Vehicle Weights and Dimensions Study

Volume 15

Graphic Representation of Heavy Vehicle Computer Simulation Model Output

Canroad Transportation
Research Corporation
1765 St. Laurent Blvd.
Ottawa, Canada K1G 3V4

ISBN: 0-919098-92-4

RTAC REPORT DOCUMENTATION FORM

Project No	Report No.	Report Date	IRRD No.
		July 25, 1986	

Project Manager

J.R. Pearson

Title and Subtitle

Volume 15 -- Graphic Representation of Heavy Vehicle Computer Simulation Model Output: A 3-D Animation Package for Computer Models

Author(s)	Corporate Affiliation(s)
R. Gagne T. Stock	Systems Laboratory Division of Mechanical Engineering National Research Council of Canada

Sponsoring/Funding Agency and Address	Performing Agency Name and Address
Canroad Transportation Research Corporation 1765 St. Laurent Blvd. Ottawa, Canada K1G 3V4	**Roads and Transportation Association of Canada** 1765 St. Laurent Blvd. Ottawa, Canada K1G 3V4

Abstract

An animation procedure has been developed which allows animation data generated by a computer model to manipulate a 3 dimensional graphic image. The graphic is generated and manipulated using the 3D graphics package MOVIE.BYU, produced by Brigham Young University. The MOVIE software was modified to allow inputs to be received from an animation file. Routines are provided to allow the computer model to change this animation file, and to create and control the process executing MOVIE. The graphics package and animation procedure run on a DEC VAX computer using a Testronix compatible graphics terminal.

Keywords

computer
animation
simulation
graphics

No. of Pages	No. of Figures	Language	Price
76		English	

Supplementary Information

DISCLAIMER

This publication is produced under the auspices of the Technical Steering Committee of the Vehicle Weights and Dimensions Study. The points of view expressed herein are exclusively those of the authors and do not necessarily reflect the opinions of the Technical Steering Committee, Canroad Transportation Research Corporation or its supporting agencies.

This report has been published for the convenience of individuals or agencies with interests in the subject area. Readers are cautioned that the use and interpretation of the data, material and findings contained herein is done at their own risk. Conclusions drawn from this research, particularly as applied to regulation, should include consideration of the broader context of Vehicle Weights and Dimension issues, some of which have been examined in other elements of the research program and are reported on in other volumes in this series.

The Technical Steering Committee will be considering the findings of these research investigations in preparing its "Final Technical Report" (Volume 1 & 2), scheduled for completion in December 1986.

PREFACE

The report which follows constitutes one volume in a series of sixteen which have been produced by contract researchers involved in the Vehicle Weights and Dimensions Study. The research procedures and findings contained herein address one or more specific technical objectives in the context of the development of a consistent knowledge base necessary to achieve the overall goal of the Study; improved uniformity in interprovincial weight and dimension regulations.

Dr. Roland Gagne and Mrs. Terry Stock of the National Research Council of Canada undertook the task of developing an interface mechanism between a commercially available animation program and the University of Michigan's computer simulation models used to predict the dynamic behaviour of heavy articulated vehicles. In pursuit of the objective of improving Canadian research capabilities in the vehicle dynamics area, it was hoped that the animation technique would make computer simulation more accessible to, and more easily understood by, the transportation community at large.

The points of view expressed herein are those of the authors and do not necessarily reflect the opinions or policies of Canroad Transportation Research Corporation or its supporting agencies.

Canroad Transportation Research Corporation gratefully acknowledges the generous support provided by the National Research Council of Canada to carry out this task, and in particular to the Division of Mechanical Engineering for donating the time and resources of the Systems Laboratory for this endeavour.

Funding to conduct the Vehicle Weights and Dimensions Study was provided to Canroad Transportation Research Corporation by:

Alberta Transportation
British Columbia Ministry of Transportation and Highways
Manitoba Highways and Transportation
New Brunswick Department of Transportation
Newfoundland Department of Transportation
Nova Scotia Department of Transportation
Ontario Ministry of Transportation and Communications
Prince Edward Island Transportation and Public Works
Ministere des Transports du Quebec
Saskatchewan Highways and Transportation
Transport Canada
Motor Vehicle Manufacturers Association
Canadian Trucking Association
Truck Trailer Manufacturers Association
Private Motor Truck Council

John Pearson, P. Eng.
Project Manager
Vehicle Weights and Dimensions Study

VEHICLE WEIGHTS AND DIMENSIONS STUDY
TECHNICAL STEERING COMMITTEE

Project Manager	John R. Pearson, Senior Programs Manager, Roads and Transportation Association of Canada
Chairman	M.F. Clark, Associate Deputy Minister (Engineering), Saskatchewan Highways and Transportation

Members

Dr. J.B.L. Robinson, Director of Technical Programs, Roads and Transportation Association of Canada

M. Brenkmann, Director, Research Program Development, Transport Canada

M.W. Harrin, Manager, Vehicle Standards Office, Ontario Ministry of Transportation and Communications

R.J. Lewis, Special Consultant, Canadian Trucking Association

M. Ouellette, Manager, Engineering, Mack Canada Inc.

R. Saddington, National Technical Advisor, Esso Petroleum Canada

W.A. Phang, Head, Pavement Research Division, Ontario Ministry of Transportation and Communications

G. Tessier, Direction de la recherche, Ministère des Transports du Québec

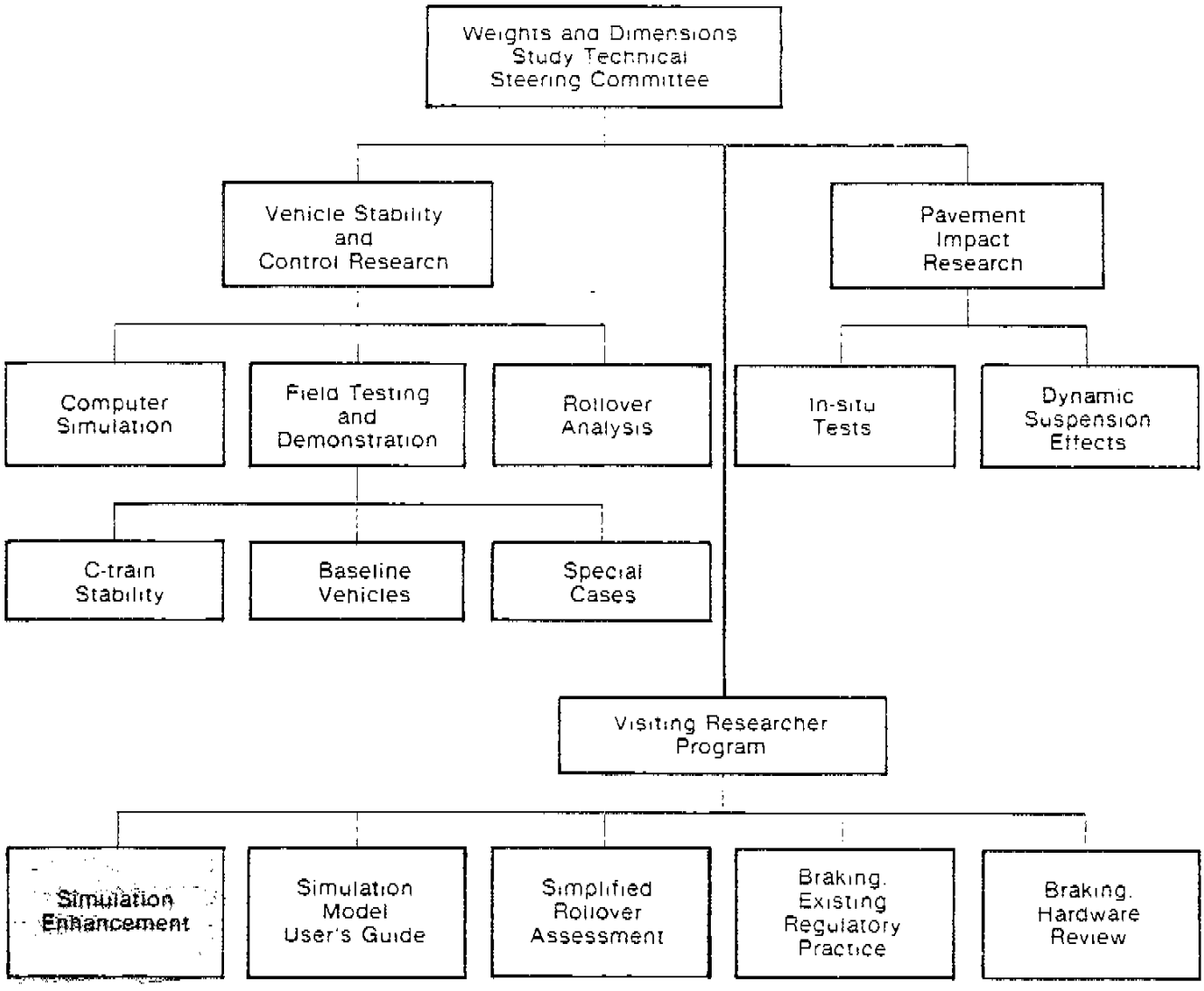
E. Welbourne, Head, Vehicle Systems, Transport Canada

R. Zink, Chief Engineer, North Dakota State Highway Department (representing AASHTO)

D.J. Kulash, Assistant Director, Special Projects, Transportation Research Board

HEAVY VEHICLE WEIGHTS AND DIMENSIONS STUDY

TECHNICAL WORK ELEMENTS OVERVIEW



Volume 15

A 3D ANIMATION PACKAGE FOR COMPUTER MODELS

Dr. R. Gagne
T. Stock

Systems Laboratory
Division of Mechanical Engineering
National Research Council of Canada

July, 1986

INTRODUCTION

When a computer model is run, there is usually much to be gained by viewing the results as the run proceeds. This is usually possible as most facilities allow interaction with programs at run time, using graphic terminals and suitable software.

Creating computer models with graphic output is usually done as one programming task, where the graphic displays are included as an integral part of the model. Creating a computer model requires quite different skills from that required to create a comuter graphic. This project attempts to separate the modelling task from the graphic task, where each is done separately, and where the interface between the two activities is clear and simple.

This project was done for a study being conducted by the Roads and Transportation Association of Canada (RTAC), the Canadian Vehicle Weights and Dimensions Study. In Canada trucking is regulated by the ten provinces and two territorial governments who exercise control over the size, weights and configurations of vehicles using their roads. The RTAC study is to provide a scientific basis for these regulations, and attempt to reduce the difficulties of interprovincial operation of vehicles. It is being supported by all governments and by many fleet operators.

Key components of the Vehicle Weights and Dimensions Study are to be computer models of vehicles which will accumulate information as it becomes available, with the goal of eventually using these models to predict the performance of any vehicle on any road. These models would be made available to regulators and operators alike, who could use them in evaluating alternatives.

An important component of any computer model, and especially models to be used by many people, is the "user friendliness" of the model -- the model must be easy to modify to the system being considered, and the results of runs must be easily understood.

The computer models chosen for the vehicle dynamics were those produced by the University of Michigan Transportation Research Institute. These models had reached a high level of maturity and were well structured so as to allow additional modules to be added.

The Systems Laboratory of the Mechanical Engineering Division of NRC was asked to install the vehicle models in their VAX computer, and asked to seek ways of improving their ease of use. This animation project was one of the results.

OVERVIEW OF ANIMATION

An overview of the procedure to animate a 3D graphic image is shown in fig. 1. The computer model of some system is run as a separate process in the VAX computer. Routines executed in the computer model are used to modify a file which contains the animation information for the image generated by a concurrent display process also running in the VAX. The graphic image is previously created using the graphic package, and the animation file contains only those commands which change the image.

To use this animation procedure, the following tasks must be done:

* Create the model of the 3D graphic which is to be animated

* Create an animation command file which contains MOVIE commands to operate on the graphic model for the animation sequence. This defines the animation variables which are to be set by the computer model.

* Include in the computer model routes which will compute the required animation variables.

* Add animation control routines to the computer model to create the display process, then repeatedly set the values of the animation variables, and synchronize the execution of the animation process.

The first two tasks involve the direct use of the 3D graphics package MOVIE.BYU. and this will be discussed first. Then the routines to set up and control a graphic process will be described.

MOVIE.BYU

MOVIE is a graphics package which provides for the creation, display and manipulation of a segmented three dimensional graphic model whose geometry is described in terms of polygonal elements each with an arbitrary number of nodes. Connections between the polygonal elements in three space is defined by common nodes for connecting elements. Solids can be used in defining the 3D image -- a routine is provided to decompose solid elements into equivalent polygonal elements.

The generated view of the 3D graphic is defined by the position of the observer, and his angle and direction of sight. The view can be as a "wire frame" model, or include the removal of hidden lines.

A number of animation operations can be performed on the 3D model, not all of which are used in this project. The model can be made up of any number of segments, each of which can be manipulated separately by the PIVOT operation, which is the main animation operation used. Another useful operation is ROTATE, which rotates the whole image about any axes.

The complete MOVIE package consists of the 6 modules summarized in Figure 2. This figure uses ellipses to denote data modules, and rectangles for program modules. It also groups the MOVIE commands as pseudo menu trees. (The program is command driven, not menu driven). It includes the additional commands added to the MOVIE package for this project and which will be described later.

Four of the modules are for the creation of graphic images, one for the playback of previously recorded images, and one for the manipulation of images.

1) UTILITY: This module is the main one used to create and edit graphics data files. It allows input of the polygons in three space that make up the graphic. These can be grouped into the separate segments which can be acted upon independently. It also allows data entry in the form of solid elements or as mathematical functions.

2) SECTION: This module is used to decompose solid elements into the equivalent polygonal elements.

3) MOSAIC: This module produces polygonal elements from contour data.

4) TITLE: This module produces polygonal elements defining 2 or 3 dimensional text.

5) DISPLAY: This is the module used for the manipulation and display of the graphics defined by the other modules. It displays graphic data files of polygonal elements using a defined observer position, and can remove hidden lines. It can translate, rotate or scale segments of the model. It has other operations, some involving color, which are not used in this project.

6) COMPOSE: This module combines images previously displayed and saved by DISPLAY.

Modifications were made to the standard MOVIE.BYU software to enhance its ability to animate linked structures. These additions were in four groups.

* An addition to the DISPLAY module to allow hard copy plots to be generated. The functions PLOT and NOPLOT turned the hard copy feature on and off.

* Another addition to the DISPLAY module was the function FILE which switched the control of DISPLAY from keyboard entries to records of any file. This allowed DISPLAY to be driven from an animation file which could be set up and altered from another program. A companion function was the HALT function which switched control back to the keyboard. A PAUSE function was added which halted the file driven control of DISPLAY until released by keyboard activity.

* A function RESCEN was added to DISPLAY to return all geometry to the initial configuration, and center the image on the display device.

* A new function ORIGIN was added to the UTILITY package to make animation of linked mechanisms easier. This allowed the segments of the image to be connected together at defined points to form a linked chain of parts. Modifications were made to the PIVOT command of display to preserve any links defined by ORIGIN.

A listing of the modified modules is contained in the appendix.

ANIMATION CONTROL

Animation of the graphic model created by MOVIE is achieved by modifying the animation file. This is done by the computer model which is generating the animation information. Routines were created to set up the display process, modify the animation file, generate one frame on the display, and finally to terminate the display process.

The model routines to set up and control the display process are:

```
*    Set up a Display Process

     CALL RUN_MOVIE (FGEOM, FDISP, FFUN, FORG)

     where

     FGEOM - geometry file defining the graphic
     FDISP - displacement file (optional)
     FFUNC - function file (optional)
     FORG - parts of origin file defining linkage
            points of a chain

*    View one frame of animation

     CALL DISPLAY_FILE (FCOMM)

     where

     FCOMM - animation file to be used.
          This file is created by the user
          using any text editor and contains
          the MOVIE commands to perform the animation
          operations.

*    Alter fields of an animation file

     CALL ALT_REC (FCOMM, IREC, ITEM, VALUE, M)

     where

     FCOMM - animation file
     IREC - integer array of record numbers to be altered
     ITEM - integer array of field numbers to be altered in corresponding
            record of IREC
     VALU - real array of new field values for the records
     M    - total number of records to be altered

*    Terminate the display process

     CALL EXIT_MOVIE
```

An interactive version of the routines to alter the animation file have also been provided. To alter fields of an animation file interactively, execute the routine

```
     CALL MOD_FILE (FCOMM)

     where

     FCOMM - animation file to be accessed
```

The MOD_FILE prompt is "::" and the commands available are:

C - Change one item in a record
D - Delete one record
H - Help
I - Insert a new record
L - List all records
P - Print one record on the screen
Q - Quit
R - Replace one record

Listings for all these routines are attached. The MOVIE.BYU package is available at modest cost from M.B. Stephenson, Civil Engineering-370 CB, Brigham Young University, Provo, UT 84602.

EXAMPLE

Figure 3 shows a graphic model of a vehicle to be animated. The vehicle consists of three parts, a tractor which pulls the vehicle, an attached semi-trailer followed by a trailer. the geometry is defined with a coordinate system which has the x axes along the center line of the vehicle, goes through the truck hitch points and has its origin on the front bumper. The y axes is perpendicular to this,and the z axes points upward. Each part of the vehicle can have roll and yaw independently of the others, but each is connected to its neighbour at the hitch points.

The new UTILITY function ORIGIN is used to connect the parts of the vehicle together. The command ORIGIN requests entry of:

* the first and last segments of the group of segments to be treated as one link
* the segment number to which this link is connected
* the x y z coordinates of the connecting point

These utilities are repeated for each link of the chain.

The UTILITY package was used to define the geometry of the vehicle, and is contained in the file TRUCK.GEO. (A discussion on how to create these geometry files for any vehicle shape is described in the next section.) The drawing consists of four parts. Part 1 is the road, part 2 the tractor, part 3 the semitrailer, and part 4 the trailer. The road is manipulated as a separate part, but the vehicle parts must be connected together at the hitch points. The ORIGIN command establishes this, and the content of the origin file TRUCK.ORG was:

```
2  2  0  0  0  0
3  3  2  5  0  0
4  4  3 15  0  0
```

This defined each of the truck parts as a separate link, connected to its neighbour at the two hitch points along the x axes.

The animation file TRUCK.ANA was created which contained the MOVIE commands required to generate one frame of the sequence, shown in figure 4. These commands were:

```
EXPL      - Invoke the EXPLODE function
1 1 00 #  - where # is the z position of the road
          - blank line terminates function
1         - scale factor for EXPLODE
PIVOT     - Invoke the PIVOT function
2 2 Y #   - where # = part 2 yaw
3 3 Y #   - where # = part 3 yaw
4 4 Y #   - where # = part 4 yaw
2 2 X #   - where # = part 2 roll
3 3 X #   - where # = part 3 roll
4 4 X #   - where # = part 4 roll
```

```
            - blank line - terminate PIVOT
ROTA        - Invoke ROTATE function
X #         - where # = rotation about X
VIEW        - send frame to screen
HALT        - return to calling program
```

The animation file of MOVIE commands thus contains 16 records. The 5th field of the second record defines the position of the road center from the middle of the vehicle front bumper. The 4th field of the sixth to eighth records contains the yaw information, and the 4th field of the ninth to eleventh records the roll information. The second field of the forteenth record defines the observer's position.

