constraint;
        FILE *tmp1,*tmp2;
        int V_NUM = 5,U_NUM = 5;
        int FLAG,ID_BLOCK,CHECK,TYPE[2000],V[5],U[5],case_n;
        int i,CHECK1,II,case_i,case_nn, r;
        int I1,I2,I3,I4;

        fpos_t file_node,file_element,file_output[2000];
        fpos_t file_pro;
        fpos_t file_mat,file_coordinate;
        char *filename;                         // FEMAP Neutral File
        char *bool;                             // YES or NO
        char gui_name[3];                       // Name id for gui
        char *case_name;                        // case name
        char case_names[3000][30];      // set_name without carriage return
        char **output_data = NULL;      // output data
        int j = 0, m = 0, k = 0;
        char *loadset_name;                     // load name
        char loadset_names[100][30];    // loadset_name without carriage return
        char *constraintset_name;               // constraint name
        char constraintset_names[100][40];      // constraintset_name without carriage return

//************************open neutral file************************//

while(flag_open_file == 0){
        filename = file_prompt();

        if(strncmp(filename, "cancel", 6) == 0)
                exit(1);
        if(*filename == '\0')
                continue;
        else if((NEU_INP = fopen(filename, "r+")) != NULL)
                        flag_open_file = 1;
}

//*************************************************************************//
// Check FEMAP neutral file
//*************************************************************************//

FLAG = 1;
while((!feof(NEU_INP)) && (FLAG == 1))
{
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);              // Move the file pointer
        if((CHECK==-1)&&(FLAG==1))
                {
                fscanf(NEU_INP,"%d",&ID_BLOCK);
                fgets(buffer,200,NEU_INP);

                switch(ID_BLOCK)
                {
                case 100:
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%d",&CHECK);
                fgets(buffer,200,NEU_INP);
                                if(CHECK==-1)
                                        {
                                        FLAG=1;
```

```
                }
        break;
case 405:
        fgetpos(NEU_INP,&file_coordinate);
        COORDINATE_NUM=0;
        label405:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
        {
                COORDINATE_NUM=COORDINATE_NUM+1;
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                goto label405;
                }
        FLAG=1;
        break;
case 475:
        //**********Process the text information*******************//
        label475:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                for(i=0;i<=II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label475;
                }
        FLAG=1;
        break;
case 410:
        //**********Process the variable information*******************//
        label410:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label410;
                }
        FLAG=1;
        break;
case 413:
//**********Process the layer information*******************//
        label413:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                goto label413;
                }
        FLAG=1;
        break;
case 470:
        //**********Process the point information*******************//
        label470:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
```

```
                goto label470;
                }
        FLAG=1;
        break;
case 471:
//***********Process the curve information*******************//
        label471:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label471;
                }
        FLAG=1;
        break;
case 472:
        //***********Process the surface information*******************//
        label472:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label472;
                }
        FLAG=1;
        break;
case 473:
        //***********Process the volume information*******************//
        label473:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label473;
                }
        FLAG=1;
        break;
case 474:
        //***********Process the boundary information*******************//
        label474:
        fscanf(NEU_INP,"%d",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                goto label474;
                }
        FLAG=1;
        break;
case 401:
        //***********Process the material information*******************//

        MATERIAL_NUM=0;
        fgetpos(NEU_INP,&file_mat);
        label401:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
        {
                MATERIAL_NUM=MATERIAL_NUM+1;

                for(i=0;i<32;i++)
```

```
                        {
                        fgets(buffer,200,NEU_INP);

                        }
                goto label401;
                }
        FLAG=1;
        break;
case 402:
        //**********Process the property information*******************//
        ELEMENT_PRO_NUM=0;
        fgetpos(NEU_INP,&file_pro);

        label402:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
        {
                ELEMENT_PRO_NUM=ELEMENT_PRO_NUM+1;
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<(float)(II/8)+1.0;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<(float)(II/5)+1.0;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                goto label402;
                }
        FLAG=1;
        break;
case 403:
        NODE_NUM=0;
        fgetpos(NEU_INP,&file_node);
        //**********Process the node information*******************//
        label403:
        fscanf(NEU_INP,"%d,",&CHECK);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                NODE_NUM=NODE_NUM+1;
                goto label403;
                }
        FLAG=1;
        break;
case 404:
        fgetpos(NEU_INP,&file_element);
        ELEMENT_NUM=0;
        ELEMENT_NUM1=0;
//**********Process the element information*******************//
        label404:
        fscanf(NEU_INP,"%d,",&CHECK);
        if(CHECK!=-1)
                {
                fscanf(NEU_INP,"%d,%d,%d,%d,",&I1,&I2,&I3,&I4);
        switch(I4)
                {
                case 0:
                        ELEMENT_NUM=ELEMENT_NUM+1;
                        break;
                case 2:
                        ELEMENT_NUM=ELEMENT_NUM+1;
                        break;
                case 3:
```

```
                                        ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                case 4:
                                        ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                case 5:
                                        ELEMENT_NUM=ELEMENT_NUM+1;
                                        break;
                                case 6:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+4;
                                        break;
                                case 7:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+5;
                                        break;
                                case 8:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+6;
                                case 9:
                                        break;
                                case 10:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+4;
                                        break;
                                case 11:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+5;
                                        break;
                                case 12:
                                        flag_solid = 1;
                                        ELEMENT_NUM=ELEMENT_NUM+6;
                                        break;
                                case 13:
                                        break;
                                }

                        ELEMENT_NUM1=ELEMENT_NUM1+1;
                        for(i = 0; i < 7; i++)
                                {
                                fgets(buffer, 200, NEU_INP);
                                }
                        if(I4 == 13)
                                {
                                CHECKD=0;
                                while(CHECKD != -1)
                                        {
                                        fscanf(NEU_INP,"%ld,",&CHECKD);
                                        fgets(buffer,200,NEU_INP);
                                        }
                                }
                        goto label404;
                        }
                        FLAG=1;
                        break;

//*********************     CONSTRAINT INFORMATION          ****************************//

                        case 406:

                        CONSTRAINT_NUM = 0;

                        label406:
                                if(CONSTRAINT_NUM > 100)
                                {
                                        printf("The number of constraint sets exceeds the default value of 100\n");
                                        exit(0);
                                }
                                fscanf(NEU_INP,"%d,",&I1);
                                fgets(buffer,200,NEU_INP);
```

```
                    if(I1 != -1)
                            {
                            CONSTRAINT_SET[CONSTRAINT_NUM].A = I1;
                            fgets(CONSTRAINT_SET[CONSTRAINT_NUM].B, 25, NEU_INP);
                            CONSTRAINT_SET[CONSTRAINT_NUM].NUM = 0;

        fgetpos(NEU_INP,&(CONSTRAINT_SET[CONSTRAINT_NUM].file_constraint));
                            fscanf(NEU_INP,"%d,",&CHECK1);

                    label4061:
                            if(CHECK1 != -1)
                                    {
                                    CONSTRAINT_SET[CONSTRAINT_NUM].NUM=
CONSTRAINT_SET[CONSTRAINT_NUM].NUM + 1;
                                    fgets(buffer,200,NEU_INP);
                                    fscanf(NEU_INP,"%d,",&CHECK1);
                                    goto label4061;
                                    }
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                    label4062:
                                    if(CHECK1 != -1)
                                    {
                                    fgets(buffer,200,NEU_INP);
                                    fscanf(NEU_INP,"%d",&II);
                                    goto label4062;
                                    }

                                    fgets(buffer,200,NEU_INP);
                                    CONSTRAINT_NUM=CONSTRAINT_NUM+1;
                            goto label406;

                            }

                    FLAG=1;
                    break;

//*********************     LOAD INFORMATION        ****************************//

                    case 407:

                            LOAD_NUM = 0;
                            label407:
                            fscanf(NEU_INP,"%d",&CHECK);                    // check for end of block
                            fgets(buffer,200,NEU_INP);
                            if(CHECK != -1)
                                    {
                                    LOAD_SET[LOAD_NUM].SET_ID = CHECK;
                                    fgets(LOAD_SET[LOAD_NUM].NAME,25,NEU_INP);

                                    for(i=0;i<19;i++)
                                            {
                                            fgets(buffer,200,NEU_INP);

                                            }
                                            fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].load_file));
                                            fscanf(NEU_INP,"%d,",&CHECK1);
                                            LOAD_SET[LOAD_NUM].NUM = 0;
                    label4071:
                                            if(CHECK1 != -1)
                                            {
                                            LOAD_SET[LOAD_NUM].NUM =
LOAD_SET[LOAD_NUM].NUM + 1;
                                            for(I1 = 0; I1 < 12; I1++)
                                                    {
                                                    fgets(buffer,200,NEU_INP);
                                                    }
                                            fscanf(NEU_INP,"%d",&CHECK1);
                                            goto label4071;
                                            }
```

```
                    fgets(buffer,200,NEU_INP);
                    LOAD_SET[LOAD_NUM].NT_NUM = 0;
                    fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].nt_file));
                    fscanf(NEU_INP,"%d,",&CHECK1);

        label4072:
                    if(CHECK1 != -1)
                            {
                            LOAD_SET[LOAD_NUM].NT_NUM =
LOAD_SET[LOAD_NUM].NT_NUM + 1;
                            fgets(buffer, 200, NEU_INP);
                            fscanf(NEU_INP, "%d,", &CHECK1);
                            goto label4072;
                            }
                    fgets(buffer,200,NEU_INP);
                    LOAD_SET[LOAD_NUM].ET_NUM = 0;
                    fgetpos(NEU_INP,&(LOAD_SET[LOAD_NUM].et_file));
                    fscanf(NEU_INP,"%d,",&CHECK1);
        label4073:
                    if(CHECK1!=-1)
                            {
    LOAD_SET[LOAD_NUM].ET_NUM=LOAD_SET[LOAD_NUM].ET_NUM+1;
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                            goto label4073;
                            }
                    fgets(buffer,200,NEU_INP);
                    LOAD_NUM=LOAD_NUM+1;
                    goto label407;
                    }
            FLAG=1;
            break;
        case 408:
            //********************Process the group information********************//
            label408:
            fscanf(NEU_INP,"%d",&CHECK);
            fgets(buffer,200,NEU_INP);
            if(CHECK!=-1)
                    {
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d,",&CHECK1);
                    while(CHECK1!=-1)
                            {
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                            }
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d,",&CHECK1);
                    while(CHECK1!=-1)
                            {
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                            }
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d,",&CHECK1);
                    while(CHECK1!=-1)
                            {
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                            }
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                    while(CHECK1!=-1)
                            {
                            fgets(buffer,200,NEU_INP);
                            fscanf(NEU_INP,"%d,",&CHECK1);
                            }
                            fgets(buffer,200,NEU_INP);
                    goto label408;
```

```
                }
        FLAG=1;
        break;
case 409:
        //********************Process the view information*********************//
        label409:
        fscanf(NEU_INP,"%d",&CHECK);
        fgets(buffer,200,NEU_INP);
        if(CHECK!=-1)
                {
                fgets(buffer,200,NEU_INP);
                for(i=0;i<9;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<8;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<4;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                for(i=0;i<3;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                fscanf(NEU_INP,"%d",&II);
                fgets(buffer,200,NEU_INP);
                for(i=0;i<II-1;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                        fscanf(NEU_INP,"%d",&CHECK1);
                        fgets(buffer,200,NEU_INP);
                while(CHECK1!=-1)
                        {
                        fscanf(NEU_INP,"%d",&CHECK1);
                        fgets(buffer,200,NEU_INP);
                        }
                goto label409;
                }
        FLAG=1;
        break;
case 411:
```

```
                    //********************Process the Report Format
information*********************//
                    label411:
                    fscanf(NEU_INP,"%d",&CHECK);
                    fgets(buffer,200,NEU_INP);
                    if(CHECK!=-1)
                              {
                              fgets(buffer,200,NEU_INP);
                              fscanf(NEU_INP,"%d",&II);
                              fgets(buffer,200,NEU_INP);
                              for(i=0;i<II-1;i++)
                                        {
                                        fgets(buffer,200,NEU_INP);
                                        }
                              fscanf(NEU_INP,"%d",&II);
                              fgets(buffer,200,NEU_INP);
                              for(i=0;i<II-1;i++)
                                        {
                                        fgets(buffer,200,NEU_INP);
                                        }
                              goto label411;
                              }
                    FLAG=1;
                    break;
          case 420:
                    //********************Process the function information*********************//
                    label420:
                    fscanf(NEU_INP,"%d",&CHECK);
                    fgets(buffer,200,NEU_INP);
                    if(CHECK!=-1)
                              {
                              fgets(buffer,200,NEU_INP);
                              fscanf(NEU_INP,"%d",&CHECK1);
                              if(CHECK1!=-1)
                                        {
                                        fgets(buffer,200,NEU_INP);
                                        }
                              goto label420;
                              }
                    FLAG=1;
                    break;
          case 412:
                    //********************Process the active data information*********************//
                    fgets(buffer,200,NEU_INP);
                    fscanf(NEU_INP,"%d",&CHECK);
                    fgets(buffer,200,NEU_INP);
                    FLAG=1;
                    break;
          case 450:
                    case_i=0;
                    label450:
                    fscanf(NEU_INP,"%d",&CHECK);
                    fgets(buffer,200,NEU_INP);
                    if(CHECK!=-1)
                    {
                              case_i=case_i+1;
                              fgets(set_name[CHECK-1],30,NEU_INP);
                              fgets(buffer,200,NEU_INP);
                              fscanf(NEU_INP,"%lg",TIME+CHECK-1);
                              fgets(buffer,200,NEU_INP);
                              fscanf(NEU_INP,"%d",&I1);
                              fgets(buffer,200,NEU_INP);
                              for(i=0;i<I1;i++)
                                        {
                                        fgets(buffer,200,NEU_INP);
                                        }
                              goto label450;
                    }
                    FLAG=1;
                    break;
```

```
                        case 451:
                                output_set_num=0;

                label451:
                                fscanf(NEU_INP,"%d",&CHECK);
                                fgets(buffer,200,NEU_INP);
                                if(CHECK!=-1)
                                {
                                        case_num[output_set_num]=CHECK;
                                        fgets(out_set_name[output_set_num],30,NEU_INP);
                                        fscanf(NEU_INP,"%lg,%lg,%lg",MIN_VALUE+output_set_num,

        MAX_VALUE+output_set_num,AMAX_VALUE+output_set_num);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);

        fscanf(NEU_INP,"%ld,%ld,%d,%d",&CHECKD,&NODE_i,&I1,TYPE+output_set_num);
                                        fgets(buffer,200,NEU_INP);
                                        fgets(buffer,200,NEU_INP);
                                        fgetpos(NEU_INP,file_output+output_set_num);
                                        output_set_num=output_set_num+1;
                                        ELEMENT_i=0;
                                        label4511:
                                        fscanf(NEU_INP,"%ld,",&CHECKD);
                                        if(CHECKD!=-1)
                                                {
                                                fgets(buffer,200,NEU_INP);
                                                        ELEMENT_i=ELEMENT_i+1;
                                                goto label4511;
                                                }
                                        fgets(buffer,200,NEU_INP);
                                        Total_num[output_set_num-1]=ELEMENT_i;
                                        goto label451;
                                }
                                FLAG=-1;
                                break;
                        }

                }
        }

//********************allocation dynamic memory for node data********************//

 NODE_P =(struct NODE_DATA *) malloc(NODE_NUM*(sizeof(NODE_DATA)+1));
 if( NODE_P == NULL )
                {
                printf("Insufficient memory available for node data\n" );
                getchar();
                }
 else{
        printf("\n");
                }
//********************allocation dynamic memory for element data********************//

