Copyright 1986 by:

Canroad Transportation Research Corporation
1765 St. Laurent Blvd.
Ottawa, Canada K1C 3V4

ISBN: 0-919098-92-4
<table>
<thead>
<tr>
<th><strong>RTAC REPORT DOCUMENTATION FORM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project No</strong></td>
</tr>
<tr>
<td>Project Manager</td>
</tr>
<tr>
<td><strong>Title and Subtitle</strong></td>
</tr>
<tr>
<td><strong>Author(s)</strong></td>
</tr>
<tr>
<td>R. Gagne</td>
</tr>
<tr>
<td>T. Stock</td>
</tr>
<tr>
<td>National Research Council of Canada</td>
</tr>
<tr>
<td><strong>Sponsoring/Funding Agency and Address</strong></td>
</tr>
<tr>
<td>Canroad Transportation Research Corporation</td>
</tr>
<tr>
<td>1765 St. Laurent Blvd.</td>
</tr>
<tr>
<td>Ottawa, Canada K1G 3V4</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
</tr>
<tr>
<td>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 Telescript compatible graphics terminal.</td>
</tr>
<tr>
<td><strong>No. of Pages</strong></td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td><strong>Supplementary information</strong></td>
</tr>
</tbody>
</table>
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 Dimensions 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.
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. Hattin, 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. Saddlington, 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
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 computer 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 axis.

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 RESCN 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 (FGEOFM, FDISP, FFUN, FORC)

where

FGEOFM - geometry file defining the graphic
FDISP - displacement file (optional)
FFUN - function file (optional)
FORC - 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
VALUE - 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.CEO. (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 0 0  # - 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 fourteenth 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')

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 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 mode 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 straightforward 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 11: Truck 6x Forty Mountain Double, 45' Semi, 27' Trailer
Figure 1.3: Truck Bed 8-Axle. 27" Semi and Trailer, R-Dolly.
Appendix A

Modified MUVT, EYU Routines

DISPLA
MUL100
FLVU1
RENDF
VIEDRA
SE1X5
NUMDIS

Modifications indicated with
CC
in columns 1 and 2
PROGRAM DISPLAY

A GENERAL PURPOSE COMPUTER GRAPHICS DISPLAY PROGRAM FOR
POLYGONAL DATA WITH LINE DRAWING AND
CONTINUOUS-TONE PHOTOIMAGE OUTPUT

TO ASK QUESTIONS CONCERNING THIS PROGRAM OR TO REPORT BUGS,
CONTACT :

DR. BRUCE J. NAY
-
HANK CHRISTIANSEN
CIVIL ENGINEERING
368 CH
BYU
PROVO, UTAH 84602
(801) 378-2812

PLEASE TRY TO RESTRICT CALLS TO THE FOLLOWING HOURS:
MWF 11:00 am TO 1:00 pm (Mountain Time Zone)
TH 9:00 am TO 11:00 am

INTERACTIVE COMMANDS ARE

ALIAS = ENABLE ANTI-ALIASING OPTION
ANIMATE = SPECIFY ANIMATION SEQUENCE
CENTER = TRANSLATES ORIGIN TO CENTER OF MODEL
COLOR = SELECT COLORS FOR BACKGROUND, PARTS, AND FRINGES
CONTOUR = SELECT CONTOUR LEVELS
DASH = SPECIFY DOTTED LINE PARAMETERS
DEVICE = SET DISPLAY DEVICE
DIFFUSE = SET DIFFUSED LIGHT INTENSITY OF INDIVIDUAL PARTS
DISTANCE = SET DISTANCE FROM OBSERVER TO MODEL
DOTTED = ENABLE OR DISABLE DOTTED LINE OPTION
DRAW = SENDS PICTURE TO DISPLAY DEVICE (NO HIDDEN LINES)
EXIT = TERMINATE PROGRAM EXECUTION
EXPLODE = SPECIFY LOCAL MOTION (EXPLOSION)
FAST = INVOKES POOR MAN'S OPTION
FEATURE = SET FEATURE ANGLE
FIELD = SPECIFY FRUSTIUM OF VISION
FILE = ENABLE COMMAND INPUT FROM DISK FILE
FLAT = USE FLAT SHADING
FRINGE = SPECIFY COLOR FRINGES FOR DISPLACEMENT OR
SCALAR FUNCTION SYSTEM
CLASS = ENABLES TRANSPARENCY OPTION AND SETS PARAMETERS
HALT = ADD INTERACTIVE COMMAND INPUT FROM DISK FILE
HAZE = ENABLES HAZE/FOG OPTION AND SETS PARAMETERS
HELP = TYPE COMMANDS
IMMUNE = SET PARTS IMMUNE TO ROTATIONS
LIGHT = SETS CONTINUOUS TONE INTENSITY PARAMETERS
LINEAR = LINEAR INTERPOLATION BETWEEN DISPLACEMENT
OR SCALAR FUNCTION FILES
MULTIPLE = ENABLE OR DISABLE PREVIOUSLY DEFINED LIGHT SOURCES
NODE = ENABLE OR DISABLE NODE NUMBERING OPTION
NOPLOT = DISABLE HARDCOPY OUTPUT OPTION
ORIGIN = SPECIFY ORIGIN (PIVOT POINT) OF PARTS
PART = SPECIFY PARTS TO BE DISPLAYED
PAUSE = PAUSE BETWEEN COMMANDS WITH OPTION FOR
ABORTING THE "FILE" COMMAND OPTION
PIVOT = ROTATE MODEL ABOUT LOCAL AXES
POLY = ENABLE OR DISABLE POLYGON NUMBERING OPTION
PLOT = ENABLE HARDCOPY OUTPUT OPTION
READ = READ NEW DATA FILES
RESCENTER = RESTORE GEOMETRY AND RECENTRE THE MODEL
RECORD = SWITCH FOR SAVING PICTURES ON DISK FOR USE WITH "COMPOSE"
RESET = READS THE RECORDED VALUES OF VARIABLES FROM DISK
RESTORE = RESTORE GEOMETRY TO INITIAL CONDITION
ROTA T = ROTATE MODEL ABOUT GLOBAL AXES
SAVE = SAVES THE CURRENT VALUE OF PARAMETERS ON DISK
SCALE = SET SCALE FACTOR FOR DISPLACEMENT FUNCTIONS
SCOPE = SET SCOPE PARAMETERS
SHADOW = ENABLES SHADOW OPTION AND SETS PARAMETERS
SHIFT = MOVE VIEWING SCREEN IN X OR Y DIRECTION
SHRINK = SET SHRINK FACTOR
SMOOTH = USE SMOOTH SHADING
SUMMARY = GIVE MAXIMUM AND MINIMUM VALUES OF DATA FILES READ
TRANSLATE = TRANSLATE LOCAL ORIGIN OF MODEL
UNIFORM = USE UNIFORM SHADING
VIEW = DISPLAY SCENE ON PRECISION DISPLAY
WARP = SET SCALE FACTOR FOR SCALAR FUNCTIONS

******************************************************************

VARIABLE DIMENSION INFORMATION FOR "COMMAND.FOR" AND "HIDDEN.FOR"

(1) ICNMAX = MAX. NO. OF ELEM.(NP MAX)*MAX. NO. OF SIDES(NSMAX)
  DIMENSION ICNMAX
(2) NFRINM = MAXIMUM NUMBER OF FRINGES
  DIMENSION CFRIN(3,NFRINM)
(3) NJMAX = MAXIMUM NUMBER OF NODES
  DIMENSION SCORD(3,NJMAX),SPEC(NJMAX),SPECI(NJMAX)
  1.U(3,NJMAX),X(3,NJMAX),XNORM(3,NJMAX),YN(3,NJMAX)
  2.YNORM(3,NJMAX)
(4) NP MAX = MAXIMUM NUMBER OF PARTS
  DIMENSION DA(3,NP MAX),DC(3,NP MAX+1),DD(3,NP MAX)
  1.DIF(NP MAX),DI RCA(NP MAX),FRING(2,NP MAX),ICOL(NP MAX)
  2.JSMOOTH(NP MAX),LASP(NP MAX+1),NFR(NP MAX),NHIGH(NP MAX)
  3.NCRROV(NP MAX),NPL(2,NP MAX),NPLS(NP MAX),POOR(NP MAX)
  4.RDG(3,NP MAX),X10(NP MAX),XNH(NP MAX),XNR(NP MAX)
  5.XX(3,NP MAX)
  6.ISHFLG(NP MAX),JSFHLG(NP MAX),XMA T(NP MAX)
  7.XMINT(NP MAX),TPWR(NP MAX)
(5) NPTMAX = MAXIMUM NUMBER OF ELEMENTS(POLYGONS)
  DIMENSION NCE NPT(NP MAX)
(6) NSMAX = MAXIMUM NUMBER OF SIDES OF POLYGONS
  DIMENSION CONT(NSMAX+1),JCOL(NSMAX),NITRI(NSMAX+1)
  1.NN(NSMAX+1),NNN(NSMAX),XX(3,(NSMAX+1))
  2.XP(3,(NSMAX+1)),XQ(3,(NSMAX+1)),AX(NSMAX)
  3.AY(NSMAX)
(7) MAXFRE = MAXIMUM SIZE OF FREE STORAGE
  DIMENSION IFREE(MAXFRE),ISEQ(MAXFRE),RSEQ(MAXFRE)
(8) MAXCNT = (2*NSMAX) AS A MINIMUM IF NO CLIPPING T AKE S PLACE,
  SO RECOMMEND APPROXIMATELY (4*NSMAX)
  DIMENSION VX(MAXCNT),VY(MAXCNT),VZ(MAXCNT),VN(MAXCNT)
  1.IC(MAXCNT),VC(MAXCNT),VIX(MAXCNT),VIT(MAXCNT)
  2.VT2(MAXCNT),VTN(MAXCNT),VT(C(MAXCNT),ITC(MAXCNT)
(9) MAXINS = 7*NUMBER OF INTERSECTION LINES
DIMENSION INT(MAXINS),RNT(MAXINS)
(10) MAXFSL = MAXIMUM LENGTH OF FREE STORAGE LIST
DIMENSION LIST(MAXFSL)
(11) MLSN = MAXIMUM NUMBER OF LIGHT SOURCES
DIMENSION SHAPE(MLSN),SHINT(MLSN),LSENAB(MLSN)
1,LSPEC(MLSN),DSCURD(MLSN),NPNT(MLSN),NSHMAX
(12) NTMAX = MAXIMUM NUMBER OF TRANSPARENT LAYERS
ZZL(NTMAX),ZZR(NTMAX)

