NASA/TM-2000-210388

User's Guide for SKETCH

David R. Hedgley, Jr.
NASA Dryden Flight Research Center
Edwards, California

December 2000

The NASA STI Program Office...in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and mission, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results...even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

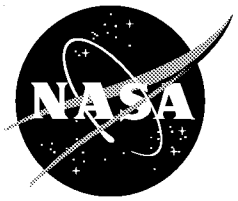
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2000-210388



User's Guide for SKETCH

David R. Hedgley, Jr.
NASA Dryden Flight Research Center
Edwards, California

National Aeronautics and
Space Administration

Dryden Flight Research Center
Edwards, California 93523-0273

December 2000

NOTICE

Use of trade names or names of manufacturers in this document does not constitute an official endorsement of such products or manufacturers, either expressed or implied, by the National Aeronautics and Space Administration.

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

ABSTRACT

A user's guide for the computer program SKETCH is presented on this disk. SKETCH solves a popular problem in computer graphics-the removal of hidden lines from images of solid objects. Examples and illustrations are included in the guide. Also included is the SKETCH program, so a user can incorporate the information into a particular software system.

README:

This file (readme) addresses the functions of all the files on this disk. The files are listed below with a description of each one.

(1) The "readme" file describes other files on the disk and contains the abstract.

(2) The "uguide" file describes how to implement the SKETCH program.

(3) The "sample" file contains an example of this FORTRAN program using a hexagonal cylinder as an illustration.

(4) The "SKETCH" file is a FORTRAN subroutine that is called by the user.

(5) A pdf file of A General Solution to the Hidden-Line Problem, by David R. Hedgley, Jr., NASA Reference Publication 1085, March 1982.

Some of this user information was also published in February 1982 as a written document, NASA Technical Memorandum 81369, User's Guide for SKETCH, by David R. Hedgley, Jr.

To make these files available to the largest number of computer users, some files appear in two formats, as a pdf file and as a Microsoft Word text file.

Note: Adobe Acrobat Reader, required for viewing pdf files, can be downloaded free of charge at www.adobe.com.

INTRODUCTION

This software package solves the hidden line problem, that problem in computer graphics which addresses the removal of hidden parts from images of solid objects. Hidden line removal is necessary to create a realistic image. As an illustration, compare figure 1 (before solution) with figure 2 (after solution).

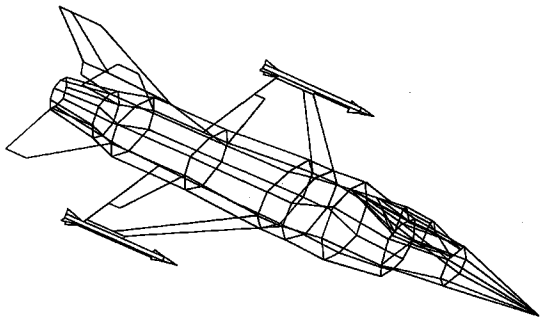


Figure 1. Before solution.

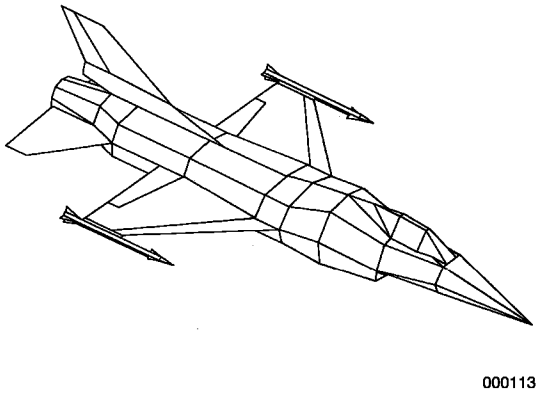


Figure 2. After solution.

This guide discusses the topics of:

1. How to decompose figures into appropriate input elements.
2. How to "feed" these elements to the hidden line package.
3. How to incorporate the package into a FORTRAN program.

INPUT ELEMENTS AND HOW TO "FEED" THEM TO THE HIDDEN LINE PACKAGE

The input elements to the hidden line package are either line segments or planar polygons. The polygons can be n-sided concave or convex and can have concave or convex holes.

The input elements are described in terms of their endpoints (in the case of line segments) or vertices (in the case of polygons). It is these endpoints or vertices, in the form of (x, y, z) triplets, that are input to the hidden line package.

As an example, note the following pentagon:

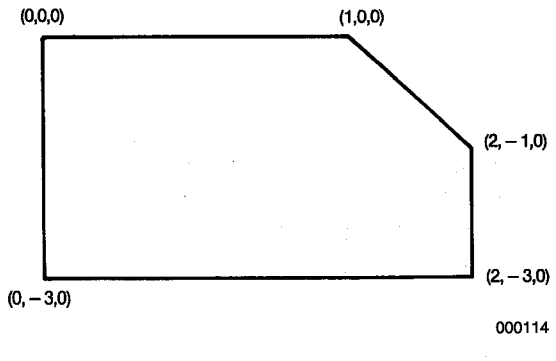


Figure 3. Pentagon example.

The triplets may be input to the package in either a clockwise (A) or counter-clockwise (B) manner:

(A) (0,0,0) (1,0,0) (2,–1,0) (2,–3,0)
(0,–3,0) (0,0,0)

or

(B) (2,–3,0) (2,–1,0) (1,0,0) (0,0,0,)
(0,–3,0) (2,–3,0)

In a given plot, composed of many such figures, any combination of clockwise and counter-clockwise may be used. It is not necessary that all figures be either clockwise or counter-clockwise.

Note in the above examples that the last point and the first point are the same. This is done to remove the difficulty that arises when the polygon has several holes.

Consider the next example, a square with a triangular hole:

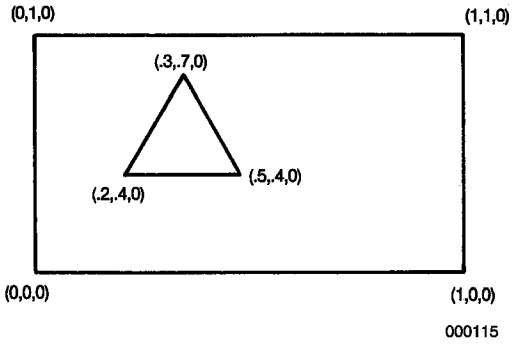


Figure 4. Square with triangular hole.

This is a polygon of seven edges and is input in the following

(0,1,0)	(0,0,0)	(1,0,0)	(1,1,0)
(0,1,0)	(.2,.4,0)	(.5,.4,0)	(.3,.7,0)
(.2,.4,0)			

Nine input triplets are required to "close-up" each sub-figure.

The actual method for inputting the triplets to the hidden line package is by means of subroutine SKETCH. The calling sequence for SKETCH is as follows:

CALL SKETCH (x,y,z,NTRP,NC)

where

x, y, z	are dimensioned arrays to hold the triplets
NTRP	is the number of triplets in a given call to SKETCH
NC is a flag:	must be set to 1 for the last call to SKETCH for a given plot must be set to 0 in all other cases

As an example, consider figure 4, the square with a triangular hole. This figure will require one call to SKETCH. The x, y, and z arrays must be dimensioned at least nine each, and will appear as follows:

x(1) = 0	y(1) = 1	z(1) = 0
x(2) = 0	y(2) = 0	z(2) = 0
x(3) = 1	y(3) = 0	z(3) = 0
x(4) = 1	y(4) = 1	z(4) = 0
x(5) = 0	y(5) = 1	z(5) = 0
x(6) = .2	y(6) = .4	z(6) = 0
x(7) = .5	y(7) = .4	z(7) = 0
x(8) = .3	y(8) = .7	z(8) = 0
x(9) = .2	y(9) = .4	z(9) = 0

NTRP, the number of triplets for this call to SKETCH, is nine. Since this is the only and last call to SKETCH, NC is set equal to 1.

Therefore:

CALL SKETCH (x,y,z,9,1)

Now consider the simple box in figure 5.

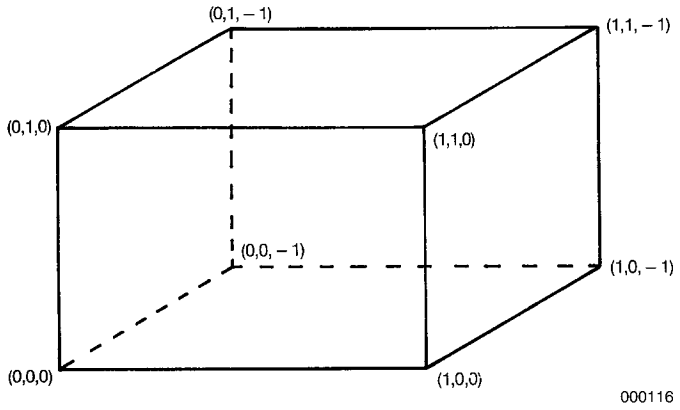


Figure 5. Simple box.

This figure will require six calls to SKETCH, one for each face of the box.

The front face is composed of the following triplets:

(0,1,0) (0,0,0) (1,0,0) (1,1,0)
(0, 1,0)

The arrays x, y, and z must be each dimensioned at least five will appear as follows:

x(l) = 0	y(l) = 1	z(l) = 0
x(2) = 0	y(2) = 0	z(2) = 0
x(3) = 1	y(3) = 0	z(3) = 0
x(4) = 1	y(4) = 1	z(4) = 0
x(5) = 0	y(5) = 1	z(5) = 0

NTRP, the number of triplets for this call to SKETCH, is five and since this is not the last call to SKETCH, NC is set equal to 0. So:

CALL SKETCH (x,y,z,5,0)

Each of the remaining five faces will be input in a similar way. On the sixth and final call to SKETCH, NC will be set to 1.

This figure can also be referred to as a figure containing six input elements.

Figure 6, a box with a hole cut in the front face, is also a figure of six input elements.