ELEMENT_TMP =(struct ELEMENT_REL *) calloc(sizeof(ELEMENT_REL),ELEMENT_NUM1);
 if( ELEMENT_TMP == NULL )
                {
                printf("Insufficient memory available for element relation data\n" );
                getchar();

                }
 else{
        printf("\n");
                }

//********************allocation dynamic memory for element p data********************//

ELEMENT_P =(struct ELEMENT_DATA *) calloc(sizeof(ELEMENT_DATA),ELEMENT_NUM);
 if( ELEMENT_P == NULL )
```

```
                    {
                    printf("Insufficient memory available for element data\n" );
                    getchar();
                    }
 else{
            printf("\n");
                    }

//*******************************Coordinate Information*****************//

COORDINATE_P=(struct COORDINATE *)calloc(sizeof(COORDINATE),COORDINATE_NUM);

if(COORDINATE_P==NULL)
{
        printf("Insufficient memory for coordinate system data\n");
        getchar();
}

fsetpos(NEU_INP,&file_coordinate);

for(I2=0;I2<COORDINATE_NUM;I2++)
{
        fscanf(NEU_INP,"%d",&I1);
        (COORDINATE_P+I2)->A=I1;
        fscanf(NEU_INP,",%d,%d",&((COORDINATE_P+I2)->B),&((COORDINATE_P+I2)->C));
        fgets(buffer,200,NEU_INP);
        fgets((COORDINATE_P+I2)->D,25,NEU_INP);

        for(I3=0;I3<3;I3++)
        {
                fscanf(NEU_INP,"%lg,",&((COORDINATE_P+I2)->E[I3]));
        }
        fgets(buffer,200,NEU_INP);
        for(I3=0;I3<3;I3++)
        {
                fscanf(NEU_INP,"%lg,",&((COORDINATE_P+I2)->F[I3]));
        }
}

//*********************************Constraint Information*********************//

if(CONSTRAINT_NUM > 0)
{
        strcpy(gui_name, "CBP");                                        // CPB name id of constraint
boolean prompt
        bool = bool_prompt(gui_name);                                   // Prompt for loading constraint information
into memory

        FLAG = 1;
        while(FLAG == 1)
        {
                if(strncmp(bool, "YES", 3) == 0)                        // If yes is pressed
                {
                        CONSTRAINT_YES = 1;

                        for(I2 = 0; I2 < CONSTRAINT_NUM; I2++)
                        {
                                CONSTRAINT_SET[I2].ID=(long int *)calloc(sizeof(long
int),CONSTRAINT_SET[I2].NUM);
                                CONSTRAINT_SET[I2].INDEX=(int
*)calloc(sizeof(int),(CONSTRAINT_SET[I2].NUM)*6);

                                if((CONSTRAINT_SET[I2].ID==NULL)||(CONSTRAINT_SET[I2].INDEX==NULL))
                                {
                                        printf("Insufficient memory for data in constraint set #%d\n", I2+1);
                                        getchar();
                                }

//*************************************************************************//
```

```
                                        fsetpos(NEU_INP,&(CONSTRAINT_SET[I2].file_constraint));

                                        for(I3 = 0; I3 < CONSTRAINT_SET[I2].NUM; I3++)
                                        {

        fscanf(NEU_INP,"%ld,%d,%d,%d,%d,%d,%d,%d,%d",(CONSTRAINT_SET[I2].ID)+I3,&I4,&I1,

        CONSTRAINT_SET[I2].INDEX+I3*6,CONSTRAINT_SET[I2].INDEX+(I3*6+1),

        CONSTRAINT_SET[I2].INDEX+(I3*6+2),CONSTRAINT_SET[I2].INDEX+(I3*6+3),

        CONSTRAINT_SET[I2].INDEX+(I3*6+4),CONSTRAINT_SET[I2].INDEX+(I3*6+5));
                                                fgets(buffer,200,NEU_INP);
                                        }
                                }

                                FLAG=0;

                                for(i = 0; i < CONSTRAINT_NUM; i++)
                                strcpy(constraintset_names[i], CONSTRAINT_SET[i].B);

                                constraintset_name = constraintset_prompt(constraintset_names, CONSTRAINT_NUM); //
Function that calls load set name prompt

                                for(i = 0; i < CONSTRAINT_NUM; i++)                                    // loop to
find CONSTRAINTSET_PICK
                                        if(strcmp(constraintset_name, constraintset_names[i]) == 0)
                                                CONSTRAINTSET_PICK = i + 1;

                                CONSTRAINTSET_NUM = CONSTRAINT_SET[CONSTRAINTSET_PICK].NUM;
                        }

                        else if(strncmp(bool, "NO", 2) == 0)            // If no is pressed
                        {
                                FLAG = 0;
                        }
                }
}

//*************************************************************************//

MATERIAL_P=(struct MATERIAL *)calloc(sizeof(MATERIAL),MATERIAL_NUM);
if(MATERIAL_P==NULL)
        {
        printf("Insufficient memory for material data\n");
        getchar();
        }
fsetpos(NEU_INP,&file_mat);
for (I2=0;I2<MATERIAL_NUM;I2++)
        {
        fscanf(NEU_INP,"%d", &I1);
        fgets(buffer,200,NEU_INP);
                        {
                (MATERIAL_P+I2)->A=I1;
                fgets((MATERIAL_P+I2)->title,25,NEU_INP);

                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Young_Modulus[I3]));
                        }
                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Shear_Modulus[I3]));
                        }
                for(I3=0;I3<3;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->Poisson_Ratio[I3]));
                        }
                for(I3=0;I3<21;I3++)
                        {
```

```
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->GMatrix[I3]));
                        }
                for(I3=0;I3<6;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->alpha[I3]));
                        }
                for(I3=0;I3<6;I3++)
                        {
                        fscanf(NEU_INP,"%lg,",&((MATERIAL_P+I2)->k[I3]));
                        }
                        fscanf(NEU_INP,"%lg,%lg,%lg,%lg,",&((MATERIAL_P+I2)->thermal_cap),
                        &((MATERIAL_P+I2)->density),&((MATERIAL_P+I2)->damping),&((MATERIAL_P+I2)-
>temperature));
                        fscanf(NEU_INP,"%lg,%lg,%lg,%lg,%lg", &((MATERIAL_P+I2)->tension_limit[0]),
                        &((MATERIAL_P+I2)->tension_limit[1]),&((MATERIAL_P+I2)->comp_limit[0]),
                        &((MATERIAL_P+I2)->comp_limit[1]),&((MATERIAL_P+I2)->shear_limit));
                for(I3=0;I3<17;I3++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                }
}

//****************************************Load Information*******************//

if(LOAD_NUM > 0)
{
        strcpy(gui_name, "LBP");
        bool = bool_prompt(gui_name);                                   // Prompt for loading constraint information
into memory

        FLAG=1;
        while(FLAG==1)
        {
                if(strncmp(bool, "YES", 3) == 0)
                {
                        LOAD_YES = 1;
                        for(I1 = 0; I1 < LOAD_NUM; I1++)
                        {

                                if(LOAD_SET[I1].NUM != 0)
                                {
                                        LOAD_SET[I1].ID = (long int *)calloc(sizeof(long
int),LOAD_SET[I1].NUM);
                                        if(LOAD_SET[I1].ID == NULL)
                                        {
                                                printf("Insufficient memory for data in load set #\n",I1+1);
                                                getchar();
                                        }
                                        LOAD_SET[I1].TYPE=(int *)calloc(sizeof(int),LOAD_SET[I1].NUM);

                                        if(LOAD_SET[I1].TYPE==NULL)
                                        {
                                                printf("Insufficient memory for data in  load set #\n",I1+1);
                                                getchar();
                                        }
                                        LOAD_SET[I1].FACE=(int *)calloc(sizeof(int),(LOAD_SET[I1].NUM)*6);

                                        if(LOAD_SET[I1].FACE==NULL)
                                        {
                                                printf("Insufficient memory for data in  load set #\n",I1+1);
                                                getchar();
                                        }
                                        LOAD_SET[I1].VALUE=(double
*)calloc(sizeof(double),(LOAD_SET[I1].NUM)*8);

                                        if(LOAD_SET[I1].VALUE==NULL)
                                        {
                                                printf("Insufficient memory for data in  load set #\n",I1+1);
                                                getchar();
```

```
                                        }

//*************************************************************************//

                                        fsetpos(NEU_INP,&(LOAD_SET[I1].load_file));

                                        for(CHECKD = 0; CHECKD < LOAD_SET[I1].NUM; CHECKD++)
                                        {
                                                fscanf(NEU_INP,"%ld,%d",LOAD_SET[I1].ID + CHECKD,
LOAD_SET[I1].TYPE + CHECKD);
                                                fgets(buffer,200,NEU_INP);

        fscanf(NEU_INP,"%lg,%lg",LOAD_SET[I1].VALUE+(CHECKD*8),LOAD_SET[I1].VALUE+(CHECKD*8+1));

                                                fgets(buffer,200,NEU_INP);

                                                for (CHECK1 = 0; CHECK1 < 6; CHECK1++)
                                                {

        fscanf(NEU_INP,"%d,%lg",LOAD_SET[I1].FACE+(CHECKD*6+CHECK1),LOAD_SET[I1].VALUE+(CHECKD*8+2
+CHECK1));
                                                        fgets(buffer,200,NEU_INP);
                                                }

                                                for(I4 = 0;I4 < 4; I4++)
                                                {
                                                        fgets(buffer,200,NEU_INP);
                                                }

                                        }
                                }

                                if(LOAD_SET[I1].NT_NUM != 0)
                                {
                                        LOAD_SET[I1].NT_ID=(long int *)calloc(sizeof(long
int),LOAD_SET[I1].NT_NUM);

                                        if(LOAD_SET[I1].NT_ID==NULL)
                                        {
                                                printf("Insufficient memory for data in  load set #\n",I1+1);
                                                getchar();
                                        }
                                        LOAD_SET[I1].NT_VALUE=(double
*)calloc(sizeof(double),LOAD_SET[I1].NT_NUM);

                                        if(LOAD_SET[I1].NT_VALUE==NULL)
                                        {
                                                printf("Insufficient memory for data in  load set #\n",I1+1);
                                                getchar();
                                        }

//*************************************************************************//

                                        fsetpos(NEU_INP,&(LOAD_SET[I1].nt_file));

                                        for(CHECKD = 0; CHECKD < LOAD_SET[I1].NT_NUM; CHECKD++)
                                        {

        fscanf(NEU_INP,"%ld,%d,%d,%lg",LOAD_SET[I1].NT_ID+CHECKD,&I3,&I4,LOAD_SET[I1].NT_VALUE+CHECK
D);
                                                fgets(buffer,200,NEU_INP);
                                        }
                                }

                                if(LOAD_SET[I1].ET_NUM != 0)
                                {
                                        LOAD_SET[I1].ET_ID=(long int *)calloc(sizeof(long
int),LOAD_SET[I1].ET_NUM);

                                        if(LOAD_SET[I1].ET_ID==NULL)
```

```
                                                {
                                                        printf("Insufficient memory for data in  load set #\n",I1+1);
                                                        getchar();
                                                }
                                                LOAD_SET[I1].ET_VALUE=(double
*)calloc(sizeof(double),LOAD_SET[I1].ET_NUM);

                                                if(LOAD_SET[I1].ET_VALUE==NULL)
                                                {
                                                        printf("Insufficient memory for data in  load set #\n",I1+1);
                                                        getchar();
                                                }

//*************************************************************************//

                                                fsetpos(NEU_INP,&(LOAD_SET[I1].et_file));

                                                for(CHECKD = 0;CHECKD < LOAD_SET[I1].ET_NUM; CHECKD++)
                                                {

        fscanf(NEU_INP,"%ld,%d,%d,%lg",LOAD_SET[I1].ET_ID+CHECKD,&I3,&I4,LOAD_SET[I1].ET_VALUE+CHECK
D);
                                                        fgets(buffer,200,NEU_INP);
                                                }
                                        }
                                }
                                FLAG = 0;

                                for(i = 0; i < LOAD_NUM; i++)
                                        strcpy(loadset_names[i], LOAD_SET[i].NAME);

                                loadset_name = loadset_prompt(loadset_names, LOAD_NUM);  // Function that calls load set name
prompt

                                for(i = 0; i < LOAD_NUM; i++)                                   // loop to find
LOADSET_PICK
                                        if(strcmp(loadset_name, loadset_names[i]) == 0)
                                                LOADSET_PICK = i + 1;

                                LOADSET_NUM = LOAD_SET[LOADSET_PICK].NUM;
                        }
                        else if(strncmp(bool, "NO", 2) == 0)
                        {
                                FLAG=0;
                        }
        }
}

//*************************************************************************//

ELEMENT_PROPERTY_P=(struct ELEMENT_PROPERTY *) calloc (sizeof(ELEMENT_PROPERTY),ELEMENT_PRO_NUM);
if(ELEMENT_PROPERTY_P==NULL)
        {
        printf("Insufficient memory for element property data\n");
        getchar();
        }
else{
        printf("\n");
        }
fsetpos(NEU_INP,&file_pro);

for (i=0;i<ELEMENT_PRO_NUM;i++)
        {
        fscanf(NEU_INP,"%d,%d,%d,%d",&CHECKD,&I1,&I2,&I3);
        (ELEMENT_PROPERTY_P+i)->A=I3;
        fgets(buffer,200,NEU_INP);
        fgets(buffer,200,NEU_INP);
        fgets(buffer,200,NEU_INP);
        fscanf(NEU_INP,"%d",&I1);
        fgets(buffer,200,NEU_INP);
```

```
        for(I1=0;I1<(float)(II/8)+1.0;I1++)
                {fgets(buffer,200,NEU_INP);
                }
        fscanf(NEU_INP,"%d",&II);
        fgets(buffer,200,NEU_INP);
        for (I3=0;I3<II;I3++)
                {
                fscanf(NEU_INP,"%lg,",&((ELEMENT_PROPERTY_P+i)->B[I3]));
                }
                fgets(buffer,200,NEU_INP);
        }

 fsetpos(NEU_INP,&file_node);

names =(struct NAMES *) malloc(sizeof(NAMES));//Dryer added 9/20/97


for (NODE_i=0;NODE_i<NODE_NUM;NODE_i++)
        {
         fscanf(NEU_INP,"%ld,",&CHECKD);
         (NODE_P+NODE_i)->A=CHECKD;
        for(i=0;i<10;i++)
                {
                fscanf(NEU_INP,"%d,",&I1);
                }
        fscanf(NEU_INP,"%lg,%lg,%lg",&X,&Y,&Z);
        (NODE_P+NODE_i)->x=X;
        (NODE_P+NODE_i)->y=Y;
        (NODE_P+NODE_i)->z=Z;
        (NODE_P+NODE_i)->dx=0.0;
        (NODE_P+NODE_i)->dy=0.0;
        (NODE_P+NODE_i)->dz=0.0;

        fgets(buffer,200,NEU_INP);
        for(i=0;i<5;i++)
                {
                (NODE_P+NODE_i)->output_data[i]=0.0;
                }
        }

//********************processing the output data********************//

        fsetpos(NEU_INP,&file_element);
        for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
                {
                fscanf(NEU_INP,"%ld",&CHECKD);
                (ELEMENT_TMP+ELEMENT_i)->A=CHECKD;
                for(i=0;i<7;i++)
                        {
                        fgets(buffer,200,NEU_INP);
                        }
                }