Code must be added to the model to set up and control the animation. FORTRAN versions of functions to do this are:

The display process is set up with the statement

```
CALL RUN_MOVIE ('TRUCK.GEO','','','TRUCK.ORG')
```

Since 8 of the 16 records must be supplied for the animation information, the user's model must contain information to supply this information and control the animation frames. The following declarations will set up pointers to the animation variables in the animation file:

```
DIMENSION IREC (8), ITEM (8), VALUE (8)
DATA IREC/2,6,7,8,9,10,11,14/
DATA ITEM/5,4,4,4,4,4,4,2/
```

Values of the animation variables will be set in the model using statements setting values in the VALUE array:

```
VALUE(1) = -6.3  ! position of road in z
VALUE(2) = 22.5  ! tractor yaw
VALUE(3) = 10.0  ! semitrailer yaw
           etc.
```

The animation file is updated to the new values of animation variables by the statement:

```
CALL ALT_REC('TRUCK.ANA',IREC,ITEM,VALUE'8)
```

The display process is instructed to generate the next animation frame with the statement:

```
CALL DISPLAY_MOVIE ('TRUCK.ANA')
```

This code is repeated for each frame of the animation sequence, using updated values of the animation variables until the model is to be stopped. The display is stopped by the statement:

```
CALL EXIT_MOVIE
```

CREATING A GEOMETRY FILE

Creating a geometry file for any vehicle shape involves the use of the UTILITY commands of the MOVIE.BYU package. It requires that the vehicle shape be broken down into the polyhedra surfaces that describe it. Each node on each surface is placed using its x,y,z position relative to an assumed origin.

A number of geometry files have been created for standard vehicle configurations and these drawings are attached. Listings or copies of these files can be obtained from the writers.

Most of the MOVIE commands are straight forward to use, but since several are involved, familiarity with the whole MOVIE package is required. Rather than attempt this, it will be assumed that reference can be made to the MOVIE manual, and only some useful hints will be given here.

As much as possible should be made of symmetry, as commands are available to reflect elements about the drawing axes. Use of an axes system passing through the center of the vehicle is suggested. With this, for example, only one wheel shape is required which can be reflected to the other side, and duplicated at each wheel position. These simple parts can be combined to form the more complex parts of the vehicle pieces.

Many other vehicle pieces are relatively standard ie, semi-trailers, dollies, trailers, etc. These too can be created as separate geometry files, which can be merged together to create a more complex drawing.

PERFORMANCE

The first version of this procedure was run on a VAX 11/780 using a Cybernex 1012 graphics terminal. This terminal is a high resolution (1000 line) black and white terminal of modest cost ($2000) with built in Textronix emulation. The quality of the graphic image was quite acceptable, but the image took some seconds to be generated. This mode of animation can be used on-line by the user at the terminal who can then monitor the run, or the generated views can be recorded on videotape or film and run later as a movie. This mode of running is how MOVIE is normally used, and from which it got its name.

Enhanced performance for on line animation would require graphics facilities with higher performance, with the computer model and the graphics model running in separate computers. This would be particularly effective if the graphics device had double frame buffering capability. This would allow one frame to be viewed as the other is drawn in the other buffer, resulting in immediate updating. An evaluation of this alternative is now being developed using an IBM PC/AT, with the Professional Graphics display as the display adapter.