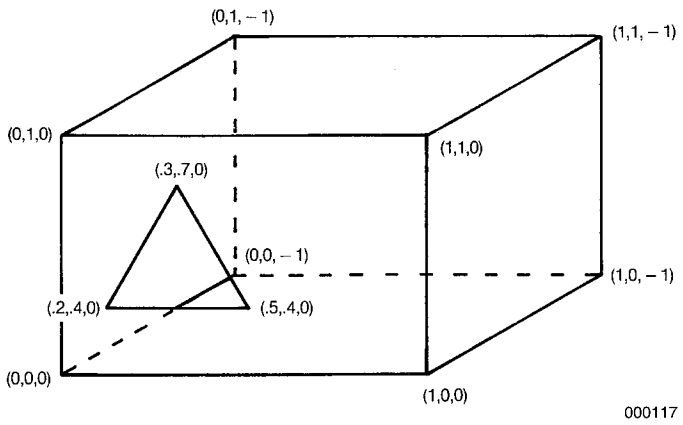


Figure 6. Box with a hole cut in the front face.

The front face would contain nine input triplets (as described in a previous example) and therefore the x, y, and z arrays would each be dimensioned at least nine. NTRP, the number of Points in the call to SKETCH for the front face, would be nine. For each of the other faces, NTRP would be five.

Now consider a transparent box, figure 7.

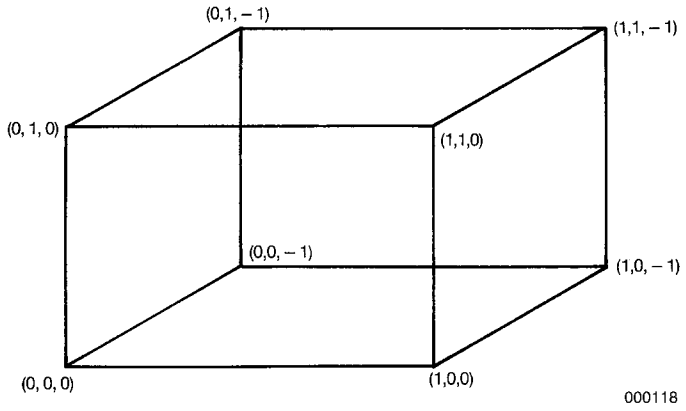


Figure 7. Transparent box.

This figure is a composite of 12 input elements, 12 line segments. It will require 12 calls to SKETCH, with the x, y, and z arrays each dimensioned at least 2 and NTRP equal to 2. For the bottom front edge:

x(1) = 0	y(1) = 0	z(1) = 0
x(2) = 1	y(2) = 0	z(1) = 0

CALL SKETCH (x,y,z,2,0)

In a similar manner all 12 of the edges will be input to SKETCH.

Note that on the 12th call to SKETCH, NC must be set equal to 1.

In the appendix, a sample figure is decomposed into input elements and "fed" to SKETCH by a sample user program. The resulting plots are also shown.

INCORPORATING THE HIDDEN LINE PACKAGE INTO THE USER'S FORTRAN PROGRAM

The programmer writing the FORTRAN code that calls the hidden line package must be aware of certain information needed by the package and must also provide information to the plot output device.

The hidden line package requires three named common blocks that must be set up in the calling program. The first is:

COMMON/GO/WORK(ICORE)

where:

ICORE = (25 + 5 * MNE + 4 * MNP)* NELEM.

MNE = Maximum number of edges that any one polygon has.

MNP = MNE + 2 + 2 * n, where n is the number of holes, if any.

NELEM = Total number of elements in one entire plot.

The second common block is:

COMMON/SCALAR/SCF, PSI, PHI, THETA, MNE, DV, MNP, ICORE

where:

SCF = Scaling factor (units/inch) for Plot Output Device.
Note: if SCF is negative, a transparent plot with no hidden lines will be produced.

PSI, PHI, THETA = The Eulerian angles (degrees) about the x-, y-, and z-axes. (See fig. 8.) The order of rotation is ψ-ϕ-θ.

MNE = (INTEGER) Defined above.

DV = A positive number that represents the distance of the viewer from the plane of the plot.

MNP = (INTEGER) Defined above.

ICORE = (INTEGER) The total core needed by the hidden line package for data.

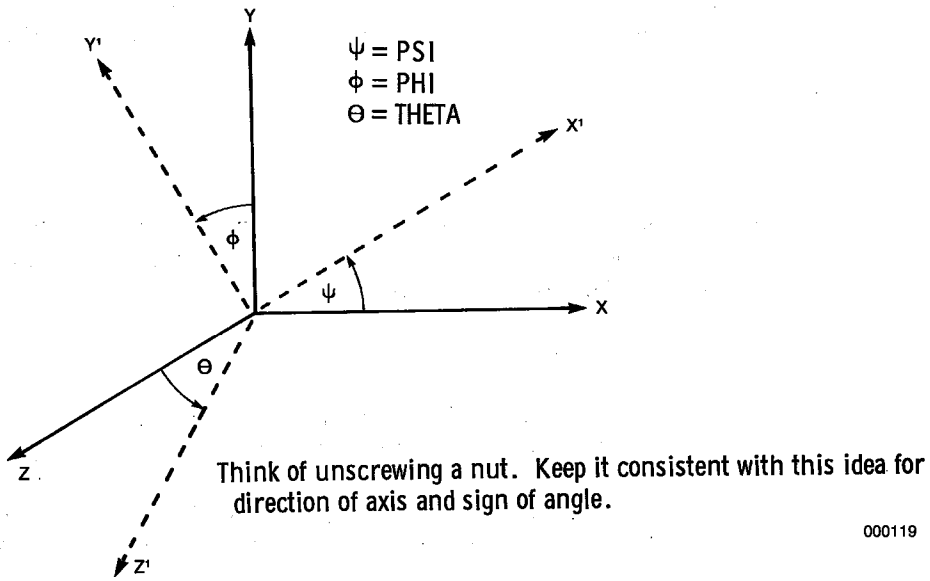


Figure 8. Axis system convention and Eulerian angles definition.

The third common block is:

COMMON/DRH/ISILH (NELEM) where:

ISILH(J) contains an integer which identifies the Jth element as a member of a family of elements.

ISILH(J) = 0 for all elements will render the picture as a typical hidden-line solution. However, if ISILH(1) = 1, ISILH(2) = 1, ISILH(3) = 2, ISILH(4) = 2, ISILH(5) = 2...etc. then a silhouette of the elements 1 and 2 will be drawn and a different silhouette of the elements 3, 4 and 5 will be drawn. In other words, many different silhouettes each comprised of a family of elements can be rendered depending on their designation. This is quite useful if the polygons are many sided and very warped. A simple subdivision of a warped polygon into its many constituent triangles can lend itself to a very attractive rendering when these triangles are designated to belong to the same family.

ISLILH(J) = a constant non-zero integer for all the elements will render a silhouette of the entire scene, since in this case, there would be only one family.

The package does not assume any environmental coherence, so, in the interest of efficiency, it might be a good idea in some cases to use a preprocessor. This preprocessor should eliminate whole planes by looking at the normals to the planes, in the case of closed polyhedra or any combination of closed polyhedra.

The hidden-line software is not output specific, but may need a few small modifications which are consistent with the user's environment. All of these possible changes can be made in the subroutine PLT and are all quite trivial.

The user must take responsibility for opening and closing the plot device. The device must be opened before the first call to SKETCH and closed after the last call to SKETCH.

Using the standard CALCOMP commands as an example there would be:

```
CALL PLOTS (0,0,1)
```

and

```
CALL PLOT (0,0,999)
```

respectively.

Requirements will, of course, vary from installation to installation.

Note: If the polygons are very warped, then errors could occur. Also, the algorithm assumes at least 32-bit integers and floating-point words.

ERROR CODES

The hidden line package returns error codes after the final call to SKETCH. The codes are found in NC of the SKETCH calling sequence and have the following definitions:

0 = Normal return

–1 = Incorrect storage allocation in the array work. The correct storage allocation is placed in ICORE by SKETCH.

–2 = DV (distance from viewer) in common scalar is incorrect. SKETCH places the correct value in DV.

–3 = Both DV and storage allocation are in error. SKETCH places the correct values in ICORE and DV.

A WORKED EXAMPLE

In this example, a hexagonal cylinder is decomposed into input elements and relayed, via a FORTRAN calling program, to the hidden line package. The resulting plot is also shown. The origin is at the center of the figure.

The hexagonal cylinder, without its x, y, z coordinates, is shown in figure A-1. It is made up of six four-sided polygons and two six-sided polygons (the top and bottom), for a total of eight faces.

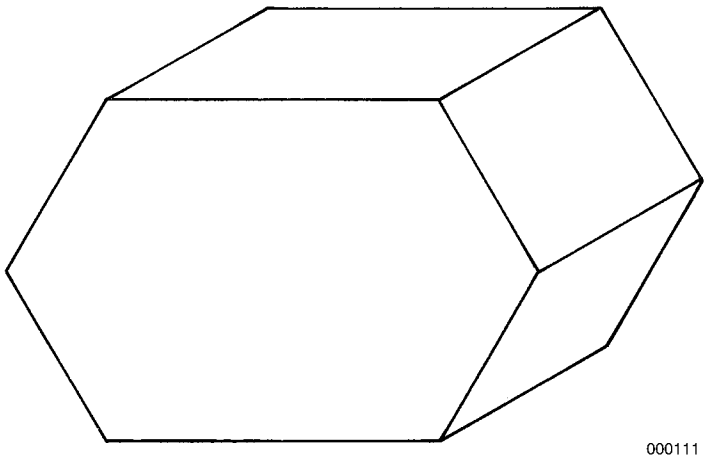


Figure A-1. Hexagonal cylinder.

The input x, y, and z coordinates for each face are as follows:

FACE 1

x	y	z
–5	0	3
–3	2	3
–3	2	–3
–5	0	–3
–5	0	3

FACE 2

x	y	z
–3	2	–3
–3	2	3
3	2	3
3	2	–3
–3	2	–3

FACE 3

x	y	z
3	2	–3
5	0	–3
5	0	3
3	2	3
3	2	–3

FACE 4

x	y	z
5	0	–3
5	0	3
3	–2	3
3	–2	–3
5	0	–3

FACE 5

x	y	z
3	–2	3
3	–2	–3
–3	–2	–3
–3	–2	3
3	–2	3

FACE 6

x	y	z
–5	0	–3
–5	0	3
–3	–2	3
–3	–2	–3
–5	0	–3

FACE 7 (Front)			FACE 8 (Back)		
x	y	z	x	y	z
–5	0	3	–5	0	–3
–3	2	3	–3	2	–3
3	2	3	3	2	–3
5	0	3	5	0	–3
3	–2	3	3	–2	–3
–3	–2	3	–3	–2	–3
–5	0	3	–5	0	–3

In the FORTRAN calling program (your program), these coordinates are read into dimensioned arrays x, y and z or any three arrays whose names are of the user's choosing.