//*********************************clear screen*********************************//

        for (i=0;i<23;i++)
        {
                printf("\n");
        }

//*****************************Case prompt*********************************//

if(case_i != 0)
{
        case_name = case_prompt(set_name, case_i);
        strncpy(names->actual_case_name, case_name, strlen(case_name));

        for(i = 0; i < case_i; i++)                                        // striping carriage return
                strncpy(case_names[i], set_name[i], strlen(set_name[i]) - 1);
```

```
        for(i = 0; i < case_i; i++)                                    // loop to find case_n
                if(strcmp(case_name, case_names[i]) == 0)
                        case_n = i + 1;

        goto labelnnn;

//*************************************************************************//

labelnnn:
case_nn = 0;
for(i = 0; i < output_set_num; i++)
        {
        if(case_num[i] == case_n)
                {
                strcpy(temp_set_name[case_nn], out_set_name[i]);
                case_set_num[case_nn] = i;
                case_nn = case_nn + 1;
                }
        }

//*********************************clear screen*******************************//

        for (i=0;i<23;i++)
        {
                printf("\n");
        }

//*************************************************************************//

for(i=0;i<case_nn;i++)
        {
        if((!strncmp(temp_set_name[i],"T1 Translation",14))||(!strncmp(temp_set_name[i],"X Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        ELEMENT_i=FindNid(CHECKD);
                        fgets(buffer,200,NEU_INP);
                        (NODE_P+ELEMENT_i)->dx=X;
                        }
                }
        if((!strncmp(temp_set_name[i],"T2 Translation",14))||(!strncmp(temp_set_name[i],"Y Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        fgets(buffer,200,NEU_INP);
                        ELEMENT_i=FindNid(CHECKD);
                        (NODE_P+ELEMENT_i)->dy=X;
                        }
                }
        if((!strncmp(temp_set_name[i],"T3 Translation",14))||(!strncmp(temp_set_name[i],"Z Translation",13)))
                {
                fsetpos(NEU_INP,file_output+case_set_num[i]);
                for (NODE_i=0;NODE_i<Total_num[case_set_num[i]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                        fgets(buffer,200,NEU_INP);
                        ELEMENT_i=FindNid(CHECKD);
                        (NODE_P+ELEMENT_i)->dz=X;
                        }
                }
        }

//*****************************Output data sets*****************************//

        for(i = 0; i < case_nn; i++){                                  // include whether output is nodal or
elemental
```

```
                if(TYPE[case_set_num[i]] == 7)
                        sprintf(temp_name[i], "(Nodal) %s", temp_set_name[i]);
                if(TYPE[case_set_num[i]] == 8)
                        sprintf(temp_name[i], "(Elemental) %s", temp_set_name[i]);
        }

        output_data = output_data_prompt(temp_name, case_nn);          // Function that calls output set name prompt
        j = 0;
        while(output_data[j + 1] != NULL && j < 10){
                for(i = 0; i < case_nn; i++)        // loop to find case_
                {
                        if(strncmp(strchr(output_data[j], ' ') + 1, temp_set_name[i], strlen(strchr(output_data[j], ' ') + 1)) ==
0)
                        {
                                if(strncmp(output_data[j], "(Elemental)", 11) == 0 && k < 5){
                                        V[k++] = i;
                                        V_NUM = V_NUM - 1;
                                }
                                else if(strncmp(output_data[j], "(Nodal)", 7) == 0 && m < 5){
                                        U[m++] = i;
                                        U_NUM = U_NUM - 1;
                                }
                        }
                }
                j = j + 1;
        }

        goto labelxxx;

//*************************************************************************//

labelxxx:

        if(V_NUM != 5)
                {

                for(i = 0; i < 5 - V_NUM; i++)
                        {
                                fsetpos(NEU_INP,file_output+case_set_num[V[i]]);
                        for(ELEMENT_i=0;ELEMENT_i<Total_num[case_set_num[V[i]]];ELEMENT_i++)
                                {
                                fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);
                                (ELEMENT_TMP+FindEid(CHECKD))->data[i]=X;
                                fgets(buffer,200,NEU_INP);
                                }
                        }
                }
        else
                {
                for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
                        {
                        for(I1=0;I1<5;I1++)
                        {
                        (ELEMENT_TMP+ELEMENT_i)->data[I1]=0.0;
                        }
                        }
                }
        if(U_NUM != 5)
        {if(U_NUM < 0){U_NUM = 0;}

        //*****************Processing the nodal output data***************************//

        for(i = 0; i < 5 - U_NUM; i++)
                {
                fsetpos(NEU_INP,file_output+case_set_num[U[i]]);
                for(NODE_i=0;NODE_i<Total_num[case_set_num[U[i]]];NODE_i++)
                        {
                        fscanf(NEU_INP,"%ld,%lg",&CHECKD,&X);

                        (NODE_P+FindNid(CHECKD))->output_data[i]=X;
```

```
                    fgets(buffer,200,NEU_INP);
                    }
                }
        }
}

fsetpos(NEU_INP, &file_element);
NODE_i=0;
for (ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM1;ELEMENT_i++)
        {
         fscanf(NEU_INP,"%ld,%d,%d,%d,%d,",&CHECKDD,&I1,&I2,&I3,&I4);
         fgets(buffer,200,NEU_INP);
        switch(I4)
                    {
                    case 0:
                            for(i=0;i<20;i++)
                                    {
                                    fscanf(NEU_INP,"%ld,",&CHECKD);
                                    ID[i]=FindNid(CHECKD);
                                    }
                            (ELEMENT_P+NODE_i)->A=2;
                            (ELEMENT_P+NODE_i)->D=CHECKDD;
                                    (ELEMENT_P+NODE_i)->B[0]=ID[0];
                                    (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                    (ELEMENT_P+NODE_i)->B[2]=-1;
                                    (ELEMENT_P+NODE_i)->B[3]=-1;
                                    (ELEMENT_P+NODE_i)->F=I3;
                    for(i=0;i<5;i++)
                                            {
                                            (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                            }
                            NODE_i=NODE_i+1;
                            fgets(buffer,200,NEU_INP);
                            break;
                    case 2:
                            for(i=0;i<20;i++)
                                    {
                                    fscanf(NEU_INP,"%ld,",&CHECKD);
                                    ID[i]=FindNid(CHECKD);
                                    }
                            (ELEMENT_P+NODE_i)->A=3;
                            (ELEMENT_P+NODE_i)->D=CHECKDD;

                            for(I1=0;I1<4;I1++)
                                    {
                                    (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                                    }
                            (ELEMENT_P+NODE_i)->F=I3;
                            for(i=0;i<5;i++)
                                            {
                                            (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                            }
                            NODE_i=NODE_i+1;

                            fgets(buffer,200,NEU_INP);
                            break;
                     case 3:
                            for(i=0;i<20;i++)
                                    {
                                    fscanf(NEU_INP,"%ld,",&CHECKD);
                                    ID[i]=FindNid(CHECKD);
                                    }
                                    (ELEMENT_P+NODE_i)->A=3;
                                    (ELEMENT_P+NODE_i)->D=CHECKDD;

                            for(I1=0;I1<4;I1++)
                                    {
                                    (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
```

```
                }
            (ELEMENT_P+NODE_i)->F=I3;
            for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
            NODE_i=NODE_i+1;
            fgets(buffer,200,NEU_INP);
            break;
        case 4:
            for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
                }
            (ELEMENT_P+NODE_i)->A=4;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            for(I1=0;I1<4;I1++)
                {
                (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                }
            (ELEMENT_P+NODE_i)->F=I3;
            for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
            NODE_i=NODE_i+1;
            fgets(buffer,200,NEU_INP);
            break;
        case 5:
            for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
                }
                (ELEMENT_P+NODE_i)->A=4;
                        (ELEMENT_P+NODE_i)->D=CHECKDD;

            for(I1=0;I1<4;I1++)
                {
                (ELEMENT_P+NODE_i)->B[I1]=ID[I1];
                }
            (ELEMENT_P+NODE_i)->F=I3;                       //revised Sept. 30,
1997
            for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
            NODE_i=NODE_i+1;
            fgets(buffer,200,NEU_INP);
            break;
        case 6:
            for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
                }
            fgets(buffer,200,NEU_INP);
            (ELEMENT_P+NODE_i)->A=3;
            (ELEMENT_P+NODE_i)->D=CHECKDD;

            (ELEMENT_P+NODE_i)->B[0]=ID[0];
            (ELEMENT_P+NODE_i)->B[1]=ID[1];
            (ELEMENT_P+NODE_i)->B[2]=ID[2];
            (ELEMENT_P+NODE_i)->B[3]=ID[3];
                    (ELEMENT_P+NODE_i)->E=1000;
            (ELEMENT_P+NODE_i)->F=I3;
            for(i=0;i<5;i++)
```

```
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[1];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[4];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        break;
case 7:
        for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
                }
        fgets(buffer,200,NEU_INP);
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[2];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;
```

```
(ELEMENT_P+NODE_i)->B[0]=ID[4];
(ELEMENT_P+NODE_i)->B[1]=ID[5];
(ELEMENT_P+NODE_i)->B[2]=ID[6];
(ELEMENT_P+NODE_i)->B[3]=ID[3];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
(ELEMENT_P+NODE_i)->B[1]=ID[1];
(ELEMENT_P+NODE_i)->B[2]=ID[5];
(ELEMENT_P+NODE_i)->B[3]=ID[4];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[1];
(ELEMENT_P+NODE_i)->B[1]=ID[2];
(ELEMENT_P+NODE_i)->B[2]=ID[5];
(ELEMENT_P+NODE_i)->B[3]=ID[4];
        (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[2];
(ELEMENT_P+NODE_i)->B[1]=ID[0];
(ELEMENT_P+NODE_i)->B[2]=ID[4];
(ELEMENT_P+NODE_i)->B[3]=ID[6];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
break;
case 8:
for(i=0;i<20;i++)
        {
        fscanf(NEU_INP,"%ld,",&CHECKD);
        ID[i]=FindNid(CHECKD);
        }
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
(ELEMENT_P+NODE_i)->B[1]=ID[3];
(ELEMENT_P+NODE_i)->B[2]=ID[2];
(ELEMENT_P+NODE_i)->B[3]=ID[1];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
```

```
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[4];
(ELEMENT_P+NODE_i)->B[1]=ID[5];
(ELEMENT_P+NODE_i)->B[2]=ID[6];
(ELEMENT_P+NODE_i)->B[3]=ID[7];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
(ELEMENT_P+NODE_i)->B[1]=ID[1];
(ELEMENT_P+NODE_i)->B[2]=ID[5];
(ELEMENT_P+NODE_i)->B[3]=ID[4];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[1];
(ELEMENT_P+NODE_i)->B[1]=ID[2];
(ELEMENT_P+NODE_i)->B[2]=ID[6];
(ELEMENT_P+NODE_i)->B[3]=ID[5];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[2];
(ELEMENT_P+NODE_i)->B[1]=ID[3];
(ELEMENT_P+NODE_i)->B[2]=ID[7];
(ELEMENT_P+NODE_i)->B[3]=ID[6];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[3];
(ELEMENT_P+NODE_i)->B[1]=ID[0];
(ELEMENT_P+NODE_i)->B[2]=ID[4];
(ELEMENT_P+NODE_i)->B[3]=ID[7];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
```

```
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                fgets(buffer,200,NEU_INP);
                break;
case 9:
                fgets(buffer,200,NEU_INP);
                fgets(buffer,200,NEU_INP);
                break;
case 10:
                for(i=0;i<20;i++)
                        {
                        fscanf(NEU_INP,"%ld,",&CHECKD);
                        ID[i]=FindNid(CHECKD);
                        }
                fgets(buffer,200,NEU_INP);
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[2];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        (ELEMENT_P+NODE_i)->E=1000;
                (ELEMENT_P+NODE_i)->F=I3;
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                (ELEMENT_P+NODE_i)->B[2]=ID[4];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        (ELEMENT_P+NODE_i)->E=1000;
                (ELEMENT_P+NODE_i)->F=I3;
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[4];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        (ELEMENT_P+NODE_i)->E=1000;
                (ELEMENT_P+NODE_i)->F=I3;
                for(i=0;i<5;i++)
                        {
                        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                        }
                NODE_i=NODE_i+1;
                (ELEMENT_P+NODE_i)->A=3;
                (ELEMENT_P+NODE_i)->D=CHECKDD;

                (ELEMENT_P+NODE_i)->B[0]=ID[1];
                (ELEMENT_P+NODE_i)->B[1]=ID[2];
                (ELEMENT_P+NODE_i)->B[2]=ID[4];
                (ELEMENT_P+NODE_i)->B[3]=ID[3];
                        (ELEMENT_P+NODE_i)->E=1000;
                (ELEMENT_P+NODE_i)->F=I3;
                for(i=0;i<5;i++)
```

```
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        break;
case 11:
        for(i=0;i<20;i++)
                {
                fscanf(NEU_INP,"%ld,",&CHECKD);
                ID[i]=FindNid(CHECKD);
                }
        fgets(buffer,200,NEU_INP);
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[3];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[1];
        (ELEMENT_P+NODE_i)->B[3]=ID[0];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=3;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[4];
        (ELEMENT_P+NODE_i)->B[1]=ID[5];
        (ELEMENT_P+NODE_i)->B[2]=ID[6];
        (ELEMENT_P+NODE_i)->B[3]=ID[3];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[0];
        (ELEMENT_P+NODE_i)->B[1]=ID[1];
        (ELEMENT_P+NODE_i)->B[2]=ID[5];
        (ELEMENT_P+NODE_i)->B[3]=ID[4];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;

        (ELEMENT_P+NODE_i)->B[0]=ID[1];
        (ELEMENT_P+NODE_i)->B[1]=ID[2];
        (ELEMENT_P+NODE_i)->B[2]=ID[6];
        (ELEMENT_P+NODE_i)->B[3]=ID[5];
                (ELEMENT_P+NODE_i)->E=1000;
        (ELEMENT_P+NODE_i)->F=I3;
        for(i=0;i<5;i++)
                {
                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                }
        NODE_i=NODE_i+1;
        (ELEMENT_P+NODE_i)->A=4;
        (ELEMENT_P+NODE_i)->D=CHECKDD;
```

```
(ELEMENT_P+NODE_i)->B[0]=ID[2];
(ELEMENT_P+NODE_i)->B[1]=ID[0];
(ELEMENT_P+NODE_i)->B[2]=ID[4];
(ELEMENT_P+NODE_i)->B[3]=ID[6];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
break;
case 12:
for(i=0;i<20;i++)
        {
        fscanf(NEU_INP,"%ld,",&CHECKD);
        ID[i]=FindNid(CHECKD);
        }
fgets(buffer,200,NEU_INP);
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
(ELEMENT_P+NODE_i)->B[1]=ID[1];
(ELEMENT_P+NODE_i)->B[2]=ID[2];
(ELEMENT_P+NODE_i)->B[3]=ID[3];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[4];
(ELEMENT_P+NODE_i)->B[1]=ID[5];
(ELEMENT_P+NODE_i)->B[2]=ID[6];
(ELEMENT_P+NODE_i)->B[3]=ID[7];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[0];
(ELEMENT_P+NODE_i)->B[1]=ID[1];
(ELEMENT_P+NODE_i)->B[2]=ID[5];
(ELEMENT_P+NODE_i)->B[3]=ID[4];
        (ELEMENT_P+NODE_i)->E=1000;
(ELEMENT_P+NODE_i)->F=I3;
for(i=0;i<5;i++)
        {
        (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
        }
NODE_i=NODE_i+1;
(ELEMENT_P+NODE_i)->A=4;
(ELEMENT_P+NODE_i)->D=CHECKDD;