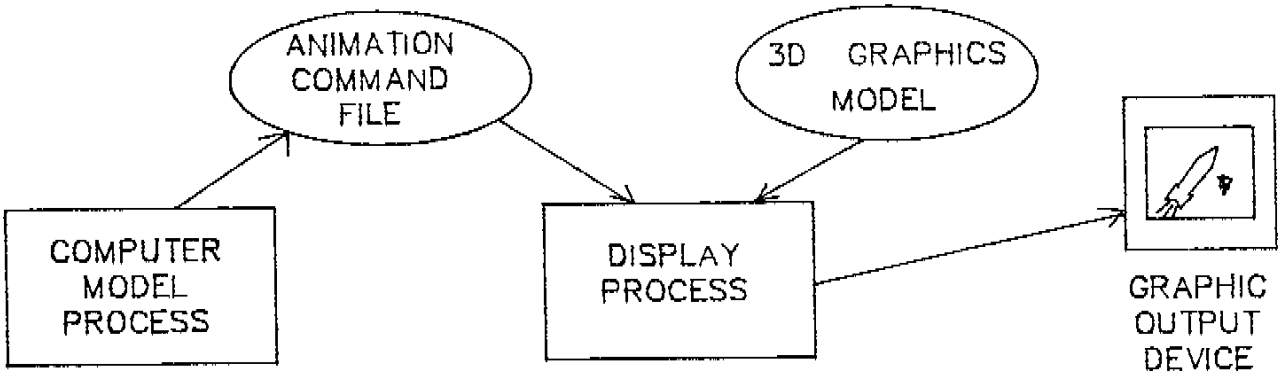


Figure 1 Overview of the animation procedure

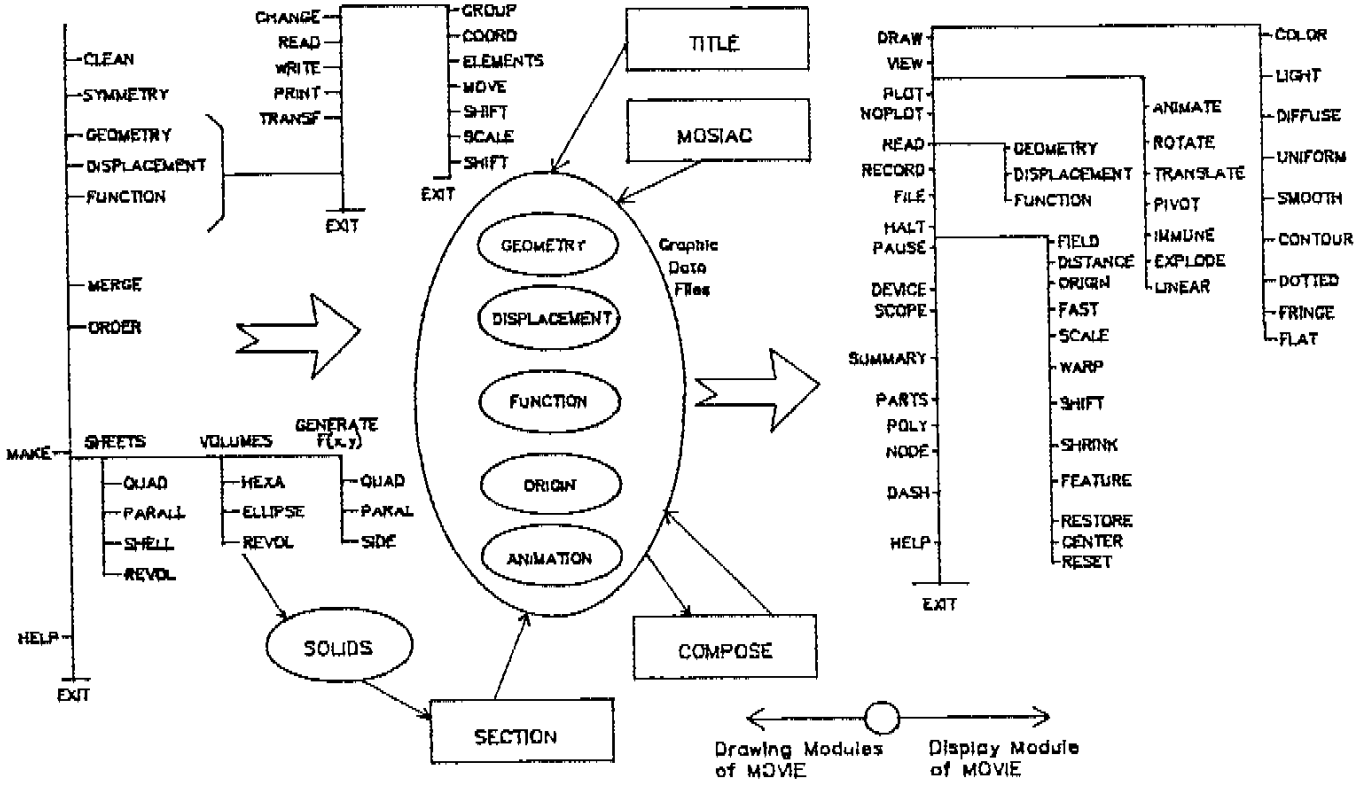


Figure 2 Overview of the MOVIE Package

Figure 3: B-Dolly Graphic Model

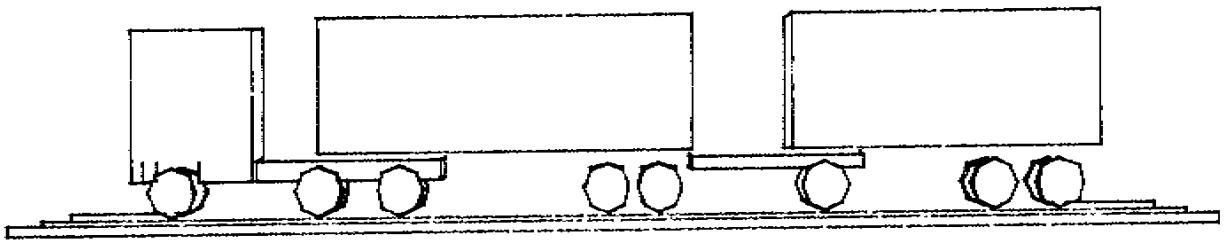


Figure 4: One Frame of Animation Sequence

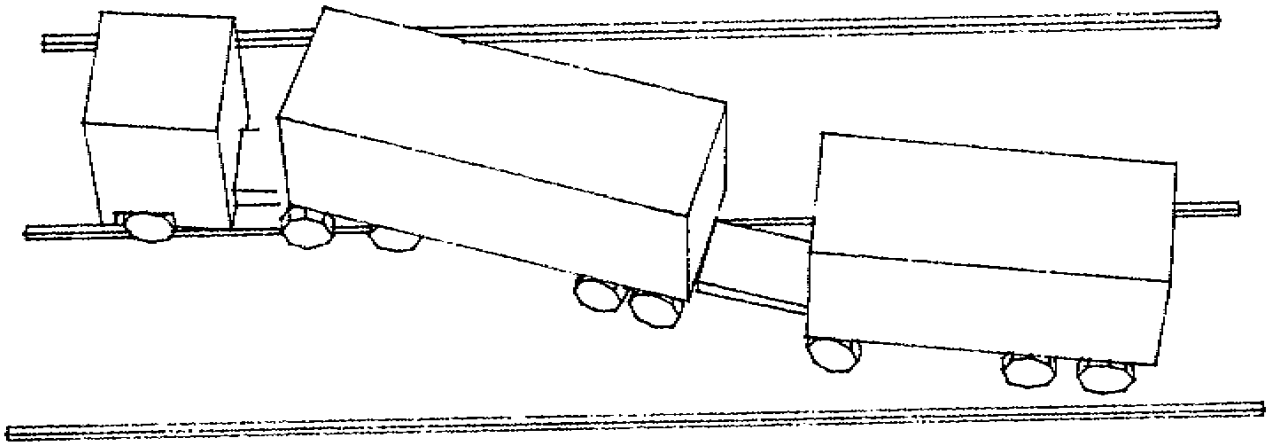


Figure 5: Truck A: 2-Axle, 16' Box

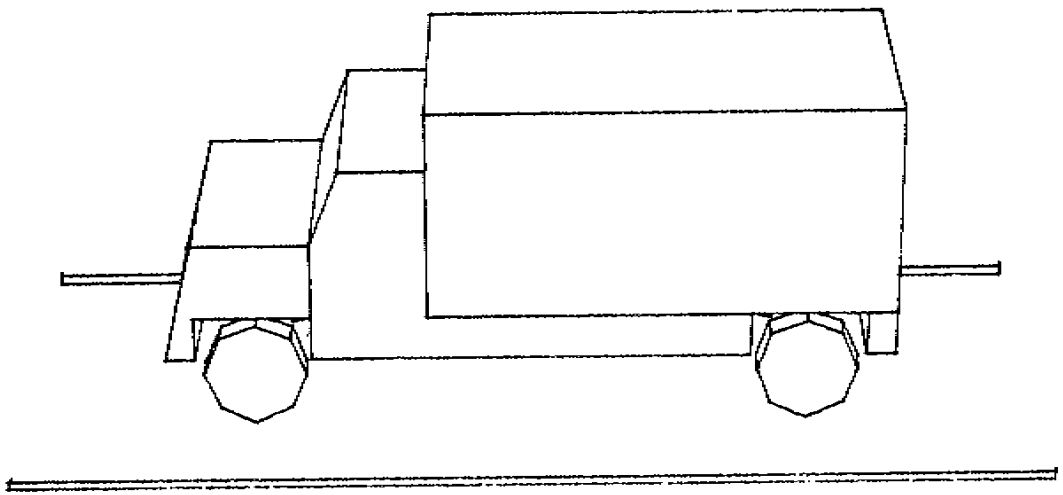


Figure 6: Truck No 3-Axle, 18' Box

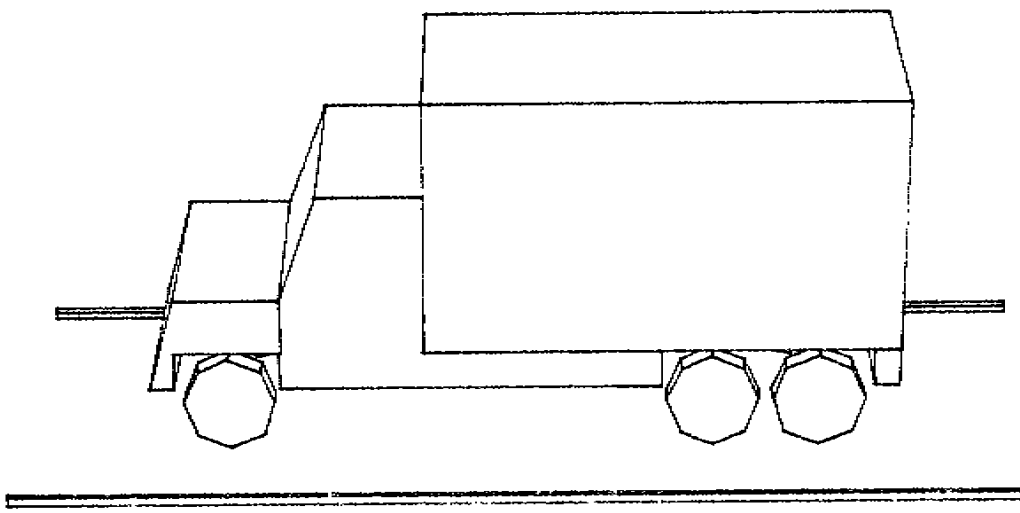


Figure 7: Truck C: 3-Axle, 27' Semi

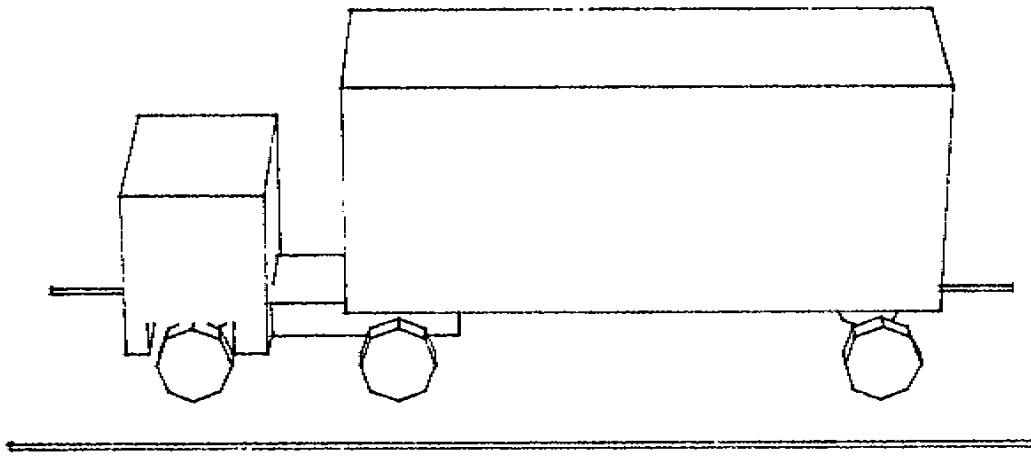


Figure 8: Truck D, 5-Axle, 45' Semi

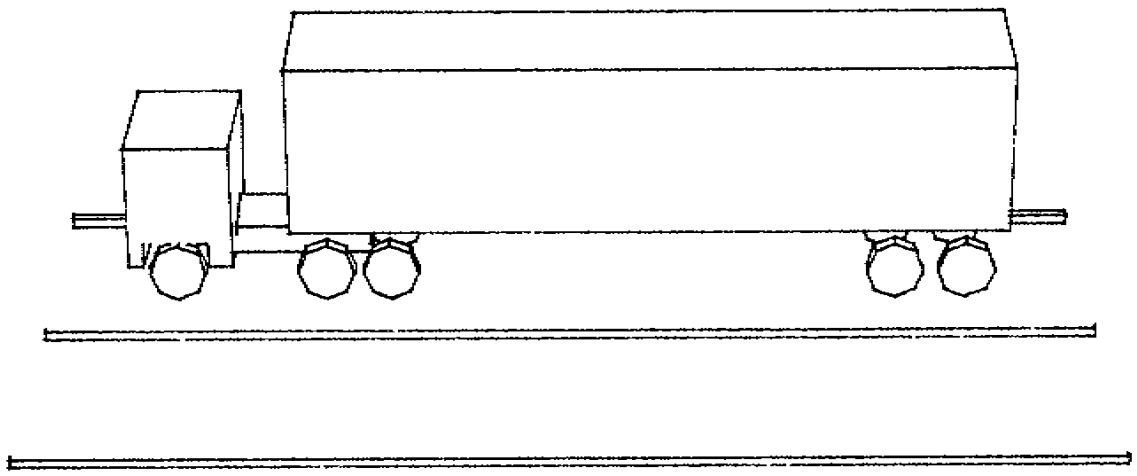


Figure 9: Truck E: 4-Axle, 45' Semi

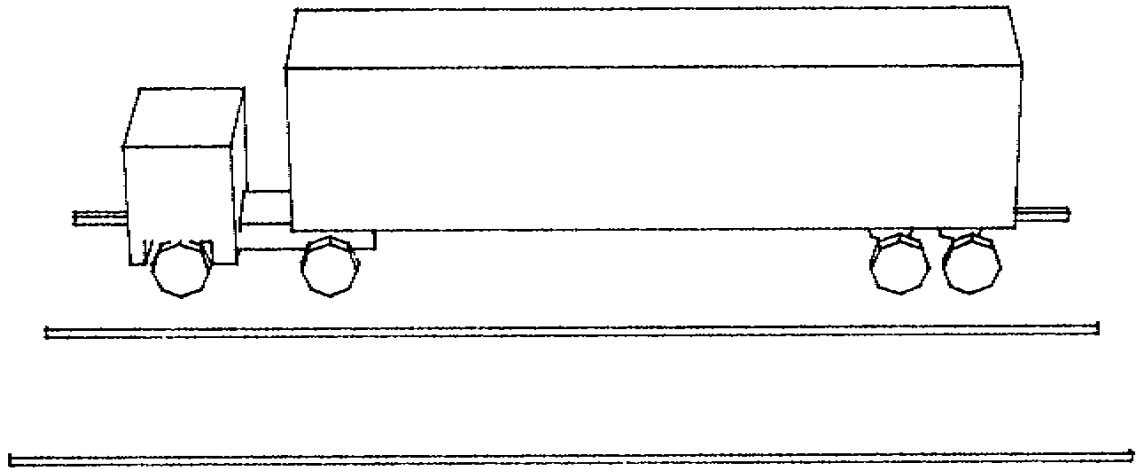


Figure 10: Truck F: 5-Axle, 27' Semi, 27' Trailer

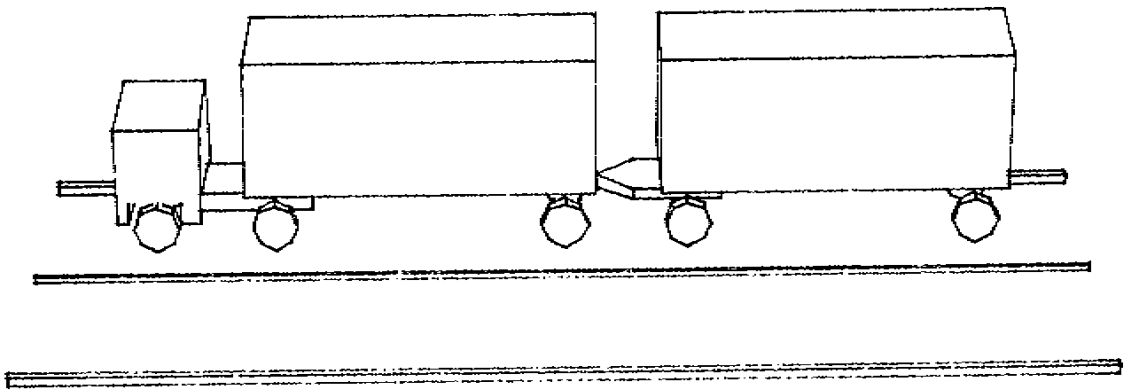


Figure 11: Truck 6: Rocky Mountain Double, 45' Semi, 27' Trailer

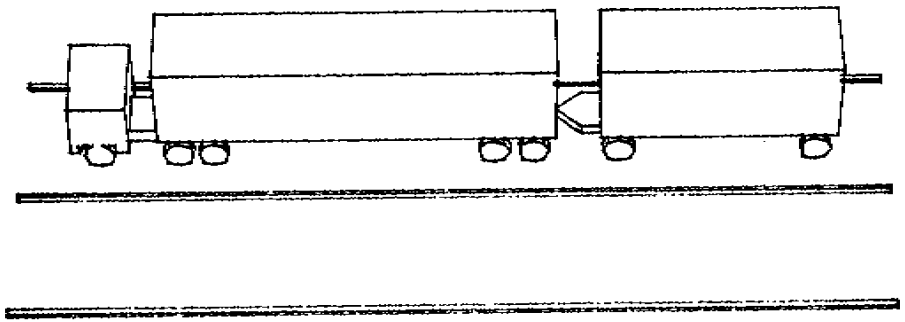


Figure 12: Truck #8: Turnpike Double, 48' Semi and Trailer

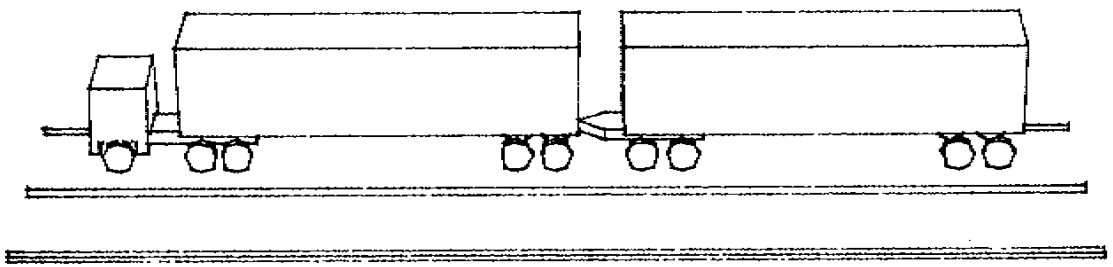
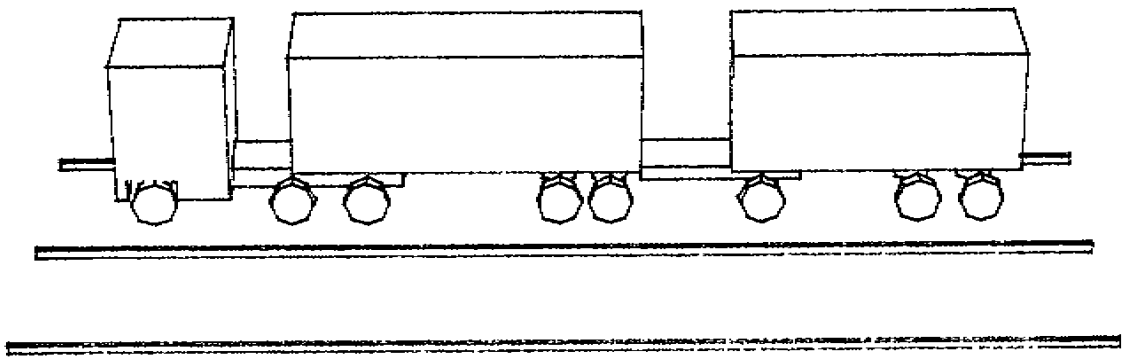