************************************************************
MAIN PROGRAM - PROCESSES INTERACTIVE COMMANDS FROM THE USER
AND CALLS APPROPRIATE SUBROUTINES
************************************************************

SUBPROGRAMS CALLED
ALIA = ENABLE THE ANTI-ALIAS FEATURE
ANIMAT = SELECT INCREMENTAL TRANSLATION, ROTATION, ETC.
CLEAR = CLEARS ARRAYS (INITIALIZATION)
COLO = SPECIFY COLORS FOR VARIOUS PARTS
CONT = SELECT CONTOUR OPTION AND SET CONTOUR LEVELS
DASHLN = ENABLE THE DOTTED LINE FEATURE
DIFF = SET DIFFUSED LIGHT INTENSITY BY PART
DIST = SPECIFY DISTANCE TO COORDINATE ORIGIN FROM OBSERVER
DSET = CHANGES DOTTED LINE PARAMETERS
EXIT = RETURNS CONTROL TO MONITOR
EXPL = EXPLOSION OF PARTS
FAST = SET DATA OPTIONS AND POOR MANS HIDDEN SURFACE REMOVAL
FEAT = SELECT FEATURE OPTION AND SET FEATURE ANGLE
FIEL = SPECIFY FRUSTRUM OF VISION AND MIN. AND MAX. X-Y CLIPPING PLANES
FILE_DP = ENABLES COMMAND INPUT FROM DISK FILE
FRIN = SELECT FRINGE OPTION AND SPECIFY FRINGED PARTS
GET_ORG = SPECIFY ORIGIN (PIVOT POINT) OF PARTS
GLAS = ENABLE TRANSPARENCY FEATURE AND ACCEPTS PARAMETERS
HALT_DP = RESTORE COMMAND INPUT TO SYS*INPUT, CLOSE FILE
HAZE = ENABLE HAZE/FOG FEATURE AND ACCEPTS PARAMETERS
HELP = GIVE AVAILABLE COMMANDS OR OPTIONS
IMMUNE = MAKES PARTS IMMUNE TO ROTATIONS
INIT = INITIALIZES NECESSARY PARAMETERS
LIGHT = SET CONTINUOUS TONE INTENSITY PARAMETER
LINE = ADD PREVIOUS DISPLACEMENTS AND SCALAR FUNCTIONS TO ARRAYS, READ NEW ARRAYS AND DIFFERENCE FOR TRANSIENT DATA
MULT = ENABLE AND DISABLE PREVIOUSLY DEFINED LIGHT SOURCES
NODPLOT = DISABLES HARDCOPY OUTPUT OPTION
PAUS_DP = PROVIDES PAUSE BETWEEN COMMANDS (USE WITH "FILE")
PIVOT = SET LOCAL ROTATION ABOUT RELATIVE ORIGIN
PLOT_DP = ENABLES HARDCOPY OUTPUT OPTION
POLNOD = ENABLE OR DISABLE NODE AND/OR POLYGON NUMBERING
READ = READS IN DATA FILES
READIN = ACCEPTS INPUT FROM USER
RECORD = ENABLES SWITCH FOR SAVING PICTURES ON DISK FOR USE WITH "COMPOSE"
RESET = READS THERecorded VALUE OF VARIABLES FROM DISK
REST = RESTORE MODEL TO ORIGINAL COORDINATE SYSTEM
(KILLS ROTATIONS AND TRANSLATIONS)
_DLAO: USER1. PTS. MOVIM. MOVEU. FOR J

RES_CEN = RESTORES GEOMETRY AND RE-CENTRES THE MODEL
ROTA = ROTATE MODEL ABOUT ORIGIN
SAVE = WRITES THE CURRENT VALUE OF VARIABLES AND LOGICALS ON
       TO DISK FOR USE WITH RESET
SCAL = SPECIFY DISPLACEMENT SCALE FACTOR
SCOP = SET SCOPE PARAMETERS
SEPART = SELECT CONTENT OF SCENE
SHADE = SELECT FLAT, SMOOTH, OR UNIFORM SHADING
SHADD = ENABLE SHADOW FEATURE AND ACCEPTS PARAMETERS
SHKR = SELECT SHRINK OPTION AND SPECIFY SHRINK FACTOR
SUMCEN = GIVE SUMMARY OF DATA READ WITH MIN./MAX. VALUES,
         OR CENTER MODEL IN VIEWING AREA
TRAN = TRANSLATE COORDINATE ORIGIN OF MODEL
VIEDRA = CALLS FOR NORMALS, LIGHT INTENSITY, ETC. NEEDED TO
         DISPLAY SCENE (EITHER VIEW OR DRAW)
WARP = SPECIFY OUT-OF-PLANE WARping SCALE FACTOR

==========================================================================

VARIABLES USED

APLOT = ENABLES ONE HARDCOPY OUTPUT OF DISPLAY
HRDCPY = LOGICAL FLAG INDICATING 'PLOT' IS ON (OR OFF)
IBAUD = TRANSMISSION RATE IN CHARACTERS/SECOND (BAUD/10)
IBUF = BUFFER ARRAY FOR MULTIPLE COMMANDS PER LINE
ICMD = INTERACTIVE COMMAND WORD STARTING LOCATION
ICODE = INITIALIZATION PARAMETER =1 FIRST TIME ONLY
IEND = NUMBER OF COMMANDS ACCEPTED ON A LINE
IPOINT = POINTER IN IBUF ARRAY
JUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE) - OUTPUT
K1 = NUMBER OF KEYWORDS FOUND
KEY = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
      ARRAY
KUNIT = DEVICE LOGICAL UNIT NUMBER FOR SYS*INPUT
LIT = COMMAND NUMBER
LUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE)
MUNIT = DEVICE LOGICAL UNIT NUMBER FOR SYS*OUTPUT
NPOL = NUMBER OF PARTS IN PREVIOUS FILE READ
WORD = ARRAY OF ACCEPTABLE COMMANDS FOR THIS ROUTINE

==========================================================================

VARIABLE DIMENSION INFORMATION FOR MAIN PROGRAM

(SET THE NINE FOLLOWING MAXIMUMS IN THE DATA STATEMENT BELOW)
(1) ICNMAX = MAX. NO. OF ELEM. (NPTMAX)*MAX. NO. OF SIDES(NSMAX)
(2) NFRIMM = MAXIMUM NUMBER OF FRINGES
(3) NMAX = MAXIMUM NUMBER OF NODES
(4) NPTMAX = MAXIMUM NUMBER OF PARTS
(5) NSMAX = MAXIMUM NUMBER OF SIDES OF POLYGONS
(6) NMAX = MAXIMUM NUMBER OF SIDES OF POLYGONS
(10) MAXFSL = MAXIMUM LENGTH OF FREE STORAGE LIST
(11) MLSN = MAXIMUM NUMBER OF LIGHT SOURCES
(12) NMAX = MAXIMUM NUMBER OF TRANSPARENCY LAYERS

==========================================================================

COMMON/BAUD/IBAUD
COMMON/CHFS/ MAXFSL
COMMON/CMLEN/ MLSN
COMMON/DEVOL/ INPUT,OUTPUT
COMMON/DEVIL/ MUNIT, JUNIT, KUNIT, LUNIT

COMMON/ENTER/ N1, N2, KEY, XNUM, K1, K2
COMMON/FRI/ NFRINM, FJRIN, GFRIN, BFRIN, RFRIN

COMMON/HCOPY/ HRDCPY, APLDT, HCEND

IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
COMMON/INTER/ ABORT

COMMON/MAXN/ NPOL, NSMAX, NJMAX, NPMAK, ICNMAX, NPTMAX
COMMON/NTCOM/ NTMAX
COMMON/PRO/ NP, NJ, ICMD, IC, IREAD, SKALE, NPT

COMMON/REST3/ PORG, SORG, IPREV

DIMENSION WORD(212), KEY(10), XNUM(40), IDUF(10)
DIMENSION BFRIN(5), GFRIN(5), RFRIN(5)

DIMENSION PORO(3,27), SORG(3,27), IPREV(27)

LOGICAL HRDCPY, APLDT, HCEND

LOGICAL FJRIN

IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
LOGICAL ABORT

INTEGER OUTPUT
CHARACTER*1 WORD

D 'S', 'A', 'V', 'E', 'E',

DATA MUST BE CHANGED IF MAXIMUM DIMENSIONING IS ALTERED

NSMAX=8
NJMAX=1007
NPMAK=27
ICNMAX=4028
NPTMAX=1007
NFRINM=11
MAXFSL=2014
NL5N=4
NTMAX=23
INPUT, OUTPUT, and baud rate are set for the VAX running VMS

Open units in case Movie is being run as a subprocess

OPEN(UNIT=5, NAME='SYS$INPUT', TYPE='OLD')
OPEN(UNIT=2, NAME='SYS$OUTPUT', TYPE='OLD')
OPEN(UNIT=6, NAME='SYS$ERROR', TYPE='OLD')

Assign I/O device units, initial plot variables

INPUT=5
OUTPUT=2
KUNIT = INPUT
LUNIT = 48
MUNIT = OUTPUT
JUNIT = 47
APLOT = .FALSE.
HRDCPY = .FALSE.

IBAUD=240
WRITE(OUTPUT, 500)
NPOL=NPMAX
CALL CLEAR
CALL READF
CALL INIT

IF on a VAX - to implement CTRL-C trap uncomment next 2 statements

CALL TRAP_INIT
ABORT = .FALSE.