In this example MNE (maximum number of edges) is six and NELEM (total number of elements in entire plot) is eight. Therefore, ICORE is calculated as follows:

ICORE = (25 + 5 * MNE + 4 * MNP)* NELEM

= (25 + 5 * 6 + 4 * 8)* 8

= 696

```
      SUBROUTINE STATUM(OJ,TMJ,XXX,TGM,RV,RVI,TGI,ZM,NNO,II,
     1H,IM,JXT,ZJ,NC,ZMI,CCC,LZ)
      DIMENSION X(50),Y(50)
      DIMENSION CCC(1),XXX(1),ZMI(1),TGM(1),RV(1),RVI(1),TGI(1),
     1NNO(1),H(1)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/DRH/ISILH(1)
      DATA GGK,F,EI/.005,.045,.2/
      OJ1=OJ
      TMJ1=TMJ
      IM=0
      J=1
      T=2.*F/J
      Y(1)=TMJ+F
      X(1)=OJ-F
      DO 97 I=1,J
      X(I+1)=X(I)+T
      Y(I+1)=TMJ+F
   97 CONTINUE
      DO 98 I=1,J
      X(J+1+I)=OJ+F
      Y(J+1+I)=Y(J+1+I-1)-T
   98 CONTINUE
      N=2*J+1
      DO 99 I=1,J
      Y(N+I)=Y(N)
      X(N+I)=X(N+I-1)-T
   99 CONTINUE
      N=3*J+1
      DO 100 I=1,J
      X(N+I)=X(N)
      Y(N+I)=Y(N+I-1)-T
  100 CONTINUE
      KS=4*J
    9 CONTINUE
C CHING
      KXX=0
      DO 80 L=1,KS
      OJ=X(L)
      TMJ=Y(L)
   12 CONTINUE
      D=EI*OJ-TMJ
      DO 65 JO=1,II
      JG=NNO(L4+JO)
      IF(ISILH(JXT).NE.ISILH(JG))GO TO 60
      IF((TMJ.GE.RV(L7+JG)).OR.(TMJ.LE.RVI(L8+JG)))GO TO 60
      IF((OJ.GE.TGI(L6+JG)).OR.(OJ.LE.TGM(L5+JG)))GO TO 60
      JS=L13+(JG-1)*LZ
      JT=L12+(JG-1)*5
      JN=0
      IF(JG.EQ.JXT)JN=H(6)
      NS=XXX(5+JT)-JN
      IB=NS*5
      DO 20 J=1,IB,5
      JJ=J+JS
      IF(CCC(JJ).NE.0)GO TO 15
      S=TMJ-CCC(JJ+3)
```

```
      S1=TMJ-CCC(JJ+4)
      DY=(-CCC(JJ+2)/CCC(JJ+1))-OJ
      GO TO 16
   15 CONTINUE
      S=OJ-CCC(JJ+3)
      S1=OJ-CCC(JJ+4)
      DY=(-CCC(JJ+2)-CCC(JJ+1)*OJ)-TMJ
   16 CONTINUE
      IF((ABS(DY).GE.GGK).OR.(S*S1.GT.0.))GO TO 20
      GO TO 80
   20 CONTINUE
      I=0
      DO 40 J=1,IB,5
      JJ=J+JS
      R=EI*CCC(JJ)+CCC(JJ+1)
      IF(R.EQ.0.)GO TO 40
      T=(-CCC(JJ+2)+CCC(JJ)*D)/R
      IF(T.LT.OJ)GO TO 40
   34 CONTINUE
      IF(CCC(JJ).NE.0.)GO TO 30
      T=EI*T-D
   30 CONTINUE
      IF((T-CCC(JJ+3))*(T-CCC(JJ+4)).GT.0.)GO TO 40
      I=I+1
   40 CONTINUE
      IF(I-(I/2)*2.EQ.0)GO TO 60
C  CHING
      IF(JG.NE.JXT)KXX=JG
      IM=1
      GO TO 80
   60 CONTINUE
   65 CONTINUE
      IM=0
      GO TO 90
  800 CONTINUE
   80 CONTINUE
      IM=1
C CHING
      IF(KXX.EQ.0)IM=0
   90 CONTINUE
      TMJ=TMJ1
      OJ=OJ1
      RETURN
      END
      SUBROUTINE SKETCH(X,Y,Z,NP,NC)
C
C
C
C     THIS SUBROUTINE SETS UP PEN MOTION INDICATORS.
C
C
      DIMENSION X(1),Y(1),Z(1)
      DIMENSION X1(160),Y1(160),Z1(160)
      COMMON/SCALAR/SCX,YAW,ROL,PIT,LZ,VP,JJJ,ICORE
      L=NP
      LI=NP
      IF(L.LE.2)GO TO 50
```

```
      LX=1
      NPX=NP
    1 NPX=NPX-1
      I=LX
      DO 8 M=I,NPX
      RX=0
      A=X(M+1)-X(M)
      B=Y(M+1)-Y(M)
      C=Z(M+1)-Z(M)
      IF(A.NE.0.)GO TO 8
      IF(B.NE.0.)GO TO 8
      IF(C.NE.0.)GO TO 8
      IX=M
      IX1=NPX
      DO 4 MX=IX,IX1
      X(MX)=X(MX+1)
      Y(MX)=Y(MX+1)
      Z(MX)=Z(MX+1)
    4 CONTINUE
      RX=1
      LX=M
      IF(LX.EQ.NPX)GO TO 10
      GO TO 1
    8 CONTINUE
   10 CONTINUE
      IF(RX.EQ.1.)NPX=NPX-1
      NP=NPX+1
      LI=NP
      IF(NP.LE.2)GO TO 50
      IX=0
      M1=0
      M=1
      IS=NP-1
   20 CONTINUE
      M=M+IX
      M1=M1+IX+1
      IF(M-1.EQ.LI)GO TO 70
C
C
C     SEARCH FOR MATCHING COORDINATES.
C
C
      DO 40 J=M,IS
      T=X(J+1)-X(M)
      U=Z(J+1)-Z(M)
      W=Y(J+1)-Y(M)
      IF(T.NE.0.)GO TO 40
      IF(W.NE.0.)GO TO 40
      IF(U.NE.0.)GO TO 40
      NP=NP+1
C
C     MATCH FOUND.....STORE COORDINATES AND SET SWITCH TO LIFT PEN
C     AND/OR END SET.
C
C
      IX=J+2-M
      IX1=J-IS+1
```

```
      I2=M1-1
      I3=M-1
      DO 30 IK=1,IX
      X1(I2+IK)=X(I3+IK)
      Y1(I2+IK)=Y(I3+IK)
      Z1(I2+IK)=Z(I3+IK)
   30 CONTINUE
      Z1(M1+IX)=-ISIGN(1,IX1)*9999.
      GO TO 20
   40 CONTINUE
   50 CONTINUE
      DO 60 J=1,LI
      X1(J)=X(J)
      Y1(J)=Y(J)
      Z1(J)=Z(J)
   60 CONTINUE
      NP=NP+1
      Z1(NP)=-9999.
   70 CONTINUE
      CALL LIN(X1,Y1,Z1,NP,NC)
      NP=L
C
C
C     RESET VALUE FOR MAXIMUM NUMBER OF EDGES IF ARGUMENT IS
C     COMPLETED.
C
C
      IF(VP.GT.0.)LZ=LZ/5
      RETURN
      END
      SUBROUTINE COEF(X,Y,Z,XXX,JXX,NC,NS,CCC,LZ)
C
C
C     THIS SUBROUTINE DETERMINES EQUATION OF LINES AND PLANES.
C
C
      DIMENSION CCC(1),XXX(1),X(1),Y(1),Z(1)
      DIMENSION ZZ(30),ZZZ(30)
      DIMENSION IB(5)
      DOUBLE PRECISION T,T1,E,F,A1,B1,C1,A2,B2,C2,COE(8)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/DRH/ISILH(1)
      DATA EPSS/.001/
      J1=JXX-1
      LE=0
      JA=L13+J1*LZ
      JF=L12+J1*5
      I=0
      J=1
   10 CONTINUE
C
C
C     SEARCH FOR MATCHING COORDINATES.
C
C
      I=I+1
      JN=I+1
```

```
      IF(X(JN)-X(I).NE.0.)GO TO 20
      IF(Y(JN)-Y(I).NE.0.)GO TO 20
      IF(Z(JN)-Z(I).NE.0.)GO TO 20
C
C
C     MATCH FOUND.....PROCEED IF LIST IS NOT EXHAUSTED.
C
C
      I=I+2
   20 CONTINUE
      IF(I.GT.NS)GO TO 70
C
C
C     DETERMINE EQUATION OF LINE-SEGMENTS.
C
C
      T=X(I+1)-X(I)
      T1=Y(I+1)-Y(I)
      IF((T.EQ.0.).AND.(T1.EQ.0.))GO TO 10
      IF(T.NE.0.)GO TO 30
   29 CONTINUE
      CCC(J+JA)=0
      CCC(J+1+JA)=1
      CCC(J+2+JA)=-X(I)
      GO TO 40
   30 CONTINUE
      CCC(J+JA)=1
      E=T1/T
      F=E*X(I)-Y(I)
      IF(DABS(E).GE.10000.)GO TO 29
      IF(DABS(E).LT.100.)GO TO 220
      YO=E*X(I+1)-F
      G=ABS(Y(I+1)-YO)
      IF(G.GT.EPSS)GO TO 29
  220 CONTINUE
      CCC(J+1+JA)=-E
      CCC(J+2+JA)=F
   40 CONTINUE
      IF(CCC(J+JA).NE.0.)GO TO 50
      CCC(J+3+JA)=Y(I)
      CCC(J+4+JA)=Y(I+1)
      GO TO 60
   50 CONTINUE
      CCC(J+3+JA)=X(I)
      CCC(J+4+JA)=X(I+1)
   60 CONTINUE
      J=J+5
      LE=LE+1
      ZZ(LE)=Z(I)
      ZZZ(LE)=Z(I+1)
      IF(LE.GT.3)GO TO 10
      IB(LE)=I
      GO TO 10