Figure 13: Truck BD: 8-Axle, 27' Semi and Trailer, B-Dolly



Appendix A

Modified MOVIE.BYU Routines

DISPLA
MULTID
PIVOT
READF
VIEDRA
GETXF
NUMDIG

Modifications indicated with
CC
in columns 1 and 2

```
      PROGRAM DISPLA
C*****************************************************************************
C
C         COMMAND.FOR          VERSION 5.3          NOVEMBER 1985
C
C        A GENERAL PURPOSE COMPUTER GRAPHICS DISPLAY PROGRAM FOR
C                POLYGONAL DATA WITH LINE DRAWING AND
C                 CONTINUOUS-TONE PHOTOIMAGE OUTPUT
C
C        TO ASK QUESTIONS CONCERNING THIS PROGRAM OR TO REPORT BUGS,
C        CONTACT :
C                          DR.  BRUCE J  NAY
C                               - OR -
C                          HANK CHRISTIANSEN
C                          CIVIL ENGINEERING
C                          368 CB          BYU
C                          PROVO, UTAH 84602
C                          (801)  378 - 2812
C
C        PLEASE TRY TO RESTRICT CALLS TO THE FOLLOWING HOURS:
C                  MWF      11:00 am TO  1:00 pm      (Mountain Time Zone)
C                  TTH       9:00 am TO 11:00 am
C
C*****************************************************************************
C
C   INTERACTIVE COMMANDS ARE
C      ALIAS      = ENABLE ANTI-ALIASING OPTION
C      ANIMATE    = SPECIFY ANIMATION SEQUENCE
C      CENTER     = TRANSLATES ORIGIN TO CENTER OF MODEL
C      COLOR      = SELECT COLORS FOR BACKGROUND, PARTS, AND FRINGES
C      CONTOUR    = SELECT CONTOUR LEVELS
C      DASH       = SPECIFY DOTTED LINE PARAMETERS
C      DEVICE     = SET DISPLAY DEVICE
C      DIFFUSE    = SET DIFFUSED LIGHT INTENSITY OF INDIVIDUAL PARTS
C      DISTANCE   = SET DISTANCE FROM OBSERVER TO MODEL
C      DOTTED     = ENABLE OR DISABLE DOTTED LINE OPTION
C      DRAW       = SENDS PICTURE TO DISPLAY DEVICE (NO HIDDEN LINES)
C      EXIT       = TERMINATE PROGRAM EXECUTION
C      EXPLODE    = SPECIFY LOCAL MOTION (EXPLOSION)
C      FAST       = INVOKES POOR MAN'S OPTION
C      FEATURE    = SET FEATURE ANGLE
C      FIELD      = SPECIFY FRUSTUM OF VISION
CC     FILE       = ENABLE COMMAND INPUT FROM DISK FILE
C      FLAT       = USE FLAT SHADING
C      FRINGE     = SPECIFY COLOR FRINGES FOR DISPLACEMENT OR
C                   SCALAR FUNCTION SYSTEM
C      GLASS      = ENABLES TRANSPARENCY OPTION AND SETS PARAMETERS
CC     HALT       = ABORT INTERACTIVE COMMAND INPUT FROM DISK FILE
C      HAZE       = ENABLES HAZE/FOG OPTION AND SETS PARAMETERS
C      HELP       = TYPE COMMANDS
C      IMMUNE     = SET PARTS IMMUNE TO ROTATIONS
C      LIGHT      = SETS CONTINOUS TONE INTENSITY PARAMETERS
C      LINEAR     = LINEAR INTERPOLATION BETWEEN DISPLACEMENT
C                   OR SCALAR FUNCTION FILES
C      MULTIPLE   = ENABLE OR DISABLE PREVIOUSLY DEFINED LIGHT SOURCES
C      NODE       = ENABLE OR DISABLE NODE NUMBERING OPTION
CC     NOPLOT     = DISABLE HARDCOPY OUTPUT OPTION
CC     ORIGIN     = SPECIFIY ORIGIN (PIVOT POINT) OF PARTS
C      PART       = SPECIFY PARTS TO BE DISPLAYED
CC     PAUSE      = PAUSE BETWEEN COMMANDS WITH OPTION FOR
```

```
C                    ABORTING THE "FILE" COMMAND OPTION
C      PIVOT       = ROTATE MODEL ABOUT LOCAL AXES
C      POLY        = ENABLE OR DISABLE POLYGON NUMBERING OPTION
CC     PLOT        = ENABLE HARDCOPY OUTPUT OPTION
C      READ        = READ NEW DATA FILES
CC     RESCENTER   = RESTORE GEOMETRY  AND  RECENTRE THE MODEL
C      RECORD      = SWITCH FOR SAVING PICTURES ON DISK FOR USE WITH
C                    "COMPOSE"
C      RESET       = READS THE RECORDED VALUES OF VARIABLES FROM DISK
C      RESTORE     = RESTORE GEOMETRY TO INITIAL CONDITION
C      ROTATE      = ROTATE MODEL ABOUT GLOBAL AXES
C      SAVE        = SAVES THE CURRENT VALUE OF PARAMETERS ON DISK
C      SCALE       = SET SCALE FACTOR FOR DISPLACEMENT FUNCTIONS
C      SCOPE       = SET SCOPE PARAMETERS
C      SHADOW      = ENABLES SHADOW OPTION AND SETS PARAMETERS
C      SHIFT       = MOVE VIEWING SCREEN IN X OR Y DIRECTION
C      SHRINK      = SET SHRINK FACTOR
C      SMOOTH      = USE SMOOTH SHADING
C      SUMMARY     = GIVE MAXIMUM AND MINIMUM VALUES OF DATA FILES READ
C      TRANSLATE   = TRANSLATE LOCAL ORIGIN OF MODEL
C      UNIFORM     = USE UNIFORM SHADING
C      VIEW        = DISPLAY SCENE ON PRECISION DISPLAY
C      WARP        = SET SCALE FACTOR FOR SCALAR FUNCTIONS
C
C*************************************************************************
C
C   VARIABLE DIMENSION INFORMATION FOR "COMMAND.FOR" AND "HIDDEN.FOR"
C
C     (1) ICNMAX = MAX. NO. OF ELEM.(NPTMAX)*MAX. NO. OF SIDES(NSMAX)
C                    DIMENSION IP(ICNMAX)
C     (2) NFRINM = MAXIMUM NUMBER OF FRINGES
C                    DIMENSION CFRIN(3,NFRINM)
C     (3) NJMAX  = MAXIMUM NUMBER OF NODES
C                    DIMENSION SCOORD(3,NJMAX),SPEC(NJMAX),SPECI(NJMAX)
C                   1,U(3,NJMAX),X(3,NJMAX),XNORM(3,NJMAX),YN(3,NJMAX)
C                   2,YNORM(3,NJMAX)
C     (4) NPMAX  = MAXIMUM NUMBER OF PARTS
C                    DIMENSION DA(3,NPMAX),DC(3,3,NPMAX+1),DD(3,3,NPMAX)
C                   1,DIF(NPMAX),DIRCA(NPMAX),FRING(2,NPMAX),ICOL(NPMAX)
C                   2,JSMOTH(NPMAX),LASP(NPMAX+1),NFR(NPMAX),NHIGH(NPMAX)
C                   3,NONROT(NPMAX),NPL(2,NPMAX),NPLS(NPMAX),POOR(NPMAX)
C                   4,RORG(3,NPMAX),XIO(NPMAX),XNH(NPMAX),XNR(NPMAX)
C                   5,XX(3,NPMAX)
C                   6,ISHFLG(NPMAX),JSHFLG(NPMAX),XMAXT(NPMAX)
C                   7,XMINT(NPMAX),TPWR(NPMAX)
CC                  8,PORG(3,NPMAX),SORG(3,NPMAX),IPREV(NPMAX)
C     (5) NPTMAX = MAXIMUM NUMBER OF ELEMENTS(POLYGONS)
C                    DIMENSION NCENPT(NPTMAX)
C     (6) NSMAX  = MAXIMUM NUMBER OF SIDES OF POLYGONS
C                    DIMENSION CONT(NSMAX+1),JCOL(NSMAX),NITRI(NSMAX+1)
C                   1,NN(NSMAX+1),NNN(NSMAX),XN(3,(NSMAX+1))
C                   2,XP(3,(NSMAX+1)),XQ(3,(NSMAX+1)),AX(NSMAX)
C                   3,AY(NSMAX)
C     (7) MAXFRE = MAXIMUM SIZE OF FREE STORAGE
C                    DIMENSION IFREE(MAXFRE),ISEG(MAXFRE),RSEG(MAXFRE)
C     (8) MAXCNT = (2*NSMAX) AS A MINIMUM IF NO CLIPPING TAKES PLACE.
C                    SO RECOMMEND APPROXIMATELY (4*NSMAX)
C                    DIMENSION VX(MAXCNT),VY(MAXCNT),VZ(MAXCNT),VN(MAXCNT)
C                   1,IC(MAXCNT),VC(MAXCNT),VTX(MAXCNT),VTY(MAXCNT)
C                   2,VTZ(MAXCNT),VTN(MAXCNT),VTC(MAXCNT),ITC(MAXCNT)
```

```
C     (9) MAXINS = 7*NUMBER OF INTERSECTION LINES
C                  DIMENSION INT(MAXINS),RNT(MAXINS)
C    (10) MAXFSL = MAXIMUM LENGTH OF FREE STORAGE LIST
C                  DIMENSION LIST(MAXFSL)
C    (11) MLSN   = MAXIMUM NUMBER OF LIGHT SOURCES
C                  DIMENSION SHASPE(MLSN),SHINI(MLSN),LSENAB(MLSN)
C                 1,LSPEC(MLSN),OSCURO(MLSN),NPNT(MLSN,NSHMAX)
C    (12) NTMAX  = MAXIMUM NUMBER OF TRANSPARENT LAYERS
C                  ZZL(NTMAX),ZZR(NTMAX)
C
C
C*****************************************************************************
C
C  MAIN PROGRAM - PROCESSES INTERACTIVE COMMANDS FROM THE USER
C      AND CALLS APPROPRIATE SUBROUTINES
C
C*****************************************************************************
C
C  SUBPROGRAMS CALLED
C     ALIA   = ENABLE THE ANTI-ALIAS FEATURE
C     ANIMAT = SELECT INCREMENTAL TRANSLATION,ROTATION,ETC.
C     CLEAR  = CLEARS ARRAYS (INITIALIZATION)
C     COLO   = SPECIFY COLORS FOR VARIOUS PARTS
C     CONT   = SELECT CONTOUR OPTION AND SET CONTOUR LEVELS
C     DASHLN = ENABLE THE DOTTED LINE FEATURE
C     DIFF   = SET DIFFUSED LIGHT INTENSITY BY PART
C     DIST   = SPECIFY DISTANCE TO COORDINATE ORIGIN FROM OBSERVER
C     DSET   = CHANGES DOTTED LINE PARAMETERS
C     EXIT   = RETURNS CONTROL TO MONITOR
C     EXPL   = EXPLOSION OF PARTS
C     FAST   = SET DATA OPTIONS AND POOR MANS HIDDEN SURFACE REMOVAL
C     FEAT   = SELECT FEATURE OPTION AND SET FEATURE ANGLE
C     FIEL   = SPECIFY FRUSTRUM OF VISION AND MIN. AND MAX. X-Y
C              CLIPPING PLANES
CC    FILE_OP= ENABLES COMMAND INPUT FROM DISK FILE
C     FRIN   = SELECT FRINGE OPTION AND SPECIFY FRINGED PARTS
CC    GET_ORG= SPECIFY ORIGIN (PIVOT POINT) OF PARTS
C     GLAS   = ENABLE TRANSPARENCY FEATURE AND ACCEPTS PARAMETERS
CC    HALT_OP= RESTORE COMMAND INPUT TO SYS$INPUT, CLOSE FILE
C     HAZE   = ENABLE HAZE/FOG FEATURE AND ACCEPTS PARAMETERS
C     HELP   = GIVE AVAILABLE COMMANDS OR OPTIONS
C     IMMUNE = MAKES PARTS IMMUNE TO ROTATIONS
C     INIT   = INITIALIZES NECESSARY PARAMETERS
C     LIGHT  = SET CONTINOUS TONE INTENSITY PARAMETER
C     LINE   = ADD PREVIOUS DISPLACEMENTS AND SCALAR FUNCTIONS TO
C              ARRAYS,READ NEW ARRAYS,AND DIFFERENCE FOR TRANSIENT
C              DATA
C     MULT   = ENABLE AND DISABLE PREVIOUSLY DEFINED LIGHT SOURCES
CC    NOPLOT = DISABLES HARDCOPY OUTPUT OPTION
CC    PAUS_OP= PROVIDES PAUSE BETWEEN COMMANDS (USE WITH "FILE")
C     PIVOT  = SET LOCAL ROTATION ABOUT RELATIVE ORIGIN
CC    PLOT_OP= ENABLES HARDCOPY OUTPUT OPTION
C     POLNOD = ENABLE OR DISABLE NODE AND/OR POLYGON NUMBERING
C     READ   = READS IN DATA FILES
C     READIN = ACCEPTS INPUT FROM USER
C     RECORD = ENABLES SWITCH FOR SAVING PICTURES ON DISK FOR USE
C              WITH "COMPOSE"
C     RESET  = READS THE RECORDED VALUE OF VARIABLES FROM DISK
C     REST   = RESTORE MODEL TO ORIGINAL COORDINATE SYSTEM
C              (KILLS ROTATIONS AND TRANSLATIONS)
```

```
CC     RES_CEN= RESTORES GEOMETRY AND RE-CENTRES THE MODEL
C      ROTA   = ROTATE MODEL ABOUT ORIGIN
C      SAVE   = WRITES THE CURRENT VALUE OF VARIABLES AND LOGICALS ON
C               TO DISK FOR USE WITH RESET
C      SCAL   = SPECIFY DISPLACEMENT SCALE FACTOR
C      SCOP   = SET SCOPE PARAMETERS
C      SEPART = SELECT CONTENT OF SCENE
C      SHADE  = SELECT FLAT, SMOOTH, OR UNIFORM SHADING
C      SHADO  = ENABLE SHADOW FEATURE AND ACCEPTS PARAMETERS
C      SHRKR  = SELECT SHRINK OPTION AND SPECIFY SHRINK FACTOR
C      SUMCEN = GIVE SUMMARY OF DATA READ WITH MIN./MAX. VALUES,
C               OR CENTER MODEL IN VIEWING AREA
C      TRAN   = TRANSLATE COORDINATE ORIGIN OF MODEL
C      VIEDRA = CALLS FOR NORMALS, LIGHT INTENSITY, ETC. NEEDED TO
C               DISPLAY SCENE (EITHER VIEW OR DRAW)
C      WARP   = SPECIFY OUT-OF-PLANE WARPING SCALE FACTOR
C
C*****************************************************************
C
C  VARIABLES USED
CC     APLOT  = ENABLES ONE HARDCOPY OUTPUT OF DISPLAY
CC     HRDCPY = LOGICAL FLAG INDICATING 'PLOT' IS ON (OR OFF)
C      IBAUD  = TRANSMISSION RATE IN CHARACTERS/SECOND (BAUD/10)
C      IBUF   = BUFFER ARRAY FOR MULTIPLE COMMANDS PER LINE
C      ICMD   = INTERACTIVE COMMAND WORD STARTING LOCATION
C      ICODE  = INITIALIZATION PARAMETER =1 FIRST TIME ONLY
C      IEND   = NUMBER OF COMMANDS ACCEPTED ON A LINE
C      IPOINT = POINTER IN IBUF ARRAY
CC     JUNIT  = DEVICE LOGICAL UNIT NUMBER (DISK FILE) - OUTPUT
C      K1     = NUMBER OF KEY WORDS FOUND
C      KEY    = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
C               ARRAY
CC     KUNIT  = DEVICE LOGICAL UNIT NUMBER FOR SYS$INPUT
C      LIT    = COMMAND NUMBER
CC     LUNIT  = DEVICE LOGICAL UNIT NUMBER (DISK FILE)
CC     MUNIT  = DEVICE LOGICAL UNIT NUMBER FOR SYS$OUTPUT
C      NPOL   = NUMBER OF PARTS IN PREVIOUS FILE READ
C      WORD   = ARRAY OF ACCEPTABLE COMMANDS FOR THIS ROUTINE
C
C*****************************************************************
C
C  VARIABLE DIMENSION INFORMATION FOR  MAIN PROGRAM
C    (SET THE NINE FOLLOWING MAXIMUMS IN THE DATA STATEMENT BELOW)
C    (1) ICNMAX = MAX. NO. OF ELEM.(NPTMAX)*MAX. NO. OF SIDES(NSMAX)
C    (2) NFRINM = MAXIMUM NUMBER OF FRINGES
C    (3) NJMAX  = MAXIMUM NUMBER OF NODES
C    (4) NPMAX  = MAXIMUM NUMBER OF PARTS
C    (5) NPTMAX = MAXIMUM NUMBER OF ELEMENTS(POLYGONS)
C    (6) NSMAX  = MAXIMUM NUMBER OF SIDES OF POLYGONS
C   (10) MAXFSL = MAXIMUM LENGTH OF FREE STORAGE LIST
C   (11) MLSN   = MAXIMUM NUMBER OF LIGHT SOURCES
C   (12) NTMAX  = MAXIMUM NUMBER OF TRANSPARENCY LAYERS
C
C*****************************************************************
C
      COMMON/BAUD/IBAUD
      COMMON/CMFSL/ MAXFSL
      COMMON/CMLSN/ MLSN
      COMMON/DEVI/ INPUT,OUTPUT
CC
```

```
      COMMON/DEVIL/ MUNIT,JUNIT,KUNIT,LUNIT
CC
      COMMON/ENTER/ N1,N2,KEY,XNUM,K1,K2
      COMMON/FRI/ NFRINM,JFRING,GFRIN,BFRIN,RFRIN
CC
      COMMON/HCOPY/ HRDCPY,APLOT,HCEND
C
C     IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
C
C      COMMON/INTER/ ABORT
C
      COMMON/MAXN/ NPOL,NSMAX,NJMAX,NPMAX,ICNMAX,NPTMAX
      COMMON/NTCOM/ NTMAX
      COMMON/PRO/ NP,NJ,ICMD,IC,IREAD,SKALE,NPT
CC
      COMMON/REST3/ PORG,SORG,IPREV
CC
      DIMENSION WORD(212),KEY(10),XNUM(40),IBUF(10)
      DIMENSION BFRIN(5),GFRIN(5),RFRIN(5)
CC
      DIMENSION PORG(3,27),SORG(3,27),IPREV(27)
      LOGICAL HRDCPY,APLOT,HCEND
CC
      LOGICAL JFRING
C
C     IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
C
C      LOGICAL ABORT
C
      INTEGER OUTPUT
      CHARACTER*1 WORD
      DATA WORD/'R','E','A','D','E','X','I','T','R','O','T','A',
     1'R','E','S','T','T','R','A','N','S','C','A','L','F','L',
     2'A','T','S','M','O','O','U','N','I','F','S','C','O','P',
     3'D','E','V','I','D','I','F','F','D','I','S','T','F','I',
     4'E','L','P','A','R','T','E','X','P','L','S','U','M','M',
     5'C','E','N','T','W','A','R','P','F','R','I','N','C','O',
     6'L','O','A','N','I','M','V','I','E','W','D','R','A','W',
     7'F','A','S','T','P','I','V','O','H','E','L','P','L','I',
     8'N','E','C','O','N','T','S','H','R','I','L','I','G','H',
     9'F','E','A','T','N','O','D','E','P','O','L','Y','I','M',
     A'M','U','S','H','I','F','D','A','S','H','D','O','T','T',
     B'R','E','C','O','A','L','I','A','G','L','A','S','S','H',
     C'A','D','H','A','Z','E','M','U','L','T','R','E','S','E',
     D'S','A','V','E',
     E'P','L','O','T','F','I','L','E','P','A','U','S','H','A',
     F'L','T','O','R','I','G','N','O','P','L','R','E','S','C'/
C
C      DATA MUST BE CHANGED IF MAXIMUM DIMENSIONING IS ALTERED
C
       NSMAX=8
       NJMAX=1007
       NPMAX=27
       ICNMAX=4028
       NPTMAX=1007
       NFRINM=11
       MAXFSL=2014
       MLSN=4
       NTMAX=23
C
```

```
C       INPUT, OUTPUT, AND BAUD RATE ARE SET FOR THE VAX RUNNING VMS
C
CC     Open units in case Movie is being run as a subprocess
CC
       OPEN(UNIT=5, NAME='SYS$INPUT', TYPE='OLD')
       OPEN(UNIT=2, NAME='SYS$OUTPUT', TYPE='OLD')
       OPEN(UNIT=6, NAME='SYS$ERROR', TYPE='OLD')
CC
CC     Assign i/o device units, initial plot variables
CC
       INPUT=5
       OUTPUT=2
       KUNIT = INPUT
       LUNIT = 48
       MUNIT = OUTPUT
       JUNIT = 47
       APLOT = .FALSE.
       HRDCPY= .FALSE.
CC
       IBAUD=240
       WRITE(OUTPUT,500)