(ELEMENT_P+NODE_i)->B[0]=ID[1];
(ELEMENT_P+NODE_i)->B[1]=ID[2];
(ELEMENT_P+NODE_i)->B[2]=ID[6];
(ELEMENT_P+NODE_i)->B[3]=ID[5];
        (ELEMENT_P+NODE_i)->E=1000;
```

```
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        for(i=0;i<5;i++)
                                                {
                                                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                }
                                        NODE_i=NODE_i+1;
                                        (ELEMENT_P+NODE_i)->A=4;
                                        (ELEMENT_P+NODE_i)->D=CHECKDD;

                                        (ELEMENT_P+NODE_i)->B[0]=ID[2];
                                        (ELEMENT_P+NODE_i)->B[1]=ID[3];
                                        (ELEMENT_P+NODE_i)->B[2]=ID[7];
                                        (ELEMENT_P+NODE_i)->B[3]=ID[6];
                                                (ELEMENT_P+NODE_i)->E=1000;
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        for(i=0;i<5;i++)
                                                {
                                                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                }
                                        NODE_i=NODE_i+1;
                                        (ELEMENT_P+NODE_i)->A=4;
                                        (ELEMENT_P+NODE_i)->D=CHECKDD;

                                        (ELEMENT_P+NODE_i)->B[0]=ID[3];
                                        (ELEMENT_P+NODE_i)->B[1]=ID[0];
                                        (ELEMENT_P+NODE_i)->B[2]=ID[4];
                                        (ELEMENT_P+NODE_i)->B[3]=ID[7];
                                                (ELEMENT_P+NODE_i)->E=1000;
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        for(i=0;i<5;i++)
                                                {
                                                (ELEMENT_P+NODE_i)->C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                }
                                        NODE_i=NODE_i+1;
                                        break;
                                case 13:

                                        for(i=0;i<20;i++)
                                                {
                                                fscanf(NEU_INP,"%ld,",&CHECKD);
                                                ID[i]=FindNid(CHECKD);
                                                }
                                        (ELEMENT_P+NODE_i)->A=4;
                                        (ELEMENT_P+NODE_i)->D=CHECKDD;

                                        (ELEMENT_P+NODE_i)->B[0]=ID[0];
                                        (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[2]=ID[1];
                                        (ELEMENT_P+NODE_i)->B[3]=ID[0];
                                        (ELEMENT_P+NODE_i)->F=I3;
                                        for(i=0;i<5;i++)
                                                        {
                                                        (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                        }
                                        NODE_i=NODE_i+1;
                                        fgets(buffer,200,NEU_INP);

                                        break;
                                }
        for(i=0;i<4;i++)
                {
                fgets(buffer,200,NEU_INP);
                }
        if(I4==13)
                                        {
                                        CHECKD=0;
                                        while(CHECKD!=-1)
                                                {
                                                fscanf(NEU_INP,"%ld,",&CHECKD);
```

```
                                                fgets(buffer,200,NEU_INP);

                                                if(CHECKD!=-1){
                                        ID[1]=FindNid(CHECKD);
                                (ELEMENT_P+NODE_i)->A=4;
                                (ELEMENT_P+NODE_i)->D=CHECKDD;

                                (ELEMENT_P+NODE_i)->B[0]=ID[0];
                                (ELEMENT_P+NODE_i)->B[1]=ID[1];
                                (ELEMENT_P+NODE_i)->B[2]=ID[1];
                                (ELEMENT_P+NODE_i)->B[3]=ID[0];
                                (ELEMENT_P+NODE_i)->F=I3;
                                for(i=0;i<5;i++)
                                                {
                                                (ELEMENT_P+NODE_i)-
>C[i]=(ELEMENT_TMP+ELEMENT_i)->data[i];
                                                }
                                NODE_i=NODE_i+1;}
                                                }
                                        }
        }

for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM;ELEMENT_i++)
        {
        flag=0;
        if((ELEMENT_P+ELEMENT_i)->E==1000)
                {
                NODE_i=ELEMENT_i+1;
                while((NODE_i<ELEMENT_NUM)&&(flag==0))
                {
                        if((ELEMENT_P+NODE_i)->E==1000)
                        {
                flag=compare(ELEMENT_i,NODE_i);

                        }
                if(flag==1)
                {
                        (ELEMENT_P+ELEMENT_i)->E=1;
                        (ELEMENT_P+NODE_i)->E=1;
                        }
                else{
                        NODE_i=NODE_i+1;
                        }
                }
                if(flag==0)
                {
(ELEMENT_P+ELEMENT_i)->E=0;
                        (ELEMENT_P+NODE_i)->E=0;
                        }
        }
}

fclose(NEU_INP);

//*************************Filter Internal Surface**********************//

if(flag_solid == 1)
{
        strcpy(gui_name, "FBP");                                        // FPB name id of constraint
boolean prompt

        bool = bool_prompt(gui_name);                                   // Prompt to filter internal surfaces

        FLAG = 1;
        while(FLAG == 1)
        {
                if(strncmp(bool, "YES", 3) == 0)                        // If yes is pressed
                {
                        tmp1 = fopen("test.tmp", "w+");
                        tmp2 = fopen("test1.tmp","w+");
```

```
                    ELEMENT_NUM_S=0;
                    for (i = 0; i < ELEMENT_NUM; i++)
                    {
                            if((ELEMENT_P + i)->E == 0)
                            {
                                    ELEMENT_NUM_S = ELEMENT_NUM_S + 1;
                                    fprintf(tmp2,
"%ld\t%d\t%ld\t%ld\t%ld\t%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\t%d\n",
                                    (ELEMENT_P+i)->D,(ELEMENT_P+i)->A,(ELEMENT_P+i)-
>B[0],(ELEMENT_P+i)->B[1],
                                    (ELEMENT_P+i)->B[2],(ELEMENT_P+i)->B[3],(ELEMENT_P+i)->C[0],
                                    (ELEMENT_P+i)->C[1],(ELEMENT_P+i)->C[2],(ELEMENT_P+i)->C[3],
                                    (ELEMENT_P+i)->C[4],(ELEMENT_P+i)->E,(ELEMENT_P+i)->F);
                                    (NODE_P + ((ELEMENT_P + i)->B[0]))->H = 1;
                                    (NODE_P + ((ELEMENT_P + i)->B[1]))->H = 1;
                                    (NODE_P + ((ELEMENT_P + i)->B[2]))->H = 1;
                                    (NODE_P + ((ELEMENT_P + i)->B[3]))->H = 1;
                            }
                    }
                    NODE_NUM_S = 0;

            for (i = 0; i < NODE_NUM; i++)
            {
                    if((NODE_P + i)->H == 1)
                    {
                            NODE_NUM_S=NODE_NUM_S + 1;
                            fprintf(tmp1,"%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\n",
                            i+1, (NODE_P + i)->x, (NODE_P + i)->y, (NODE_P + i)->z,
                            (NODE_P + i)->dx,(NODE_P + i)->dy, (NODE_P + i)->dz, (NODE_P + i)-
>output_data[0],
                            (NODE_P + i)->output_data[1], (NODE_P + i)->output_data[2],
                            (NODE_P + i)->output_data[3], (NODE_P + i)->output_data[4],
                            (NODE_P + i)->H);
                    }
            }
            rewind(tmp2);
            rewind(tmp1);
            ELEMENT_NUM=ELEMENT_NUM_S;
            NODE_NUM=NODE_NUM_S;
            free(ELEMENT_P);
            ELEMENT_P=(struct ELEMENT_DATA *)calloc(sizeof (ELEMENT_DATA),ELEMENT_NUM);
            free(NODE_P);
            NODE_P=(struct NODE_DATA *)calloc (sizeof(NODE_DATA),NODE_NUM);

            for (i=0;i<NODE_NUM;i++)
            {
                    fscanf(tmp1,"%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\n",
                    &((NODE_P+i)->A),&((NODE_P+i)->x),&((NODE_P+i)->y),
                    &((NODE_P+i)->z),&((NODE_P+i)->dx),&((NODE_P+i)->dy),
                    &((NODE_P+i)->dz),&((NODE_P+i)->output_data[0]),
                    &((NODE_P+i)->output_data[1]),&((NODE_P+i)->output_data[2]),
                    &((NODE_P+i)->output_data[3]),&((NODE_P+i)->output_data[4]),
                    &((NODE_P+i)->H));
            }
            for (i=0;i<ELEMENT_NUM;i++)
            {
                    fscanf(tmp2,"%ld\t%d\t%ld\t%ld\t%ld\t%ld\t%lg\t%lg\t%lg\t%lg\t%lg\t%d\t%d\n",
                    &((ELEMENT_P+i)->D),&((ELEMENT_P+i)->A),&IA,&IB,&IC,&IE,&((ELEMENT_P+i)-
>C[0]),
                    &((ELEMENT_P+i)->C[1]),&((ELEMENT_P+i)->C[2]),&((ELEMENT_P+i)->C[3]),
                    &((ELEMENT_P+i)->C[4]),&((ELEMENT_P+i)->E),&((ELEMENT_P+i)->F));
                    (ELEMENT_P+i)->B[0]=FindNid(IA+1);
                    (ELEMENT_P+i)->B[1]=FindNid(IB+1);
                    (ELEMENT_P+i)->B[2]=FindNid(IC+1);
                    (ELEMENT_P+i)->B[3]=FindNid(IE+1);
            }
            fclose(tmp1);
            fclose(tmp2);
            FLAG=0;
            }
```

```
                else if(strncmp(bool, "NO", 2) == 0)          // If no is pressed
                        FLAG = 0;
        }
}

//************************print to file - node data***************************//

fp = fopen("node.lst", "w+");
for(NODE_i=0;NODE_i<NODE_NUM;NODE_i++)
{
fprintf(fp,"%ld                %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg %10.5lg
%10.5lg\n",
        (NODE_P+NODE_i)->A,(NODE_P+NODE_i)->x,(NODE_P+NODE_i)->y,(NODE_P+NODE_i)->z,
        (NODE_P+NODE_i)->dx,(NODE_P+NODE_i)->dy,(NODE_P+NODE_i)->dz,
        (NODE_P+NODE_i)->output_data[0],
        (NODE_P+NODE_i)->output_data[1],(NODE_P+NODE_i)->output_data[2],
        (NODE_P+NODE_i)->output_data[3],(NODE_P+NODE_i)->output_data[4]);
}
fclose(fp);

//*****************************print to file - element data*********************************//

fp1=fopen("element.lst","w+");
for(ELEMENT_i=0;ELEMENT_i<ELEMENT_NUM;ELEMENT_i++)
{
fprintf(fp1,"%ld    %ld              %ld              %ld              %ld              %ld %10.5lg
%10.5lg %10.5lg %10.5lg %10.5lg %ld %ld\n",
        (ELEMENT_P+ELEMENT_i)->D,
        (ELEMENT_P+ELEMENT_i)->A,
        (ELEMENT_P+ELEMENT_i)->B[0],
        (ELEMENT_P+ELEMENT_i)->B[1],
        (ELEMENT_P+ELEMENT_i)->B[2],
        (ELEMENT_P+ELEMENT_i)->B[3],
        (ELEMENT_P+ELEMENT_i)->C[0],
        (ELEMENT_P+ELEMENT_i)->C[1],
        (ELEMENT_P+ELEMENT_i)->C[2],
        (ELEMENT_P+ELEMENT_i)->C[3],
        (ELEMENT_P+ELEMENT_i)->C[4],
        (ELEMENT_P+ELEMENT_i)->E,
        (ELEMENT_P+ELEMENT_i)->F);

}
fclose(fp1);

//************************print to file - info data***************************//

fp2=fopen("inf.lst","w+");
fprintf(fp2,"%ld  %ld  %ld\n",NODE_NUM,ELEMENT_NUM,ELEMENT_NUM1);
for(i=0;i<5-U_NUM;i++)
{
        fprintf(fp2,"%ld %10.5lg %10.5lg %10.5lg %s",Total_num[case_set_num[U[i]]-1],MIN_VALUE[case_set_num[U[i]]-
1],MAX_VALUE[case_set_num[U[i]]-1],AMAX_VALUE[case_set_num[U[i]]-1],out_set_name[case_set_num[U[i]]-1]);
        strncpy(names->actual_set_name[i],out_set_name[case_set_num[U[i]]],strlen(out_set_name[case_set_num[U[i]]]));
}
for(i=0;i<5-V_NUM;i++)
{
        fprintf(fp2,"%ld %10.5lg %10.5lg %10.5lg %s",Total_num[case_set_num[V[i]]-1],MIN_VALUE[case_set_num[V[i]]-
1],MAX_VALUE[case_set_num[V[i]]-1],AMAX_VALUE[case_set_num[V[i]]-1],out_set_name[case_set_num[V[i]]-1]);
        strncpy(names->actual_set_name[5+i],out_set_name[case_set_num[V[i]]],strlen(out_set_name[case_set_num[V[i]]]));
}

fclose(fp2);

//*****************************print to file - LOAD data*********************************//

        if(LOAD_YES == 1){
                fp_load = fopen("load.lst", "w+");

                for(i = 0; i < LOAD_NUM; i++){
                        fprintf(fp_load, "%d   %s", LOAD_SET[i].SET_ID, LOAD_SET[i].NAME);
```

```
                        for(q = 0; q < LOAD_SET[i].NUM; q++){
                                fprintf(fp_load, "%ld  %d\n", LOAD_SET[i].ID[q], LOAD_SET[i].TYPE[q]); //NODE
OR ELEMENT ID

                                for (r = 0; r < 6; r++)
                                        fprintf(fp_load, "%d   %lg\n", LOAD_SET[i].FACE[q*6+r],
LOAD_SET[i].VALUE[q*8+2+r]);
                        }

                }
                fclose(fp_load);
        }

//*****************************print to file - CONSTRAINT data*********************************//

        if(CONSTRAINT_YES == 1){
                fp_constraint = fopen("constraint.lst", "w+");

                for(I2 = 0; I2 < CONSTRAINT_NUM; I2++){
                        fprintf(fp_constraint, "%d          %s", CONSTRAINT_SET[I2].A, CONSTRAINT_SET[I2].B);

                        for(I3 = 0; I3 < CONSTRAINT_SET[I2].NUM; I3++)
                        {
                                fprintf(fp_constraint, "%ld        %d        %d        %d        %d        %d
        %d\n", CONSTRAINT_SET[I2].ID[I3],
                                CONSTRAINT_SET[I2].INDEX[I3*6], CONSTRAINT_SET[I2].INDEX[I3*6+1],
                                CONSTRAINT_SET[I2].INDEX[I3*6+2],CONSTRAINT_SET[I2].INDEX[I3*6+3],
                                CONSTRAINT_SET[I2].INDEX[I3*6+4],CONSTRAINT_SET[I2].INDEX[I3*6+5]);
                        }
                }
        }
} // end of main loop

//*************************************************************************//
// Function:  FindNid
// Inputs:    Entity ID - ID of node or element for output
// Outputs:   Node ID
// Date revised and comments:
//*************************************************************************//

long int FindNid(long int u)
{
        long int NL,NH,Ntmp;

        if(u == 0)
                return Ntmp = -1;

        if((NODE_P + NODE_NUM)->A == u)
                Ntmp = NODE_NUM;
        else
        {
                NH = NODE_NUM - 1;
                NL = 0;
                while(NL <= NH)
                {
                        Ntmp = (NH + NL) / 2;
                        if(u < (NODE_P + Ntmp)->A)
                                NH = Ntmp - 1;
                        else if((NODE_P + Ntmp)->A < u)
                                        NL = Ntmp + 1;
                        else
                                return Ntmp;
                }
         }
         return Ntmp;
}

//*************************************************************************//
// Function:  FindEid
```