READ INPUT command string for processing

WRITE(OUTPUT, 510)
IPOINT=0
ICMD=0
NJ=1
NZ=0

Change dimension of WORD array

NW=212
CALL READIN(WORD, NW)
KEND=K1
IF(K1 EQ 0) GO TO 250
DO 20 I=1,K1
20 IBUF(I)=KEY(I)

IF on a VAX - to implement CTRL-C trap uncomment the next five statements and comment the 30 CONTINUE statement

30 IF (ABORT) THEN
   IF (KEYBD) WRITE(OUTPUT, 520)
   ABORT = .FALSE
   GO TO 10
ENDIF

30 CONTINUE
IPOINT=IPOINT+1
IF(IPOINT.GT.IEND) GO TO 10
ICMD=1BUF(IPOINT)
LIT=(ICMD+3)/4

C
LIT 1 ="READ" LIT 13 ="DIST" LIT 25 ="FAST" LIT 37 ="DASH"
LIT 2 ="EXIT" LIT 14 ="FIEL" LIT 26 ="PIVO" LIT 38 ="DOTT"
LIT 3 ="ROTA" LIT 15 ="PART" LIT 27 ="HELP" LIT 39 ="RECO"
LIT 4 ="REST" LIT 16 ="EXPL" LIT 28 ="LINE" LIT 40 ="ALIA"
LIT 5 ="TRAN" LIT 17 ="SUMM" LIT 29 ="CONT" LIT 41 ="GLAS"
LIT 6 ="SCAL" LIT 18 ="CENT" LIT 30 ="SHRI" LIT 42 ="SHAD"
LIT 7 ="FLAT" LIT 19 ="WARP" LIT 31 ="LIGH" LIT 43 ="HAZE"
LIT 8 ="SMOD" LIT 20 ="FRIN" LIT 32 ="FEAT" LIT 44 ="MULT"
LIT 9 ="UNIF" LIT 21 ="COLO" LIT 33 ="NODE" LIT 45 ="RESE"
LIT 10 ="SCOP" LIT 22 ="ANIM" LIT 34 ="POLY" LIT 46 ="SAVE"
LIT 11 ="DEVI" LIT 23 ="VIEW" LIT 35 ="IMMU"
LIT 12 ="DIFF" LIT 24 ="DRAW" LIT 36 ="SHIF"

GO TO (40, 50, 60, 70, 80, 90, 100, 100, 100, 110, 120, 130
1, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240
2, 250, 260, 270, 280, 290, 300, 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
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 FIEM
GO TO 30
150 ICODE=0
CALL SEPART(ICODE)
GO TO 30
160 CALL EXPL
GO TO 30
170 IPERZ=0
CALL SUMCEN(IPHERZ)
GO TO 30
180 CALL WARP
GO TO 30
190 CALL FRIN
GO TO 30
200 CALL COLDO
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 CLAS
   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’)"
    SEE POSSIBLE INCOMPATIBILITIES (2)
510 FORMAT(’>> ’)
    IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT
520 FORMAT(’<OPERATION ABORTED’)
C
END
SUBROUTINE MULTDD(II)

**MULTDD** - MULTIPLIES COORDINATES BY LOCAL ROTATION TRANSFORMATION MATRIX

SUBROUTINE CALLED BY

POINTS = GETS COORDINATES OF NODES FOR A POLYGON

**VARIABLES USED**

DD = TRANSFORMATION MATRIX
IPART = PART NUMBER
PORG = ARRAY OF PIVOTAL POINTS CURRENT POSITION
ROI = RELATIVE ORIGIN BY PART
XP = COORDINATE ARRAY FOR POLYGON
YY = INCREMENTAL TRANSLATION ARRAY

**VARIABLE DIMENSION INFORMATION FOR SUBROUTINE MULTDD**

(4) NPMAX = MAXIMUM NUMBER OF PARTS
DIMENSION DD(3,3,NPMAX),PORG(3,NPMAX)

(6) NSMAX = MAXIMUM NUMBER OF SIDES OF POLYGONS
DIMENSION XP(3,(NSMAX+1))

**COMMON/COORD/ XP**
COMMON/FYSL/FUN,YY,SLINR
COMMON/NEME/ NEDGE, MEDGE, IPART
COMMON/REST1/ ROI, DD

Addition for change in "PIVO" option

COMMON/REST3/ PORG, SORG, IPREV
DIMENSION PORG(3,27), SORG(3,27), IPREV(27)

DIMENSION DD(3,3,27), RORG(3,27)
DIMENSION XP(3,9)
DIMENSION YY(3), FUN(3)

X1=XP(1,II)-PORG(1,IPART)+YY(1)
X2=XP(2,II)-PORG(2,IPART)+YY(2)
X3=XP(3,II)-PORG(3,IPART)+YY(3)
XP(1,II)=DD(1,1,IPART)*X1+DD(2,1,IPART)*X2+DD(3,1,IPART)*X3
  + PORG(1,IPART)-YY(1)
XP(2,II)=DD(1,2,IPART)*X1+DD(2,2,IPART)*X2+DD(3,2,IPART)*X3
  + PORG(2,IPART)-YY(2)
XP(3,II)=DD(1,3,IPART)*X1+DD(2,3,IPART)*X2+DD(3,3,IPART)*X3
  + PORG(3,IPART)-YY(3)
RETURN
END
SUBROUTINE PIVOT

SUBROUTINE PIVOT - SET LOCAL ROTATION ABOUT RELATIVE ORIGIN

SUBROUTINE CALLED BY

MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND CALLS APPROPRIATE SUBROUTINES

SUBPROGRAMS CALLED

READIN = ACCEPTS INPUT FROM THE USER

ROTAT = CALCULATES GLOBAL ROTATION TRANSFORMATION MATRIX

VARIABLES USED

DD = LOCAL TRANSFORMATION MATRICES BY PART

IA = AXIS NUMBER

KEY = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD ARRAY

N1 = INPUT FLAG, =1 WHEN KEY WORDS SOUGHT =0 WHEN KEY WORDS NOT SOUGHT

N2 = INPUT FLAG, =1 WHEN NUMBERS WANTED =0 WHEN NUMBERS NOT WANTED

NW = DIMENSION OF WORD ARRAY

RORC = RELATIVE ORIGIN ARRAY FOR LOCAL ROTATIONS

WORD = ARRAY OF ACCEPTABLE KEY WORDS

XNUM = ARRAY OF REAL NUMBERS ACCEPTED

VARIABLE DIMENSION INFORMATION FOR SUBROUTINE PIVOT

(4) NPMAX = MAXIMUM NUMBER OF PARTS

DIMENSION DD(3,3,NPMAX), RORC(3,NPMAX)

COMMON/DEVI/ INPUT, OUTPUT

COMMON/ENTER/N1, N2, KEY, XNUM, K1, K2

COMMON/REST1/ RORC, DD

DIMENSION KEY(10), XNUM(40), WORD(12)

DIMENSION DD(3,3,27), RORC(3,27)

INTEGER OUTPUT

CHARACTER*1 WORD


NW=12

WRITE(OUTPUT, 80)

10 WRITE(OUTPUT, 80)

N1=1

N2=1

CALL READIN(WORD, NW)

I1=XNUM(1)

I2=XNUM(2)
\texttt{x2=xnum(3)}
\texttt{iA=KEY(1)}

\texttt{iA=1 ="X"}
\texttt{iA=2 ="Y"}
\texttt{iA=3 ="Z"}

\texttt{i3=(iA+3)/4}
\texttt{if (i1.eq.0) return}
\texttt{do 20 i=i1,i2}
\texttt{isafe=i}
\texttt{20 call rotat(ii,i3,x2,isafe)}
\texttt{go to 10}

\texttt{60 format(’ <parts ii/i2, axis, angle>’)}
\texttt{see possible incompatibilities (2)}

\texttt{80 format(’ &gt;&gt; ’)}
\texttt{end}
**SUBROUTINE READF**

**SUBROUTINE READF - READS IN DATA FILES**

**SUBROUTINE CALLED BY**

- **MAIN** = PROCESSES INTERACTIVE COMMANDS FROM THE USER
- **AND CALLS APPROPRIATE SUBROUTINES**

**SUBPROGRAMS CALLED**

- **CLSFL** = CLOSES FILE IUNIT
- **LASTC** = FLAGS CONNECTIVITY ARRAY FOR PART LIMITS AND DETERMINES IF POLYGON CENTER POINTS SPECIFIED
- **OPNFIL** = REQUESTS FILENAME AND OPENS FILE FOR I/O
- **SEPART** = SELECTS CONTENT OF SCENE