   70 CONTINUE
C
C
C     DETERMINE EQUATION OF PLANE.
```

```
C
      J=(J-1)/5
      XXX(JF+5)=J
      IF(NS.LE.3)GO TO 120
      A1=X(3)-X(1)
      B1=Y(3)-Y(1)
      C1=Z(3)-Z(1)
      A2=X(2)-X(1)
      B2=Y(2)-Y(1)
      C2=Z(2)-Z(1)
      COE(1)=B1*C2-B2*C1
      COE(2)=C1*A2-C2*A1
      COE(3)=A1*B2-A2*B1
      COE(4)=COE(1)*X(1)+COE(2)*Y(1)+COE(3)*Z(1)
      COE(4)=-COE(4)
      DO 110 J=1,4
  110 XXX(JF+J)=COE(J)
      IF(COE(3).NE.0.)GO TO 140
      J=1
      DO 25 K=1,LE
      CCC(JA+J)=ZZ(K)
      CCC(JA+J+1)=ZZZ(K)
      J=J+5
   25 CONTINUE
      IF(COE(1).NE.0.)I=1
      IF(COE(2).NE.0.)I=2
      P=COE(I)
      DO 26 K=1,4
   26 XXX(JF+K)=XXX(JF+K)/P
      GO TO 140
  120 CONTINUE
      XXX(JF+5)=1
      DO 130 IX=1,2
  130 XXX(JF+IX)=Z(IX)
      XXX(JF+3)=0
  140 CONTINUE
      RETURN
      END
      SUBROUTINE LIN(X,Y,Z,NP,NC)
C
C
C
C     THIS SUBROUTINE IS THE EXECUTIVE.
C
C
      INTEGER XIND(1),RCT(200)
      DIMENSION RRX(20)
      DIMENSION XXX(1),CCC(1)
      DIMENSION TGM(1),IN(1),ZM(1),ZMI(1),
     1TGMT(1),TGI(1),NNO(1),RV(1),RVI(1),NOCT(1)
      DIMENSION NGX(15),IADR(200)
      DIMENSION X21(500),Y21(500),Z21(500),IIA(500)
      DIMENSION XI(1000),YI(1000),ZI(1000),DI(1000)
      DIMENSION U(6),V(6),W(6),X(1),Y(1),Z(1),X1(10),Y1(10),Z1(10),H(15)
      DIMENSION ZMIN(1),YMIN(1),IN1(1),IN2(1),ICOUNT(150)
      DIMENSION XE(150),YE(150),XU(150),YU(150)
      DIMENSION I2(2),I3(2)
```

```
      DIMENSION COORD(1)
      DIMENSION SNDT(1)
      DIMENSION IND(1)
      DIMENSION REX(20)
      DIMENSION NEH(1),KEEP(1)
      DIMENSION IBEG(200),IEND(200),ICT(200),ICCT(200)
      DIMENSION XB(70),YB(70),ZB(70)
      DIMENSION YSUM(3),SUM1(3,3)
      COMMON/GO/WORK(1)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/SCALAR/SCX,YAW,ROLL,PIT,LZ, VP,JJJ,ICORE
      COMMON/HEDG/JAT,ME,JT,D4,D2,D1,D3,NS
      COMMON/DRH/ISILH(1)
      COMMON/ISIL/ISIL
      COMMON/INDX/JT1,JO
      EQUIVALENCE(WORK(1),XXX(1),CCC(1),IN(1),TGMT(1))
      EQUIVALENCE(WORK(1),ZM(1),ZMI(1),NNO(1),
     1TGM(1),TGI(1),RV(1),RVI(1),NOCT(1))
      EQUIVALENCE(WORK(1),IN1(1),IN2(1),YMIN(1),ZMIN(1))
      EQUIVALENCE(WORK(1),COORD(1))
      EQUIVALENCE(WORK(1),SNDT(1))
      EQUIVALENCE (WORK(1),IND(1),XIND(1))
      EQUIVALENCE(WORK(1),KEEP(1),NEH(1))
      IF(VP.LT.0.)GO TO 20
      HXX=.005
      AVA=0
      ISIL=0
      HX1=.001
      LC=10**6
      IXXX=0
      IF(SCX.LT.0.)IXXX=1
      SCX=ABS(SCX)
C
C     INITIALIZE VARIABLES.
C
      LZ=LZ*5
      AXMIN=10.**6
      AXMAX=-10.**6
      SW1=0
      VTX=VP+10
      SW=0
      IDAV=0
C
C     CALCULATE MAXIMUM ALLOWABLE ELEMENTS.
C
      IABC=ICORE/(25+LZ+4*JJJ)
      ISAVE=NC
      NC=IABC
      L5=0
      L6=NC
      L7=2*NC
      L8=3*NC
      L2=4*NC
      L3=5*NC
      L4=6*NC
      L00=7*NC
      L01=8*NC
```

```
      L1=9*NC
      L0=10*NC
      L9=11*NC
      L10=12*NC
      L11=13*NC
      L15=14*NC
      L16=15*NC
      L17=16*NC
      L18=19*NC
      L12=20*NC
      L13=25*NC
      L14=L13+LZ*NC
      DO 10 J=1,NC
      RVI(L8+J)=10**6
      TGM(L5+J)=10**6
      RV(L7+J)=-RVI(L8+J)
      TGI(L6+J)=-TGM(L5+J)
      NOCT(L9+J)=0
      ZM(L2+J)=RV(L7+J)
      ZMI(L3+J)=RVI(L8+J)
      XIND(L16+J)=0
      IND(L15+J)=J
      KEEP(L18+J)=0
   10 CONTINUE
      NC=ISAVE
      IK=0
      IKT=0
      PI=3.1416/180.
      IJBB=0
      VP=-VP
C
C     STORE EULERIAN ANGLES.
C
      XX=YAW*PI
      YY=ROLL*PI
      ZZ=PIT*PI
      COSY=COS(YY)
      SINY=SIN(YY)
      COSZ=COS(ZZ)
      SINZ=SIN(ZZ)
      COSX=COS(XX)
      SINX=SIN(XX)
   20 CONTINUE
      NT=NP-1
      IKK=IK+1
      IK=IK+1
C
C
C     SET ERROR CODES, IF NECESSARY.
C
C
      IF(IKK.LE.IABC)GO TO 30
      SW=1
   30 CONTINUE
      IF(NC.EQ.0)GO TO 40
      IDAV=1
      NC=-SW1
```

```
      IF(SW.EQ.0.)GO TO 50
      ICORE=(25+LZ+4*JJJ)*IKK
      NC=-(SW+SW1)
   40 CONTINUE
   50 CONTINUE
      DO 60 J=1,NP
      X21(J)=X(J)
      Y21(J)=Y(J)
      Z21(J)=Z(J)
      IIA(J)=0
   60 CONTINUE
C
C
C     STORE COORDINATES AND SET PEN POSITION WHENEVER ABS(Z)=9999.
C
C
      DO 70 J=1,NT
      IF(Z21(J).NE.9999.)GO TO 70
      IIA(J)=1
      IXU=J-2
      IBB=J-ISIGN(1,IXU)
      X21(J)=X21(IBB)
      Y21(J)=Y21(IBB)
      Z21(J)=Z21(IBB)
   70 CONTINUE
      IIA(NP)=1
      Z21(NP)=Z21(NT)
      Y21(NP)=Y21(NT)
      X21(NP)=X21(NT)
      JXX=IKK
      I=1
      VL=ABS(VP)
C
C
C     LOOP THAT DOES THE THREE DIMENSIONAL TRANSFORMATION ON THE
C     COORDINATES.
C
      JV=L14+(IKK-1)*4*JJJ
      JT=1
      JX2=JXX+L2
      JX3=JXX+L3
      JX4=JXX+L4
      JX5=JXX+L5
      JX6=JXX+L6
      JX7=JXX+L7
      JX8=JXX+L8
      JV1=JV+1
      JV2=JV+2
      JV3=JV+3
      DO 90 J=1,NP
      XJ=X21(J)
      YJ=Y21(J)
      ZJ=Z21(J)
      X21(J)=ZJ*(COSY*SINX)+XJ*(COSY*COSX)-YJ*SINY
      TW=YJ*COSY*COSZ
      TZ=XJ*(SINZ*SINX+SINY*COSZ*COSX)
      TY=ZJ*(-SINZ*COSX+SINY*COSZ*SINX)
```

```
      Y21(J)=TZ+TW+TY
      PT=YJ*COSY*SINZ
      PK=ZJ*(COSZ*COSX+SINY*SINZ*SINX)
      PS=XJ*(-COSZ*SINX+SINY*SINZ*COSX)
      Z21(J)=PK+PS+PT
      RV(JX7)=AMAX1(RV(JX7),Y21(J))
      RVI(JX8)=AMIN1(RVI(JX8),Y21(J))
      TGI(JX6)=AMAX1(TGI(JX6),X21(J))
      TGM(JX5)=AMIN1(TGM(JX5),X21(J))
      ZM(JX2)=AMAX1(ZM(JX2),Z21(J))
      ZMI(JX3)=AMIN1(ZMI(JX3),Z21(J))
      COORD(JV+JT)=X21(J)
      COORD(JV1+JT)=Y21(J)
      COORD(JV2+JT)=Z21(J)
      COORD(JV3+JT)=IIA(J)
      JT=JT+4
   90 CONTINUE
      NOCT(L9+IKK)=NOCT(L9+IKK)+NP
      NS=NP
      AVA=AVA+(TGI(JX6)-TGM(JX5))*(RV(JX7)-RVI(JX8))
      IF(IXXX.EQ.1)GO TO 95
C
C
C     CALL SUBROUTINE WHICH CALCULATES BOTH THE EQUATIONS OF THE LINE
C     SEGMENTS AND POLYGONS.
C
      CALL COEF(X21,Y21,Z21,XXX,JXX,NC,NS,CCC,LZ)
C
C     CHECKS TO SEE IF ALL ELEMENTS(SETS) HAVE BEEN PASSED.
C
   95 CONTINUE
      IF(IDAV.EQ.1)GO TO 100