```
// Inputs:
// Outputs:
// Date revised and comments:
//*************************************************************************//

long int FindEid(long int u)
{
        long int NL, NH, Ntmp;

        if(u == 0)
                return Ntmp = -1;
        if((ELEMENT_TMP + ELEMENT_NUM1)->A == u)
                Ntmp = ELEMENT_NUM1;
        else
        {
                NH = ELEMENT_NUM1 - 1;
                NL = 0;
                while(NL <= NH)
                {
                        Ntmp = (NH + NL) / 2;
                        if(u < (ELEMENT_TMP + Ntmp)->A)
                                NH = Ntmp - 1;
                        else if((ELEMENT_TMP + Ntmp)->A < u)
                                NL = Ntmp + 1;
                        else
                                return Ntmp;
                }
        }
        return Ntmp;
}

//revised on Oct. 22,1997

//*************************************************************************//
// Function: Compare
// Inputs:   element_i, node_i
// Outputs:  Flag
// Date revised and comments:  Oct. 22,1997
//*************************************************************************//

int compare(long int ELEMENT_i, long int NODE_i)
{
        int FLAG, C[4], i, j;

        for(i = 0; i < 4; i++)
                C[i] = 0;

        for(i = 0; i < 4; i++)
        {
                j = 3;
                FLAG = 0;
                while((j >= 0) && (FLAG == 0))
                {
                        if((ELEMENT_P + ELEMENT_i)->B[i] == (ELEMENT_P + NODE_i)->B[j])
                        {
                                FLAG = 1;
                                C[i] = 1;
                        }
                        j = j-1;
                }
        }

        if((C[0] == 1) && (C[1] == 1) && (C[2] == 1) && (C[3] == 1))
                return FLAG = 1;
        else
                return FLAG = 0;
}

//*************************************************************************//
```

```
//*************************************************************************//
DVET Release 2.2/11/98 for WindowsNT Workstation
inside1120.h
11 February 1998
Copyright 1998
Dual Incorporated/University of Central Florida

        extern float        xeulfloat,yeulfloat,zeulfloat;
        extern float        q0float,q1float,q2float,q3float;//quat req
//*************************************************************************//
```

```
//*************************************************************************//
DVET Release 2.2/11/98 for WindowsNT Workstation
inside1120.c
11 February 1998
Copyright 1998
Dual Incorporated
// Headers
//
#include <windows.h>
#include <stdio.h>
#include <conio.h>

//
// Definitions
//
#define MESSAGE_COUNT     10
#define MESSAGE_TEXT      "The owls are not what they seem."

//
// Global data
//
HANDLE      hDevice;
            DWORD       dwErrorCode;
            DWORD       dwBytesRead;
            TCHAR       szBuffer[100];
            OVERLAPPED  overlap;
            BOOL        status;
            DWORD       i;
            int         n;
            float       xeulfloat,yeulfloat,zeulfloat;
            float       q0float,q1float,q2float,q3float;//quat req


//
// Forward declarations of local functions...
//
DWORD
Writer(char *lpvParam);

static      VOID
ErrorMessage(
              LPTSTR lpOrigin,
              DWORD dwMessageId
);

//
VOID        __cdecl
insideinit()
{
            hDevice = CreateFile("\\\\.\\InSide1",
                                 GENERIC_READ | GENERIC_WRITE,
                                 0,
                                 NULL,
                                 OPEN_EXISTING,
                                 FILE_FLAG_OVERLAPPED,
                                 NULL);

            if (hDevice == INVALID_HANDLE_VALUE) {
                        dwErrorCode = GetLastError();
                        ErrorMessage("CreateFile", dwErrorCode);
//                      ExitProcess(dwErrorCode);
            }
            //
            // Set up asynch operation (otherwise, we won't
            // be able to use the same handle in two threads).
            //
            overlap.Offset = 0;
            overlap.OffsetHigh = 0;
            overlap.hEvent = CreateEvent(NULL,
                                         TRUE,
```

```
                                    FALSE,
                                    NULL);


status = ReadFile(hDevice,
                  szBuffer,
                  100,
                  &dwBytesRead,
                  &overlap);
if (!status &&
   GetLastError() != ERROR_IO_PENDING) {
        dwErrorCode = GetLastError();
        ErrorMessage("ReadFile", dwErrorCode);

        goto Error;
}
//
// Wait for the read to complete
//
status = GetOverlappedResult(
                                    hDevice,
                                    &overlap,
                                    &dwBytesRead,
                                    TRUE);
if (!status) {
        dwErrorCode = GetLastError();
        ErrorMessage("ReadFile", dwErrorCode);
        goto Error;
}
szBuffer[dwBytesRead] = '\0';
printf("Msg #%2d [%d bytes]: %s\n", i + 1, dwBytesRead, szBuffer);

Writer("S");
Sleep(200);

//
// Set up asynch operation (otherwise, we won't
// be able to use the same handle in two threads).
//
overlap.Offset = 0;
overlap.OffsetHigh = 0;
overlap.hEvent = CreateEvent(NULL,
                                    TRUE,
                                    FALSE,
                                    NULL);


status = ReadFile(hDevice,
                  szBuffer,
                  100,
                  &dwBytesRead,
                  &overlap);
if (!status &&
   GetLastError() != ERROR_IO_PENDING) {
        dwErrorCode = GetLastError();
        ErrorMessage("ReadFile", dwErrorCode);

        goto Error;
}
//
// Wait for the read to complete
//
status = GetOverlappedResult(
                                    hDevice,
                                    &overlap,
                                    &dwBytesRead,
                                    TRUE);
if (!status) {
        dwErrorCode = GetLastError();
        ErrorMessage("ReadFile", dwErrorCode);
```

```
                goto Error;
        }
        szBuffer[dwBytesRead] = '\0';
        printf("Msg #%2d [%d bytes]: %s\n", i + 1, dwBytesRead, szBuffer);


        Sleep(100);
        Writer("B1\n");
        Sleep(100);

        //Writer("O1,2,20,1\n"); will just give quaternions (10 bytes in ReadFile())
//      Writer("O1,11,1\n");//quat req

        //Writer("O1,2,20,1\n"); will give both position and quaternions
        // You have to increase data field in the
        // ReadFile(hDevice,szBuffer,10,&dwBytesRead,&overlap); from 10 to 18 bytes
//      Writer("O1,2,20,1\n");

        Sleep(100);
        Writer("P");
        Sleep(100);


        Error://Dryer test commented out

        //
        // All done with this; get rid of it
        //
        printf("done insideinit()\n");

///     CloseHandle(overlap.hEvent);//Dryer test commented out

        //
        // Hang around until the writer is finished
        //
///     CloseHandle(hDevice);//Dryer test commented out
//      ExitProcess(ERROR_SUCCESS);

}

insidetick()
{
        short   xeulshort,yeulshort,zeulshort;
        short   q0short,q1short,q2short,q3short;//quat req
                //
                // Set up asynch operation (otherwise, we won't
                // be able to use the same handle in two threads).
                //
                overlap.Offset = 0;
                overlap.OffsetHigh = 0;
                overlap.hEvent = CreateEvent(NULL,
                                             TRUE,
                                             FALSE,
                                             NULL);


                status = ReadFile(hDevice,
                                  szBuffer,
///                               10,//quat req
                                  16,
                                  &dwBytesRead,
                                  &overlap);
                if (!status &&
                   GetLastError() != ERROR_IO_PENDING) {
                        dwErrorCode = GetLastError();
                        ErrorMessage("ReadFile", dwErrorCode);

                        goto Error;
                }
                //
```

```
                    // Wait for the read to complete
                    //
                    status = GetOverlappedResult(
                                                hDevice,
                                                &overlap,
                                                &dwBytesRead,
                                                TRUE);
                    if (!status) {
                            dwErrorCode = GetLastError();
                            ErrorMessage("ReadFile", dwErrorCode);
                            goto Error;
                    }
                    szBuffer[dwBytesRead] = '\0';
///                 printf("Msg #%2d [%d bytes]: %s\n", i + 1, dwBytesRead, szBuffer);
///                 printf("type  %c", szBuffer[0]);   // type

///                 printf("station  %c ", szBuffer[1]);            // station

///                 printf("%d ", (*(unsigned short *)( szBuffer + 2)) ); // xpos
///                 printf("%d ", (*(unsigned short *)( szBuffer + 4)) );
///                 printf("%d\n", (*(unsigned short *)( szBuffer + 6)) );

///                 printf("%d ", (*(short *)( szBuffer + 8)) ); // x angle
///                 printf("%d ", (*(short *)( szBuffer + 10)) );
///                 printf("%d\n", (*(short *)( szBuffer + 12)) );

                    xeulshort=(*(unsigned short *)( szBuffer + 8)); // x angle short
                    xeulfloat=(((float)(xeulshort))/65536.0)*360.0; // x angle float degrees
                    yeulshort=(*(unsigned short *)( szBuffer + 10)); // y angle short
                    yeulfloat=(((float)(yeulshort))/65536.0)*360.0; // y angle float degrees
                    zeulshort=(*(unsigned short *)( szBuffer + 12)); // z angle short
                    zeulfloat=(((float)(zeulshort))/65536.0)*360.0; // z angle float degrees

///                 printf("x y z euler is %d %d %d\n",xeulshort,yeulshort,zeulshort);
///                 printf("x y z euler is %f %f %f\n",xeulfloat,yeulfloat,zeulfloat);

//                  printf("cr %x  ", *(unsigned char *)( szBuffer + 14));
//                  printf("lf %x\n", *(unsigned char *)( szBuffer + 15));

/*                  printf("%d ", (*(short *)( szBuffer + 2)) ); //q0
                    printf("%d ", (*(short *)( szBuffer + 4)) ); //q1
                    printf("%d ", (*(short *)( szBuffer + 6)) );//q2
                    printf("%d\n", (*(short *)( szBuffer + 8)) ); //q3
*/
/*                  q0short=(*(unsigned short *)( szBuffer + 2)); // q0 short
                    q0float=(((float)(q0short))/65536.0); // q0 float
                    q1short=(*(unsigned short *)( szBuffer + 4)); // q1 short
                    q1float=(((float)(q1short))/65536.0); // q1 float
                    q2short=(*(unsigned short *)( szBuffer + 6)); // q2 short
                    q2float=(((float)(q2short))/65536.0); // q2 float
                    q3short=(*(unsigned short *)( szBuffer + 8)); // q3 short
                    q3float=(((float)(q3short))/65536.0); // q3 float
*/
///                 printf("q0s q1s q2s q3s is %d %d %d %d\n",q0short,q1short,q2short,q3short);
///                 printf("q0f q1f q2f q3f is %f %f %f %f\n",q0float,q1float,q2float,q3float);


                    Writer("P");

Error://Dryer test commented out

                //
                // All done with this; get rid of it
                //
;///             printf("done insidetick()\n");

///             CloseHandle(overlap.hEvent);//Dryer test commented out

                //
                // Hang around until the writer is finished
```

```
        //
///     CloseHandle(hDevice);//Dryer test commented out
//      ExitProcess(ERROR_SUCCESS);
}

//
// This function writes messages to the device
//
DWORD
Writer(char *p)
{

        DWORD       dwBytesWritten;
        DWORD       dwErrorCode;
        OVERLAPPED    overlap;
        BOOL        status;
        DWORD       i;

        //
        // Set up accoutrements of overlapped I/O
        //
        overlap.Offset = 0;
        overlap.OffsetHigh = 0;
        overlap.hEvent = CreateEvent(NULL,
                                     TRUE,
                                     FALSE,
                                     NULL);

        //
        // Wait 10 seconds before starting
        //


        status = WriteFile(hDevice,
                           p,
                           strlen(p),
                           &dwBytesWritten,
                           &overlap);
        if (!status && GetLastError() != ERROR_IO_PENDING) {
                dwErrorCode = GetLastError();
                ErrorMessage("WriteFile", dwErrorCode);
//              ExitThread(dwErrorCode);
        }
        //
        // Wait for the write to complete
        //
        status = GetOverlappedResult(
                                     hDevice,
                                     &overlap,
                                     &dwBytesWritten,
                                     TRUE);
        if (!status) {
                dwErrorCode = GetLastError();
                ErrorMessage("WriteFile", dwErrorCode);
                CloseHandle(overlap.hEvent);
//              ExitThread(dwErrorCode);
        }
///     printf("dwBytesWritten %d\n", dwBytesWritten);
        //
        // Introduce a little timing delay. This is
        // necessary because the driver is using the
        // FIFO timeout as a way to detect end of
        // transmission.
        //
///     Sleep(100);


        //
        // Get rid of Event object
        //
```

```
        CloseHandle(overlap.hEvent);

        return (ERROR_SUCCESS);
}


//
// This helper routine converts a system-service error
// code into a text message and prints it on StdOutput
//
static      VOID
ErrorMessage(LPTSTR lpOrigin,
             DWORD dwMessageId
)
{
        LPTSTR      msgBuffer;      // string returned from system

        DWORD       cBytes;         // length of returned string

        cBytes = FormatMessage(
                                    FORMAT_MESSAGE_FROM_SYSTEM |
                                    FORMAT_MESSAGE_ALLOCATE_BUFFER,
                                    NULL,
                                    dwMessageId,
                                    MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US),
                                    (TCHAR *) & msgBuffer,
                                    500,
                                    NULL);
        if (msgBuffer) {
                msgBuffer[cBytes] = TEXT('\0');
                printf("Error: %s -- %s\n", lpOrigin, msgBuffer);
                LocalFree(msgBuffer);
        } else {
                printf("FormatMessage error: %d\n", GetLastError());
        }
}
```

```
//*************************************************************************//
DVET Release 2.2/11/98 for WindowsNT Workstation
prompts.c
11 February 1998
Copyright 1998
Dual Incorporated
//
// DESCRIPTION: This module contains functions that interface to the
//                                        'graphical' part of this program. This currently only relates to
//                                        the code that brings up the
//
// DEPENDENCIES:  congui.dlg
//
//
//*************************************************************************//

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <windows.h>
#include "ConGUI.h"

/*************************Declarations*********************************/

int GetBool (char ***pargv, char *);
extern BOOL CenterWindow (HWND hwnd);
BOOL APIENTRY BOOLDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam);
int GetName (char ***pargv, char set_name[3000][30], int);
BOOL APIENTRY CPDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam);
int GetOutputData (char ***pargv, char temp_set_names[3000][40], int);
BOOL APIENTRY ODDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam);
int GetLoadSetName (char ***pargv, char loadset_name[100][30], int);
BOOL APIENTRY LSDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam);
int GetConstraintSetName (char ***pargv, char constraintset_name[100][40], int);
BOOL APIENTRY CSDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam);


char case_names[3000][30], output_data_names[3000][40], load_names[100][40], constraint_names[100][40];
int case_amt, output_data_amt, load_amt, constraint_amt;

//*************************************************************************//
// Function:  GetBool
// Inputs:  none
// Outputs:  YES or NO
// Date revised and comments:
//*************************************************************************//

int GetBool (char ***pargv, char *gui_name)
{
   int ret;
   HANDLE hinst;
   HWND hwnd;
   char szFile[80];

   hinst = GetModuleHandle (NULL);
   hwnd = GetFocus();

   ret = DialogBoxParam (hinst, gui_name, NULL, BOOLDlgProc, (LPARAM)pargv);

   if (-1 == ret) {
      ret = GetLastError();
      printf ("Unable to create dialog: %d\n", ret);
      GetModuleFileName (hinst, szFile, sizeof(szFile));
      printf ("hinst = %d\n", hinst);
      printf ("hwnd = %d\n", hwnd);
      printf ("File = %s\n", szFile);
      return FALSE;

   }
```

```
  return ret;
}

//*************************************************************************//
// Function:  BOOLDlgProc - Procedure that handles inputs, commands to the
//                                                              dialog box
// Inputs:  window handle
// Outputs:
// Date revised and comments:
//*************************************************************************//

BOOL APIENTRY BOOLDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
  int wmId;
  static char ***pargv;
  static char **argv;

  int iCtrl = 420, argc;
  char *cmd;
  char *cmdline;
        int max_no_file = 30;

  switch (msg) {
    case WM_INITDIALOG:
      // We need to initialize stuff in the dialog box...

      pargv = (char ***)lParam;
      argv = *pargv;
      CenterWindow (hdlg);

      return (TRUE);

    case WM_DESTROY:
      break;

    case WM_COMMAND:
      wmId = LOWORD(wParam);
      switch (wmId) {

        case IDYES:
          cmd = cmdline = (char *)GlobalAlloc (GPTR, 128);
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                        strcpy(cmd, "YES");
                                cmd += strlen(cmd);
                        cmd[0] = 0;
                                argv[++argc] = ++cmd;
                } // if (cmdline)...