**VARIABLES USED**

- **BFRIN** = BLUE FRINGE ARRAY
- **CFRIN** = RED, BLUE, GREEN FRINGE INTENSITY BY FRINGE NUMBER
- **ICNMAX** = MAXIMUM NUMBER OF ELEMENTS TIMES MAXIMUM NUMBER OF SIDES OF POLYGONS (NPTMAX * NSMAX), OR SUITABLE REDUCTION IF MOSTLY LOWER ORDER ELEMENTS USED
- **DIF** = DIFFUSED LIGHT ARRAY BY PARTS
- **CFRIN** = GREEN FRINGE ARRAY
- **IBAD** = ERROR PARAMETER
- **ICODE** = INITIALIZATION PARAMETER, 1 FIRST TIME ONLY
- **ICOL** = RED, BLUE, GREEN INTENSITY BY PARTS
- **IERROR** = 1 ON SUCCESSFUL COMPLETION
- **= 0** ON EMPTY FILE SPECIFICATION
- **= -1** ON FAILURE
- **IP** = TOTAL CONNECTIVITY ARRAY
- **IPB** = BACKGROUND COLOR
- **IREAD** = 1 FOR INPUT FILE
- **= -1** FOR OUTPUT FILE
- **ISPEC** = IF .TRUE., SCALAR FUNCTION FILE INCLUDED
- **IUNIT** = DEVICE LOGICAL NUMBER
- **NCON** = LENGTH OF IP ARRAY
- **NJ** = NUMBER OF JOINTS OR NODES
- **NUMAX** = MAXIMUM NUMBER OF NODES
- **NP** = NUMBER OF PARTS
- **NPFL** = PARTS ARRAY
- **NPMAX** = MAXIMUM NUMBER OF PARTS
- **NPOL** = NUMBER OF PARTS IN PREVIOUS FILE READ
- **NPI** = NUMBER OF ELEMENTS (POLYGONS)
- **NPTMAX** = MAXIMUM NUMBER OF ELEMENTS (POLYGONS)
- **NTR** = IF .TRUE., INTERIOR POLYGON
- **RFIN** = RED FRINGE ARRAY
- **SPEC** = SCALAR FUNCTION ARRAY
- **U** = DISPLACEMENT ARRAY
- **X** = TOTAL COORDINATE ARRAY
C VARIABLE DIMENSION INFORMATION FOR SUBROUTINE READF
C
C   (1) ICNMAX = MAX. NO. OF ELEM. (NPTMAX) MAX. NO. OF SIDES (NSMAX)
C       DIMENSION IP(ICNMAX)
C   (2) NFRIM = MAXIMUM NUMBER OF FRINGES
C       DIMENSION CFRIN(3, NFRIM)
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
C COMMON/CIP/  IP
C COMMON/CLIP3/KB, YB, ZB, KB, CB, XE, YE, ZE, KE, CE, LAS, ISHARE, NTR
C            1, ITR, ITR2
C COMMON/DEVI/ INPUT, OUTPUT
C COMMON/FRI/  NFRINM, JFRING, GFRIN, BFRIN, RFRIN
C COMMON/INTENS/ IPH, IPL, IPB, IFX
C COMMON/LEF/  DIF, ICOL
C COMMON/LOGI/ ISPOTH, IPOOR, IMIX, DIRC, ISPEC, ISMA, LINEAR, I1 IHLR, IFRING
C COMMON/MAXN/ NPOL, NSMAX, NJMAX, NPMAX, ICNMAX, NPTMAX
C COMMON/PAAR/ NPL, NPLS
C COMMON/PRO/NP, NJ, ICMD, IC, IREAD, SKALE, NPT
C COMMON/GFR1O/CONT, IDV, IEAD, SHOSHR, LBLSPC
C COMMON/SPUX/ SPEC, SPEC1, U, X
C COMMON/VCOL/ NFRING, CFRIN
CC
CC Addition for change in "PIVOT" option
CC
C COMMON/REST3/ PORG, SORG, IPREV
C DIMENSION PORG(3,27), SORG(3,27), IPREV(27)
CC
C DIMENSION IP(4028)
C DIMENSION CFRIN(3,11)
C DIMENSION SPEC(1007), SPEC1(1007), U(3, 1007), X(3, 1007)
C DIMENSION DIF(27), ICOL(27), NPL(2, 27), NPLS(27)
C DIMENSION BFRIN(5), GFRIN(5), RFRIN(5)
C LOGICAL ISPEC, IBA, NTR, ISHARE, LAS, JFRING, DIRC, IFRING, IHLR
C LOGICAL IMIX, IPOOR, ISMA, LINEAR, CONT, SHOSHR
C INTEGER OUTPUT
C IREAD = 1
C BFRIN(1)=1
C BFRIN(2)=1.
C BFRIN(3)=0.
C BFRIN(4)=0.
C BFRIN(5)=0.
C GFRIN(1)=0.
C GFRIN(2)=1.
C GFRIN(3)=1.
C GFRIN(4)=1.
C GFRIN(5)=0
C RFRIN(1)=0.
C RFRIN(2)=0.
C RFRIN(3)=0.
C RFRIN(4)=0.
C RFRIN(5)=1.
C
READ GEOMETRY FILE

10 CALL OEPNFI('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. NPT, AND. NP, LE. NPMAX) WRITE(OUTPUT, 260)
   NPTM=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. NPTM) WRITE(OUTPUT, 240) NPT, NPTM
   IF(NCON, GT. NCONM) WRITE(OUTPUT, 270) NCON, NCONM
   IF(NP, GT. NPMAX, OR. NJ, GT. NJMAX, OR. NPT, GT. NPTM) STOP
   IF(NCON, GT. NCONM) 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, 250) (IP(I), I=1, NCON)
   WRITE(OUTPUT, 250) NP, NJ, NPT
   STOP

READ DISPLACEMENT FILE

50 CALL CLSFIL(IUNIT)
60 IF(NP, EQ. 0) GO TO 10
   CALL OEPNFI('DISP', IUNIT, IREAD, IERROR)
   IF(IERROR) 60, 80, 70
70 READ(IUNIT, 210) ((U(I, J), I=1, 3), J=1, NJ)

READ SPECIAL FUNCTION FILE

CALL CLSFIL(IUNIT)
80 CONTINUE
90 CALL OEPNFI('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
   IBADEL=. FALSE.
   NI=. FALSE.
   DO 120 I=1, NP
      DIF(I)=0.15
120 CONTINUE
   ICOL(I)=1077215
   IPB=0
   DO 130 I=1, 5
      CFRIN(I, I)=CFRIN(I)
      CFRIN(I, I)=RFVIN(I)
   130 CFRIN(I, I)=RFVIN(I)
   ICODE=1
   CALL SEPAR1(ICODE)

READ RELATIVE ORIGIN FILE for change in "PIVOT" option

DO 140 J = 1, NP
   IPREV(J) = 0
   DO 140 I = 1, 3
140 DO 140
SORG(I, J) = 0.0
140 CONTINUE
150 CALL OPNFL( '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, ',
   I15, ' elements. >' )
260 FORMAT( ' <WARNING: new file has more parts than old file>' )
270 FORMAT( ' <CONN. LENGTH(', I7, ' ) exceeds max. of(', I7, ', ') >' )
280 FORMAT( ' <ERROR: data format problems>' )
END
**SUBROUTINE VIEDRA**

- Calls for normals, light intensity, etc.
- Needed to display scene (either view or draw)
- Also increments displacements, rotations, translations, etc. for animation

**SUBROUTINE CALLED BY**

- **MAIN** = Processes interactive commands from the user and calls appropriate subroutines

**SUBPROGRAMS CALLED**

- **AAALIAS** = Buffers segment information for anti-aliasing
- **BDNFRM** = Performs frame initialization procedures
- **CRORE** = Draws global coordinate axes
- **ENDFRM** = Ends frame, returns to alpha-numeric mode
- **FRNBAR** = Draws fringe bar when bar is .TRUE.
- **HIDDEN** = Determines visible segments
- **INTHID** = Initializes the hidden process
- **MOD_ONC** = Computes current position of pivotal points
- **NUMON** = Sends visible node coordinates on
- **OPNFILE** = Requests filename and opens file for I/O
- **PART** = Processes polygons by part, subdivides warped polygons on edge
- **POLNUM** = Sends center coordinates of visible polygons on
- **READIN** = Accepts input from the user
- **ROTAT** = Calculates rotation transformation matrix
- **SHINIT** = Initializes parameters for shadows

**VARIABLES USED**

- **ANISHR** = Delta shrink factor for animate option
- **ANTIAL** = If .TRUE., anti-aliasing enabled
- **BACKZ** = General back z perspective plane
- **BAR** = If .TRUE., then fringe bar is drawn
- **CHI** = Highest scalar function value
- **CLO** = Lowest scalar function value
- **CMD** = Interactive command word
- **CONTNS** = If .TRUE., contours exist
- **CPF** = Vibrations/frame
- **DA** = Local rotations array by parts
- **DAMP** = Damping factor for smooth animation
- **DC** = Global transformation matrix
- **DD** = Local transformation matrices by part
- **DDELTA** = Position scale factor in animated sequence
- **DDOZ** = Change in distance to origin in animated sequence
- **DELTA** = Local motion scale factor
- **DIRC** = .TRUE. for clockwise orientation of polygons
  = .FALSE. for counterclockwise orientation of polygons
- **DIRCA** = Data direction by parts array
  = 0.0 for clockwise
  = 1.0 for counterclockwise
- **DOZ** = Distance to origin from observer
DR = TOTAL ROTATION IN ANIMATED SEQUENCE
DT = TOTAL TRANSLATION IN ANIMATED SEQUENCE
FRONTZ = GENERAL FRONT Z PERSPECTIVE PLANE
IBAD = ERROR PARAMETER
IC = 1 FOR COLOR, 2 FOR BLACK AND WHITE
IDVICE = DISPLAY DEVICE NUMBER
IFACT = FACTOR USED TO PACK DATA
IFR1 = FIRST SCENE IN SEQUENCE SENT TO DISPLAY
IFR2 = LAST SCENE IN SEQUENCE SENT TO DISPLAY
IFRING = .TRUE. FOR FRINGES, .FALSE. FOR NO FRINGES
IHLINE = HIDDEN LINE REMOVAL (LOGICAL VARIABLE)
IPART = PART NUMBER
IPERZ = FLAG, USED FOR Z PERSPECTIVE PURPOSES
IPH = .TRUE. DISPLAYS ALL PICTURES IN SEQUENCE
     .FALSE. MODIFIES GEOMETRY BUT DOES NOT DISPLAY
IPOOR = IF .TRUE. INVIKES POOR MANS HIDDEN LINE/SURFACE
     REMOVAL
ISCLIP = 0 FOR NON-SHADOW POLYGONS, = 1 FOR SHADOW POLYGONS
ISET = 0 INITIALLY FOR A FRAME, THEREAFTER = 1
ISHARE = EDGE SHARING
ISMA = IF .TRUE., SMOOTH ANIMATION
ISMOOTH = -1 FOR SMOOTH SHADING
       0 FOR FLAT SHADING
       1 FOR UNIFORM SHADING
ITSHDN = FLAG USED TO TURN OFF SHADOWS TEMPORARY IF IN LINE
       DRAWING MODE
JCONT = 0 IF LIGHT HAS NOT BEEN SPECIFIED, = 1 OTHERWISE
JLAST = LOCATION OF LAST NODE OF PREVIOUS ELEMENT IN TOTAL
       CONNECTIVITY ARRAY
JSMOOTH = SMOOTH BY PARTS ARRAY
       -1 FOR SMOOTH
       0 FOR FLAT
       1 FOR UNIFORM
K1 = NUMBER OF KEY WORDS FOUND
K2 = NUMBER OF NUMBERS FOUND
KEY = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
       ARRAY
LINEAR = TRANSIENT DATA (LOGICAL VARIABLE)
N1 = INPUT FLAG, =1 WHEN KEY WORDS SOUGHT
    =0 WHEN KEY WORDS NOT SOUGHT
N2 = INPUT FLAG, =1 WHEN NUMBERS WANTED
    =0 WHEN NUMBERS NOT WANTED
NFRAME = NUMBER OF FRAMES TO BE GENERATED
NJ = NUMBER OF JOINTS OR NODES
NNUM = .TRUE. IF NODE NUMBERING ENABLED
NONROT = .TRUE. PART IS IMMUNE TO GLOBAL ROTATIONS
       FALSE. PART RotateS
NP = NUMBER OF PARTS
NPL = PARTS ARRAY
NPLS = DISPLAY PARTS ARRAY
       1 TO DISPLAY
       0 DO NOT DISPLAY
NTR = IF .TRUE., INTERIOR POLYGON
NUM = .TRUE. IF EITHER NODE OR POLYGON NUMBERING ENABLED
NW = DIMENSION OF WORD ARRAY
PNUM = .TRUE. IF POLYGON NUMBERING ENABLED
POOR = POOR MANS BY PART ARRAY
       0.0 FOR NO POOR MANS
       1.0 FOR POOR MANS
SFDEL = DISPLACEMENT SCALE FACTOR IN ANIMATED SEQUENCE
SHSHR = EDGE SHARE PARAMETER (LOGICAL VARIABLE)
SHRK = SHRINK FACTOR
SKALE = DISPLACEMENT SCALE FACTOR
SLNRM = TRANSIENT DATA INCREMENT FACTOR
SHDO = IF .TRUE. SHADOWS ENABLED
SPEC = SCALAR FUNCTION ARRAY
SPECI = SECOND SCALAR FUNCTION ARRAY IN LINEAR OPTION
WORD = ARRAY OF ACCEPTABLE KEY WORDS
WR = .TRUE. IF RECORD IS ON
XNORM = NORMALS ARRAY FOR SMOOTH SHADING WITH CONTINUOUS
        TONE OUTPUT, OR SCREEN COORDINATES AND VISIBLE FLAG
        FOR NODE OR POLYODON NUMBERING IN LINE DRAWING MODE
XNUM = ARRAY OF REAL NUMBERS ACCEPTED
XO = TRANSLATION ARRAY
ZB = EYE BACK Z PERSPECTIVE PLANE
ZEF = EYE FRONT Z PERSPECTIVE PLANE
ZFB = ZEB TIMES TANAL
ZPF = ZEF TIMES TANAL
ZSPRED = Z PERSPECTIVE MAPPING FACTOR