       NPOL=NPMAX
       CALL CLEAR
       CALL READF
       CALL INIT
C
C      IF ON A VAX - TO IMPLEMENT CRTL-C TRAP UNCOMMENT NEXT 2 STATEMENTS
C
C       CALL TRAP_INIT
C       ABORT = .FALSE.
C
C      READ INPUT COMMAND STRING FOR PROCESSING
C
   10  WRITE(OUTPUT,510)
       IPOINT=0
       IEND=0
       ICMD=0
       N1=1
       N2=0
CC
CC     Change dimension of WORD array
CC
       NW=212
       CALL READIN(WORD,NW)
       IEND=K1
       IF(K1 EQ 0) GO TO 250
       DO 20 I=1,K1
   20  IBUF(I)=KEY(I)
C
C      IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT THE NEXT
C      FIVE STATEMENTS AND COMMENT THE 30 CONTINUE STATEMENT
C
C    30 IF (ABORT) THEN
C          IF (KEYBD) WRITE(OUTPUT,520)
C          ABORT = .FALSE
C          GO TO 10
C       ENDIF
C
   30  CONTINUE
       IPOINT=IPOINT+1
```

```
      IF(IPOINT.GT.IEND) GO TO 10
      ICMD=IBUF(IPOINT)
      LIT=(ICMD+3)/4
C
C     LIT  1 ="READ"   LIT 13 ="DIST"   LIT 25 ="FAST"   LIT 37 ="DASH"
C     LIT  2 ="EXIT"   LIT 14 ="FIEL"   LIT 26 ="PIVO"   LIT 38 ="DOTT"
C     LIT  3 ="ROTA"   LIT 15 ="PART"   LIT 27 ="HELP"   LIT 39 ="RECO"
C     LIT  4 ="REST"   LIT 16 ="EXPL"   LIT 28 ="LINE"   LIT 40 ="ALIA"
C     LIT  5 ="TRAN"   LIT 17 ="SUMM"   LIT 29 ="CONT"   LIT 41 ="GLAS"
C     LIT  6 ="SCAL"   LIT 18 ="CENT"   LIT 30 ="SHRI"   LIT 42 ="SHAD"
C     LIT  7 ="FLAT"   LIT 19 ="WARP"   LIT 31 ="LIGH"   LIT 43 ="HAZE"
C     LIT  8 ="SMOO"   LIT 20 ="FRIN"   LIT 32 ="FEAT"   LIT 44 ="MULT"
C     LIT  9 ="UNIF"   LIT 21 ="COLO"   LIT 33 ="NODE"   LIT 45 ="RESE"
C     LIT 10 ="SCOP"   LIT 22 ="ANIM"   LIT 34 ="POLY"   LIT 46 ="SAVE"
C     LIT 11 ="DEVI"   LIT 23 ="VIEW"   LIT 35 ="IMMU"
C     LIT 12 ="DIFF"   LIT 24 ="DRAW"   LIT 36 ="SHIF"
CC
CC    LIT 47 ="PLOT"   LIT 48 ="FILE"   LIT 49 ="PAUS"
CC    LIT 50 ="HALT"   LIT 51 ="ORIG"   LIT 52 ="NOPL"
CC    LIT 53 ="RESC"
C
      GO TO (40,50,60,70,80,90,100,100,100,110,110,120,130
     1,140,150,160,170,170,180,190,200,210,220,220,230,240
     2,250,260,270,280,290,300,310,310,320,330,340,350,360
     3,370,380,390,400,410,420,430
     4,435,440,445,450,455,460,465), LIT
   40 CALL READF
      GO TO 30
   50 STOP
   60 CALL ROTA
      GO TO 30
   70 CALL REST
      GO TO 30
   80 CALL TRAN
      GO TO 30
   90 CALL SCAL
      GO TO 30
  100 CALL SHADE
      GO TO 30
  110 CALL SCOP
      GO TO 30
  120 CALL DIFF
      GO TO 30
  130 CALL DIST
      GO TO 30
  140 CALL FIEL
      GO TO 30
  150 ICODE=0
      CALL SEPART(ICODE)
      GO TO 30
  160 CALL EXPL
      GO TO 30
  170 IPERZ=0
      CALL SUMCEN(IPERZ)
      GO TO 30
  180 CALL WARP
      GO TO 30
  190 CALL FRIN
      GO TO 30
  200 CALL COLO
```

```
      GO TO 30
  210 CALL ANIMAT
      GO TO 30
  220 CALL VIEDRA
      GO TO 30
  230 CALL FAST
      GO TO 30
  240 CALL PIVOT
      GO TO 30
  250 CALL HELP
      GO TO 30
  260 CALL LINE
      GO TO 30
  270 CALL CONT
      GO TO 30
  280 CALL SHRKR
      GO TO 30
  290 CALL LIGHT
      GO TO 30
  300 CALL FEAT
      GO TO 30
  310 CALL POLNOD
      GO TO 30
  320 CALL IMMUNE
      GO TO 30
  330 CALL TRAN
      GO TO 30
  340 CALL DSET
      GO TO 30
  350 CALL DASHLN
      GO TO 30
  360 CALL RECORD
      GO TO 30
  370 CALL ALIA
      GO TO 30
  380 CALL GLAS
      GOTO 30
  390 CALL SHADO
      GOTO 30
  400 CALL HAZE
      GOTO 30
  410 CALL MULT
      GOTO 30
  420 CALL RESETQ
      GOTO 30
  430 CALL SAVE
      GO TO 30
CC
CC    Additional options added
CC
  435 CALL PLOT_OP
      GO TO 30
  440 CALL FILE_OP
      GO TO 30
  445 CALL PAUS_OP
      GO TO 30
  450 CALL HALT_OP
      GO TO 30
  455 CALL GET_ORG
      GO TO 30
```

```
  460 CALL NOPLOT
      GO TO 30
  465 CALL RES_CEN
      GO TO 30
C
  500 FORMAT(' <MOVIE SYSTEM DISPLAY>')
C
C     SEE POSSIBLE INCOMPATIBILITIES (2)
C
  510 FORMAT(' >> ')
C
C     IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
C
  520 FORMAT(' <OPERATION ABORTED>')
C
      END
```

```
      SUBROUTINE MULTDD(II)
C
C*****************************************************************************
C
C  SUBROUTINE MULTDD - MULTIPLIES COORDINATES BY LOCAL ROTATION
C     TRANSFORMATION MATRIX
C
C*****************************************************************************
C
C  SUBROUTINE CALLED BY
C     POINTS = GETS COORDINATES OF NODES FOR A POLYGON
C
C*****************************************************************************
C
C  VARIABLES USED
C     DD       = TRANSFORMATION MATRIX
C     IPART    = PART NUMBER
C     PORG     = ARRAY OF PIVOTAL POINTS CURRENT POSITION
C     RORG     = RELATIVE ORIGIN BY PART
C     XP       = COORDINATE ARRAY FOR POLYGON
C     YY       = INCREMENTAL TRANSLATION ARRAY
C
C*****************************************************************************
C
C  VARIABLE DIMENSION INFORMATION FOR SUBROUTINE MULTDD
C    (4) NPMAX   = MAXIMUM NUMBER OF PARTS
C                  DIMENSION DD(3,3,NPMAX),RORG(3,NPMAX)
C                                         ,PORG(3,NPMAX)
C    (6) NSMAX   = MAXIMUM NUMBER OF SIDES OF POLYGONS
C                  DIMENSION XP(3,(NSMAX+1))
C
C*****************************************************************************
C
      COMMON/COOR/ XP
      COMMON/FYSL/FUN,YY,SLINR
      COMMON/NEME/ NEDGE,MEDGE,IPART
      COMMON/REST1/ RORG,DD
CC
CC    Addition for change in "PIVOT" option
CC
      COMMON/REST3/ PORG,SORG,IPREV
      DIMENSION PORG(3,27),SORG(3,27),IPREV(27)
CC
      DIMENSION DD(3,3,27),RORG(3,27)
      DIMENSION XP(3,9)
      DIMENSION YY(3),FUN(3)
C
      X1=XP(1,II)-RORG(1,IPART)+YY(1)
      X2=XP(2,II)-RORG(2,IPART)+YY(2)
      X3=XP(3,II)-RORG(3,IPART)+YY(3)
      XP(1,II)=DD(1,1,IPART)*X1+DD(2,1,IPART)*X2+DD(3,1,IPART)*X3
     1         + PORG(1,IPART)-YY(1)
      XP(2,II)=DD(1,2,IPART)*X1+DD(2,2,IPART)*X2+DD(3,2,IPART)*X3
     1         + PORG(2,IPART) -YY(2)
      XP(3,II)=DD(1,3,IPART)*X1+DD(2,3,IPART)*X2+DD(3,3,IPART)*X3
     1         + PORG(3,IPART)-YY(3)
      RETURN
      END
```

```
      SUBROUTINE PIVOT
C
C*****************************************************************
C
C  SUBROUTINE PIVOT - SET LOCAL ROTATION ABOUT RELATIVE ORIGIN
C
C*****************************************************************
C
C  SUBROUTINE CALLED BY
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER
C               AND CALLS APPROPRIATE SUBROUTINES
C
C*****************************************************************
C
C  SUBPROGRAMS CALLED
C     READIN = ACCEPTS INPUT FROM THE USER
C     ROTAT  = CALCULATES GLOBAL ROTATION TRANSFORMATION MATRIX
C
C*****************************************************************
C
C  VARIABLES USED
C     DD      = LOCAL TRANSFORMATION MATRICES BY PART
C     IA      = AXIS NUMBER
C     KEY     = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
C               ARRAY
C     N1      = INPUT FLAG, =1 WHEN KEY WORDS SOUGHT
C                           =0 WHEN KEY WORDS NOT SOUGHT
C     N2      = INPUT FLAG, =1 WHEN NUMBERS WANTED
C                           =0 WHEN NUMBERS NOT WANTED
C     NW      = DIMENSION OF WORD ARRAY
C     RORG    = RELATIVE ORIGIN ARRAY FOR LOCAL ROTATIONS
C     WORD    = ARRAY OF ACCEPTABLE KEY WORDS
C     XNUM    = ARRAY OF REAL NUMBERS ACCEPTED
C
C*****************************************************************
C
C  VARIABLE DIMENSION INFORMATION FOR SUBROUTINE PIVOT
C    (4) NPMAX  = MAXIMUM NUMBER OF PARTS
C                 DIMENSION DD(3,3,NPMAX),RORG(3,NPMAX)
C
C*****************************************************************
C
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/ENTER/N1,N2,KEY,XNUM,K1,K2
      COMMON/REST1/ RORG,DD
      DIMENSION KEY(10),XNUM(40),WORD(12)
      DIMENSION DD(3,3,27),RORG(3,27)
      INTEGER OUTPUT
      CHARACTER*1 WORD
      DATA WORD/'X','#','#','#','Y','#','#','#',
     1'Z','#','#','#'/
C
      NW=12
      WRITE(OUTPUT,60)
   10 WRITE(OUTPUT,80)
      N1=1
      N2=1
      CALL READIN(WORD,NW)
      I1=XNUM(1)
      I2=XNUM(2)
```

```
      X2=XNUM(3)
      IA=KEY(1)
C
C     IA=1    ="X"
C     IA=2    ="Y"
C     IA=3    ="Z"
C
      I3=(IA+3)/4
      IF (I1.EQ.0) RETURN
      DO 20 I=I1,I2
         ISAFE=I
   20 CALL ROTAT(DD,I3,X2,ISAFE)
      GO TO 10
C
   60 FORMAT(' <PARTS I1/I2, AXIS, ANGLE>')
C
C     SEE POSSIBLE INCOMPATIBILITIES (2)
C
   80 FORMAT(' >>> ')
      END
```

```
      SUBROUTINE READF
C
C*****************************************************************************
C
C  SUBROUTINE READF - READS IN DATA FILES
C
C*****************************************************************************
C
C  SUBROUTINE CALLED BY
C     MAIN   = PROCESSES INTERACTIVE COMMANDS FROM THE USER
C              AND CALLS APPROPRIATE SUBROUTINES
C
C*****************************************************************************
C
C  SUBPROGRAMS CALLED
C     CLSFIL = CLOSES FILE IUNIT
C     LASTC  = FLAGS CONNECTIVITY ARRAY FOR PART LIMITS AND
C              DETERMINES IF POLYGON CENTER POINTS SPECIFIED
C     OPNFIL = REQUESTS FILENAME AND OPENS FILE FOR I/O
C     SEPART = SELECTS CONTENT OF SCENE
C
C*****************************************************************************
C
C  VARIABLES USED
C     BFRIN  = BLUE FRINGE ARRAY
C     CFRIN  = RED,BLUE,GREEN FRINGE INTENSITY BY FRINGE NUMBER
C     ICNMAX = MAXIMUM NUMBER OF ELEMENTS TIMES MAXIMUM NUMBER OF
C              SIDES OF POLYGONS (NPTMAX * NSMAX), OR SUITABLE
C              REDUCTION IF MOSTLY LOWER ORDER ELEMENTS USED
C     DIF    = DIFFUSED LIGHT ARRAY BY PARTS
C     GFRIN  = GREEN FRINGE ARRAY
C     IBAD   = ERROR PARAMETER
C     ICODE  = INITIALIZATION PARAMETER, 1  FIRST TIME ONLY
C     ICOL   = RED,BLUE,GREEN INTENSITY BY PARTS
C     IERROR = 1  ON SUCESSFUL COMPLETION
C            = 0  ON EMPTY FILE SPECIFICATION
C            = -1 ON FAILURE
C     IP     = TOTAL CONNECTIVITY ARRAY
C     IPB    = BACKGROUND COLOR
C     IREAD  = 1  FOR INPUT FILE
C            = -1 FOR OUTPUT FILE
C     ISPEC  = IF .TRUE., SCALAR FUNCTION FILE INCLUDED
C     IUNIT  = DEVICE LOGICAL NUMBER
C     NCON   = LENGTH OF IP ARRAY
C     NJ     = NUMBER OF JOINTS OR NODES
C     NJMAX  = MAXIMUM NUMBER OF NODES
C     NP     = NUMBER OF PARTS
C     NPL    = PARTS ARRAY
C     NPMAX  = MAXIMUM NUMBER OF PARTS
C     NPOL   = NUMBER OF PARTS IN PREVIOUS FILE READ
C     NPT    = NUMBER OF ELEMENTS (POLYGONS)
C     NPTMAX = MAXIMUM NUMBER OF ELEMENTS (POLYGONS)
C     NTR    = IF .TRUE., INTERIOR POLYGON
C     RFRIN  = RED FRINGE ARRAY
C     SPEC   = SCALAR FUNCTION ARRAY
C     U      = DISPLACEMENT ARRAY
C     X      = TOTAL COORDINATE ARRAY
C
C*****************************************************************************
C
```

```
C  VARIABLE DIMENSION INFORMATION FOR SUBROUTINE READF
C     (1) ICNMAX = MAX.  NO.  OF ELEM.(NPTMAX)*MAX.  NO.  OF SIDES(NSMAX)
C                  DIMENSION IP(ICNMAX)
C     (2) NFRINM = MAXIMUM NUMBER OF FRINGES
C                  DIMENSION CFRIN(3,NFRINM)
C     (3) NJMAX  = MAXIMUM NUMBER OF NODES
C                  DIMENSION SPEC(NJMAX),SPEC1(NJMAX),U(3,NJMAX)
C                 1,X(3,NJMAX)
C     (4) NPMAX  = MAXIMUM NUMBER OF PARTS
C                  DIMENSION DIF(NPMAX),ICOL(NPMAX),NPL(2,NPMAX)
C                 1,NPLS(NPMAX)
C
C*******************************************************************************
C
      COMMON/CIP/ IP
      COMMON/CLIP3/XB,YB,ZB,KB,CB,XE,YE,ZE,KE,CE,LAS,ISHARE,NTR
     1,ITR1,ITR2
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/FRI/ NFRINM,JFRING,GFRIN,BFRIN,RFRIN
      COMMON/INTENS/ IPH,IPL,IPB,IFX
      COMMON/LEF/ DIF,ICOL
      COMMON/LOGI/ ISMOTH,IPOOR,IMIX,DIRC,ISPEC,ISMA,LINEAR,
     1 IHLR,IFRING
      COMMON/MAXN/ NPOL,NSMAX,NJMAX,NPMAX,ICNMAX,NPTMAX
      COMMON/PAAR/ NPL,NPLS
      COMMON/PRO/NP,NJ,ICMD,IC,IREAD,SKALE,NPT
      COMMON/QFORIO/CONTRS,IDVICE,IBAD,SHOSHR,LBLSPC
      COMMON/SPUX/ SPEC,SPEC1,U,X
      COMMON/VCOL/ NFRING,CFRIN
CC
CC    Addition for change in "PIVOT" option
CC
      COMMON/REST3/ PORG,SORG,IPREV
      DIMENSION PORG(3,27),SORG(3,27),IPREV(27)
CC
      DIMENSION IP(4028)
      DIMENSION CFRIN(3,11)
      DIMENSION SPEC(1007),SPEC1(1007),U(3,1007),X(3,1007)
      DIMENSION DIF(27),ICOL(27),NPL(2,27),NPLS(27)
      DIMENSION BFRIN(5),GFRIN(5),RFRIN(5)
      LOGICAL ISPEC,IBAD,NTR,ISHARE,LAS,JFRING,DIRC,IFRING,IHLR
      LOGICAL IMIX,IPOOR,ISMA,LINEAR,CONTRS,SHOSHR
      INTEGER OUTPUT
      IREAD = 1
      BFRIN(1)=1
      BFRIN(2)=1.
      BFRIN(3)=0.
      BFRIN(4)=0.
      BFRIN(5)=0.
      GFRIN(1)=0.
      GFRIN(2)=1.
      GFRIN(3)=1.
      GFRIN(4)=1.
      GFRIN(5)=0
      RFRIN(1)=0.
      RFRIN(2)=0.
      RFRIN(3)=0
      RFRIN(4)=1.
      RFRIN(5)=1.
C
```

```
C      READ GEOMETRY FILE
C
   10 CALL OPNFIL('GEOM',IUNIT,IREAD,IERROR)
      IF(IERROR) 10,60,20
   20 READ(IUNIT,200) NP,NJ,NPT,NCON,NTEST
      IF(NP.LE.0.OR.NTEST.NE.0) GO TO 30
      IF(NP.GT.NPOL.AND.NP.LE.NPMAX) WRITE(OUTPUT,260)
      NPOL=NP
      IF(NCON.NE.0) GO TO 40
   30 WRITE(OUTPUT,280)
      STOP
   40 IF(NP.GT.NPMAX) WRITE(OUTPUT,220) NP,NPMAX
      IF(NJ.GT.NJMAX) WRITE(OUTPUT,230) NJ,NJMAX
      IF(NPT.GT.NPTMAX) WRITE(OUTPUT,240) NPT,NPTMAX
      IF(NCON.GT.ICNMAX) WRITE(OUTPUT,270) NCON,ICNMAX
      IF(NP.GT.NPMAX.OR.NJ.GT.NJMAX.OR.NPT.GT.NPTMAX) STOP
      IF(NCON.GT.ICNMAX) STOP
      READ(IUNIT,200) ((NPL(I,J),I=1,2),J=1,NP)
      READ(IUNIT,210) ((X(I,J),I=1,3),J=1,NJ)
      READ(IUNIT,200) (IP(I),I=1,NCON)
      WRITE(OUTPUT,250) NP,NJ,NPT
C
C      READ DISPLACEMENT FILE
C
   50 CALL CLSFIL(IUNIT)
   60 IF(NP.EQ.0) GO TO 10
      CALL OPNFIL('DISP',IUNIT,IREAD,IERROR)
      IF(IERROR) 60,80,70
   70 READ(IUNIT,210) ((U(I,J),I=1,3),J=1,NJ)
C
C      READ SPECIAL FUNCTION FILE
C
      CALL CLSFIL(IUNIT)
   80 CONTINUE
   90 CALL OPNFIL('FUNC',IUNIT,IREAD,IERROR)
      ISPEC=.FALSE.
      IF(IERROR) 90,110,100
  100 ISPEC=.TRUE.
      READ(IUNIT,210) (SPEC(I),I=1,NJ)
      CALL CLSFIL(IUNIT)
  110 CONTINUE
      CALL LASTC
      IBAD=.FALSE.
      NTR=.FALSE.
      DO 120 I=1,NP
         DIF(I)=0.15
  120 ICOL(I)=16777215
      IPB=0
      DO 130 I=1,5
         CFRIN(1,I)=CFRIN(1)
         CFRIN(2,I)=BFRIN(I)
  130 CFRIN(3,1)=RFRIN(I)
      ICODE=1
      CALL SEPART(ICODE)
CC
CC    READ RELATIVE ORIGIN FILE for change in "PIVOT" option
CC
      DO 140 J = 1,NP
         IPREV(J) = 0
         DO 140 I = 1,3
```

```
_DUAU: [USER1.PTS.NEW]SHOVHOBJ.FOR;2

            SORG(I,J) = 0.0
  140 CONTINUE
  150 CALL OPNFIL('ORIG',IUNIT,IREAD,IERROR)
      IF (IERROR) 150,180,160
  160 READ(IUNIT,*,END=170) I1,I2,I3,XC,YC,ZC
      DO 165 J = I1,I2
         IPREV(J)  = I3
         SORG(1,J) = XC
         SORG(2,J) = YC
         SORG(3,J) = ZC
  165 CONTINUE
      GO TO 160
  170 CALL CLSFIL(IUNIT)
  180 CONTINUE
      RETURN