          EndDialog(hdlg, argc);
          return (TRUE);

        case IDNO:
                        cmd = cmdline = (char *)GlobalAlloc (GPTR, 128);
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                                strcpy(cmd, "NO");
                                cmd += strlen(cmd);
                        cmd[0] = 0;
                                argv[++argc] = ++cmd;
                } // if (cmdline)...

          EndDialog(hdlg, 0);
          return (TRUE);
      }
      break;
```

```
  }
  return (FALSE);

  lParam; // unreferenced formal parameter
}

//************************************************************************//
// Function:  GetName
// Inputs:  array containing case numbers
// Outputs:  case number
// Date revised and comments:
//************************************************************************//

int GetName (char ***pargv, char set_name[3000][30], int no_cases)
{
  int ret, i;
  HANDLE hinst;
  HWND hwnd;
  char szFile[80];

  hinst = GetModuleHandle (NULL);
  hwnd = GetFocus();

        case_amt = no_cases;

        for(i = 0; i < case_amt; i++)
                strncpy(case_names[i], set_name[i], strlen(set_name[i]) - 1);          // removes the carriage return

  ret = DialogBoxParam (hinst, "CP", NULL, CPDlgProc, (LPARAM)pargv);

  if (-1 == ret) {
    ret = GetLastError();
    printf ("Unable to create dialog: %d\n", ret);
    GetModuleFileName (hinst, szFile, sizeof(szFile));
    printf ("hinst = %d\n", hinst);
    printf ("hwnd = %d\n", hwnd);
    printf ("File = %s\n", szFile);
    return FALSE;

  }
  return ret;
}

//************************************************************************//
// Function:  CPDlgProc - Procedure that handles inputs, commands to the
//                                                          dialog box
// Inputs:  window handle
// Outputs:
// Date revised and comments:
//************************************************************************//

BOOL APIENTRY CPDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
  int wmId;
  static char ***pargv;
  static char **argv;

  int i, item, index, iCtrl = 402, argc;
  char *cmd;
  char *cmdline;
        int max_no_file = 30;

  switch (msg) {
    case WM_INITDIALOG:
      // We need to initialize stuff in the dialog box...

      pargv = (char ***)lParam;
      argv = *pargv;
      CenterWindow (hdlg);
```

```
                                i = 0;
                                while(i < case_amt){
                                        index = SendDlgItemMessage (hdlg, iCtrl, CB_ADDSTRING, 0,
(DWORD)(LPSTR)case_names[i]);
        SendDlgItemMessage (hdlg, iCtrl, CB_SETITEMDATA, index, i);
                                        if (i == 0)
          SendDlgItemMessage (hdlg, iCtrl, CB_SETCURSEL, index, 0);
                                        i++;
                                }
      return (TRUE);

    case WM_DESTROY:
      break;

    case WM_COMMAND:
      wmId = LOWORD(wParam);
      switch (wmId) {

        case IDOK:
          cmd = cmdline = (char *)GlobalAlloc (GPTR, 128);
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                                                index = SendDlgItemMessage(hdlg, iCtrl, CB_GETCURSEL, 0,
0);
                                                item = SendDlgItemMessage (hdlg, iCtrl, CB_GETITEMDATA,
index, 0);
            wsprintf ((LPSTR)cmd, "%s", (LPSTR)case_names[item]);
                                                cmd += strlen(cmd);
            cmd[0] = 0;
            argv[++argc] = ++cmd;
                                        } // if (cmdline)...

          EndDialog(hdlg, argc);
          return (TRUE);

        case IDCANCEL:
                                                EndDialog(hdlg, 0);
          return (TRUE);
      }
      break;
  }
  return (FALSE);

  lParam; // unreferenced formal parameter
}

//*************************************************************************//
// Function:  GetOutputData
// Inputs:  array containing case numbers
// Outputs:  case number
// Date revised and comments:
//*************************************************************************//

int GetOutputData (char ***pargv, char temp_set_name[3000][40], int no_output_data)
{
  int ret, i;
  HANDLE hinst;
  HWND hwnd;
  char szFile[80];

  hinst = GetModuleHandle (NULL);
  hwnd = GetFocus();

        output_data_amt = no_output_data;

        for(i = 0; i < output_data_amt; i++)
                strncpy(output_data_names[i], temp_set_name[i], strlen(temp_set_name[i]) - 1);                  // removes
the carriage return
```

```
  ret = DialogBoxParam (hinst, "ODP", NULL, ODDlgProc, (LPARAM)pargv);

  if (-1 == ret) {
     ret = GetLastError();
     printf ("Unable to create dialog: %d\n", ret);
     GetModuleFileName (hinst, szFile, sizeof(szFile));
     printf ("hinst = %d\n", hinst);
     printf ("hwnd = %d\n", hwnd);
     printf ("File = %s\n", szFile);
     return FALSE;

  }
  return ret;
}

//*************************************************************************//
// Function:  CPDlgProc - Procedure that handles inputs, commands to the
//                                                              dialog box
// Inputs:  window handle
// Outputs:
// Date revised and comments:
//*************************************************************************//

BOOL APIENTRY ODDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
  int wmId;
  static char ***pargv;
  static char **argv;

  int i, items[10], no_items, index = 10, iCtrl = 402, argc;          // index - maximum number of items selected from list box
  char *cmd;
  char *cmdline;
          int max_no_file = 30;

  switch (msg) {
    case WM_INITDIALOG:
      // We need to initialize stuff in the dialog box...

      pargv = (char ***)lParam;
      argv = *pargv;
      CenterWindow (hdlg);

      i = 0;
                              while(i < output_data_amt){
                                      index = SendDlgItemMessage (hdlg, iCtrl, LB_ADDSTRING, 0,
(DWORD)(LPSTR)output_data_names[i]);
          i++;
                              }
      return (TRUE);

    case WM_DESTROY:
      break;

    case WM_COMMAND:
      wmId = LOWORD(wParam);
      switch (wmId) {
        case IDOK:
          cmd = cmdline = (char *)GlobalAlloc (GPTR, 400);                    // Increased memory space
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                                                                    no_items = SendDlgItemMessage (hdlg, iCtrl,
LB_GETSELITEMS, index, (DWORD)(LPINT)items);
                                                                    i = 0;
                                                                    while(i < no_items){
                                                                            wsprintf ((LPSTR)cmd, "%s",
(LPSTR)output_data_names[(items[i])]);
                                                                            cmd += strlen(cmd);
```

```
                                                    cmd[0] = 0;
                                                        argv[++argc] = ++cmd;
                                                            i++;
                                                    }
                                            } // if (cmdline)...

            EndDialog(hdlg, argc);
            return (TRUE);

          case IDCANCEL:
                                                    EndDialog(hdlg, 0);
            return (TRUE);
        }
        break;
  }
  return (FALSE);

  lParam; // unreferenced formal parameter
}

//*************************************************************************//
// Function: GetLoadSetName
// Inputs: array containing case numbers
// Outputs: case number
// Date revised and comments:
//*************************************************************************//

int GetLoadSetName (char ***pargv, char loadset_names[100][30], int no_load)
{
  int ret, i;
  HANDLE hinst;
  HWND hwnd;
  char szFile[80];

  hinst = GetModuleHandle (NULL);
  hwnd = GetFocus();

          load_amt = no_load;

          for(i = 0; i < load_amt; i++)
                  strncpy(load_names[i], loadset_names[i], strlen(loadset_names[i]) - 1);                    // removes the
carriage return

  ret = DialogBoxParam (hinst, "LSP", NULL, LSDlgProc, (LPARAM)pargv);

  if (-1 == ret) {
    ret = GetLastError();
    printf ("Unable to create dialog: %d\n", ret);
    GetModuleFileName (hinst, szFile, sizeof(szFile));
    printf ("hinst = %d\n", hinst);
    printf ("hwnd = %d\n", hwnd);
    printf ("File = %s\n", szFile);
    return FALSE;

  }
  return ret;
}

//*************************************************************************//
// Function: CPDlgProc - Procedure that handles inputs, commands to the
//                                                    dialog box
// Inputs: window handle
// Outputs:
// Date revised and comments:
//*************************************************************************//

BOOL APIENTRY LSDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
  int wmId;
  static char ***pargv;
```

```
  static char **argv;

  int i, item, index, iCtrl = 402, argc;
  char *cmd;
  char *cmdline;
          int max_no_file = 30;

  switch (msg) {
    case WM_INITDIALOG:
      // We need to initialize stuff in the dialog box...

      pargv = (char ***)lParam;
      argv = *pargv;
      CenterWindow (hdlg);

                                  i = 0;
                                  while(i < load_amt){
                                          index = SendDlgItemMessage (hdlg, iCtrl, CB_ADDSTRING, 0,
(DWORD)(LPSTR)load_names[i]);
        SendDlgItemMessage (hdlg, iCtrl, CB_SETITEMDATA, index, i);
                                          if (i == 0)
          SendDlgItemMessage (hdlg, iCtrl, CB_SETCURSEL, index, 0);
                                          i++;
                                  }
      return (TRUE);

    case WM_DESTROY:
      break;

    case WM_COMMAND:
      wmId = LOWORD(wParam);
      switch (wmId) {

        case IDOK:
          cmd = cmdline = (char *)GlobalAlloc (GPTR, 400);
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                                                  index = SendDlgItemMessage(hdlg, iCtrl, CB_GETCURSEL, 0,
0);
                                                  item = SendDlgItemMessage (hdlg, iCtrl, CB_GETITEMDATA,
index, 0);
            wsprintf ((LPSTR)cmd, "%s", (LPSTR)load_names[item]);
                                                  cmd += strlen(cmd);
            cmd[0] = 0;
            argv[++argc] = ++cmd;
                                          } // if (cmdline)...

          EndDialog(hdlg, argc);
          return (TRUE);

        case IDCANCEL:
                                                  EndDialog(hdlg, 0);
          return (TRUE);
      }
      break;
  }
  return (FALSE);

  lParam; // unreferenced formal parameter
}

//*************************************************************************//
// Function:  GetConstraintSetName
// Inputs:  array containing constraint names
// Outputs:  case number
// Date revised and comments:
//*************************************************************************//
```

```
int GetConstraintSetName (char ***pargv, char constraintset_names[100][40], int no_constraint)
{
   int ret, i;
   HANDLE hinst;
   HWND hwnd;
   char szFile[80];

   hinst = GetModuleHandle (NULL);
   hwnd = GetFocus();

            constraint_amt = no_constraint;

            for(i = 0; i < constraint_amt; i++)
                     strncpy(constraint_names[i], constraintset_names[i], strlen(constraintset_names[i]) - 1);                    // removes
the carriage return

   ret = DialogBoxParam (hinst, "CSP", NULL, CSDlgProc, (LPARAM)pargv);

   if (-1 == ret) {
      ret = GetLastError();
      printf ("Unable to create dialog: %d\n", ret);
      GetModuleFileName (hinst, szFile, sizeof(szFile));
      printf ("hinst = %d\n", hinst);
      printf ("hwnd = %d\n", hwnd);
      printf ("File = %s\n", szFile);
      return FALSE;

   }
   return ret;
}

//*************************************************************************//
// Function:  CSDlgProc - Procedure that handles inputs, commands to the
//                                                                  dialog box
// Inputs:  window handle
// Outputs:
// Date revised and comments:
//*************************************************************************//

BOOL APIENTRY CSDlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
   int wmId;
   static char ***pargv;
   static char **argv;

   int i, item, index, iCtrl = 402, argc;
   char *cmd;
   char *cmdline;
            int max_no_file = 30;

   switch (msg) {
      case WM_INITDIALOG:
         // We need to initialize stuff in the dialog box...

         pargv = (char ***)lParam;
         argv = *pargv;
         CenterWindow (hdlg);

                                    i = 0;
                                    while(i < constraint_amt){
                                             index = SendDlgItemMessage (hdlg, iCtrl, CB_ADDSTRING, 0,
(DWORD)(LPSTR)constraint_names[i]);
            SendDlgItemMessage (hdlg, iCtrl, CB_SETITEMDATA, index, i);
                                             if (i == 0)
               SendDlgItemMessage (hdlg, iCtrl, CB_SETCURSEL, index, 0);
                                             i++;
                                    }
         return (TRUE);

      case WM_DESTROY:
```

```
        break;

    case WM_COMMAND:
      wmId = LOWORD(wParam);
      switch (wmId) {

        case IDOK:
          cmd = cmdline = (char *)GlobalAlloc (GPTR, 400);
          argv[0] = cmdline;
          argc = 0;

          if (cmdline) {
                                                    index = SendDlgItemMessage(hdlg, iCtrl, CB_GETCURSEL, 0,
0);
                                                    item = SendDlgItemMessage (hdlg, iCtrl, CB_GETITEMDATA,
index, 0);
            wsprintf ((LPSTR)cmd, "%s", (LPSTR)constraint_names[item]);
                                                    cmd += strlen(cmd);
            cmd[0] = 0;
            argv[++argc] = ++cmd;
                                          } // if (cmdline)...