      GO TO 400
  100 CONTINUE
      AVA=AVA/IKK
      DO 1301 J=1,200
      ICCT(J)=0
      ICT(J)=0
      RCT(J)=J-1
      IBEG(J)=1
      IEND(J)=0
 1301 CONTINUE
      AMAXX=-999999.
      AMAXY=-999999.
      AMINX=999999.
      AMINY=999999.
      DO 1400 J=1,IKK
      AMAXX=AMAX1(AMAXX,TGI(L6+J))
      AMAXY=AMAX1(AMAXY,RV(L7+J))
      AMINX=AMIN1(AMINX,TGM(L5+J))
      AMINY=AMIN1(AMINY,RVI(L8+J))
 1400 CONTINUE
      IAUG=50+((IKK/10000)*2)
      TMAX=(AMAXX-AMINX)*(AMAXY-AMINY)
      IBL=TMAX/AVA
      IBL=IBL/4
C
```

```
C     DETERMINES THE NUMBER OF GRID POINTS IN THE GRID.
C
C
      IF(IXXX.EQ.1)GO TO 19990
      EN=IKK
      K=(ALOG(EN)/ALOG(2.))+.01
      K=K+IAUG
      K=MIN0(K,IBL)
      IF(K.LE.1)K=1
      IF(K.GE.199)K=199
      K=MAX0(K,4)
      T=K
      R=(T**.5)
      KS=R+.5
      S=T/KS
      MS=S+.5
      N=KS*MS
      MND=N+1
      XMD=MND
      T=3./(MND-1)
      IGY=T*IKK
      K=KS
      K1=MS
      CRX=(AMAXX-AMINX)/K
      CRY=(AMAXY-AMINY)/K1
C
C
C     DETERMINES THE RELEVANT ELEMENTS VIA THE GRID BLOCKS.
C
      DO 93 J=1,IKK
      IA=0
      XMAT=TGI(L6+J)
      XMIT=TGM(L5+J)
      YMAT=RV(L7+J)
      YMIT=RVI(L8+J)
      M=0
      DO 91 I=1,K1
      A1=YMAT-(I*CRY+AMINY)
      A=A1+CRY
      B1=YMIT-(I*CRY+AMINY)
      B=B1+CRY
      A2=A*A1
      B2=B*B1
      DO 92 L=1,K
      M=M+1
      S1=XMAT-(L*CRX+AMINX)
      S=S1+CRX
      R1=XMIT-(L*CRX+AMINX)
      R=R1+CRX
      IF((S.LT.0.).OR.(R1.GT.0.))GO TO 92
      IF((A.LT.0.).OR.(B1.GT.0.))GO TO 92
      IF((S*S1.GT.0.).OR.(R*R1.GT.0.))GO TO 94
      IF((A2.GT.0.).OR.(B2.GT.0.))GO TO 94
      GO TO 93
   94 CONTINUE
      ICCT(M)=ICCT(M)+1
   92 CONTINUE
```

```
  91 CONTINUE
  93 CONTINUE
     IADR(1)=0
     MM=K*K1
     I8=3*IKK
     DO 8977 I=2,MM
     IADR(I)=ICCT(I-1)+IADR(I-1)
     IF(IADR(I)+ICCT(I).LE.I8)GO TO 8977
     DO 6666 K9=I,MM
     ICCT(K9)=-1
6666 CONTINUE
     GO TO 5555
8977 CONTINUE
5555 CONTINUE
     DO 191 J=1,MM
     IF(ICCT(J).GE.0)ICCT(J)=0
 191 CONTINUE
     DO 3 J=1,IKK
     IA=0
     XMAT=TGI(L6+J)
     XMIT=TGM(L5+J)
     YMAT=RV(L7+J)
     YMIT=RVI(L8+J)
     J16=L16+J
     M=0
     DO 1 I=1,K1
     A1=YMAT-(I*CRY+AMINY)
     A=A1+CRY
     B1=YMIT-(I*CRY+AMINY)
     B=B1+CRY
     A2=A*A1
     B2=B*B1
     DO 2 L=1,K
     M=M+1
     S1=XMAT-(L*CRX+AMINX)
     S=S1+CRX
     R1=XMIT-(L*CRX+AMINX)
     R=R1+CRX
     IF((S.LT.0.).OR.(R1.GT.0.))GO TO 2
     IF((A.LT.0.).OR.(B1.GT.0.))GO TO 2
     IF((S*S1.GT.0.).OR.(R*R1.GT.0.))GO TO 4
     IF((A2.GT.0.).OR.(B2.GT.0.))GO TO 4
     XIND(J16)=M
     GO TO 3
   4 CONTINUE
     IA=IA+1
     IF(IA.LE.4)GO TO 8000
     XIND(J16)=0
     GO TO 8001
8000 CONTINUE
     XIND(J16)=XIND(J16)+M*(MND**(IA-1))
8001 CONTINUE
     IF(ICCT(M).LT.0)GO TO 2
     ICCT(M)=ICCT(M)+1
     JK=IADR(M)+ICCT(M)+L17
     NEH(JK)=J
   2 CONTINUE
```

```
    1 CONTINUE
    3 CONTINUE
      CALL VSRT1(XIND(L16+1),IK,IND(L15+1))
      SW=0
      L=1
      DO 5 I=1,IKK
   11 CONTINUE
      IF(XIND(L16+I).NE.RCT(L))GO TO 6
      SW=SW+1
      IF(SW.EQ.1.)LT=I
      ICT(L)=ICT(L)+1
      GO TO 5
    6 CONTINUE
      IF(SW.NE.0.)GO TO 8
      L=L+1
      GO TO 11
    8 CONTINUE
      IBEG(L)=LT
      IEND(L)=LT+ICT(L)-1
      SW=0
      IF(XIND(L16+I).GE.MND)GO TO 2110
      L=L+1
      GO TO 11
    5 CONTINUE
      IBEG(L)=LT
      IEND(L)=LT+ICT(L)-1
 2110 CONTINUE
      DO 2111 J=1,IKK
      SNDT(L4+J)=IND(L15+J)
 2111 CONTINUE
      CALL VSRTR(SNDT(L4+1),IK,XIND(L16+1))
      EN=IKK
      IGX=(ALOG(EN)/ALOG(2.))+1.
      DO 105 J=1,IGX
      RRX(J)=2**(IGX-J)
  105 CONTINUE
19990 CONTINUE
      IJ=0
      X1(3)=(AMAXX+AMINX)/2
      Y1(3)=(AMAXY+AMINY)/2
      X1(4)=SCX
      Y1(4)=SCX
      IF(IXXX.EQ.1)GO TO 18880
      DO 115 J=1,IKK
      IN(L11+J)=J
      IN1(L0+J)=J
      TGMT(L10+J)=TGM(L5+J)
      YMIN(L1+J)=RVI(L8+J)
  115 CONTINUE
C
C     CALL SUBROUTINE WHICH WILL SORT ON X,Y AND Z.
C
      CALL VSRTR(TGMT(L10+1),IK,IN(L11+1))
      CALL VSRTR(YMIN(L1+1),IK,IN1(L0+1))
      H(8)=0
18880 CONTINUE
      L10W=L10-1
```

```
      L1W=L1-1
      L01W=L01-1
      TZQ=.05*IKK
      DO 395 J=1,IKK
      N9=0
      IF(ISILH(J).NE.0)N9=2
      IF((IJBB.EQ.0).AND.(J.GE.TZQ))N9=0
      N7=J-1
      JXT=J
      KS=IKK
      JJ=L14+N7*4*JJJ
      JH=1
      II=0
      IXR=NOCT(L9+J)
      NIT=0
      JT=L12+5*N7
      JT1=JT
      JO=L13+LZ*N7
      IF(IXXX.EQ.1)GO TO 200
      NS=XXX(5+JT)
      DO 1910 IC=1,NS
 1910 IIA(400+IC)=0
      D1=XXX(JT+1)
      D2=XXX(JT+2)
      D3=XXX(JT+3)
      D4=XXX(JT+4)
      IF(D3.EQ.0.)GO TO 108
      Q1=D1/D3
      Q2=D2/D3
      Q4=D4/D3
  108 CONTINUE
      I9=0
      NG=NS*5
      I=0
      JD=1
      JW=JJ+1
      JW1=JJ+2
      DO 121 I=1,NS
      XE(I)=COORD(JJ+JD)
      YE(I)=COORD(JW+JD)
      DI(I)=COORD(JW1+JD)
      JD=JD+4
  121 CONTINUE
  122 CONTINUE
C     THOSE OTHER ELEMENTS WHICH COULD POSSIBLY HIDE SOME PORTION
C     OF THE GIVEN ELEMENT.
C
C
C
      K=2**IGX
      K1=K
      K2=K
C
C     DO LOGARITHMIC SEARCH TO DETERMINE RELEVANT ELEMENTS.
C
      TGII=TGI(L6+J)
      RVV=RV(L7+J)
```

```
      ZMII=ZMI(L3+J)
      TGMM=TGM(L5+J)
      RVII=RVI(L8+J)
      S=-1
      DO 131 I=1,IGX
      K=K+SIGN(RRX(I),S)
      IF(K.GT.IKK)K=IKK
      S=TGII-TGMT(L10+K)
      IF(S*(TGII-TGMT(L10W+K)).LT.0.)GO TO 132
  131 CONTINUE
      K=IKK
  132 CONTINUE
      S=-1
      DO 133 I=1,IGX
      K1=K1+SIGN(RRX(I),S)
      IF(K1.GT.IKK)K1=IKK
      S=RVV-YMIN(L1+K1)
      IF(S*(RVV-YMIN(L1W+K1)).LT.0.)GO TO 134
  133 CONTINUE
      K1=IKK
  134 CONTINUE
C
C     RETRIEVE THE RELEVANT ELEMENTS DETERMINED FROM SCHEME 1.
C
 8181 CONTINUE
      IR=XIND(L16+J)
      IF(IR.EQ.0)GO TO 1270
      VX=IR
      T=ALOG(VX)
      IF(IR.LE.LC)GO TO 1800
      E=LC
      LG=IR/LC
      MU=MOD(IR,LC)