C
  200 FORMAT(16I5)
  210 FORMAT(6E12.5)
  220 FORMAT(' <NP .GT.NPMAX : ',2I5,'>')
  230 FORMAT(' <NJ .GT.NJMAX : ',2I5,'>')
  240 FORMAT(' <NPT.GT.NPTMAX: ',2I5,'>')
  250 FORMAT(' <READ: ',I5,' PARTS; ',I5,' COORDINATES; ',
     1I5,' ELEMENTS.>')
  260 FORMAT(' <WARNING: NEW FILE HAS MORE PARTS THAN OLD FILE>',
     1' <ISSUE "RESTORE" COMMAND> ')
  270 FORMAT(' <CONN. LENGTH(',I7,') EXCEEDS MAX. OF(',I7,')>')
  280 FORMAT(' <ERROR: DATA FORMAT PROBLEMS>')
      END
```

```
_DUA0:[USER1.FTS.MOVT]MOVMODS.FOR;7                                    14-JUL

      SUBROUTINE VIEDRA
C
C*************************************************************************
C
C  SUBROUTINE VIEDRA - CALLS FOR NORMALS, LIGHT INTENSITY, ETC.
C                      NEEDED TO DISPLAY SCENE (EITHER VIEW OR DRAW)
C                      ALSO INCREMENTS DISPLACEMENTS, ROTATIONS,
C                      TRANSLATIONS, ETC  FOR ANIMATION
C
C*************************************************************************
C
C  SUBROUTINE CALLED BY
C     MAIN   = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND
C              CALLS APPROPRIATE SUBROUTINES
C
C*************************************************************************
C
C  SUBPROGRAMS CALLED
C     AALIAS = BUFFERS SEGMENT INFORMATION FOR ANTI-ALIASING
C     BGNFRM = PERFORMS FRAME INITIALIZATION PROCEDURES
C     COARRO = DRAWS GLOBAL COORDINATE AXES
C     ENDFRM = ENDS FRAME, RETURNS TO ALPHA-NUMERIC MODE
C     FRNBAR = DRAWS FRINGE BAR WHEN BAR IS .TRUE.
C     HIDDEN = DETERMINES VISIBLE SEGMENTS
C     INTHID = INITIALIZES THE HIDDEN PROCESS
C     MOD_ORG= COMPUTE CURRENT POSITION OF PIVOTAL POINTS
C     NODNUM = SENDS VISIBLE NODE COORDINATES ON
C     OPNFIL = REQUESTS FILENAME AND OPENS FILE FOR I/O
C     PART   = PROCESSES POLYGONS BY PART, SUBDIVIDES WARPED
C              POLYGONS ON EDGE
C     POLNUM = SENDS CENTER COORDINATES OF VISIBLE POLYGONS ON
C     READIN = ACCEPTS INPUT FROM THE USER
C     ROTAT  = CALCULATES ROTATION TRANSFORMATION MATRIX
C     SHINIT = INITIALIZES PARAMETERS FOR SHADOWS
C
C*************************************************************************
C
C  VARIABLES USED
C     ANISHR = DELTA SHRINK FACTOR FOR ANIMATE OPTION
C     ANTIAL = IF .TRUE. ANTI-ALIASING ENABLED
C     BACKZ  = GENERAL BACK Z PERSPECTIVE PLANE
C     BAR    = IF .TRUE. THEN FRINGE BAR IS DRAWN
C     CHI    = HIGHEST SCALAR FUNCTION VALUE
C     CLO    = LOWEST SCALAR FUNCTION VALUE
C     ICMD   = INTERACTIVE COMMAND WORD
C     CONTRS = IF .TRUE., CONTOURS EXIST
C     CPF    = VIBRATIONS/FRAME
C     DA     = LOCAL ROTATIONS ARRAY BY PARTS
C     DAMP   = DAMPING FACTOR FOR SMOOTH ANIMATION
C     DC     = GLOBAL TRANSFORMATION MATRIX
C     DD     = LOCAL TRANSFORMATION MATRICES BY PART
C     DDELTA = POSITION SCALE FACTOR IN ANIMATED SEQUENCE
C     DDOZ   = CHANGE IN DISTANCE TO ORIGIN IN ANIMATED SEQUENCE
C     DELTA  = LOCAL MOTION SCALE FACTOR
C     DIRC   = .TRUE. FOR CLOCKWISE ORIENTATION OF POLYGONS
C            = .FALSE. FOR COUNTERCLOCKWISE ORIENTATION OF POLYGONS
C     DIRCA  = DATA DIRECTION BY PARTS ARRAY
C            = 0.0  FOR CLOCKWISE
C            = 1.0  FOR COUNTERCLOCKWISE
C     DOZ    = DISTANCE TO ORIGIN FROM OBSERVER
```

```
C      DR      = TOTAL ROTATION IN ANIMATED SEQUENCE
C      DT      = TOTAL TRANSLATION IN ANIMATED SEQUENCE
C      FRONTZ  = GENERAL FRONT Z PERSPECTIVE PLANE
C      IBAD    = ERROR PARAMETER
C      IC      = 1  FOR COLOR, 2  FOR BLACK AND WHITE
C      IDVICE  = DISPLAY DEVICE NUMBER
C      IFACT   = FACTOR USED TO PACK DATA
C      IFR1    = FIRST SCENE IN SEQUENCE SENT TO DISPLAY
C      IFR2    = LAST SCENE IN SEQUENCE SENT TO DISPLAY
C      IFRING  = .TRUE. FOR FRINGES, .FALSE. FOR NO FRINGES
C      IHLR    = HIDDEN LINE REMOVAL (LOGICAL VARIABLE)
C      IPART   = PART NUMBER
C      IPERZ   = FLAG, USED FOR Z PERSPECTIVE PURPOSES
C      IPM     =  TRUE. DISPLAYS ALL PICTURES IN SEQUENCE
C              = .FALSE. MODIFIES GEOMETRY BUT DOES NOT DISPLAY
C      IPOOR   = IF .TRUE. INVOKES POOR MANS HIDDEN LINE/SURFACE
C                REMOVAL
C      ISCLIP  = 0 FOR NON-SHADOW POLYGONS, = 1 FOR SHADOW POLYGONS
C      ISET    = 0 INITIALLY FOR A FRAME, THEREAFTER = 1
C      ISHARE  = EDGE SHARING
C      ISMA    = IF .TRUE., SMOOTH ANIMATION
C      ISMOTH  = -1  FOR SMOOTH SHADING
C              =  0  FOR FLAT SHADING
C              =  1  FOR UNIFORM SHADING
C      ITSHDO  = FLAG USED TO TURN OFF SHADOWS TEMPORARY IF IN LINE
C                DRAWING MODE
C      JCONT   = 0 IF LIGHT HAS NOT BEEN SPECIFIED, = 1 OTHERWISE
C      JLAST   = LOCATION OF LAST NODE OF PREVIOUS ELEMENT IN TOTAL
C                CONNECTIVITY ARRAY
C      JSMOTH  = SMOOTH BY PARTS ARRAY
C              = -1  FOR SMOOTH
C              =  0  FOR FLAT
C              =  1  FOR UNIFORM
C      K1      = NUMBER OF KEY WORDS FOUND
C      K2      = NUMBER OF NUMBERS FOUND
C      KEY     = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
C                ARRAY
C      LINEAR  = TRANSIENT DATA (LOGICAL VARIABLE)
C      N1      = INPUT FLAG, =1 WHEN KEY WORDS SOUGHT
C                            =0 WHEN KEY WORDS NOT SOUGHT
C      N2      = INPUT FLAG, =1 WHEN NUMBERS WANTED
C                            =0 WHEN NUMBERS NOT WANTED
C      NFRAME  = NUMBER OF FRAMES TO BE GENERATED
C      NJ      = NUMBER OF JOINTS OR NODES
C      NNUM    =  TRUE. IF NODE NUMBERING ENABLED
C      NONROT  =  TRUE. PART IS IMMUNE TO GLOBAL ROTATIONS
C                 FALSE. PART ROTATES
C      NP      = NUMBER OF PARTS
C      NPL     = PARTS ARRAY
C      NPLS    = DISPLAY PARTS ARRAY
C              = 1  TO DISPLAY
C              = 0  DO NOT DISPLAY
C      NTR     = IF  TRUE., INTERIOR POLYGON
C      NUM     = .TRUE. IF EITHER NODE OR POLYGON NUMBERING ENABLED
C      NW      = DIMENSION OF WORD ARRAY
C      PNUM    = .TRUE. IF POLYGON NUMBERING ENABLED
C      POOR    = POOR MANS BY PART ARRAY
C              = 0.0  FOR NO POOR MANS
C              = 1.0  FOR POOR MANS
C      SFDEL   = DISPLACEMENT SCALE FACTOR IN ANIMATED SEQUENCE
```

```
C      SHOSHR = EDGE SHARE PARAMETER (LOGICAL VARIABLE)
C      SHRK   = SHRINK FACTOR
C      SKALE  = DISPLACEMENT SCALE FACTOR
C      SLINR  = TRANSIENT DATA INCREMENT FACTOR
C      SHDO   = IF .TRUE. SHADOWS ENABLED
C      SPEC   = SCALAR FUNCTION ARRAY
C      SPECI  = SECOND SCALAR FUNCTION ARRAY IN LINEAR OPTION
C      WORD   = ARRAY OF ACCEPTABLE KEY WORDS
C      WR     = .TRUE. IF RECORD IS ON
C      XNORM  = NORMALS ARRAY FOR SMOOTH SHADING WITH CONTINUOUS
C               TONE OUTPUT, OR SCREEN COORDINATES AND VISIBLE FLAG
C               FOR NODE OR POLYGON NUMBERING IN LINE DRAWING MODE
C      XNUM   = ARRAY OF REAL NUMBERS ACCEPTED
C      XO     = TRANSLATION ARRAY
C      ZEB    = EYE BACK Z PERSPECTIVE PLANE
C      ZEF    = EYE FRONT Z PERSPECTIVE PLANE
C      ZPB    = ZEB TIMES TANAL
C      ZPF    = ZEF TIMES TANAL
C      ZSPRED = Z PERSPECTIVE MAPPING FACTOR
C
C*****************************************************************
C
C  VARIABLE DIMENSION INFORMATION FOR SUBROUTINE VIEDRA
C    (3) NJMAX  = MAXIMUM NUMBER OF NODES
C                 DIMENSION SPEC(NJMAX),SPECI(NJMAX),U(3,NJMAX)
C                1,X(3,NJMAX),XNORM(3,NJMAX)
C    (4) NPMAX  = MAXIMUM NUMBER OF PART
C   (11) MLSN   = MAXIMUM NUMBER OF LIGHT SOURCES
C                 DIMENSION DA(3,NPMAX),DC(3,3,NPMAX+1),DD(3,3,NPMAX)
C                1,DIRCA(NPMAX),JSMOTH(NPMAX),NPL(2,NPMAX)
C                2,NPLS(NPMAX),POOR(NPMAX),RORG(3,NPMAX),NONROT(NPMAX)
C                 DIMENSION NHIGH(NPMAX,MLSN),XIO(NPMAX,MLSN),XNH(NPMAX,MLSN)
C                1,XNR(NPMAX,MLSN),XLSI(MLSN),XL(3,MLSN),INFIN(MLSN)
C
C*****************************************************************
C
      COMMON/ANI/ DA,DT,DR,CPF,IFR1,IFR2,DDOZ,DDELTA,IPM,NFRAME,
     1  SFDEL,ANISHR
      COMMON/CANTI/ANTIAL
      COMMON/CFIRST/IFIRST
      COMMON/CLIP3/ XB,YB,ZB,KB,CB,XE,YE,ZE,KE,CE,LAS,ISHARE,NTR,
     1ITR1,ITR2
      COMMON/COLBAR/ BAR,BEGRNG,ENDRNG,BAROUT
      COMMON/CONLEV/ CHI,CLO,NCONLV,CLEVEL
      COMMON/CSHDO/ SHDO
      COMMON/CUT/ WR,FACT,IUNIT
      COMMON/DELSH/ DELTA,SHRK
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/FFF/ IYMIN,NOSHDO
      COMMON/ENTER/N1,N2,KEY,XNUM,K1,K2
      COMMON/FYSL/ FUN,YY,SLINR
      COMMON/INTENS/ IPH,IPL,IPB,IFX
C
C     IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
C
C      COMMON/INTER/ ABORT
C
      COMMON/LABEL/NUM,NNUM,PNUM
      COMMON/LAIF/ JLAST,IFLAG
      COMMON/LOGI/ISMOTH,IPOUR,IMIX,DIRC,ISPEC,ISMA,LINEAR,
```

```
      1   IHLR, IFRING
       COMMON/MOSAIC/ MOSAIC
       COMMON/NEME/ NEDGE, MEDGE, IPART
       COMMON/PAAR/ NPL, NPLS
       COMMON/PARA/ XNR, XNH, XIO, INFIN, NHIGH, XL, JCONT, XLSI
       COMMON/PERZ/ FRONTZ, BACKZ, ISET, ZEF
       COMMON/PGNCNT/IPOLY
       COMMON/POORPT/ POOR, DIRCA
       COMMON/PRO/ NP, NJ, ICMD, IC, IREAD, SKALE, NPT
       COMMON/QFORIO/ CONTRS, IDVICE, IBAD, SHOSHR, LBLSPC
       COMMON/RESTO/ XO, DC
       COMMON/REST1/ RORG, DD
       COMMON/ROTIM/ NONROT
       COMMON/SAVCON/ CONSAV
       COMMON/SCLIP/ ISCLIP, ZPF, ZPB
       COMMON/SCMR/ DOZ, FIOLD, FIELD, TANAL, RES
       COMMON/SMOTH/ JSMOTH
       COMMON/SPUX/ SPEC, SPEC1, U, X
       COMMON/XNO/ XNORM
       COMMON/ZFIXER/ ZLO, ZHI, ZSPRED, CURZEE
       DIMENSION S1(3)
       DIMENSION KEY(10), XNUM(40), WORD(4)
       DIMENSION FUN(3), YY(3)
       DIMENSION XNORM(3, 1007)
       DIMENSION DA(3, 27), DC(3, 3, 28), DD(3, 3, 27), DIRCA(27)
      1, JSMOTH(27), NPL(2, 27), NPLS(27), POOR(27), RORG(3, 27), NONROT(27)
       DIMENSION XO(3), DT(3), DR(3)
       DIMENSION NHIGH(27, 4), XIO(27, 4), XNH(27, 4), XNR(27, 4), XLSI(4),
      1XL(3, 4), INFIN(4)
       DIMENSION CLEVEL(26), LIT(26)
       DIMENSION SPEC(1007), SPEC1(1007), U(3, 1007), X(3, 1007)
       LOGICAL LINEAR, ISMA, IHLR, IBAD, IFRING, DIRC, IPOOR
       LOGICAL IPM, ISHARE, CONTRS, MOSAIC, WR, INFIN, NHIGH
       LOGICAL SHOSHR, NTR, NNUM, PNUM, NUM, NONROT, LAS, IMIX, ISPEC
       LOGICAL SHDO, ANTIAL, IFIRST, ITSHDO, CURZEE, CONSAV
       LOGICAL BAR, BAROUT
C
C      IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
C
C       LOGICAL ABORT
C
       INTEGER OUTPUT
       CHARACTER*1 WORD, LIT
       DATA WORD/'A', '#', '#', '#'/
       DATA LIT/'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
      1'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'/
       DATA S1/0., 0., 0./, A1/0./, IA1/-1/
C
       IFIRST=.TRUE.
       IHLR=.FALSE.
C
C      DO NOT ALLOW CONTOURS TO BE ON WHILE IN CONTINUOUS TONE MODE
C
       IF (IDVICE.GE.0.AND.CONTRS) THEN
          CONSAV = .TRUE.
          CONTRS = .FALSE.
       ELSEIF (IDVICE.LT.0.AND.CONSAV) THEN
          CONTRS = .TRUE
          CONSAV = .FALSE
       ENDIF
```

```
C
C       TURN OFF SHADOWS TEMP.  IF IN LINE DRAWING MODE.
C
        ITSHDO=.FALSE.
        IF((IDVICE.LT.0).AND.SHDO) THEN
          ITSHDO=.TRUE.
          SHDO=.FALSE.
        ENDIF
C
C       ICMD 93 = "DRAW"
C
        IF(IDVICE.EQ.1.AND.ICMD.EQ.93) ICMD=89
C
C       ICMD 89 = "VIEW"
C
        IF(ICMD.EQ.89) IHLR=.TRUE.
C
C       CALL LIGHT IF IT HAS NOT BEEN CALLED ALREADY
C
        IF((JCONT.EQ.0).AND.(IDVICE.GT.0).AND.IHLR) CALL LIGHT
C
        ISHARE=((IDVICE.LT.0).OR.(.NOT.IHLR))
        IF(.NOT.CONTRS) GO TO 30
        CHI=SPEC(1)
        CLO=SPEC(1)
        DO 10 I=2,NJ
           X1=SPEC(I)
           IF(X1.LT.CLO) CLO=X1
        IF(X1.GT.CHI) CHI=X1
   10   CONTINUE
        IF(.NOT.LINEAR) GO TO 30
        DO 20 I=1,NJ
           X1=SPEC(I)+SPEC1(I)
           IF(X1.LT.CLO) CLO=X1
        IF(X1.GT.CHI) CHI=X1
   20   CONTINUE
   30   SLINR=0.0
        XMAGN=SKALE
        AMPZ=1.0
        MFRAME=NFRAME
        IF(NFRAME.EQ.0) MFRAME=1
        XFRAME=FLOAT(MFRAME)
        DO 205 IIMOVE=1,MFRAME
           IF(NFRAME.EQ.0) GO TO 100
C
C       INCREMENT DISPLACEMENTS, ROTATIONS, TRANSLATIONS, ETC. FOR ANIMATE
C
           XIMOVE=FLOAT(IIMOVE)
           XMAGN=XMAGN+SPDEL
           SKALE=XMAGN
           IF(LINEAR) SLINR=XIMOVE/XFRAME
           IF(LINEAR) SKALE=XMAGN*SLINR
           IF(CPF.EQ.0.0) GO TO 60
           ANG=360.0*CPF*XIMOVE
           SKALE=XMAGN*SIN(ANG*.017453)
   60      AMP=180.0*XIMOVE/XFRAME
           AMP=COS(AMP*.017453)
           DAMP=0.5*(AMPZ-AMP)
           IF(.NOT.ISMA) DAMP=1.0/XFRAME
           AMPZ=AMP
```

```
          DOZ=DOZ+DDOZ*DAMP
          SHRK=SHRK+ANISHR*DAMP
          DELTA=DELTA+DDELTA*DAMP
          DO 90 I=1,3
             ISAFE=I
             XO(I)=XO(I)+DT(I)*DAMP
             DO 80 J=1,NP
                JSAFE=J
                IF(NONROT(JSAFE)) GO TO 70
                DDD=DR(I)*DAMP
                IF(DDD.NE.0.0) CALL ROTAT(DC,ISAFE,DDD,JSAFE)
   70           DDD=DA(I,J)*DAMP
             IF(DDD.NE.0.0) CALL ROTAT(DD,ISAFE,DDD,JSAFE)
   80        CONTINUE
             DDD=DR(I)*DAMP
             JSAFE=NP+1
          IF(DDD.NE.0.0) CALL ROTAT(DC,ISAFE,DDD,JSAFE)
   90     CONTINUE
          IF( NOT.IPM) GO TO 200
          IF(IIMOVE.LT.IFR1.OR.IIMOVE.GT.IFR2) GO TO 200
C
C      PROCESS PARTS INDIVIDUALLY
C
  100     IERROR=1
          IYMIN=IFX
          IOPNFG=0
          BAROUT = .FALSE.
          IF(.NOT.WR.OR.(IHLR.AND.(IDVICE.GE.0))) GO TO 50
          CALL OPNFIL('PICT',IUNIT,-2,IERROR)
          IF(IERROR.EQ.1) IOPNFG=-1
          IF(IERROR.EQ.0) GO TO 40
          FACT=1024./FLOAT(IFX)
          GO TO 50
   40     WR=.FALSE.
   50     CALL BGNFRM
          IF(IHLR) CALL INTHID
          IF(IBAD) GO TO 220
          IF(SHDO) CALL SHINIT
          DO 110 I=1,NJ
  110     XNORM(3,I)=0.
          IPERZ=1
          ISET=0
          ISCLIP = 0
          CALL SUMCEN(IPERZ)
          SPR = FRONTZ-BACKZ
          IF (SPR.LT.1E-6) SPR = 1.0
          DELZ = 0.005*SPR
          ZEF=DOZ-(FRONTZ+DELZ)
          ZEB=DOZ-(BACKZ-DELZ)
          IF(ZEF.LT.0.0) ZEF = DELZ
          ZPF = ZEF*TANAL
          ZPB = ZEB*TANAL
          ZSPRED=32767.*ZEB/(ZEB-ZEF)
CC
CC      Addition for change in "PIVOT" option
CC
          CALL MOD_ORG
CC
          DO 120 I=1,NP
             IPART=I
```

```
            IPOOR=.FALSE.
            DIRC=.FALSE.
            X1=POOR(IPART)
            X2=DIRCA(IPART)
            IF(X1.EQ.1.0) IPOOR=.TRUE.
            IF(X2.EQ.0.0) DIRC=.TRUE.
            IF (IDVICE.LT.0) THEN
               ISMOTH = 0
            ELSE
               ISMOTH=JSMOTH(IPART)
            ENDIF
            SHOSHR=.TRUE.
            IF(ISMOTH.EQ.-1) SHOSHR=.FALSE.
            IF(NPLS(I).EQ.0)GO TO 120
            CALL PART
C
C       IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT THE NEXT
C       FOUR STSTEMENTS
C
C             IF (ABORT) THEN
C                CALL ENDFRM