******************************************************************************

VARIABLE DIMENSION INFORMATION FOR SUBROUTINE VIEDRA
(3) NJMAX = MAXIMUM NUMBER OF NODES
         DIMENSION SPEC(NJMAX), SPEC1(NJMAX), U(3, NJMAX)
         1, X(3, NJMAX), XNORM(3, NJMAX)
(4) NPMAX = MAXIMUM NUMBER OF PART
(11) MLSN = MAXIMUM NUMBER OF LIGHT SOURCES
         DIMENSION DA(3, NPMAX), DC(3, 3, NPMAX+1), DD(3, 3, NPMAX)
         1, DIRCA(NPMAX), JSMOTH(NPMAX), NPL(2, NPMAX)
         2, NPLS(NPMAX), POOR(NPMAX), RORG(3, NPMAX), NONRDT(NPMAX)
         1, XNHR(NPMAX, MLSN), XCH(NPMAX, MLSN), XNH(NPMAX, MLSN)
         1, XNRR(NPMAX, MLSN), XLSI(MLSN), XL(3, MLSN), INF(MLSN)

******************************************************************************

COMMON/ANI/ DA, DT, DR, CPF, IFRI, IFR2, DDOZ, DDELT, IFRM, NFRAME,
  1  SFDEL, ANISHR
COMMON/CANTI/ANTIAL
COMMON/CFIRST/IFIRST
COMMON/CLIPF/ XB, YB, ZB, KB, CB, XE, YE, ZE, KE, CE, LAS, ISHARE, NTR,
          1ITR1, ITR2
COMMON/COLBAR/ BAR, BEGBO, ENDO, BAROUT
COMMON/CONLEV/ CHI, CLO, NCONLEV, CLEVEL
COMMON/CSSHH/ SHDO
COMMON/CUT/ WR, FACT, IUNIT
COMMON/DELS/ DELTA, SHRK
COMMON/DEV1/ INPUT, OUTPUT
COMMON/FF/ YMIN, NOCHDO
COMMON/ENTER/ N1, N2, KEY, XNUM, K1, K2
COMMON/FYSL/ FUN, YY, SLNRM
COMMON/INTENS/ IFH, IPL, IPB, IFX

IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMMENT NEXT STATEMENT

COMMON/INTER/ ABORT

COMMON/LABEL/NUM, NNUM, PNUN
COMMON/LAIF/ JLAST, IFLAG
COMMON/LGI/ ISMOTH, IPOUN, IMIX, DIRC, ISPEC, ISMA, LINEAR,
1  IFLR, IFRING
COMMON/MOSAIC/ MOSAIC
COMMON/NEDGE/ NEDGE, MEDGE, IPART
COMMON/FAAR/ NFL, NFLS
COMMON/ PARA/ XNR, XNH, XIO, INFIN, NHIGH, XL, JCONT, XLSI
COMMON/PERRY/ FRONTZ, BACKZ, ISET, IEF
COMMON/PGNCNTL/ IPOLY
COMMON/Poor/ PPOOR, DIRCA
COMMON/PRO/ NP, NJ, ICHE, IC, IREAD, SKALE, NPT
COMMON/QFORD/ CONTRS, IDVICE, IBAD, SHOSHR, LBLSPC
COMMON/RESTO/ X0, DC
COMMON/REST1/ RORG, DD
COMMON/Rotm/ NONROT
COMMON/SavCON/ CONSAV
COMMON/SCLIP/ ISCLIP, ZPF, ZPB
COMMON/SCMR/ DZ, FIOLED, FIELD, TANAL, RES
COMMON/Smoth/ JSMOTH
COMMON/SPUX/ SPEC, SPEC1, U, X
COMMON/XNO/ XNORM
COMMON/ZFIXER/ ZLO, ZHI, ZSPED, CURZE
DIMENSION SI(3)
DIMENSION KEY(10), XNUM(40), WORD(4)
DIMENSION FUN(3), YY(3)
DIMENSION XNORM(3, 1007)
DIMENSION DA(3, 27), DC(3, 27), DD(3, 27), DIRCA(27)
1. JSMOTH(27), NPL(2, 27), NFLS(27), PPOOR(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, IHRL, 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, ITSHD, CURIZE, 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
1'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z,'
DATA SI/O., O., O., A1/O., IAIV/1
C
IFIRST=. TRUE.
IHRL=. FALSE.
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
TURN OFF SHADOWS TEMP. IF IN LINE DRAWING MODE.

ITSHDD=.FALSE.
IF((IDVICE LT.0) AND. SHDD) THEN
  ITSHDD=. TRUE.
  SHDD=. FALSE.
ENDIF

ICMD 93 = "DRAW"

IF(IDVICE EQ.1. AND. ICMD.EQ.93) ICMD=89

ICMD 89 = "VIEW"

IF(ICMD.EQ.89) IHLR=. TRUE.

CALL LIGHT IF IT HAS NOT BEEN CALLED ALREADY

IF((JCONT.EQ.0) AND. (IDVICE.GT.0), AND. IHLR) CALL LIGHT

ISHARE=((IDVICE.LT.0). OR. (. NOT. IHLR))
IF(. NOT. CONTROS) GO TO 30
CLO=SPEC(I)

DO 10 I=2,NJ
   XI=SPEC(I)
      IF(XI.LT. CLO) CLO=XI
      IF(XI.GT. CHI) CHI=XI

10 CONTINUE

IF(. NOT. LINEAR) GO TO 30

DO 20 I=1,NJ
   XI=SPEC(I)+SPEC(I)
      IF(XI.LT. CLO) CLO=XI
      IF(XI.GT. CHI) CHI=XI

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,NFRAME
   IF(NFRAME.EQ.0) GO TO 100

   INCREDMENT DISPLACEMENTS, ROTATIONS, TRANSLATIONS, ETC. FOR ANIMATE

   XMOVE=FLOAT(IIMOVE)
   XMAGN=XMAGN+SFUEL.
   SKALE=XMAGN
   IF(LINEAR) SLINR*XMOVE/XFRAME
   IF(LINEAR) SKALE=XMAGN*SLINR
   IF(CPF.EQ.0,0) GO TO 60
   ANG=360.0*CPF*XMOVE
   SKALE=XMAGN*SIN(ANG*.017453)

205 CONTINUE

60 AMP=180.0*XMOVE/XFRAME
AMP=COS(AMP*.017453)
DAMP=.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=1
  X0(I)=X0(I)+DT(I)*DAMP
DO 80 J=1,NP
  JSafe=J
  IF(NONROT(JSAFE)) GO TO 70
  DDD=DRI(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=DRI(I)*DAMP
  JSafe=NP+1
  IF(DDD NE. 0.0) CALL ROTAT(DC, ISAFE, DDD, JSAFE)
90  CONTINUE
IF( N.OT. IPM ) GO TO 200
IF( IMOVE GT. IFR1 OR IMOVE. GT. IFR2 ) GO TO 200
C PROCESS PARTS INDIVIDUALLY
100  IERROR=1
    IYMIN=IFX
    IOPNF=0
    DAROUT = .FALSE.
    IF(. NOT. WR OR (IHLR AND (IDVICE.EQ.0))) GO TO 50
    CALL OPNFIL('PICT', IUNIT, -2, IERROR)
    IF(IERROR EQ 1) IOPNF=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
70  DO 110 I=1,NJ
110  XNORM(3,I)=0.
    IPERZ=1
    ISET=0
    ISCLIP = 0
    CALL SUMCEN(IPERZ)
    SPR = FRUNT7-RECK7
    IF( SPR LT 1E-6) SPR = 1.0
    DELZ = 0.005*SPR
    ZEF=DOZ-(FRUNTZ+DELZ)
    ZEB=DOZ-(RECKZ-DELZ)
    IF(ZEF,LT.0 0) ZEF = DELZ
    ZPF = ZEF*TANAL
    ZPB = ZEB*TANAL
    ZSPRED=32767.*ZEB/(ZEB-ZEF)
CC Addition for change in "PIVOT" option
CC CALL MOD_ORD
CC DO 120 I=1,NP
    IPart=I
IPOOR=.FALSE.
DIRC=.FALSE.
XI=POOR(IPART)
X2=DIRCA(IPART)
IF(X1.EQ.1.0) IPOOR=.TRUE.
IF(X2.EQ.0.0) DIRC=.TRUE.
IF(IDVICE.LT.0) THEN
  JSMOTH = 0
ELSE
  JSMOTH=JSMOTH(IPART)
ENDIF
SHOSHR=.TRUE.
IF(JSMOTH.EQ.-1) SHOSHR=.FALSE.
IF(NPLO(I).EQ.0) GO TO 120
CALL PART