      UX=LG+(MU/E)
      T=ALOG(UX)+ALOG(E)
 1800 CONTINUE
      IXT=0
      IEXP=(T/ALOG(XMD))+1
      DO 8004 L=1,IEXP
      JCZ=MND**(IEXP-L)
      IV=IR/JCZ
      IR=IR-IV*JCZ
      IV=IV+1
      IV1=IV-1
      IF(ICCT(IV1).EQ.0)GO TO 4000
      IF(ICCT(IV1).GT.0)GO TO 4001
      GO TO 1270
 4001 CONTINUE
      KE=ICCT(IV1)
      IL=0
      JTT=IADR(IV1)+L17
      JJG=L4+IXT
      DO 4003 I=1,KE
      KV=NEH(I+JTT)
      IF(KEEP(L18+KV).EQ.J)GO TO 4003
      IL=IL+1
      NNO(JJG+IL)=KV
```

```
      KEEP(L18+KV)=J
 4003 CONTINUE
      IXT=IXT+IL
 4000 CONTINUE
      IX=IBEG(IV)
      IX1=IEND(IV)
      LJ=L4+IXT-IX+1
      DO 1170 I=IX,IX1
 1170 NNO(LJ+I)=IND(L15+I)
      IXT=IXT+IX1-IX+1
 8004 CONTINUE
      KS=IXT
 1270 CONTINUE
      IM=MIN0(K,K1)
C
C     PICK MINIMUM COUNT FROM BOTH SCHEMES.
C
      IF(KS.LT.IM)GO TO 129
      IF(IM.EQ.K)GO TO 1001
      IF(IM.EQ.K1)GO TO 1002
      KS=I1
      IJ1=L00+IKK+1
      DO 1003 I=1,KS
 1003 NNO(L4+I)=IN2(IJ1-I)
      GO TO 129
 1001 CONTINUE
      KS=K
      DO 1004 I=1,KS
 1004 NNO(L4+I)=IN(L11+I)
      GO TO 129
 1002 CONTINUE
      KS=K1
      DO 1006 I=1,KS
 1006 NNO(L4+I)=IN1(L0+I)
  129 CONTINUE
      DO 170 I=1,KS
      JB=NNO(L4+I)
      IF(ISILH(J).EQ.0)GO TO 265
      IF(ISILH(J).NE.ISILH(JB))GO TO 265
      IF((RVV.LT.RVI(L8+JB)).OR.(RVII.GT.RV(L7+JB)))GO TO 170
      IF((TGMM.GT.TGI(L6+JB)).OR.(TGII.LT.TGM(L5+JB)))GO TO 170
      JS=L12+(JB-1)*5
      IF(XXX(3+JS).EQ.0)GO TO 170
      IF(J.EQ.JB)GO TO 166
      GO TO 2399
  265 CONTINUE
      IF((TGMM.GE.TGI(L6+JB)).OR.(TGII.LE.TGM(L5+JB)))GO TO 170
      IF((RVV.LE.RVI(L8+JB)).OR.(RVII.GE.RV(L7+JB)))GO TO 170
      IF(ZMII.GE.ZM(L2+JB))GO TO 170
      IF(J.EQ.JB)GO TO 170
      JS=L12+(JB-1)*5
      IF(XXX(3+JS).EQ.0.)GO TO 170
 2399 CONTINUE
      IF(D3.EQ.0.)GO TO 166
      NV=XXX(5+JS)
C
C     TEST TO SEE IF ALL VERTICES LIE EITHER BEHIND OR IN FRONT OF
```

```
C     THE GIVEN POLYGON.
C
      IT=0
      JD=1
      JI=L14+(JB-1)*4*JJJ
      JI1=JI+1
      JI2=JI+2
      DO 145 M=1,NV
      XU(M)=COORD(JI+JD)
      YU(M)=COORD(JI1+JD)
      XI(M)=COORD(JI2+JD)
      JD=JD+4
      ZS1=-(Q4+Q2*YU(M)+Q1*XU(M))
      IF(ABS(XI(M)-ZS1).LT.HXX)GO TO 145
      IT=IT+1
      ICOUNT(IT)=0
      IF(XI(M).GT.ZS1)ICOUNT(IT)=1
  145 CONTINUE
C
C
C     TESTS FOR SEMI-RELEVANT PLANES.  THAT IS, NEGATIVE INDEXES
C     INDICATE ELEMENT IS TO BE USED FOR VISIBILITY TEST, BUT NOT FOR
C     INTERSECTION LINE DETERMINATION.
C
C
C     IF ALL EDGES HAVE ALREADY BEEN DRAWN,THEN DISREGARD THE FOLLOWING C
C
      IF((ISILH(J).NE.ISILH(JB)).AND.(ISILH(J).NE.0))GO TO 1212
      IF(J.LT.JB)GO TO 1650
      IF(I9.EQ.NS)GO TO 1650
      MG=0
      NV1=NV-1
      NS1=NS-1
      DO 1630 L=1,NS
      DO 1580 IX=1,NV
      IF(ABS(XI(IX)-DI(L)).GT.HXX)GO TO 1580
      IF((ABS(YU(IX)-YE(L)).GT.HXX).OR.(ABS(XU(IX)-XE(L)).GT.
     1HXX))GO TO 1580
      IF(MG.EQ.2)GO TO 1640
      MG=MG+1
      I2(MG)=L
      I3(MG)=IX
      GO TO 1630
 1580 CONTINUE
 1630 CONTINUE
 1640 CONTINUE
      IF(MG.NE.2)GO TO 1650
      IR=IABS(I3(2)-I3(1))
      IR1=IABS(I2(2)-I2(1))
      IF((IR.NE.1).AND.(IR.NE.NV1))GO TO 1650
      IF((IR1.NE.1).AND.(IR1.NE.NS1))GO TO 1650
      IX=MAX0(I2(1),I2(2))
      IF(IR1.EQ.1)IX=IX-1
      IIA(400+IX)=1
      I9=I9+1
 1650 CONTINUE
 1212 CONTINUE
```

```
      L=0
      DO 150 M=1,IT
  150 L=L+ICOUNT(M)
      IF((ISILH(J).NE.0).AND.(ISILH(JB).EQ.ISILH(J)))GO TO 165
      IF(L.EQ.0)GO TO 170
      IF(II.GT.N9)GO TO 165
C
C
C     INTERROGATE THE RELATIONSHIP OF THE CANDIDATE POLYGON TO THE
C     GIVEN POLYGON BY DETERMINING IF THE PROJECTION OF ONE POLYGON
C     CAN BE SEPARATED BY AN EDGE FROM THE OTHERS PROJECTION
C
C
      JK=L13+(JB-1)*LZ
      E3=XXX(3+JS)
      E1=XXX(1+JS)
      E4=XXX(4+JS)
      E2=XXX(2+JS)
      SD=0
      I3(1)=JK
      I3(2)=JO
      I2(1)=NV*5
      I2(2)=NS*5
      DO 164 KU=1,2
      IS=I3(KU)
      IB=I2(KU)
      DO 163 LL=1,IB,5
  151 CONTINUE
      IF(SD.EQ.1.)GO TO 152
      A=D2*E3-E2*D3
      B=D1*E3-E1*D3
      C=D4*E3-E4*D3
      GO TO 153
  152 CONTINUE
      A=CCC(LL+IS)
      B=CCC(LL+IS+1)
      C=CCC(LL+IS+2)
  153 CONTINUE
      IF((A.EQ.0.).AND.(B.EQ.0.))GO TO 162
      IF(A.NE.0.)GO TO 154
      A=0
      C=C/B
      B=1
      GO TO 155
  154 CONTINUE
      B=B/A
      C=C/A
      A=1
  155 CONTINUE
      M=0
      R1=0
      DO 158 IX=1,NV
      M=M+1
      YG=YU(M)
      IF(A.NE.0.)GO TO 156
      DY=-C/B
      YG=XU(M)
```

```
      GO TO 157
  156 CONTINUE
      DY=-C-B*XU(M)
  157 IF(ABS(DY-YG).LT.HXX)GO TO 158
      R=YG-DY
      IF(R*R1.LT.0.)GO TO 162
      R1=R
  158 CONTINUE
      M=0
      R2=0
      DO 161 IX=1,NS
      M=M+1
      YG=YE(M)
      IF(A.NE.0.)GO TO 159
      DY=-C/B
      YG=XE(M)
      GO TO 160
  159 CONTINUE
      DY=-C-B*XE(M)
  160 IF(ABS(DY-YG).LT.HXX)GO TO 161
      R=YG-DY
      IF(R*R2.LT.0.)GO TO 162
      R2=R
  161 CONTINUE
      IF(R1*R2.LT.0.)GO TO 170
  162 CONTINUE
      IF(SD.NE.0.)GO TO 163
      SD=1
      GO TO 151
  163 CONTINUE
  164 CONTINUE
  165 CONTINUE
      IF((L.EQ.IT).OR.(L.EQ.0))JB=-JB
  166 CONTINUE
      II=II+1
      NNO(L4+II)=JB
  170 CONTINUE
      IF(II.LE.2)IJBB=1
      JAT=-4
C  CHING
C       IF(IXR.LE.3)GO TO 200
      IF(II.EQ.0)GO TO 190
C
C
C     CALL SUBROUTINE WHICH SOLVES FOR THE LINES OF INTERSECTION,IF ANY,
C     OF THE JTH ELEMENT WITH OTHER ELEMENTS.
C
C
      CALL SOLVE(IXR,J,XXX,CCC,II,NNO,NIT,X21,Y21,Z21,IIA,NC,ZM,ZMI,LZ)
  190 CONTINUE
  200 CONTINUE
      IJ1=JJ+1
      IJ2=JJ+2
      IJ3=JJ+3
      DO 210 JM=1,IXR
      X21(JM)=COORD(JH+JJ)
      Y21(JM)=COORD(JH+IJ1)
```

```
      Z21(JM)=COORD(JH+IJ2)
      IIA(JM)=COORD(JH+IJ3)
      JH=JH+4
  210 CONTINUE
      IXR=IXR+3*NIT
      IF(II.EQ.0)GO TO 220
      IF(IXXX.NE.1)GO TO 240