C                RETURN
C             ENDIF
C
  120     CONTINUE
          IF(IHLR) CALL HIDDEN
C
C       IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT THE NEXT
C       FOUR STATEMENTS
C
C          IF (ABORT) THEN
C             CALL ENDFRM
C             RETURN
C          ENDIF
C
          IF(IBAD) GO TO 220
          IF(.NOT.NUM) GO TO 130
          IF(NNUM) CALL NODNUM
          IF(PNUM) CALL POLNUM
  130     CONTINUE
          IF(ANTIAL) CALL AALIAS(A1,S1,A1,A1,S1,A1,IA1)
          CALL COARRO
          IF ((BAR.AND.IHLR.AND.IFRING).AND.IDVICE.GT.0) CALL FRNBAR
          CALL ENDFRM
          IF(.NOT.CONTRS.OR.LBLSPC.GE.9000.OR..NOT.IHLR) GO TO 150
          DO 140 K=1,NCONLV
  140     WRITE(OUTPUT,270) LIT(K),CLEVEL(K)
  150     IF(IIMOVE.GT.1) GO TO 180
          IF(IDVICE.GT.0) GO TO 160
          GO TO 170
  160     IF(IC.EQ.1) WRITE(OUTPUT,240)
          IF(IC.EQ.2) WRITE(OUTPUT,250)
  170     IF(NFRAME.LT.1) GO TO 200
  180     IF(IDVICE.EQ.-3.OR.IDVICE.EQ.-2) GO TO 190
          IF(IIMOVE.EQ.NFRAME) GO TO 190
          WRITE(OUTPUT,290) IIMOVE,NFRAME
          NW=4
          N1=1
          N2=0
          CALL READIN(WORD,NW)
```

```
          IF(K1.EQ.1) GO TO 210
          GO TO 200
  190     WRITE(OUTPUT,260) IIMOVE,NFRAME
  200 IF(IERROR.EQ.0) WR= TRUE.
  205 CONTINUE
  210 NFRAME=0
      IF(ITSHDO) SHDO=.TRUE.
      SKALE=XMAGN
      LINEAR=.FALSE.
      IF(IOPNFG.EQ.1) CALL CLSFIL(IUNIT)
      RETURN
  220 CALL ENDFRM
      WRITE(OUTPUT,230)
      NFRAME=0
      SKALE=XMAGN
      LINEAR=.FALSE.
      IBAD=.FALSE.
      NTR=.FALSE.
      IF(IOPNFG.EQ.1) CALL CLSFIL(IUNIT)
      RETURN
C
  230 FORMAT(' <HIDDEN FAILURE!>')
  240 FORMAT(' <COLOR PASS>')
  250 FORMAT(' <BLACK AND WHITE PASS>')
  260 FORMAT(' <',I3,'/',I3,'>')
  270 FORMAT(' <',A1,' =',F9.3,'>')
C
C     SEE POSSIBLE INCOMPATIBILITIES (2)
C
  290 FORMAT(' <',I3,'/',I3,'>')
      END
```

```
      SUBROUTINE GETXT(SPACE,NS)
C
C*************************************************************************
C
C  SUBROUTINE GETXT - RETRIEVES LINE OF ENTERED TEXT
C
C*************************************************************************
C
C  SUBROUTINE CALLED BY
C     READIN = ACCEPTS LINE OF TEXT, SEPARATES INTO WORDS
C              OR NUMBERS AND SENDS ON FOR PROCESSING
C
C*************************************************************************
C
C  SUBPROGRAMS CALLED
C     CLSFIL = CLOSES FILE AFTER I/O
C     DELSTR = DELETES SPECIFIED RECEIVED TEXT STRING
C     FNAME  = GATHERS THE CHARACTERS THAT MAKES UP THE FILENAME AND
C              COMBINES THEM INTO A WORD
C     LENTXT = DETERMINES THE LENGTH OF THE TEXT STRING
C
C*************************************************************************
C
C  VARIABLES USED
C     I      = POSITION OF THE CHARACTER IN THE LINE
C     INPUT  = DEVICE UNIT NUMBER TO READ, PASSED FROM COMMON/DEVI/
C     IUNIT  = DEVICE LOGICAL NUMBER
C     LENTXT = LENGTH OF THE TEXT STRING (PASSED FROM FUNCTION STAT.)
C     LINE   = LOGICAL REDIRECTION FLAG
C     LREAD  = LOGICAL REDIRECTION FLAG FOR READING
CC    LUNIT  = DEVICE LOGICAL UNIT NUMBER (DISK FILE) - INPUT
C     LWRITE = LOGICAL REDIRECTION FLAG FOR WRITING
C     NCHAR  = NUMBER OF CHARACTERS IN LINE OF TEXT ENTERED
C     NS     = ALLOWABLE SEPARATORS (4 OF THEM)
C     NULL   = CHARACTER EQUIVELANT TO ZERO (0)
C     OUTPUT = DEVICE UNIT NUMBER TO WRITE, PASSED FROM COMMON/DEVI/
C     SPACE  = ARRAY OF ALLOWABLE SEPARATORS
C     TEXT   = ARRAY OF ENTERED TEXT
C     XNAME  = USER'S SPECIFIED FILENAME FOR I/O
C
C*************************************************************************
C
      CHARACTER*1 SPACE(NS),TEXT(72)
      CHARACTER*12 XNAME
      INTEGER OUTPUT
      LOGICAL LINE,LREAD,LWRITE
      SAVE XNAME,LUNOLD
      COMMON/DEVI/ INPUT,OUTPUT
CC
      COMMON/DEVIL/ MUNIT,JUNIT,KUNIT,LUNIT
CC
      COMMON/TEXTO/NCHAR
      COMMON/TEXT1/TEXT
      DATA IUNIT/3/,NULL/0/
      DATA LINE,LREAD,LWRITE/3*.FALSE./
C
C     GET LINE OF TEXT
C
      NCHAR=72
```

```
      READ(INPUT,1100,END=400) (TEXT(I),I=1,NCHAR)
      NCHAR=LENTXT(TEXT,NCHAR)
      IF (INPUT.EQ.IUNIT) WRITE(OUTPUT,1800) (TEXT(I),I=1,NCHAR)
C
C     SCAN INPUT LINE FOR SPECIAL REDIRECTION CHARACTERS.
C
      LINE=.FALSE.
      DO 200 I=1,NCHAR
C
C        OUTPUT TO FILE.
C
         IF(TEXT(I).EQ.'>') THEN
C
C           CLOSE FILE IF OPEN.
C
            IF(LWRITE) THEN
               WRITE(OUTPUT,1200) XNAME
               IF(I-1 GT.1) THEN
                  WRITE(IUNIT,1100) (TEXT(J),J=1,I-1)
                  LINE=.TRUE.
                  CALL DELSTR(TEXT,I,I+1,NCHAR)
               ELSE IF(NCHAR.EQ.1) THEN
                  WRITE(OUTPUT,1700)
                  NCHAR=72
                  READ(INPUT,1100,END=400) (TEXT(J),J=1,NCHAR)
                  NCHAR=LENTXT(TEXT,NCHAR)
               ELSE
                  CALL DELSTR(TEXT,I,I+1,NCHAR)
               END IF
               CALL CLSFIL(IUNIT)
               LWRITE=.FALSE.
               GO TO 300
C
C           OPEN FILE IF READ REDIRECTION NOT IN USE.
C
            ELSE IF(.NOT.LREAD) THEN
               CALL FNAME(I,SPACE,NS,XNAME,L)
               OPEN(IUNIT,FILE=XNAME(:L),STATUS='NEW',ERR=130)
               REWIND IUNIT
               WRITE(OUTPUT,1300) XNAME
               LWRITE=.TRUE.
               IF(NCHAR.GT.I) THEN
                  WRITE(IUNIT,1100) (TEXT(J),J=I,NCHAR)
                  LINE=.TRUE.
               ELSE IF(NCHAR.LE.1) THEN
                  WRITE(OUTPUT,1700)
                  NCHAR=72
                  READ(INPUT,1100,END=400) (TEXT(J),J=1,NCHAR)
                  NCHAR=LENTXT(TEXT,NCHAR)
               END IF
               GO TO 300
C
C              ERROR OPENING REDIRECTION OUTPUT FILE.
C
130            CONTINUE
                  WRITE(OUTPUT,1600)
                  NCHAR=I-1
               GO TO 300
C
C           ERROR READ REDIRECTION ALREADY IN USE.
```

```
C
            ELSE IF(LREAD) THEN
               WRITE(OUTPUT,1400)
               NCHAR=I-1
               GO TO 300
            END IF
C
C        INPUT FROM FILE
C
         ELSE IF(TEXT(I).EQ.'<') THEN
C
C           CLOSE FILE IF OPEN
C
            IF(LREAD) THEN
               CALL CLSFIL(IUNIT)
               LREAD=.FALSE.
               WRITE(OUTPUT,1200) XNAME
               CALL DELSTR(TEXT,I,I+1,NCHAR)
               INPUT=LUNOLD
               IF(NCHAR.LE.1) THEN
                  WRITE(OUTPUT,1700)
                  NCHAR=72
                  READ(INPUT,1100,END=400) (TEXT(J),J=1,NCHAR)
                  NCHAR=LENTXT(TEXT,NCHAR)
               END IF
               GO TO 300
C
C           OPEN FILE IF WRITE REDIRECTION NOT IN USE.
C
            ELSE IF( NOT.LWRITE) THEN
               CALL FNAME(I,SPACE,NS,XNAME,L)
               OPEN(IUNIT,FILE=XNAME(:L),STATUS='OLD',ERR=170)
               REWIND IUNIT
               LREAD=.TRUE.
               WRITE(OUTPUT,1500) XNAME
               LUNOLD = INPUT
               INPUT=IUNIT
               IF(NCHAR.LE.1) THEN
                  NCHAR=72
                  READ(INPUT,1100,END=400) (TEXT(J),J=1,NCHAR)
                  NCHAR=LENTXT(TEXT,NCHAR)
                  IF (INPUT.EQ.IUNIT) WRITE(OUTPUT,1800)
     $               (TEXT(J),J=1,NCHAR)
               END IF
               GO TO 300
C
C              ERROR OPENING REDIRECTION OUTPUT FILE.
C
170            CONTINUE
               WRITE(OUTPUT,1600)
               NCHAR=I-1
               GO TO 300
C
C           ERROR WRITE REDIRECTION ALREADY IN USE.
C
            ELSE IF(LWRITE) THEN
               WRITE(OUTPUT,1400)
               NCHAR=I-1
               GO TO 300
            END IF
```

```
         END IF
200    CONTINUE
300    CONTINUE
C
C      IF OUTPUT REDIRECTION AND LINE NOT PREVIOUSLY WRITTEN,
C      THEN WRITE THIS LINE.
C
       IF(LWRITE.AND  NOT LINE) WRITE(IUNIT,1100) (TEXT(I),I=1,NCHAR)
       RETURN
C
C      END-OF-FILE ON INPUT
C
400    CONTINUE
CC
CC     If input was from an animation (command) file send on
CC     the "HALT" option to close file and reset input device
CC
       IF (INPUT.EQ.LUNIT) THEN
          TEXT(1) = 'H'
          TEXT(2) = 'A'
          TEXT(3) = 'L'
          TEXT(4) = 'T'
          RETURN
       ENDIF
CC
       IF(LREAD) THEN
          CALL CLSFIL(IUNIT)
          LREAD=.FALSE.
          WRITE(OUTPUT,1200) XNAME
          WRITE(OUTPUT,1700)
          NCHAR=72
          INPUT=LUNOLD
          READ(INPUT,1100,END=400) (TEXT(J),J=1,NCHAR)
          NCHAR=LENTXT(TEXT,NCHAR)
       ELSE
          STOP
       END IF
       RETURN
C
1100   FORMAT(72A1)
1200   FORMAT(' <CLOSE FILE: ',A,'>')
1300   FORMAT(' <INPUT TO FILE: ',A,'>')
1400   FORMAT(' <LOGICAL UNIT IN USE.  REST OF LINE IGNORED.>')
1500   FORMAT(' <INPUT FROM FILE: ',A,'>')
1600   FORMAT(' <ERROR:  OPEN FILE!  REST OF LINE IGNORED.>')
1700   FORMAT(' INPUT>> ')
1800   FORMAT(' ',72A)
C
       END
```

```
      SUBROUTINE NUMDIG(WORD,N,L1,L2)
C
C*****************************************************************************
C
C  SUBROUTINE NUMDIG - ACCEPTS NUMBER DIGITS AND CONVERTS TO REAL
C                      NUMBER
C
C*****************************************************************************
C
C  SUBROUTINE CALLED BY
C     READIN = ACCEPTS LINE OF TEXT, SEPARATES INTO WORDS
C              OR NUMBERS AND SENDS ON FOR PROCESSING
C     WORDS  = COMPARES TEXT WORD SENT TO IT WITH ACCEPTABLE KEY WORDS
C
C*****************************************************************************
C
C  SUBPROGRAMS CALLED - NONE
C
C*****************************************************************************
C
C  VARIABLES USED
C     FAC    = MULTIPLICATION FACTOR WHICH CHANGES NUMBER WITH
C              EXPONENT TO A REAL NUMBER WITHOUT EXPONENT
C     ICOUNT = COUNTER FOR NUMBER OF DIGITS IN A NUMBER OR EXPONENT
C     ID     = NUMBER OF DECIMALS FOUND IN A NUMBER (FOR ERROR PURPOSE)
C     IDOT   = 1 AT THE TIME A DECIMAL IS FOUND, THEN RESET TO ZERO
C     IE     = EXPONENT FLAG, =1 IF EXPONENT EXISTS
C                             =0 IF NO EXPONENT EXISTS
C     ILOC   = NUMBER OF DIGITS TO LEFT OF DECIMAL
C     IRIGHT = NUMBER OF DIGITS TO RIGHT OF DECIMAL
C     ISIGN  = 1 FOR POSITIVE NUMBERS
C              -1 FOR NEGATIVE NUMBERS
C     J      = DIMENSION OF NUMT ARRAY
C     JCOUNT = COUNTER FOR RECOGNIZING NUMBER OR EXPONENT DIGIT
C     K2     = NUMBER OF NUMBERS FOUND
C     KEY    = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
C              ARRAY
C     LE     = STARTING LOCATION OF EXPONENT
C     LT     = TEMPORARY STARTING LOCATION OF A TEXT WORD
C     MM     = 10 TO THE CORRECT POWER USED TO CALCULATE NUMBER
C              FROM ITS DIGITS
C     MSIGN  = 1 FOR POSITIVE EXPONENT
C              -1 FOR NEGATIVE EXPONENT
C     MUL    = EXPONENT VALUE
C     MULT   = ARRAY OF DIGITS IN EXPONENT
C     MULTN  = NUMBER OF DIGITS IN EXPONENT
C     N2     = INPUT FLAG, =1 WHEN NUMBERS WANTED
C                          =0 WHEN NUMBERS NOT WANTED
C     NR     = VALUE OF THE NUMBER
C     NUMT   = ARRAY OF DIGITS IN NUMBER
C     NUMTN  = NUMBER OF DIGITS IN NUMBER
C     TEXT   = ARRAY OF ENTERED CHARACTERS
C     WORD   = ARRAY OF ACCEPTABLE COMMANDS FOR ACTIVE ROUTINE
C     X      = ARRAY OF ACCEPTABLE NUMBER DIGITS
C     XNUM   = ARRAY OF REAL NUMBERS ACCEPTED
C
C*****************************************************************************
C
      CHARACTER*1 TEXT,X,WORD
      COMMON/ENTER/ N1,N2,KEY,XNUM,K1,K2
```

```
      COMMON/TEXTO/NCHAR
      COMMON/TEXT1/TEXT
      DIMENSION KEY(10),XNUM(40),TEXT(72),WORD(*)
      DIMENSION NUMT(15),MULT(5),X(15)
      DATA X/'1','2','3','4','5','6','7','8','9','0',
     1'+','-','E','e','.'/
C
C     IF NO NUMBERS SOUGHT - RETURN.
C
      IF(N2.EQ.0) RETURN
C
C     INITIALIZE FLAGS AND COUNTERS.
C
      LT=L1
      IE=0
      NUMTN=0
      MULTN=0
      MUL=0
      ISIGN=1
      MSIGN=1
      IDOT=0
      ILOC=-1
      NR=0
      ID=0
C
C     GET SIGN OF NUMBER
C
      IF(TEXT(LT).EQ.X(11)) LT=LT+1
      IF(TEXT(LT).NE.X(12)) GO TO 10
      ISIGN=-1
      LT=LT+1
   10 ICOUNT=0
      DO 60 I=LT,L2
         JCOUNT=0
C
C        IF NOT EXPONENT DIGITS - JUMP.
C
         IF(IE.NE.1) GO TO 20
C
C        GET SIGN OF EXPONENT.
C
         IF(TEXT(I).EQ.X(11)) GO TO 80
         IF(TEXT(I).EQ.X(12)) GO TO 70
         LE=I
         GO TO 90
C
C        IF DECIMAL HAS BEEN FOUND - SET IDOT BACK TO ZERO.
C
   20    IF(IDOT.EQ.0) GO TO 30
         IDOT=0
         GO TO 40
C
C        CHECK FOR DECIMAL POINT.
C
   30    IF(TEXT(I).NE.X(15)) GO TO 40
         IDOT=1
         ID=ID+1
         ILOC=ICOUNT
         GO TO 60
C
```

```
C         CHECK FOR EXPONENT CHARACTER E.
C
   40     IF(TEXT(I) .NE. X(13) .AND. TEXT(I) .NE. X(14)) GO TO 50
          IE=1
          GO TO 60
C
C         IDENTIFY NUMBER DIGIT AND LOAD INTO NUMT.
C
   50     IF(JCOUNT.GT.10) RETURN
          JCOUNT=JCOUNT+1
          IF(TEXT(I).NE.X(JCOUNT)) GO TO 50
          ICOUNT=ICOUNT+1
          IF(JCOUNT.EQ.10) JCOUNT=0
          NUMT(ICOUNT)=JCOUNT
          NUMTN=ICOUNT
   60 CONTINUE
      GO TO 120
   70 MSIGN=-1
   80 LE=I+1
C
C     IDENTIFY EXPONENT DIGIT AND LOAD INTO MULT.
C
   90 ICOUNT=0
      DO 110 I=LE,L2
         JCOUNT=0
  100    IF(JCOUNT.GT.10) RETURN
         JCOUNT=JCOUNT+1
         IF(TEXT(I).NE.X(JCOUNT)) GO TO 100
         ICOUNT=ICOUNT+1
CC
CC    Make sure a zero is handled correctly
CC
         IF (JCOUNT.EQ.10)  JCOUNT=0
CC
         MULT(ICOUNT)=JCOUNT
  110 MULTN=ICOUNT
C
C     IF NO NUMBER DIGITS RECOGNIZED - RETURN.
C
  120 IF(NUMTN.EQ.0) RETURN
C
C     IF MORE THAN ONE DECIMAL POINT FOUND IN NUMBER - RETURN.
C
      IF(ID.GT.1) RETURN
      K2=K2+1
C
C     IF NO DECIMAL POINT FOUND - ASSUME AT END OF NUMBER
C
      IF(ILOC.EQ.-1) ILOC=NUMTN
C
C     IF NO EXPONENT - JUMP.
C
      IF(MULTN.EQ.0) GO TO 140
      MM=1
C
C     EXPONENT IS CALCULATED, ASSIGNED ITS SIGN AND ALTERED TO
C     REFLECT LOCATION OF DECIMAL POINT.  (EFFECTIVELY MOVING
C     DECIMAL POINT TO END OF NUMBER)
C
      DO 130 I=1,MULTN
```

```
            J=MULTN+1-I
            MUL=MUL+MULT(J)*MM
  130 MM=10*MM
      MUL=MSIGN*MUL
  140 IRIGHT=NUMTN-ILOC
      MUL=MUL-IRIGHT
      MM=1
      DO 150 I=1,NUMTN
          J=NUMTN+1-I
          NR=NR+NUMT(J)*MM
C
C     NUMBER VALUE CALCULATED FROM ITS DIGITS.
C
  150 MM=10*MM
      NR=ISIGN*NR
C
C     APPROPRIATE MULTIPLICATION FACTOR(DUE TO EXPONENT) CALCULATED.
C
      IF(MUL) 160,180,170
  160 MUL=-MUL
      FAC=1./(10.**MUL)
      GO TO 190
  170 FAC=10.**MUL
      GO TO 190
  180 FAC=1.
C
C     MULTIPLICATION FACTOR IS APPLIED TO NUMBER AND RESULTING NUMBER
C     IS LOADED INTO XNUM ARRAY.
C
  190 XNUM(K2)=FLOAT(NR)*FAC
      RETURN
      END
```