          EndDialog(hdlg, argc);
          return (TRUE);

        case IDCANCEL:
                                                EndDialog(hdlg, 0);
          return (TRUE);
      }
      break;
  }
  return (FALSE);

  lParam; // unreferenced formal parameter
}
```

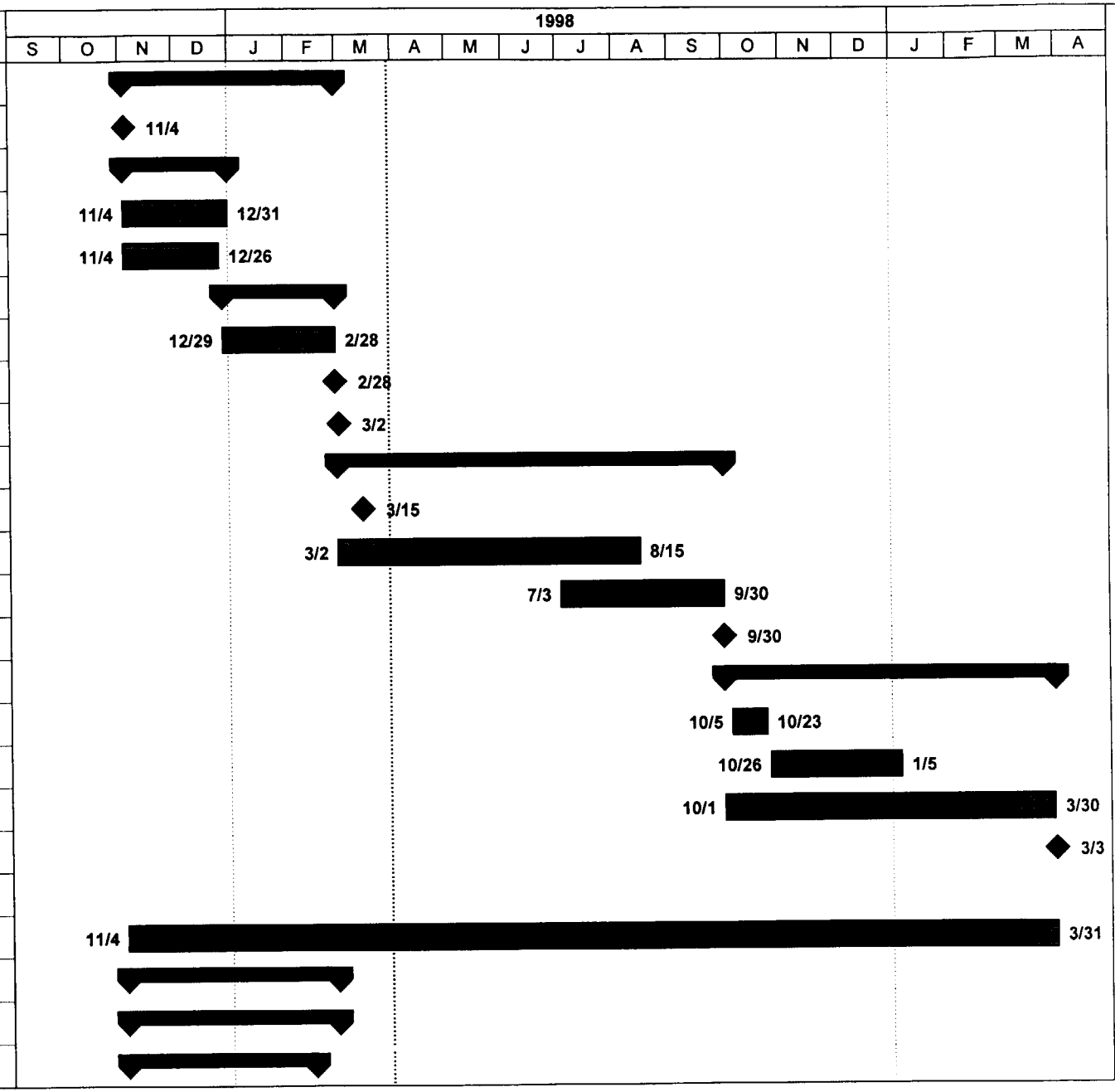
```
* User name:   BSTEVENSON (83)              Queue:  DUAL_ORLANDO/HP5_SIQ
* File name:                                Server: HP5SIMX
* Directory:
* Description: MS Project - ITT0498.MPP
*              March 30, 1998                              1:08pm
```