  220 CONTINUE
      DO 230 JM=1,IXR
      IF(IXXX.EQ.1)GO TO 1993
      IF(JM.GE.IXR-1)GO TO 1993
      IF(IIA(400+JM).EQ.1)IIA(JM+1)=1
 1993 CONTINUE
      X1(2)=X21(JM)
      Y1(2)=Y21(JM)
      IM=IABS(IIA(JM))
      CALL PLT(X1,Y1,IJ,IM)
      IF((JM.NE.IXR).AND.(IIA(JM+1).EQ.0))CALL PLT(X1,Y1,IJ,0)
  230 CONTINUE
      GO TO 390
  240 CONTINUE
      JX=1
  250 CONTINUE
C
C
C     PLOTS IF IIA(JX+1) IS EQUAL TO 1.
C
      IF((IIA(JX).EQ.0).AND.(IIA(JX+1).EQ.0))GO TO 260
      IF(IIA(JX+1).NE.-1)GO TO 251
      IIA(JX+1)=0
      IM=1
      JAT=JAT+5
      JX=JX-1
      GO TO 252
  251 CONTINUE
      IM=IIA(JX+1)
  252 CONTINUE
      X1(2)=X21(JX+1)
      Y1(2)=Y21(JX+1)
      CALL PLT(X1,Y1,IJ,IM)
      JX=JX+2
      IF(JX.GE.IXR)GO TO 390
      GO TO 250
  260 CONTINUE
      IJ=0
      JAT=JAT+5
      ME=0
      IF(JX.GT.NS)GO TO 255
      IF(IIA(400+JX).EQ.1)GO TO 381
  255 CONTINUE
C
C     CALL SUBROUTINE WHICH DETERMINES THE POINTS OF INTERSECTIONS
C     OF THE LINES OF THE JTH SET WITH THE RELEVANT LINES AND PLANES
C     OF OTHER ELEMENTS.
C
      H(6)=NIT
      CALL CHECK(XXX,CCC,NNO,J,II,NC,XI,YI,NGX,ZM,ZMI,RV,RVI,TGM,TGI,
```

```
     1ZI,LZ)
      NG=NGX(1)+2
      IMB=JX+1
      XI(1)=X21(JX)
      YI(1)=Y21(JX)
      ZI(1)=Z21(JX)
      XI(NG)=X21(IMB)
      YI(NG)=Y21(IMB)
      ZI(NG)=Z21(IMB)
  880 CONTINUE
 1101 CONTINUE
C
C
C     THE FOLLOWING CODE SORTS THE INTERSECTION POINTS IN ASCENDING
C     ORDER OF OCCURRENCE AND THEN SHRINKS THE LIST IF REDUNDANCY EXISTS.
C
      IF(NG.LE.3)GO TO 340
      NI=NG-2
      NII=NI
      DO 270 M=1,NG
      DI(M)=(XI(M)-XI(1))**2+(YI(M)-YI(1))**2
  270 CONTINUE
      DO 290 M=2,NI
      DO 280 MX=2,NII
      IF(DI(MX).LE.DI(MX+1))GO TO 280
      IW=MX+1
      HOLD=DI(MX)
      HOLD1=XI(MX)
      HOLD2=YI(MX)
      HOLD3=ZI(MX)
      XI(MX)=XI(IW)
      YI(MX)=YI(IW)
      ZI(MX)=ZI(IW)
      DI(MX)=DI(IW)
      DI(IW)=HOLD
      XI(IW)=HOLD1
      YI(IW)=HOLD2
      ZI(IW)=HOLD3
  280 CONTINUE
      NII=NII-1
  290 CONTINUE
      LX=1
      NPX=NG
  300 NPX=NPX-1
      I=LX
      DO 320 M=I,NPX
      RX=0
      T=SQRT((XI(M)-XI(M+1))**2+(YI(M)-YI(M+1))**2)
      IF(T.GT.HX1)GO TO 320
      IX=M
      IX1=NPX
      DO 310 MX=IX,IX1
      IW=MX+1
      XI(MX)=XI(IW)
      YI(MX)=YI(IW)
      ZI(MX)=ZI(IW)
  310 CONTINUE
```

```
      RX=1
      LX=M
      IF(LX.EQ.NPX)GO TO 330
      GO TO 300
  320 CONTINUE
  330 CONTINUE
      IF(RX.EQ.1.)NPX=NPX-1
      NG=NPX+1
  340 CONTINUE
C
C
C     THIS CODE DETERMINES THE STATUS(VISIBILITY) OF EVERY OTHER POINT
C     AS SUGGESTED BY THE THEOREM IN TECHNICAL REPORT, NASA/RP-1085.
C
      DO 350 L=1,NG,2
      OJ=XI(L)
      TMJ=YI(L)
      ZJ=ZI(L)
      CALL STATUS(OJ,TMJ,XXX,TGM,RV,RVI,TGI,ZM,NNO,II,H,IM,JXT,
     1ZJ,NC,ZMI,CCC,LZ)
      DI(L)=IM
  350 CONTINUE
      JII=NG-1
      DO 370 L=1,NG,2
      IF(L.EQ.NG)GO TO 370
      IF(L.EQ.JII)GO TO 360
      IF(DI(L)+DI(L+2).NE.2.)GO TO 360
      DI(L+1)=DI(L)
      GO TO 370
  360 CONTINUE
      MN=L+1
      OJ=XI(MN)
      TMJ=YI(MN)
      ZJ=ZI(MN)
      CALL STATUS(OJ,TMJ,XXX,TGM,RV,RVI,TGI,ZM,NNO,II,H,IM,
     1JXT,ZJ,NC,ZMI,CCC,LZ)
      DI(MN)=IM
  370 CONTINUE
C
C
C     THE FOLLOWING CODE ACTUALLY PLOTS THE POINTS ON A GIVEN LINE
C     GOVERNED BY THE VALUE(IM) RETURNED BY STATUS SUBROUTINE.
C     1 MEANS INVISIBLE,...0 MEANS VISIBLE.
C
C
  379 CONTINUE
      DO 380 L=1,NG
      X1(2)=XI(L)
      Y1(2)=YI(L)
      IM=DI(L)
      CALL PLT(X1,Y1,IJ,IM)
      IF(L.EQ.NG)GO TO 380
      IF(DI(L)+DI(L+1).GT.0.)GO TO 380
      H(8)=1
      MN=L+1
      OJ=(XI(L)+XI(MN))/2
      TMJ=(YI(L)+YI(MN))/2
```

```
      ZJ=(ZI(L)+ZI(MN))/2
      CALL STATUS(OJ,TMJ,XXX,TGM,RV,RVI,TGI,ZM,NNO,II,H,IM,JXT,ZJ,NC,
     1ZMI,CCC,LZ)
      H(8)=0
      IF(IM.EQ.0)GO TO 380
      X1(2)=OJ
      Y1(2)=TMJ
      CALL PLT(X1,Y1,IJ,IM)
  380 CONTINUE
  381 CONTINUE
      JX=JX+1
      GO TO 250
  390 CONTINUE
C
C     DECREMENTS THE COUNT OF THE NUMBER OF LINES IN THE JTH SET
C     SINCE THE LINES OF INTERSECTIONS WERE ADDED TO THIS ELEMENT
C     BY THE SUBROUTINE SOLVE.
C
      XXX(5+JT)=XXX(5+JT)-NIT
  395 CONTINUE
  400 CONTINUE
      RETURN
      END
      SUBROUTINE STATUS(OJ,TMJ,XXX,TGM,RV,RVI,
     1TGI,ZM,NNO,II,H,IM,JXT,ZJ,NC,ZMI,CCC,LZ)
C
C
C
C     THIS SUBROUTINE DETERMINES THE VISIBILITY OF AN ARBITRARY POINT
C     BY DRAWING A LINE FROM THE POINT IN QUESTION TO INFINITY AND
C     COUNTING THE NUMBER OF TIMES IT CROSSES THE BOUNDARIES OF A
C     RELEVANT ELEMENT.
C
C
C
      DIMENSION CCC(1),XXX(1)
      DIMENSION ZMI(1),TGM(1),RV(1),RVI(1),
     1TGI(1),ZM(1),NNO(1),H(1)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/DRH/ISILH(1)
      DATA GGK,GGJ,EI/.005,.015,.2/
      IM=0
      D=EI*OJ-TMJ
      DO 60 JO=1,II
      JG=NNO(L4+JO)
C
C     PRELIMINARY CHECK TO SEE IF THE POINT IS OUTSIDE THE BOUNDARY
C     BOXES IN THE X,Y,Z DIMENSIONS.
C
      IF((TMJ.GE.RV(L7+JG)).OR.(TMJ.LE.RVI(L8+JG)))GO TO 60
      IF((OJ.GE.TGI(L6+JG)).OR.(OJ.LE.TGM(L5+JG)))GO TO 60
      IF(ZJ.GE.ZM(L2+JG))GO TO 60
      JT=L12+(JG-1)*5
      ZS=-(XXX(4+JT)+XXX(2+JT)*TMJ+XXX(1+JT)*OJ)/XXX(JT+3)
      IF(ABS(ZJ-ZS).LT.GGJ)GO TO 60
      IF((ZJ.GT.ZS).OR.(JXT.EQ.JG))GO TO 60
      IB=XXX(5+JT)*5
```

```
      JS=L13+(JG-1)*LZ
      DO 20 J=1,IB,5
      JJ=JS+J
      IF(CCC(JJ).EQ.0)GO TO 15
      S=OJ-CCC(JJ+3)
      S1=OJ-CCC(JJ+4)
      DY=(-CCC(JJ+2)-CCC(JJ+1)*OJ)-TMJ
      GO TO 16
   15 CONTINUE
      DY=(-CCC(JJ+2)/CCC(JJ+1))-OJ
      S=TMJ-CCC(JJ+3)
      S1=TMJ-CCC(JJ+4)
   16 CONTINUE
      IF((ABS(DY).LT.GGK).AND.(S*S1.LE.0.))GO TO 61
   20 CONTINUE
C
C
C     THE FOLLOWING CODE COUNTS THE INTERSECTIONS OF BOUNDARIES
C     OF A GIVEN ELEMENT WITH THE INFINITE LINE AND CHECKS,IF INSIDE
C     OF THE BOUNDARY, WHETHER OR NOT THE POINT IS BEHIND OR IN FRONT