120 CONTINUE
IF(IHLR) CALL HIDDEN

IF ON A VAX - TO IMPLEMENT CTRL-C TRAP UNCOMENT THE NEXT FOUR STATEMENTS

IF (ABORT) THEN
  CALL ENDFRM
  RETURN
ENDIF

IF(IBAD) GO TO 220
IF(.NOT.NUM) GO TO 130
IF(NNUM) CALL NODNUM
IF(PNUM) CALL PNUM

130 CONTINUE
IF(AANTIAL) 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, NONLY

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(I.C.EQ.1) WRITE(OUTPUT,240)
IF(I.C.EQ.2) WRITE(OUTPUT,250)
170 IF(NFRAME.LT.1) GO TO 200
180 IF(IDVICE.EQ.-3.OR.IDVICE.EQ.-2) GO TO 170
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) IINOVE,NFRAME
200   IF(IERROR.EQ.0) WR= TRUE.
205   CONTINUE
210   NFRAME=0
      IF(ITSHDO) SHDO= TRUE.
      SKALE=XMAXN
      LINEAR=.FALSE.
      IF(IOPNFG.EQ.1) CALL CLSFIL(IUNIT)
      RETURN
220   CALL ENDFRM
      WRITE(OUTPUT,230)
      NFRAME=0
      SKALE=XMAXN
      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, 'I3.' )
270   FORMAT( '<A1., =', F9.3, 'I3.' )

C

SEE POSSIBLE INCOMPATIBILITIES (2)

C
290   FORMAT( '<I3.' , I3, 'I3.' )
END
SUBROUTINE GETXT(SPACE, NS)

C**************************************************************
C SUBROUTINE GETXT - RETRIEVES LINE OF ENTERED TEXT
C**************************************************************
C SUBROUTINE CALLED BY
C READIN = ACCEPTS LINE OF TEXT, SPLIT INTO SEPARATE WORDS
C OR NUMBERS AND SENDS ON FOR PROCESSING
C**************************************************************
C SUBPROGRAMS CALLED
C CLSFIL = CLOSES FILE AFTER I/O
C DELSTR = DELETES SPECIFIED RECEIVED TEXT STRING
C FNAME = GATHERS THE CHARACTERS THAT MAKE UP THE FILENAME AND
C COMBINES THEM INTO A WORD
C LENTXT = DETERMINES THE LENGTH OF THE TEXT STRING
C**************************************************************
C VARIABLES USED
C I = POSITION OF THE CHARACT IN THE LINE
C INPUT = DEVICE UNIT NUMBER TO READ, PASSED FROM COMMON/DEVI/
C JUNIT = 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
C 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 EQUIVALENT 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 CHARACTER*1 SPACE(NS), TEXT(72)
C CHARACTER*12 XNAME
C INTEGER OUTPUT
C LOGICAL LINE, LREAD, LWRITE
C SAVE XNAME, LUNOLD
C COMMON/DEVI/ INPUT, OUTPUT
C COMMON/DEVIL/ JUNIT, KUNIT, LUNIT
C COMMON/TEXTO/NCHAR
C COMMON/TEXTL/ TEXT
C DATA JUNIT/3/, NULL/0/
C DATA LINE, LREAD, LWRITE/3*, FALSE /
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)
SCAN INPUT LINE FOR SPECIAL REDIRECTION CHARACTERS.
LINE=.FALSE.
DO 200 I=1, NCHAR
OUTPUT TO FILE.
IF(TEXT(I).EQ. '>') THEN
CLOSE FILE IF OPEN.
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
OPEN FILE IF READ REDIRECTION NOT IN USE.
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. 1) 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
ERROR OPENING REDIRECTION OUTPUT FILE.
CONTINUE
WRITE(OUTPUT, 1600)
NCHAR=I-1
GO TO 300
ERROR READ REDIRECTION ALREADY IN USE.
ELSE IF(LREAD) THEN
    WRITE(OUTPUT, 1400)
    NCHAR=I-1
    GO TO 300
END IF

INPUT FROM FILE

ELSE IF(TEXT(I), EQ,'<') THEN

CLOSE FILE IF OPEN

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=LENXTX(TEXT, NCHAR)
    END IF
    GO TO 300

OPEN FILE IF WRITE REDIRECTION NOT IN USE

ELSE IF(.NOT.LWRITE) THEN
    CALL FNNAME(I, SPACE, NS, XNAME, L)
    OPEN(IUNIT, FILE=XNAME(:L), STATUS='OLD', ERR=170)
    REWIND IUNIT
    LREAD=.TRUE.
    WRITE(OUTPUT, 1500) XNAME
    LUNOLD = INPUT
    IF(NCHAR LE 1) THEN
        NCHAR=72
        READ(INPUT, 1100, END=400) (TEXT(J), J=1, NCHAR)
        NCHAR=LENXTX(TEXT, NCHAR)
        IF(INPUT, EQ, IUNIT) WRITE(OUTPUT, 1800)
            (TEXT(J), J=1, NCHAR)
    END IF
    GO TO 300

ERROR OPENING REDIRECTION OUTPUT FILE.

170 CONTINUE
    WRITE(OUTPUT, 1600)
    NCHAR=I-1
    GO TO 300

ERROR WRITE REDIRECTION ALREADY IN USE.

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 C 400 CONTINUE
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) = 'Y'
RETURN
ENDIF
CC IF(LREAD) THEN
CALL CLSFIL(IUNIT)
LREAD= FALSE
WRITE(OUTPUT, 1200) XNAME
WRITE(OUTPUT, 1700)
NCHAR = 72
INPUT = LUNID.D
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( '<ERROR: OPEN FILE! REST OF LINE IGNORED.>' )
1600 FORMAT( '<ERROR: OPEN FILE! REST OF LINE IGNORED.>' )
1700 FORMAT( '<INPUT>> ' )
1800 FORMAT( '<72A>' )
C END
SUBROUTINE NUMDIG(WORD, N, LI, L2)

C***********************************************************************
C SUBROUTINE NUMDIG - ACCEPTS NUMBER DIGITS AND CONVERTS TO REAL
C NUMBER
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 SUBPROGRAMS CALLED - NONE
C***********************************************************************

VARIABLES USED
FAC = MULTIPLICATION FACTOR WHICH CHANGES NUMBER WITH
      EXPONENT TO A REAL NUMBER WITHOUT EXPONENT
ICOUNT = COUNTER FOR NUMBER OF DIGITS IN A NUMBER OR EXPONENT
ID = NUMBER OF DECIMALS FOUND IN A NUMBER (FOR ERROR PURPOSE)
IDOT = 1 AT THE TIME A DECIMAL IS FOUND, THEN RESET TO ZERO
IE = EXPONENT FLAG, =1 IF EXPONENT EXISTS
     =0 IF NO EXPONENT EXISTS
ILOC = NUMBER OF DIGITS TO LEFT OF DECIMAL
IRIGHT = NUMBER OF DIGITS TO RIGHT OF DECIMAL
ISIGN = 1 FOR POSITIVE NUMBERS
         -1 FOR NEGATIVE NUMBERS
J = DIMENSION OF NUM ARRAY
JCOUNT = COUNTER FOR RECOGNIZING NUMBER OR EXPONENT DIGIT
K2 = NUMBER OF NUMBERS FOUND
KEY = ARRAY OF ACCEPTED KEY WORD STARTING LOCATIONS IN WORD
      ARRAY
LE = STARTING LOCATION OF EXPONENT
LT = TEMPORARY STARTING LOCATION OF A TEXT WORD
MM = 10 TO THE CORRECT POWER USED TO CALCULATE NUMBER
     FROM ITS DIGITS
MSIGN = 1 FOR POSITIVE EXPONENT
         -1 FOR NEGATIVE EXPONENT
MUL = EXPONENT VALUE
MULT = ARRAY OF DIGITS IN EXPONENT
MULTN = NUMBER OF DIGITS IN EXPONENT
N2 = INPUT FLAG, =1 WHEN NUMBERS WANTED
     =0 WHEN NUMBERS NOT WANTED
NR = VALUE OF THE NUMBER
NUMT = ARRAY OF DIGITS IN NUMBER
NUMTN = NUMBER OF DIGITS IN NUMBER
TEXT = ARRAY OF ENTERED CHARACTERS
WORD = ARRAY OF ACCEPTABLE COMMANDS FOR ACTIVE ROUTINE
X = ARRAY OF ACCEPTABLE NUMBER DIGITS
XNUM = ARRAY OF REAL NUMBERS ACCEPTED

CHARACTER*1 TEXT, X, WORD
COMMON/ENTER/ N1, N2, KEY, XNUM, K1, K2
COMMON/TXT1/TEXT
DIMENSION KEY(10), XNUM(40), TEXT(72), WORD(*)
DIMENSION NUMT(15), MUL(5), X(15)
DATA X/1', '2', '3', '4', '5', '6', '7', '8', '9', '0',
     '+', '-', 'E', 'e', '.'/

C IF NO NUMBERS SOUGHT - RETURN.
C
C IF(N2.EQ.0) RETURN
C
C INITIALIZE FLAGS AND COUNTERS.
C
LT=L1
IE=0
NUMIN=0
MULIN=0
MUL=0
ISIGN=1
MSIGN=1
IDOT=0
ILOC=-1
NR=0
ID=0

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
DQ 60 I=LT,L2
   JCOUNT=0

C IF NOT EXPONENT DIGITS - JUMP.
C
IF(IE.NE.1) GO TO 20

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 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 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
CHECK FOR EXPONENT CHARACTER E.

40 IF (TEXT(I) NE. X(13). AND TEXT(I) NE. X(14)) GO TO 50
   IE=1
   GO TO 60

IDENTIFY NUMBER DIGIT AND LOAD INTO NUMT.

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
   CONTINUE
   GO TO 120
   MSIGN=-1
   LE=I+1

IDENTIFY EXPONENT DIGIT AND LOAD INTO MULT.

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

Make sure a zero is handled correctly

IF (JCOUNT EQ. 10) JCOUNT=0

MULT(ICOUNT)=JCOUNT
   MULTN=ICOUNT

IF NO NUMBER DIGITS RECOGNIZED - RETURN.

120 IF (NUMTN EQ 0) RETURN

IF MORE THAN ONE DECIMAL POINT FOUND IN NUMBER - RETURN.