```
BBBB   SSS  TTTTT EEEEE V   V EEEEE N   N  SSS   OOO  N   N
B   B S   S   T   E     V   V E     N   N S   S O   O N   N
B   B S       T   E     V   V E     NN  N S     O   O NN  N
BBBB   SSS    T   EEEE  V   V EEEE  N N N  SSS  O   O N N N
B   B      S  T   E     V   V E     N  NN     S O   O N  NN
B   B S   S   T   E      V V  E     N   N S   S O   O N   N
BBBB   SSS    T   EEEEE   V   EEEEE N   N  SSS   OOO  N   N
```

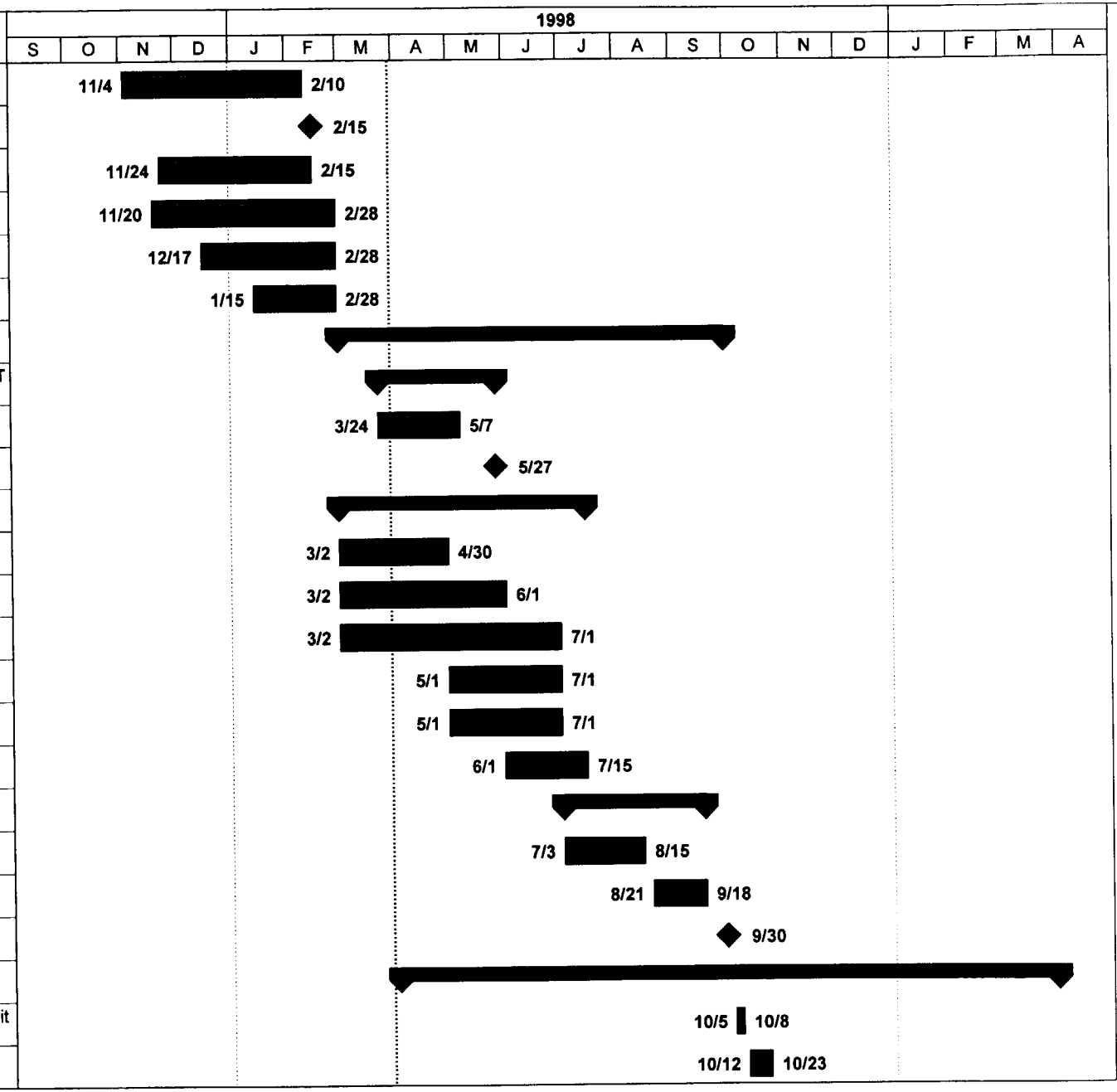
```
L      SSS  TTTTT
L     S   S   T
L     S       T    ::
L      SSS    T    ::
L         S   T
L     S   S   T    ::
LLLLL  SSS    T    ::
```

S3B ITT REHOST, INSTALLATION AND SUPPORT

ID	Task Name
1	**PROGRAM MILESTONES**
2	Program Start-up Under S3 D.O.
3	**Initial Tasking**
4	Phase I Requirements Document (Dr
5	Interface Design Document (Draft)
6	**Phase I Effort**
7	Complete Phase I
8	Draft SRF Installation Report
9	Contract Award for New D. O.
10	**Phase II Effort**
11	Full ITT Requirements Document
12	Complete ITT Rehost to SRF
13	Test ITT on SRF
14	Submit Test Report & Software
15	**Phase III Effort**
16	Install ITT Hardware On WST
17	HSI ITT On WST
18	Test ITT
19	Acceptance & Test Results Report
20	
21	PROGRAM SCHEDULE
22	**PHASE I**
23	**Rehost ITT Acoustic S/W onto SRF**
24	**Phase I Requirements Document**



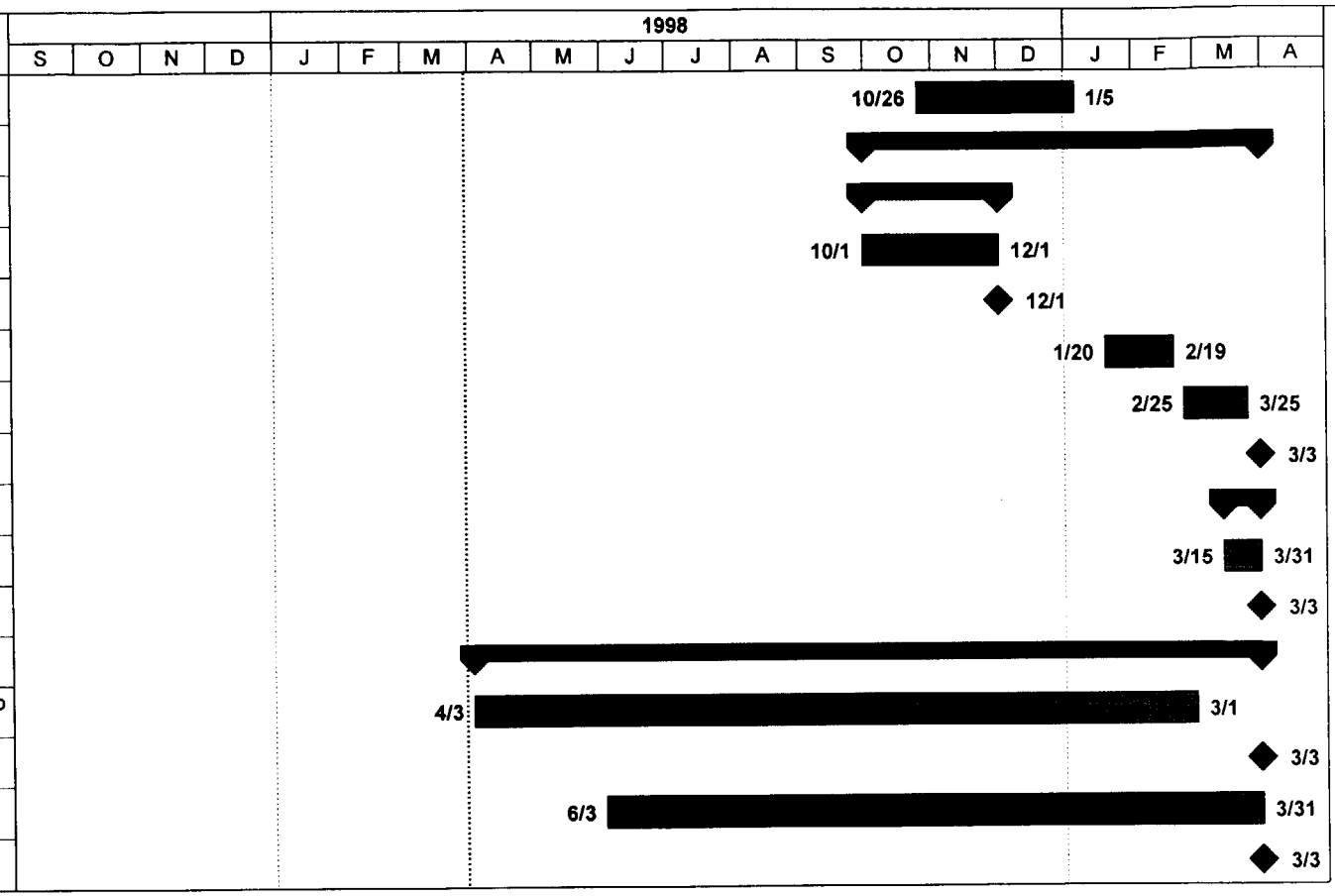
S3B ITT REHOST, INSTALLATION AND SUPPORT

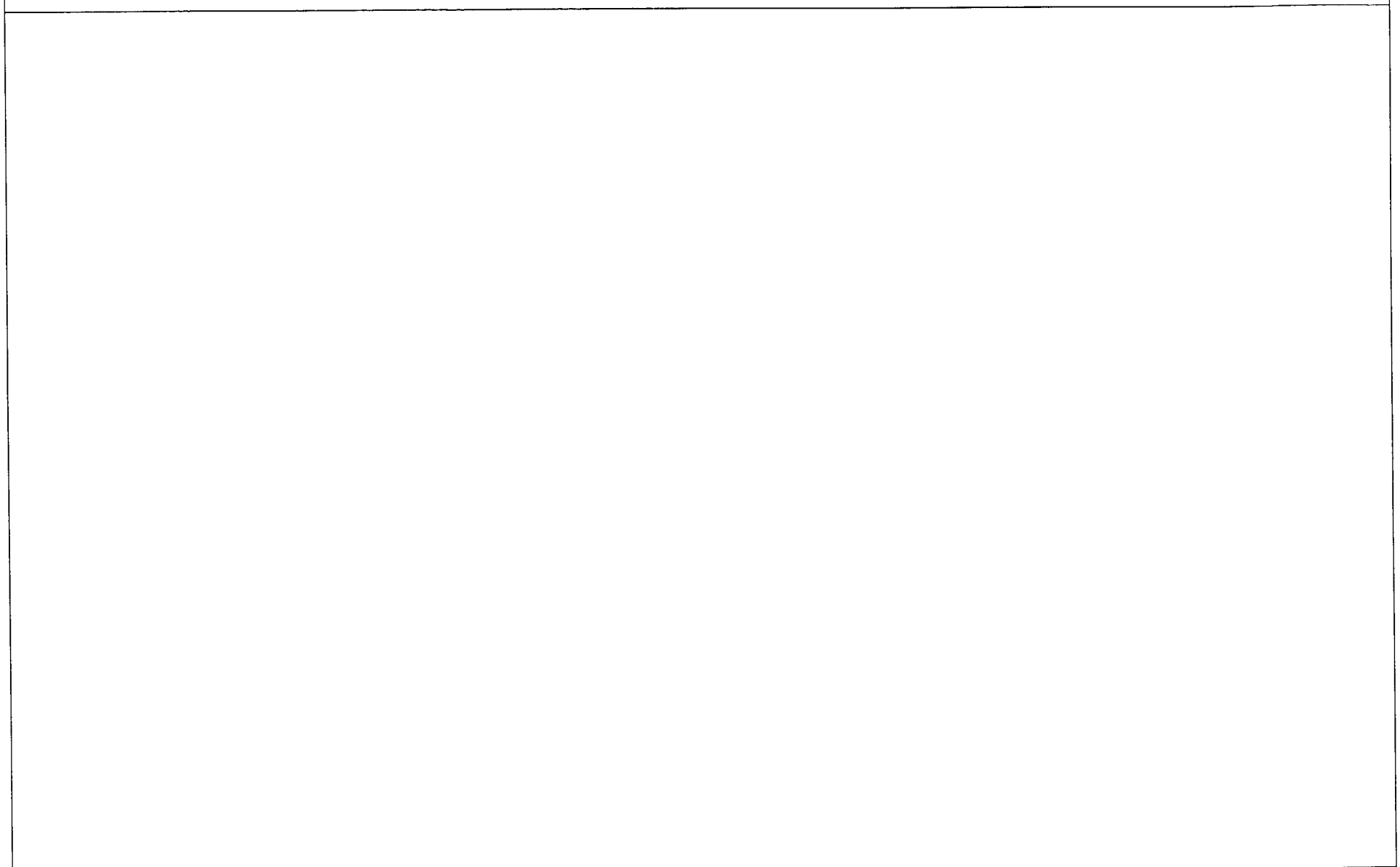


ID	Task Name
25	Develop Phase I Requirements
26	Submit Phase I Requirements D
27	Port Existing H1200 S/W
28	Develop Communications Process
29	Add Required Functionality
30	Test and Evaluate S/W Rehost
31	PHASE II
32	**Full ITT Requirements for SRF and WST**
33	Develop Requirements Document
34	Submit Requirements Document
35	**Complete ITT Rehost to SRF**
36	Continue T & E of S/W Rehost
37	Search Stores Instructor Control
38	Data Monitor Process
39	Search Stores Interface
40	Search Stores Display
41	IEER Status Tableau
42	**Test Full ITT on SRF**
43	Contractor Testing
44	Government Testing
45	Provide Configured S/W
46	PHASE III
47	Move ITT Hardware Suite #2 to Trainer Sit
48	Install ITT on S3B WST#1

S3B ITT REHOST, INSTALLATION AND SUPPORT

ID	Task Name
49	HSI of ITT onto WST#1
50	**Testing**
51	**Test Procedures**
52	Develop Test Procedures
53	Submit Test Procedures
54	Contractor Test
55	Government Test
56	Acceptance
57	**Test Results Report**
58	Develop Report
59	Submit Report
60	**Documentation**
61	Develop Requisite S/W Documentatio
62	Submit S/W Documentation
63	Develop O&M and User Manuals
64	Submit Requisite Manuals

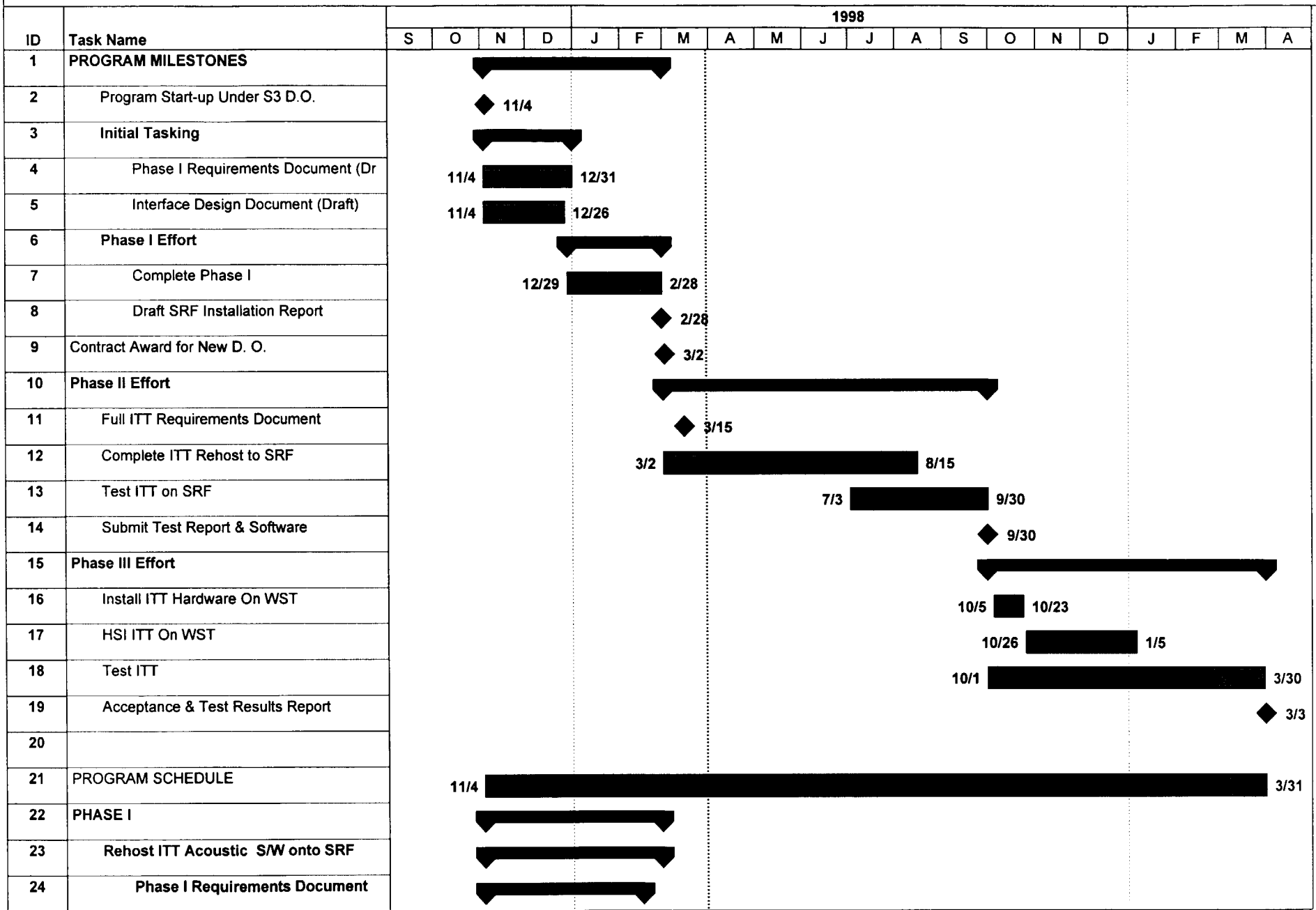




Project:
Date: 3/30/98

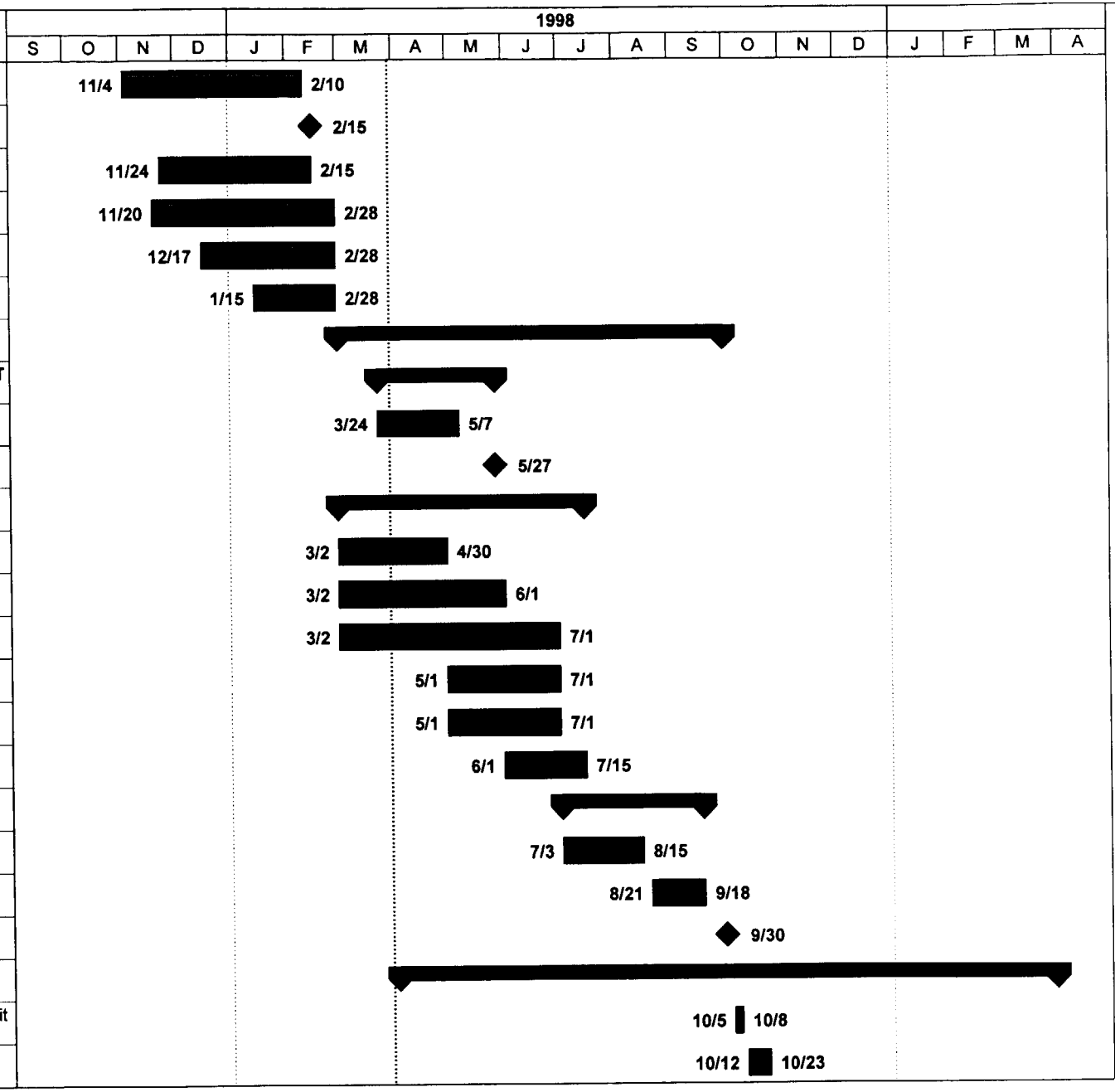
Task
Progress
Milestone
Summary
Rolled Up Task
Rolled Up Milestone
Rolled Up Progress

S3B ITT REHOST, INSTALLATION AND SUPPORT



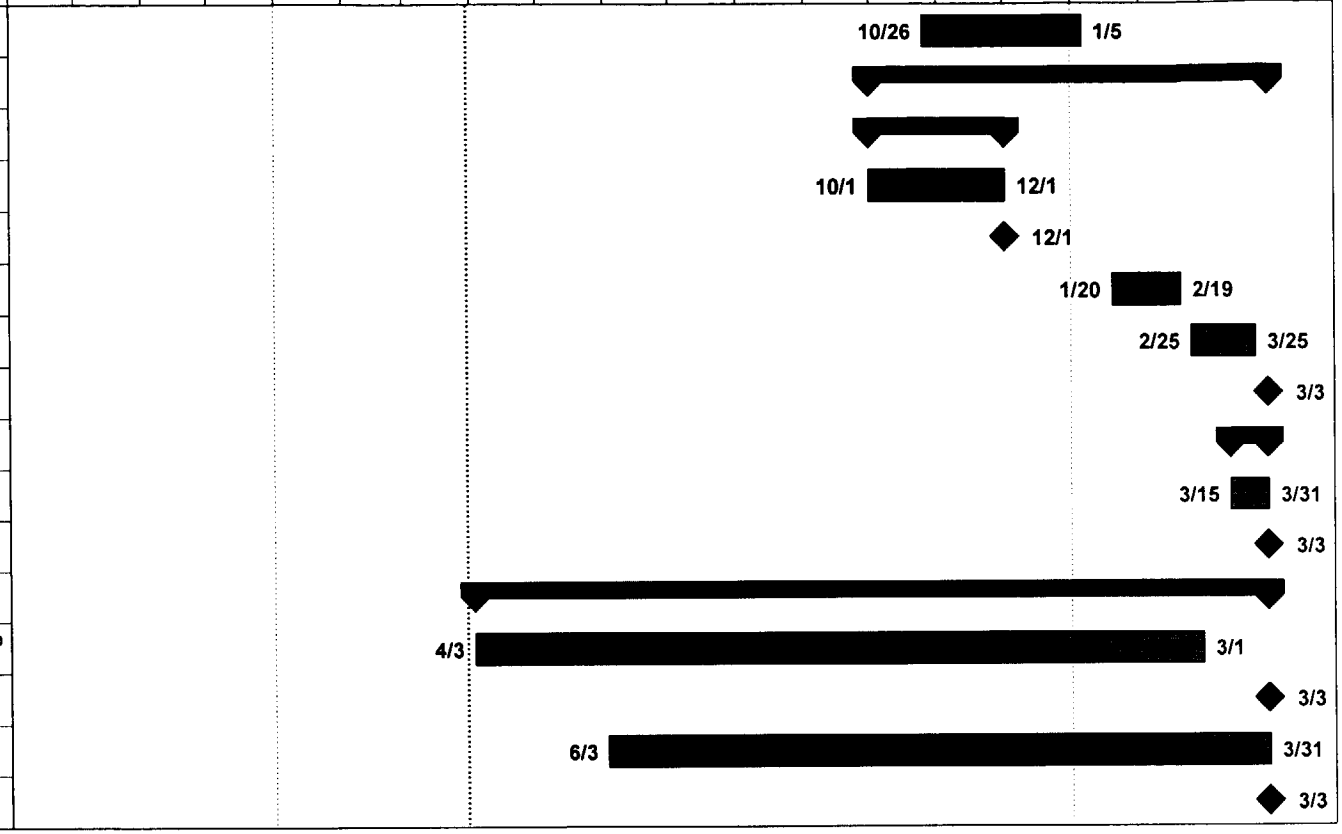
S3B ITT REHOST, INSTALLATION AND SUPPORT

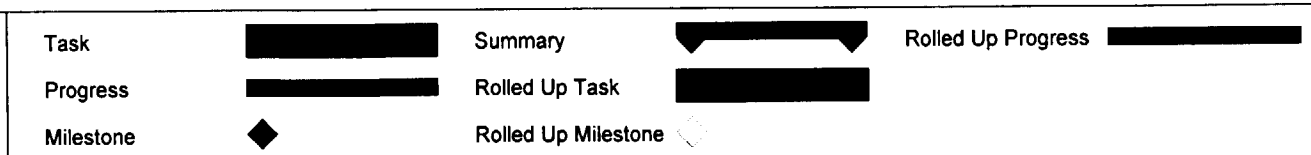
ID	Task Name	
25	Develop Phase I Requirements	11/4 – 2/10
26	Submit Phase I Requirements D	2/15
27	Port Existing H1200 S/W	11/24 – 2/15
28	Develop Communications Process	11/20 – 2/28
29	Add Required Functionality	12/17 – 2/28
30	Test and Evaluate S/W Rehost	1/15 – 2/28
31	**PHASE II**	
32	**Full ITT Requirements for SRF and WST**	
33	Develop Requirements Document	3/24 – 5/7
34	Submit Requirements Document	5/27
35	**Complete ITT Rehost to SRF**	
36	Continue T & E of S/W Rehost	3/2 – 4/30
37	Search Stores Instructor Control	3/2 – 6/1
38	Data Monitor Process	3/2 – 7/1
39	Search Stores Interface	5/1 – 7/1
40	Search Stores Display	5/1 – 7/1
41	IEER Status Tableau	6/1 – 7/15
42	**Test Full ITT on SRF**	
43	Contractor Testing	7/3 – 8/15
44	Government Testing	8/21 – 9/18
45	Provide Configured S/W	9/30
46	**PHASE III**	
47	Move ITT Hardware Suite #2 to Trainer Sit	10/5 – 10/8
48	Install ITT on S3B WST#1	10/12 – 10/23



S3B ITT REHOST, INSTALLATION AND SUPPORT

ID	Task Name
49	HSI of ITT onto WST#1
50	**Testing**
51	**Test Procedures**
52	Develop Test Procedures
53	Submit Test Procedures
54	Contractor Test
55	Government Test
56	Acceptance
57	**Test Results Report**
58	Develop Report
59	Submit Report
60	**Documentation**
61	Develop Requisite S/W Documentatio
62	Submit S/W Documentation
63	Develop O&M and User Manuals
64	Submit Requisite Manuals





Project:
Date: 3/30/98