Appendix B

New NOVTEL/BYU Routines

FILE OP
GET ORG
HALT OP
MOV ORG
PAUSE OP
PLOT OP
NOPLOT
RES OPN

```
      SUBROUTINE FILE_OP
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE FILE_OP - ENABLES COMMAND INPUT FROM DISK FILE            C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE CALLED BY                                                 C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND        C
C               CALLS APPROPRIATE SUBROUTINES                           C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBPROGRAMS CALLED                                                   C
C     OPEN    = SYSTEM OPEN FILE ROUTINE                                C
C     RES_CEN = RESTORES MODEL TO ORIGINAL COORDINATE SYSTEM            C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  VARIABLES USED                                                       C
C     INPUT  = DEVICE LOGICAL UNIT NUMBER FOR INPUT                     C
C     JUNIT  = DEVICE LOGICAL UNIT NUMBER (DISK FILE)  OUTPUT           C
C     KUNIT  = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)                   C
C     LUNIT  = DEVICE LOGICAL UNIT NUMBER (DISK FILE)  INPUT            C
C     MUNIT  = DEVICE LOGICAL UNIT NUMBER (SYS$OUTPUT)                  C
C     OUTPUT = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT                    C
C     XNAME  = NAME OF USERS COMMAND FILE (DEFAULT EXT .DAT)            C
C                                                                       C
C***********************************************************************C
C
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/DEVIL/ MUNIT,JUNIT,KUNIT,LUNIT
      CHARACTER*12 BLANK,XNAME
      INTEGER OUTPUT
      DATA BLANK/'            '/
C
   10 WRITE(OUTPUT,50)
      XNAME = BLANK
      READ (INPUT,60)  XNAME
      IF (XNAME.NE.BLANK)  THEN
          OPEN(UNIT=LUNIT,FILE=XNAME,TYPE='OLD',ERR=10)
          OPEN(UNIT=JUNIT,FILE='PROMPT.DUM',TYPE='NEW',
     1          DISPOSE='DELETE')
          INPUT = LUNIT
          OUTPUT = JUNIT
          CALL RES_CEN
      ENDIF
      RETURN
   50 FORMAT(21H+<READ COMMAND FILE> $)
   60 FORMAT(A12)
      END
```

```
      SUBROUTINE GET_ORG
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE GET_ORG - INPUTS RELATIVE ORIGIN OF PARTS (PIVOT POINT)   C
C                       FOLLOWING PROMPT (>>>) USER ENTERS VALUES FOR   C
C                       I1 AND I2 (RANGE OF PARTS), I3 (PART ON WHICH   C
C                       I1 TO I2 PIVOT) & X,Y,Z (PIVOT POINT COORDS).   C
C                       WHERE POSITION OF PARTS I1/I2 IS NOT AFFECTED   C
C                       BY THE PIVOTS OF ANY OTHER PART, ENTER A ZERO   C
C                       FOR PART I3.  A BLANK INPUT (CARRIAGE RETURN)   C
C                       WILL TRANSFER CONTROL BACK TO CALLING PROGRAM   C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE CALLED BY                                                 C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND        C
C               CALLS APPROPROPRIATE SUBROUTINES                        C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINES CALLED                                                   C
C     READIN = ACCEPTS INPUT FROM THE USER                              C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  VARIABLES USED                                                       C
C     IPREV  = ARRAY OF CONNECTING PART NUMBERS                         C
C     N1     = INPUT FLAG, =1 WHEN KEY WORDS SOUGHT                     C
C                          =0 WHEN KEY WORDS NOT SOUGHT                 C
C     N2     = INPUT FLAG, =1 WHEN NUMBERS WANTED                       C
C                          =0 WHEN NUMBERS NOT WANTED                   C
C     NW     = DIMENSION OF WORD ARRAY                                  C
C     OUTPUT = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT                    C
C     PROMPT = IF .TRUE. PROMPT FOR USER INPUT (LOCAL VAR.)             C
C     SORG   = ARRAY OF SPECIFIED PIVOTAL POINTS                        C
C     XNUM   = ARRAY OF REAL NUMBERS ACCEPTED                           C
C                                                                       C
C***********************************************************************C

      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/ENTER/ N1,N2,KEY,XNUM,K1,K2
      COMMON/REST3/ PORG,SORG,IPREV
      DIMENSION KEY(10),XNUM(40)
      DIMENSION PORG(3,27),SORG(3,27),IPREV(27)
      INTEGER OUTPUT
      LOGICAL PROMPT

      WRITE(OUTPUT,70)
      PROMPT = .TRUE.
      NW = 0
      N1 = 0
      N2 = 1

      DO WHILE (PROMPT)
         WRITE(OUTPUT,80)
         CALL READIN(WORD,NW)
         I1 = XNUM(1)
         IF (I1 .EQ. 0) THEN
            PROMPT = .FALSE.
```

```
            ELSE
                 I2 = XNUM(2)
                 DO J = I1,I2
                      IPREV(J) = XNUM(3)
                      DO I = 1,3
                           SORG(I,J) = XNUM(I+3)
                      ENDDO
                 ENDDO
            ENDIF
       ENDDO
       RETURN

 70    FORMAT(29H+DEFINE CONNECTIVITY OF PARTS /
       1       43H <PARTS RANGE I1/I2, PART I3, PIVOT COORDS>/)
 80    FORMAT(5H+>>> ,$)
       END
```

```
      SUBROUTINE HALT_OP
C
C*************************************************************************C
C                                                                         C
C  SUBROUTINE HALT_OP - TERMINATES COMMAND INPUT FROM DISK FILE           C
C                                                                         C
C*************************************************************************C
C                                                                         C
C  SUBROUTINE CALLED BY                                                   C
C     MAIN     = PROCESSES INTERACTIVE COMMANDS FROM THE USER            C
C                AND CALLS APPROPRIATE SUBROUTINES                        C
C     PAUS_OP  = PROCESSES THE "PAUSE" OPTION                            C
C                                                                         C
C*************************************************************************C
C                                                                         C
C  SUBPROGRAMS CALLED                                                     C
C     CLOSE   = SYSTEM CLOSE FILE ROUTINE                                C
C                                                                         C
C*************************************************************************C
C                                                                         C
C  VARIABLES USED                                                         C
C     INPUT   = DEVICE LOGICAL UNIT NUMBER FOR INPUT                     C
C     JUNIT   = DEVICE LOGICAL UNIT NUMBER (DISK FILE)  OUTPUT           C
C     KUNIT   = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)                   C
C     LUNIT   = DEVICE LOGICAL UNIT NUMBER (DISK FILE)  INPUT            C
C     MUNIT   = DEVICE LOGICAL UNIT NUMBER (SYS$OUTPUT)                  C
C     OUTPUT  = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT                    C
C                                                                         C
C*************************************************************************C
C
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/DEVIL/ MUNIT,JUNIT,KUNIT,LUNIT
      INTEGER OUTPUT
C
      CLOSE(LUNIT)
      CLOSE(JUNIT)
      OUTPUT = MUNIT
      INPUT = KUNIT
      RETURN
      END
```

```
      SUBROUTINE MOD_ORG
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE MOD_ORG - COMPUTES CURRENT POSITION OF PIVOTAL POINTS     C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  SUBROUTINE CALLED BY                                                 C
C     VIEDRA = CALLS FOR NORMALS, LIGHT INTENSITY, ETC. NEEDED TO       C
C              DISPLAY SCENE (EITHER VIEW OR DRAW)                      C
C                                                                       C
C***********************************************************************C
C                                                                       C
C  VARIABLES USED                                                       C
C     DD      = LOCAL TRANSFORMATION MATRIX                             C
C     IPREV   = ARRAY OF CONNECTING PART NUMBERS                        C
C     NP      = NUMBER OF PARTS                                         C
C     PORG    = ARRAY OF PIVOTAL POINTS CURRENT POSITION                C
C     RORG    = RELATIVE ORIGIN OF PART                                 C
C     SORG    = SPECIFIED PIVOTAL POINT OF PART                         C
C                                                                       C
C***********************************************************************C

      COMMON/PRO  / NP, NJ, ICMD, IC, IREAD, SKALE, NPT
      COMMON/REST1/ RORG, DD
      COMMON/REST3/ PORG, SORG, IPREV
      DIMENSION DD(3,3,27), RORG(3,27)
      DIMENSION PORG(3,27), SORG(3,27), IPREV(27)

      DO I = 1, NP
         DO J = 1, 3
            RORG(J, I) = SORG(J, I)
         ENDDO
         J = IPREV(I)
         IF (J EQ. 0) THEN
            DO J = 1, 3
               PORG(J, I) = RORG(J, I)
            ENDDO
         ELSE
            X1 = RORG(1, I) - RORG(1, J)
            X2 = RORG(2, 1) - RORG(2, J)
            X3 = RORG(3, I) - RORG(3, J)
            PORG(1, I) = DD(1, 1, J)*X1 + DD(2, 1, J)*X2 +
     1                   DD(3, 1, J)*X3 + PORG(1, J)
            PORG(2, I) = DD(1, 2, J)*X1 + DD(2, 2, J)*X2 +
     2                   DD(3, 2, J)*X3 + PORG(2, J)
            PORG(3, I) = DD(1, 3, J)*X1 + DD(2, 3, J)*X2 +
     3                   DD(3, 3, J)*X3 + PORG(3, J)
         ENDIF
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE PAUS_OP
C
C*****************************************************************C
C                                                                 C
C  SUBROUTINE PAUS_OP - WAITS FOR INPUT AT TERMINAL KEYBOARD      C
C                       IF AN 'A' IS INPUT - HALT_OP IS EXECUTED  C
C                       TO CLOSE USER'S COMMAND FILE AND TO SET   C
C                       SYS$INPUT AS THE COMMAND-INPUT DEVICE     C
C                       . OTHERWISE..  NO CHANGES  (IF COMMAND    C
C                       INPUT IS FROM A DISK FILE A "PAUSE" WILL  C
C                       PERMIT USER TO READY THE PLOTTER  ..OR.   C
C                       ENTER "A" TO ABORT FILE-COMMAND INPUT)    C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  SUBROUTINE CALLED BY                                           C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND  C
C               CALLS APPROPRIATE SUBROUTINES                     C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  SUBPROGRAMS CALLED                                             C
C    HALT_OP = TERMINATES COMMAND INPUT FROM DISK FILE            C
C    READIN  = ACCEPTS INPUT FROM THE USER                        C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  VARIABLES USED                                                 C
C     INSAV   = TEMPORARY STORAGE OF LOGICAL UNIT NUMBER (LOCAL VAR)  C
C     K1      = NUMBER OF KEYWORDS FOUND                          C
C     N1      = INPUT FLAG = 1 WHEN KEYWORD SOUGHT                C
C     N2      = INPUT FLAG = 0 WHEN NUMBERS NOT WANTED            C
C     NW      = DIMENSION OF WORD ARRAY                           C
C     WORD    = ARRAY OF ACCEPTABLE WORDS                         C
C                                                                 C
C*****************************************************************C
C
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/ENTER/ N1,N2,KEY,XNUM,K1,K2
      INTEGER OUTPUT
      DIMENSION KEY(10),XNUM(40),WORD(4)
      CHARACTER*1 WORD
      DATA WORD/'A','#','#','#'/
C
      WRITE(6,100)
      INSAV = INPUT
      INPUT = 6
      NW = 4
      N1 = 1
      N2 = 0
      CALL READIN(WORD,NW)
      IF (K1.NE.1) THEN
          INPUT = INSAV
      ELSE
          CALL HALT_OP
      ENDIF
      RETURN
  100 FORMAT(5H+>>  ,$)
      END
```

```
      SUBROUTINE PLOT_OP
C
C*********************************************************************C
C                                                                     C
C  SUBROUTINE PLOT_OP - ENABLES SWITCH FOR HARD COPY PLOTS.           C
C                       PROCESSES INTERACTIVE COMMAND  "PLOT"         C
C                       WHEN COMMAND INPUT IS VIA 'SYS$INPUT' AND     C
C                       USER REQUESTS A VIEW OR DRAW. THE PICTURE     C
C                       WILL BE DISPLAYED ON THE TERMINAL SCREEN.     C
C                       IF THIS HARD-COPY SWITCH IS ENABLED, USER     C
C                       WILL THEN BE GIVEN THE OPTION OF PLOTTING     C
C                       THE DISPLAY ON THE PLOTTER.  WHEN COMMAND     C
C                       INPUT IS FROM A DISK FILE PICTURE WILL BE     C
C                       DISPLAYED ON THE PLOTTER IF 'PLOT' SWITCH     C
C                       IS ON, BUT ON THE TERMINAL SCREEN IF OFF.     C
C                                                                     C
C*********************************************************************C
C                                                                     C
C  SUBROUTINE CALLED BY                                               C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND      C
C               CALLS APPROPRIATE SUBROUTINES                         C
C                                                                     C
C*********************************************************************C
C                                                                     C
C  VARIABLES USED                                                     C
C     APLOT  = ENABLES ONE HARD COPY OUTPUT                           C
C     HRDCPY = LOGICAL FLAG INDICATING PLOT IS ON (OR OFF)            C
C     KUNIT  = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)                 C
C                                                                     C
C*********************************************************************C
C
      COMMON/HCOPY/ HRDCPY,APLOT,HCEND
      COMMON/DEVI/ INPUT,OUTPUT
      COMMON/DEVIL/ MUNIT,JUNIT,KUNIT,LUNIT
      LOGICAL HRDCPY,APLOT,HCEND,KEYBD
      INTEGER OUTPUT
C
      HRDCPY = .TRUE.
      IF (INPUT.EQ.KUNIT) THEN
          WRITE(OUTPUT,20)
          APLOT = .FALSE.
      ELSE
          APLOT = .TRUE.
      ENDIF
      RETURN
C
   20 FORMAT(16H+<PLOT ENABLED> ,/)
      END
```

```
      SUBROUTINE NOPLOT
C
C*************************************************************************C
C                                                                         C
C  SUBROUTINE NOPLOT  - DISABLES SWITCH FOR HARD COPY PLOTS.              C
C                       PROCESSES INTERACTIVE COMMAND  "NOPLot"           C
C                                                                         C
C*************************************************************************C
C                                                                         C
C  SUBROUTINE CALLED BY                                                   C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND          C
C               CALLS APPROPRIATE SUBROUTINES                             C
C                                                                         C
C*************************************************************************C
C                                                                         C
C  VARIABLES USED                                                         C
C     APLOT   = ENABLES ONE HARD COPY OUTPUT                              C
C     HRDCPY  = LOGICAL FLAG INDICATING PLOT IS ON (OR OFF)               C
C                                                                         C
C*************************************************************************C
C
      COMMON/HCOPY/ HRDCPY, APLOT, HCEND
      COMMON/DEVI/ INPUT, OUTPUT
      LOGICAL HRDCPY, APLOT, HCEND
      INTEGER OUTPUT
C
      HRDCPY = .FALSE.
      APLOT  = .FALSE.
      WRITE(OUTPUT,30)
      RETURN
C
   30 FORMAT(17H<CPLOT DISABLED> ,/)
      END
```

```
      SUBROUTINE RES_CEN
C
C*****************************************************************C
C                                                                 C
C  SUBROUTINE RES_CEN - RESTORES MODEL TO ORIGINAL COORDINATE SYSTEM  C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  SUBROUTINE CALLED BY                                           C
C     FILE_OP = ENABLES COMMAND INPUT FROM USERS DISK FILE        C
C     MAIN    = PROCESSES INTERACTIVE COMMANDS FROM USER          C
C     PAUS_OP = WAITS FOR KEYBOARD INPUT                          C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  SUBPROGRAMS CALLED                                             C
C     REST   = RESTORES MODEL TO ORIGINAL COORDINATE SYSTEM       C
C              (KILLS ROTATIONS AND TRANSFORMATIONS)              C
C     SUMCEN = CENTERS MODEL IN VIEWING AREA                      C
C                                                                 C
C*****************************************************************C
C                                                                 C
C  VARIABLES USED                                                 C
C     ICMD   = INTERACTIVE COMMAND WORD STARTING LOCATION         C
C                                                                 C
C*****************************************************************C
C
      COMMON/PRO/ NP,NJ,ICMD,IC,IREAD,SKALE,NPT
C
      CALL REST
C
C     ICMD 69 = "CENT"
C
      ICMD = 69
      CALL SUMCEN
      RETURN
      END
```

Appendix C

Animation Control Routines

RUN_MOVIE
DISPLAY_FILE
EXIT_MOVIE
ALL_REC

```
      SUBROUTINE RUN_MOVIE (FGEOM,FDISP,FFUNC,FORIG)
C
C     RUN_MOVIE will create and run the "MOVIE" subprocess.  Communication
C     between this program and MOVIE will be via two mailboxes ("ABOX" and
C     "BBOX" created here).  The names (subroutine arguments) of geometry,
C     displacement, function and the parts-origin files are sent to MOVIE.
C     MOVIE's prompt (">>") for command input is read and control returned
C     to the calling program.  DISPLAY_FILE will output the "FILE" command
C     to MOVIE and the name of the disk-file (this entry's argument) which
C     contains the commands (and data if required) to be executed by MOVIE.
C     After commands (normally terminated by a "HALT") have been processed
C     MOVIE's prompt (">>") is read & control returned to calling program.
C     Arguments (char. variables) are : FGEOM - name of geometry file
C                                       FDISP - name of displacement file
C                                       FFUNC - name of function file
C                                       FORIG - name of parts-origin file
C                                       FCOMM - name of user-commands file
C
C     Call  RUN_MOVIE      once to initiate the MOVIE subprocess
C     Call  DISPLAY_FILE   whenever it is desired to have a set
C                          of commands processed by MOVIE.
C     Call  EXIT_MOVIE     to delete the MOVIE ("READIT") subprocess
C
      INTEGER*2  CH, CA
      INTEGER*4  SYS$DELPRC, SYS$CREPRC, SYS$QIOW
      INTEGER*4  SYS$CREMBX, SYS$TRNLOG, IS
      EXTERNAL   SS$_NORMAL, IO$_WRITEVBLK, IO$_READVBLK
      CHARACTER*8  TA, TB, TM*7, TT*32, FNAM*12, MIN*18, MXX*21
      CHARACTER*4  MA/'ABOX'/, MB/'BBOX'/, OUTC/'FILE'/
      CHARACTER*(*)  FGEOM, FDISP, FFUNC, FORIG, FCOMM
      LOGICAL  MOVERR

      IS = SYS$DELPRC(,'READIT')

C     Create Mailboxes:   "ABOX"  output from this process
C                         "BBOX"  input to this process
C
      IS = SYS$CREMBX(,CH,,,,,MB)
      IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
      IS = SYS$CREMBX(,CA,,,,,MA)
      IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))

C     Translate logical
C
      IS = SYS$TRNLOG('BBOX',,TT,,,)
      TB = TT(INDEX(TT,'_')+1:INDEX(TT,':')-1)
      IS = SYS$TRNLOG('ABOX',,TT,,,)
      TA = TT(INDEX(TT,'_')+1:INDEX(TT,':')-1)
      IS = SYS$TRNLOG('SYS$COMMAND',,TT,,,)
      TM = TT(INDEX(TT,'_')+1:INDEX(TT,':')-1)

C     Create process "READIT" with  SYS$INPUT   "ABOX"
C                                   SYS$OUTPUT  "BBOX"
C                                   SYS$ERROR   TERMINAL
C
      IS= SYS$CREPRC(ID,'[FTS_MOVIE]MOVIE.EXE',TA,TB,TM,,,'READIT',4,,,)
      IF (IS .NE. %LOC(SS$_NORMAL))  TYPE 100, IS

C                . . .assume MOVIE process is running...
C
```

```
C      Read two records from the mailbox "BBOX",
C      Output geometry file_name to mailbox "ABOX"

       DO I = 1,2
          IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5                  %VAL(18),,,,)
          IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
       ENDDO
       FNAM = FGEOM
       IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(FNAM),
     5               %VAL(12),,,,)
       IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))

C      Read two records from the mailbox "BBOX'
C      Output displacement file_name to mailbox "ABOX"

       DO I = 1,2
          IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5                  %VAL(18),,,,)
          IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
       ENDDO
       FNAM = FDISP
       IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(FNAM),
     5               %VAL(12),,,,)
       IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))

C      Input one record from the mailbox "BBOX"
C      Output function file_name to mailbox "ABOX"

       IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5               %VAL(18),,,,)
       IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
       FNAM = FFUNC
       IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(FNAM),
     5               %VAL(12),,,,)
       IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))

C      Input one record from the mailbox "BBOX"
C      Output origin-of-parts file_name to mailbox "ABOX"

       IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5               %VAL(18),,,,)
       IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
       FNAM = FORIG
       IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(FNAM),
     5               %VAL(12),,,,)
       IF ( NOT. IS)  CALL LIB$STOP(%VAL(IS))


C      Input records from the mailbox "BBOX"
C      until prompt (" >>") for command input is read

       MIN(2:3) = '##'
       DO WHILE (MIN(2:3).NE.'>>')
          IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5                  %VAL(4),,,,)
          IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
       ENDDO
       RETURN
```

```
      ENTRY DISPLAY_FILE (FCOMM)
C                                       Instruct  MOVIE  to process the
C                                       commands in user's file 'FCOMM'.
C                                       Wait until all commands executed


C     Output interactive command  "FILE" to "ABOX"

      IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(OUTC),
     5              %VAL(4),,,,)
      IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))

C     Input 1 record from the mailbox "BBOX"
C     Output commands file_name to mailbox "ABOX"

      IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MXX),
     5              %VAL(21),,,,)
      IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
      FNAM = FCOMM
      IS = SYS$QIOW(%VAL(3),%VAL(CA),IO$_WRITEVBLK,,,,%REF(FNAM),
     5              %VAL(12),,,,)
      IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))


C     MOVIE's outputs to mailbox inhibitted until "HALT" from file
C     or end-of-file  ( then the prompt  " >>"  should be output )

C     Input records from the mailbox "BBOX"
C     until prompt (" >>") for command input is read

      MIN(2:3) = '##'
      DO WHILE (MIN(2:3).NE.'>>')
         IS = SYS$QIOW(%VAL(3),%VAL(CH),IO$_READVBLK,,,,%REF(MIN),
     5                 %VAL(4),,,,)
         IF (.NOT. IS)  CALL LIB$STOP(%VAL(IS))
      ENDDO
      RETURN
```

```
      ENTRY EXIT_MOVIE

      IS = SYS$DELPRC(,'READIT')
      RETURN

100   FORMAT (' SUBPROCESS NOT CREATED, IS=',Z8)
300   FORMAT (A)
500   FORMAT (' Error in Movie command(s)'/ I3,
     1        ' command line(s) ignored')
      END
```

```
      SUBROUTINE ALT_REC (FCOMM, IREC, ITEM, VALU, M)

C                                                        Alter item in record(s)
C     FCOMM = animation command file to be altered
C             (character variable)
C      IREC = integer array of record numbers to be altered
C      ITEM = integer array of item numbers to be altered
C             in the corresponding record number
C      VALU = real array of new field values
C         M = total number of records to be altered

C     The content of the ith ITEM in the ith IREC of the commands-data
C     file (FCOMM) will be replaced by the ith VALU.  An invalid record
C     or item number will cause that change to be ignored.  On exit the
C     records in the input file will have been overwritten, i.e. no new
C     version of the file is created
C
      CHARACTER*(*)  FCOMM
      DIMENSION IREC(1), ITEM(1), VALU(1)
      CHARACTER*50  TEXT, HOLD, VAL*12

C     Transfer all records from the input file into a working file

      IU1 = 21
      IU2 = 22
      OPEN(UNIT=IU2, FILE=FCOMM, TYPE='OLD')
      OPEN(UNIT=IU1, FILE='WORK.DUM', TYPE='NEW',
     1  ACCESS='DIRECT', RECL=50)
      IRT = 0
 10   READ(IU2,100,END=20) TEXT
      IRT = IRT+1
      WRITE(IU1'IRT) TEXT
      GO TO 10
 20   CLOSE(IU2)

      DO 30 I = 1,M
         IR = IREC(I)
         READ(IU1'IR)  TEXT
         IT = ITEM(I)
         J = 0
         HOLD = ' '
         IF (IT.NE 1) THEN
C
C     Count chars prior to the item to be changed
            DO L = 2,IT
               J = J+1
               DO WHILE (TEXT(J:J).EQ. ' ')
                  J = J+1
               ENDDO
               DO WHILE ((TEXT(J:J).NE. ' ').AND.(TEXT(J:J).NE. ',')
     1               .AND.(TEXT(J:J).NE. ';').AND.(TEXT(J:J).NE. '.'))
                  J = J+1
               ENDDO
            ENDDO
C
C     Save items, ahead of specified item, in holding location
            HOLD(1:J) = TEXT(1:J)
         ENDIF
C
C     Convert real value to character form and save
```

```
            ENCODE(12,200,VAL)  VALU(I)
            K = J+1
            JJ = J+12
            HOLD(K:JJ) = VAL(1:12)
            J = JJ+1
C
C       Locate the last char in the item
            DO WHILE ((K.LT.50).AND.(TEXT(K:K).EQ.' '))
               K = K+1
            ENDDO
            DO WHILE ((K.LT.50) AND.
     2             ((TEXT(K:K).NE.' ').AND.(TEXT(K:K).NE.','))
     3          .AND.(TEXT(K:K).NE.';').AND (TEXT(K:K).NE.':')))
               K = K+1
            ENDDO
C
C       Determine which chars are to be picked up from the record
C       & where they are to be placed in the new record   Save in
C       holding location then over-write record with new contents
            IF (J.GE.K) THEN
               JJ = 50
               KK = 50 - J + K
            ELSE
               KK = 50
               JJ = 50 - K + J
            ENDIF
            HOLD(J:JJ) = TEXT(K:KK)
            WRITE(IU1'IR)  HOLD
     30 CONTINUE
C
C       Overwrite all records of input file (i.e. same version)
C       then delete the working file.
        OPEN(UNIT=IU2, FILE=FCOMM, TYPE='OLD')
        DO 40 IR = 1,IRT
           READ(IU2,100) HOLD
           READ(IU1'IR) TEXT
           REWRITE(IU2,100) TEXT
     40 CONTINUE
        CLOSE(IU2)
        CLOSE(IU1,DISPOSE='DELETE')
        RETURN
    100 FORMAT(A50)
    200 FORMAT(E12.5)
        END
```

Appendix D

Utility Routines

WRITE_ANA_FILE

```
      PROGRAM WRITE_ANA_FILE
      CHARACTER TEXT*50, FNAM*31

C   Prepare the animation file (interactive commands and
C   required inputs) to be used with Movie  This program
C   adds the necessary blanks at the end of each line to
C   meet with the requirements of subr ALT_REC (50 char)

 10   WRITE(6,100)
      READ (6,200)  FNAM
      IF (FNAM(1:1).EQ.' ')  CALL EXIT
      OPEN(UNIT=10, NAME=FNAM, TYPE='NEW')
      WRITE(6,300)

 20   WRITE(6,400)
      READ (6,500) TEXT
      IF (TEXT(1:1).EQ.'*') THEN
          CLOSE(UNIT=10)
          GO TO 10
      ELSE
          WRITE(10,500)  TEXT
          GO TO 20
      ENDIF

100   FORMAT(//' Enter file name : '$)
200   FORMAT(A31)
300   FORMAT(/' Enter a command line following each prompt'
     1       /' Type  *  to indicate end of file'/)
400   FORMAT(' >> '$)
500   FORMAT(A50)
      END
```