IF (ID, GT. 1) RETURN
   K2=K2+1

IF NO DECIMAL POINT FOUND - ASSUME AT END OF NUMBER

IF (ILOC EQ -1) ILOC=NUMTN

IF NO EXPONENT - JUMP.

IF (MULTN EQ 0) GO TO 140
   MNP=1

EXponent IS calculated, assigned its sign and altered to
reflect location of decimal point. (Effectively moving
decimal point to end of number)

DO 130 I=1, MULTN
   130
J=MULTN+1-I
MUL=MUL+MULT(J)*MM

130 MM=10*MM
MUL=M$IGN*MUL

140 IRIGHT=NUMTN-ILOC
MUL=MUL+IRIGHT
MM=I
DO 150 I=1,NUMTN
    J=NUMTN+I-1
    NR=NR+NUMT(J)*MM

C NUMBER VALUE CALCULATED FROM ITS DIGITS.

150 MM=10*MM
NR=ISIGN*NR

C APPROPRIATE MULTIPLICATION FACTOR (DUE TO EXPONENT) CALCULATED.

IF(MUL) 160, 180, 170

160 MUL=-MUL
FAC=1./(10.0**MUL)
GO TO 190

170 FAC=10.0**MUL
GO TO 190

180 FAC=1.

C MULTIPLICATION FACTOR IS APPLIED TO NUMBER AND RESULTING NUMBER
C IS LOADED INTO XNUM ARRAY.

190 XNUM(K2)=FLOAT(NR)*FAC
RETURN
END
Appendix B

New NUVTE, BYU Routines

FILE_OP
GET_OP
HALT_OP
MON_OP
PROG_OP
PUSH_OP
PUT OP
NUFLOP
NEG_CFN
SUBROUTINE FILE_OP

C******************************************************************************
C
C SUBROUTINE FILE_OP - ENABLES COMMAND INPUT FROM DISK FILE
C******************************************************************************
C
C SUBROUTINE CALLED BY
C    MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND
C           CALLS APPROPRIATE SUBROUTINES
C******************************************************************************
C
C SUBPROGRAMS CALLED
C    OPEN = SYSTEM OPEN FILE ROUTINE
C    RES_CEN = RESTORES MODEL TO ORIGINAL COORDINATE SYSTEM
C******************************************************************************
C
C VARIABLES USED
C    INPUT = DEVICE LOGICAL UNIT NUMBER FOR INPUT
C    JUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE) OUTPUT
C    KUNIT = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)
C    LUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE) INPUT
C    MUNIT = DEVICE LOGICAL UNIT NUMBER (SYS$OUTPUT)
C    OUTPUT = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT
C    XNAME = NAME OF USFRS COMMAND FILE (DEFAULT EXT .DAT)
C******************************************************************************
C
COMMON/DEV1/ INPUT, OUTPUT
COMMON/DEV1/ MUNIT, JUNIT, KUNIT, LUNIT
CHARACTER*12 BLANK, XNAME
INTEGER OUTPUT
DATA BLANK/" '/

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',
       DISPOSE='DELETE')
       INPUT = LUNIT
       OUTPUT = JUNIT
       CALL RES_CEN
   ENDIF
   RETURN
50 FORMAT(21H<READ COMMAND FILE> *)
60 FORMAT(A12)
END
SUBROUTINE GET ORG

SUBROUTINE GET ORG - INPUTS RELATIVE ORIGIN OF PARTS (PIVOT POINT)

FOllowing prompt ("~") USER ENTERS VALUES FOR

I1 AND I2 (RANGE OF PARTS), I3 (PART ON WHICH

I1 TO I2 PIVOT) & X,Y,Z (PIVOT POINT COORDS).

WHERE POSITION OF PARTS I1/I2 IS NOT AFFECTED
BY THE PIVOTS OF ANY OTHER PART, ENTER A ZERO
FOR PART I3. A BLANK INPUT (CARRIAGE RETURN)

WILL TRANSFER CONTROL BACK TO CALLING PROGRAM

SUBROUTINE CALLED BY

MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND
CALLS APPROPRIATE SUBROUTINES

SUBROUTINES CALLED

READIN = ACCEPTS INPUT FROM THE USER

VARIABLES USED

IPREV = ARRAY OF CONNECTING PART NUMBERS
N1 = INPUT FLAG. =1 WHEN KEY WORDS Sought
=0 WHEN KEY WORDS NOT Sought
N2 = INPUT FLAG. =1 WHEN NUMBERS WANTED
=0 WHEN NUMBERS NOT WANTED
NW = DIMENSION OF WORD ARRAY
OUTPUT = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT
PROMPT = IF .TRUE. PROMPT FOR USER INPUT (LOCAL VAR.)
SORG = ARRAY OF SPECIFIED PIVOTAL POINTS
XNUM = ARRAY OF REAL NUMBERS ACCEPTED

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(WURD,NW)
    I1 = XNUM(I)
    IF (I1 .EQ. 0) THEN
        PROMPT = .FALSE
ELSE
    I2 = XNUM(2)
    DO J = I1, I2
        IPREV(J) = XNUM(3)
        DO I = 1, 3
            SORD(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 SUBROUTINE HALT_OP - TERMINATES COMMAND INPUT FROM DISK FILE
C**************************************************************************************************C
C
C SUBROUTINE CALLED BY
C      MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER
C      PAUS_OP = PROCESSES THE "PAUSE" OPTION
C
C**************************************************************************************************C
C
C SUBPROGRAMS CALLED
C      CLOSE = SYSTEM CLOSE FILE ROUTINE
C
C**************************************************************************************************C
C
C VARIABLES USED
C      INPUT = DEVICE LOGICAL UNIT NUMBER FOR INPUT
C      JUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE) OUTPUT
C      KUNIT = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)
C      LUNIT = DEVICE LOGICAL UNIT NUMBER (DISK FILE) INPUT
C      MUNIT = DEVICE LOGICAL UNIT NUMBER (SYS$OUTPUT)
C      OUTPUT = DEVICE LOGICAL UNIT NUMBER FOR OUTPUT
C
C**************************************************************************************************C

COMMON/DEV1/ INPUT, OUTPUT
COMMON/DEV1/ MUNIT, JUNIT, KUNIT, LUNIT
INTEGER OUTPUT

CLOSE(LUNIT)
CLOSE(JUNIT)
OUTPUT = MUNIT
INPUT = KUNIT
RETURN
END
SUBROUTINE MOD_ORG

C******************************************************************************
C
C SUBROUTINE MOD_ORG - COMPUTES CURRENT POSITION OF PIVOTAL POINTS
C
C******************************************************************************
C
C SUBROUTINE CALLED BY
C
VIEDRA = CALLS FOR NORMALS, LIGHT INTENSITY, ETC. NEEDED TO
C
DISPLAY SCENE (EITHER VIEW OR DRAW)
C
C******************************************************************************
C
C VARIABLES USED
C
DD = LOCAL TRANSFORMATION MATRIX
C
IPREV = ARRAY OF CONNECTING PART NUMBERS
C
NP = NUMBER OF PARTS
C
PORC = ARRAY OF PIVOTAL POINTS CURRENT POSITION
C
RORC = RELATIVE ORIGIN OF PART
C
SORC = SPECIFIED PIVOTAL POINT OF PART
C
C******************************************************************************
C
COMMON/PRO / NP, NJ, ICMD, IC, IRCAD, SKALE, NPT
COMMON/REST1/ RORC, DD
COMMON/REST3/ PORC, SORC, IPREV
DIMENSION DD(3,3,27), RORC(3,27)
DIMENSION PORC(3,27), SORC(3,27), IPREV(27)

DO I = 1, NP
   DO J = 1, 3
      RORC(J, I) = SORC(J, I)
   ENDDO
   J = IPREV(I)
   IF (J EQ. 0) THEN
      DO J = 1, 3
         PORC(J, I) = RORC(J, I)
      ENDDO
   ELSE
      X1 = RORC(1, I) - RORC(1, J)
      X2 = RORC(2, I) - RORC(2, J)
      X3 = RORC(3, I) - RORC(3, J)
      PORC(1, I) = DD(1, 1, J)*X1 + DD(2, 1, J)*X2 +
                   DD(3, 1, J)*X3 + PORC(1, J)
      PORC(2, I) = DD(1, 2, J)*X1 + DD(2, 2, J)*X2 +
                   DD(3, 2, J)*X3 + PORC(2, J)
      PORC(3, I) = DD(1, 3, J)*X1 + DD(2, 3, J)*X2 +
                   DD(3, 3, J)*X3 + PORC(3, J)
   ENDDO
ENDDO
RETURN
END
SUBROUTINE PAUS.OP

C*******************************************************************************C
C SUBROUTINE PAUS.OP - WAITS FOR INPUT AT TERMINAL KEYBOARD
C IF AN 'A' IS INPUT - HALT.OP IS EXECUTED TO CLOSE USER'S COMMAND FILE AND TO SET
C SYS$INPUT AS THE COMMAND-INPUT DEVICE.
C OTHERWISE... NO CHANGES (IF COMMAND
C INPUT IS FROM A DISK FILE A "PAUSE" WILL
C PERMIT USER TO READY THE PLOTTING ... OR.
C ENTER "A" TO ABORT FILE-COMMAND INPUT)
C*******************************************************************************C
C
C SUBROUTINE CALLED BY
C MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND CALLS APPROPRIATE SUBROUTINES
C
C*******************************************************************************C
C SUBPROGRAMS CALLED
C HALT.OP = TERMINATES COMMAND INPUT FROM DISK FILE
C READIN = ACCEPTS INPUT FROM THE USER
C
C*******************************************************************************C
C VARIABLES USED
C INSav = TEMPORARY STORAGE OF LOGICAL UNIT NUMBER (LOCAL VAR)
C K1 = NUMBER OF KEYWORDS FOUND
C N1 = INPUT FLAG = 1 WHEN KEYWORD SOUGHT
C N2 = INPUT FLAG = 0 WHEN NUMBERS NOT WANTED
C NW = DIMENSION OF WORD ARRAY
C WORD = ARRAY OF ACCEPTABLE WORDS
C
C*******************************************************************************C
C COMMON/DEVI/, INPUT, OUTPUT
C COMMON/ENTER/, N1, N2, KEY, XNUM, K1, K2
C INTEGER OUTPUT
C DIMENSION KEY(10), XNUM(40), WORD(NW)
C CHARACTER*1 WORD
C DATA WORD/"A", "+", "-", "/

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

SUBROUTINE PLOT_OP - ENABLES SWITCH FOR HARD COPY PLOTS.

APPLICATION COMMAND "PLOT"

WHEN COMMAND INPUT IS VIA 'SYS$INPUT' AND
USER REQUESTS A VIEW OR DRAW THE PICTURE
WILL BE DISPLAYED ON THE TERMINAL SCREEN.

IF THIS HARD-COPY SWITCH IS ENABLED, USER
WILL THEN BE GIVEN THE OPTION OF PLOTTING
THE DISPLAY ON THE PLOTTER. WHEN COMMAND
INPUT IS FROM A DISK FILE PICTURE WILL BE
DISPLAYED ON THE PLOTTER IF 'PLOT' SWITCH
IS ON, BUT ON THE TERMINAL SCREEN IF OFF.

SUBROUTINE CALLED BY

MAIN - PROCESSES INTERACTIVE COMMANDS FROM THE USER AND
CALLS APPROPRIATE SUBROUTINES

VARIABLES USED

APLOT = ENABLES ONE HARD COPY OUTPUT
HRDCPY = LOGICAL FLAG INDICATING PLOT IS ON (OR OFF)
KUNIT = DEVICE LOGICAL UNIT NUMBER (SYS$INPUT)

COMMON/HCOPY, HRDCPY, APLLOT, HCPN
COMMON/DEVI/ INPUT, OUTPUT
COMMON/DEVIL/ MUNIT, JUNIT, KUNIT, LUNIT
LOGICAL HRDCPY, APLLOT, HCPN, KEYRD
INTEGER OUTPUT

HRDCPY = .TRUE.
IF (INPUT EQ. KUNIT) THEN
    WRITE(OUTPUT,20)
    APLLOT = .FALSE.
ELSE
    APLLOT = .TRUE.
ENDIF
RETURN

20 FORMAT(16H<PLOT ENABLED>, /)
END
SUBROUTINE NOPLOT
C
*******************************************************************************
C SUBROUTINE NOPLOT - DISABLES SWITCH FOR HARD COPY PLOTS
C PROCESSES INTERACTIVE COMMAND "NOPLOT"
C*******************************************************************************

C SUBROUTINE CALLED BY
C MAIN = PROCESSES INTERACTIVE COMMANDS FROM THE USER AND
C CALLS APPROPRIATE SUBROUTINES
C*******************************************************************************
C
C VARIABLES USED
C APLLOT = ENABLES ONE HARD COPY OUTPUT
C HRDCPY = LOGICAL FLAG INDICATING PLOT IS ON (OR OFF)
C*******************************************************************************

C
COMMON/HCOPY/, HRDCPY, APLLOT, HCEND
COMMON/DEVI/, INPUT, OUTPUT
LOGICAL HRDCPY, APLLOT, HCEND
INTEGER OUTPUT

C
HRDCPY = .FALSE.
APLOT = .FALSE.
WRITE(OUTPUT,30)
RETURN

C
30 FORMAT(17H+(PLOT DISABLED) ,/)
END
SUBROUTINE RES_CLN

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

Animation Control Routines

RUN/movie
DISPLAY/TILE
EXIT/movie
ALL REC
SUBROUTINE RUN_MOVIE (FGEOM, FDISP, FFUNC, FORIG)

RUN_MOVIE will create and run the "MOVIE" subprocess. Communication between this program and MOVIE will be via two mailboxes ("ABOX" and "BBOX" created here). The names (subroutine arguments) of geometry, displacement, function, and the parts-origin files are sent to MOVIE. MOVIE's prompt (">>>") for command input is read and control returned to the calling program. DISPLAY_FILE will output the "FILE" command to MOVIE and the name of the disk-file (this entry's argument) which contains the commands (and data if required) to be executed by MOVIE. After commands (normally terminated by a "HALT") have been processed MOVIE's prompt (">>>") is read & control returned to calling program.

Arguments (char. variables) are:
- FGEOM - name of geometry file
- FDISP - name of displacement file
- FFUNC - name of function file
- FORIG - name of parts-origin file
- FCOMM - name of user-commands file

Call RUN_MOVIE once to initiate the MOVIE subprocess.
Call DISPLAY_FILE whenever it is desired to have a set of commands processed by MOVIE.
Call EXIT_MOVIE to delete the MOVIE ("READIT") subprocess.

INTEGER*2 CH, CA
INTEGER*4 SYS$DELP RC, SYS$CREPRC, SYS$Q1OW
INTEGER*4 SYS$CREMBX, SYS$TRNLOG, IS
EXTERNAL SS$NORMAL, IO$WRITEVLK, IO$READVLK
CHARACTER*8 TA, TB, TM*7, TT*32, FNAM*12, MIN*10, MXX*21
CHARACTER*4 MA/"ABOX'/, MB/"BBOX'/, OUTC/"FILE'/
CHARACTER*(*) FGEOM, FDISP, FFUNC, FORIG, FCOMM
LOGICAL MOVERR

IS = SYS$DELP RC(,'READIT')

Create Mailboxes:
- "ABOX" output from this process
- "BBOX" input to this process

IS = SYS$CREMBX(CH, CA, MA)
IF (.NOT. IS) CALL LIB$STOP(XVAL(IS))
IS = SYS$CREMBX(CH, CA, MA)
IF (.NOT. IS) CALL LIB$STOP(XVAL(IS))

Translate logical

IS = SYS$TRNLOG('BBOX', TT,)
TB = TT(INDEX(TT,'_'):INDEX(TT,''))
IS = SYS$TRNLOG('ABOX', TB,)
TA = TT(INDEX(TT,'_'):INDEX(TT,''))
IS = SYS$TRNLOG('SYS$COMMAND', TB,)
TM = TT(INDEX(TT,'_'):INDEX(TT,''))

Create process "READIT" with
- SYS$INPUT "ABOX"
- SYS$OUTPUT "BBOX"
- SYS$ERROR TERMINAL

IS = SYS$CREPRC(ID,'IFTS MOVIE)MOVIE.EXE', TA, TB, TM, 'READIT', 4,...
IF (IS .NE. %LOC(SS$NORMAL)) TYPE 100, IS

... assume MOVIE process is running...
Read two records from the mailbox "BBX"
Output geometry file_name to mailbox "ABOX"

DO I = 1, 2
   IS = SYS$IOV(%VAL(3), %VAL(CH), IO$_READVLK, ..., %REF(MIN),
   %VAL(18), ...)
   IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))
ENDDO
FNAM = FGEOIM
IS = SYS$IOV(%VAL(3), %VAL(CA), IO$_WRITEVLK, ..., %REF(FNAM),
   %VAL(12), ...)
IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))

Read two records from the mailbox "BBX"
Output displacement file_name to mailbox "ABOX"

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

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

IS = SYS$IOV(%VAL(3), %VAL(CH), IO$_READVLK, ..., %REF(MIN),
   %VAL(18), ...)
IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))
FNAM = FFUNC
IS = SYS$IOV(%VAL(3), %VAL(CA), IO$_WRITEVLK, ..., %REF(FNAM),
   %VAL(12), ...)
IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))

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

IS = SYS$IOV(%VAL(3), %VAL(CH), IO$_READVLK, ..., %REF(MIN),
   %VAL(18), ...)
IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))
FNAM = FORIC
IS = SYS$IOV(%VAL(3), %VAL(CA), IO$_WRITEVLK, ..., %REF(FNAM),
   %VAL(12), ...)
IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))

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

MIN(2:3) = '##'
DO WHILE (MIN(2:3).NE. '>>')
   IS = SYS$IOV(%VAL(3), %VAL(CH), IO$_READVLK, ..., %REF(MIN),
   %VAL(4), ...)
   IF (.NOT. IS) CALL LIB$STOP(%VAL(IS))
ENDDO
RETURN
ENTRY DISPLAY_FILE (FCOMM)

C Instruct MOVIE to process the commands in user's file 'FCOMM'.
C Wait until all commands executed

C Output interactive command "FILE" to "ABOX"

IS = SYS$QIDW (%VAL(3), %VAL(CA), IO$_WRITEVBLK, ... %REF(QUIC)),
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$QIDW (%VAL(3), %VAL(CH), IO$_READVBLK, ... %REF(MXX)),
5 %VAL(21), ...)
IF (. NOT. IS) CALL LIB$STOP (%VAL(IS))
FNAM = FCOMI
IS = SYS$QIDW (%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 inhibited until "HALT" from file 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$QIDW (%VAL(3), %VAL(CH), IO$_READVBLK, ... %REF(MIN),
5 %VAL(4), ...)
   IF (. NOT. IS) CALL LIB$STOP (%VAL(IS))
ENDDO
RETURN
ENTRY EXIT_MOPIE

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
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
C CHARACTER(*) FCOMM
C DIMENSION IREC(1), ITEM(1), VALU(1)
C CHARACTER*50 TEXT, HOLD, VAL*12
C
C Transfer all records from the input file into a working file
C IU1 = 21
C IU2 = 22
C OPEN(UNIT=IU2, FILE=FCOMM, TYPE='OLD')
C OPEN(UNIT=IU1, FILE='WORK.DUM', TYPE='NEW',
C 1 ACCESS='DIRECT', RECL=50)
C IRT = 0
C 10 READ(IU2,100,END=20) TEXT
C IRT = IRT+1
C WRITE(IU1,'IRT) TEXT
C GO TO 10
C 20 CLOSE(IU2)
C
DO 30 I = 1, M
C IR = IREC(I)
C READ(IU1,IR) TEXT
C IT = ITEM(I)
C J = 0
C HOLD = , ,
C IF (IT.NE.1) THEN
C
C Count chars prior to the item to be changed
C DO L = 2, IT
C J = J+1
C DO WHILE (TEXT(J:J).EQ. ', ')
C J = J+1
C ENDDO
C DO WHILE ((TEXT(J:J).NE. ', ', AND. (TEXT(J:J).NE. ', ',
C 1 AND (TEXT(J:J).NE. '; ', AND. (TEXT(J:J).NE. ', '))
C J = J+1
C ENDDO
C ENDDO
C
C Save items, ahead of specified item, in holding location
C HOLD(J:1) = TEXT(J:1)
C 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. 
   (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(AS0)
200 FORMAT(E12.5)
END
Appendix D

Utility Routines

WRITE AND 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