C     OF THE ELEMENT.
C
C
      I=0
      DO 40 J=1,IB,5
      JJ=JS+J
      R=EI*CCC(JJ)+CCC(JJ+1)
      IF(R.EQ.0.)GO TO 40
      T=(-CCC(JJ+2)+CCC(JJ)*D)/R
      IF(T.LT.OJ)GO TO 40
      IF(CCC(JJ).NE.0.)GO TO 30
      T=EI*T-D
   30 CONTINUE
      IF((T-CCC(JJ+3))*(T-CCC(JJ+4)).GT.0.)GO TO 40
   35 CONTINUE
      I=I+1
   40 CONTINUE
      IF(I-(I/2)*2.EQ.0)GO TO 60
   50 CONTINUE
      IM=1
      GO TO 70
   61 CONTINUE
      IF(H(8).NE.1)GO TO 60
      IF((ISILH(JXT).EQ.ISILH(JG)).AND.(ISILH(JXT).NE.0))GO TO 60
      IF(ZJ.LT.ZMI(L3+JG))GO TO 50
   60 CONTINUE
      IF((ISILH(JXT).EQ.0).OR.(H(8).NE.1.))GO TO 70
      CALL STATUM(OJ,TMJ,XXX,TGM,RV,RVI,TGI,ZM,NNO,II,H,IM,JXT,
     1ZJ,NC,ZMI,CCC,LZ)
   70 CONTINUE
      RETURN
      END
      SUBROUTINE PLT(X1,Y1,IJ,IM)
C
C
C     PLOTS POINTS GOVERNED BY THE VALUE OF IM.
C
```

```
C
C     NOTE THAT CALL PLOT(X,Y,2) MEANS MOVE PEN FROM THE CURRENT
C     POSITION TO THE POINT,(X,Y),WITH THE PEN DOWN.
C
C
C     CALL PLOT(X,Y,3) MEANS MOVE THE PEN FROM THE CURRENT POSITION
C     TO THE POINT,(X,Y), WITH THE PEN UP.
C
C
      DIMENSION X1(1),Y1(1)
      IF(IM.EQ.1)GO TO 20
      IF(IJ.EQ.0)GO TO 10
C
C     HERE IS WHERE THE POINTS ARE DRAWN. YOU MUST ALSO CHECK TO BE
C     SURE THAT THE POINT AT '10 CONTINUE' IS INCLUDED WHENEVER
C     POINTS ARE DRAWN; IT WILL BE THE FIRST IN THE SEQUENCE.
C
      CALL PLOT((X1(2)-X1(3))/X1(4),(Y1(2)-Y1(3))/Y1(4),2)
      GO TO 30
   10 CONTINUE
      CALL PLOT((X1(2)-X1(3))/X1(4),(Y1(2)-Y1(3))/Y1(4),3)
      IJ=1
      GO TO 30
   20 CONTINUE
      IJ=0
   30 CONTINUE
      RETURN
      END
      SUBROUTINE CHECK(XXX,CCC,NNO,J,II,NC,XI,YI,
     1NGX,ZM,ZMI,RV,RVI,TGM,TGI,ZI,LZ)
C
C
C
C     THIS SUBROUTINE SOLVES FOR THE POINTS OF INTERSECTION ON THE
C     LINES OF THE JTH ELEMENT WITH OTHER LINES AND PLANES(RELEVANT).
C
C
      DIMENSION CCC(1),XXX(1)
      DIMENSION RV(1),RVI(1),TGM(1),TGI(1),ZM(1),ZMI(1),
     1NNO(1),NGX(15),XI(1),YI(1),ZI(1)
      DOUBLE PRECISION T8
      COMMON/DAVE/XCC(800)
      COMMON/HEDG/JS,M,JT,VX,VX1,VX2,VX3,NN
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/DRH/ISILH(1)
      DATA EEX,EXP/.015,.005/
      NGX(1)=0
      XCC3=XCC(JS+3)
      XCC4=XCC(JS+4)
      XCC1=XCC(JS+1)
      XCC2=XCC(JS+2)
      IF(NN.EQ.1)GO TO 5
      IF(VX3.NE.0.)GO TO 5
      A=XXX(JT+2)
      B=XXX(JT+1)
      C=XXX(JT+4)
      Z1=XCC(JS)
```

```
      Z2=XCC1
      IF(A.EQ.0.)GO TO 1
      Y1=-XCC3*B-C
      Y2=-XCC4*B-C
      X1=XCC3
      X2=XCC4
      GO TO 15
    1 CONTINUE
      Y1=XCC3
      Y2=XCC4
      X1=-C
      X2=X1
      GO TO 15
    5 CONTINUE
      A=XCC(JS)
      B=XCC1
      C=XCC2
      IF(A.EQ.0.)GO TO 20
      Y1=-XCC3*B-C
      Y2=-XCC4*B-C
      X1=XCC3
      X2=XCC4
      GO TO 30
   20 CONTINUE
      Y1=XCC3
      Y2=XCC4
      X1=-XCC2
      X2=X1
   30 CONTINUE
      IF(NN.NE.1)GO TO 40
      Z1=XXX(1+JT)
      Z2=XXX(2+JT)
      GO TO 50
   40 CONTINUE
      Z1=-(VX+VX1*Y1+VX2*X1)/VX3
      Z2=-(VX+VX1*Y2+VX2*X2)/VX3
   50 CONTINUE
   15 CONTINUE
      AL=X2-X1
      BL=Y2-Y1
      CL=Z2-Z1
      EG=AMIN1(Z1,Z2)
      EGX=AMAX1(X1,X2)
      EGX1=AMIN1(X1,X2)
      EGY=AMAX1(Y1,Y2)
      EGY1=AMIN1(Y1,Y2)
      EGZ=AMAX1(Z1,Z2)
      IF(AL.EQ.0.)GO TO 51
      BLA=BL/AL
      BLAX=(BLA)*X1
      CLA=CL/AL
      CLAX=(CLA)*X1
      GO TO 52
   51 CONTINUE
      IF(BL.EQ.0.)GO TO 52
      ALB=AL/BL
      ALBY=(ALB)*Y1
```

```
      CLB=CL/BL
      CLBY=(CLB)*Y1
   52 CONTINUE
C
C
C     THIS CODE DETERMINES THE POINTS OF INTERSECTIONS ON THE LINES OF
C     JTH ELEMENT RESULTING FROM THE INTERSECTION OF THE PLANES WITH
C     THESE LINES.
C
C
      DO 170 JR=1,II
      KM=L4+JR
      LG=NNO(KM)
      NNO(KM)=IABS(LG)
      LE=NNO(KM)
      IF(J.EQ.LE)GO TO 170
      IF(EGX.LT.TGM(LE+L5))GO TO 170
      IF(EGX1.GT.TGI(LE+L6))GO TO 170
      IF(EGY.LT.RVI(LE+L8))GO TO 170
      IF(EGY1.GT.RV(LE+L7))GO TO 170
      IF((ISILH(J).EQ.ISILH(LE)).AND.(ISILH(J).NE.0))GO TO 1172
      IF(EG.GT.ZM(L2+LE))GO TO 170
 1172 CONTINUE
      JE=L13+LZ*(LE-1)
      JU=L12+5*(LE-1)
      AC=XXX(1+JU)
      BC=XXX(2+JU)
      CC=XXX(3+JU)
      D=XXX(4+JU)
      G=1./CC
      NK=XXX(5+JU)*5
      IF((LG.LT.0).OR.(EGZ.LE.ZMI(L3+LE)))GO TO 80
      IF((AL.EQ.0).AND.(BL.EQ.0))GO TO 80
      IF(AL.EQ.0)GO TO 60
      VU=AC+BC*BLA+CC*CLA
      IF(VU.EQ.0.)GO TO 80
      XP=BC*BLAX+CC*CLAX-D
      XP=XP-BC*Y1-CC*Z1
      XP=XP/VU
      T1=(XP-X1)/AL
      YP=T1*BL+Y1
      GO TO 70
   60 CONTINUE
      VU=BC+AC*ALB+CC*CLB
      IF(VU.EQ.0.)GO TO 80
      YP=AC*ALBY+CC*CLBY-D
      YP=YP-CC*Z1-AC*X1
      YP=YP/VU
      T1=(YP-Y1)/BL
      XP=T1*AL+X1
   70 CONTINUE
      IF((XP-TGM(LE+L5))*(XP-TGI(LE+L6)).GT.0.)GO TO 80
      IF((YP-RV(LE+L7))*(YP-RVI(LE+L8)).GT.0.)GO TO 80
      T=XP
      IF(A.EQ.0.)T=YP
      IF((T-XCC3)*(T-XCC4).GE.0.)GO TO 80
      ZP=T1*CL+Z1
```

```
      S=ZP-ZM(L2+LE)
      S1=ZP-ZMI(L3+LE)
      IF((ABS(S).LT.EEX).OR.(ABS(S1).LT.EEX))GO TO 56
      IF(S*S1.GT.0.)GO TO 80
   56 CONTINUE
C
C     STORES INTERSECTIONS.
C
      M=M+1
      NQ=M+1
      XI(NQ)=XP
      YI(NQ)=YP
      ZI(NQ)=ZP
   80 CONTINUE
C
C     THIS CODE DETERMINES INTERSECTION POINTS OF LINES WITH LINES.
C
      JE1=JE+1
      JE2=JE+2
      JE3=JE+3
      JE4=JE+4
      DO 160 JV=1,NK,5
      B1=CCC(JV+JE1)
      A1=CCC(JV+JE)
      T8=A1*B-B1*A
      IF(T8.EQ.0.)GO TO 160
      C1=CCC(JV+JE2)
      XO=(C1*A-C*A1)/T8
      IF(A.NE.0.)GO TO 90
      YO=-C1-B1*XO
      GO TO 100
   90 CONTINUE
      YO=-C-B*XO
  100 CONTINUE
      T=XO
      IF(A.EQ.0.)T=YO
      IF((T-XCC3)*(T-XCC4).GE.0.)GO TO 160
      T=XO
      IF(A1.EQ.0.)T=YO
      S1=T-CCC(JV+JE4)
      S=T-CCC(JV+JE3)
      IF((ABS(S).LE.EXP).OR.(ABS(S1).LE.EXP))GO TO 110
      IF(S*S1.GT.0.)GO TO 160
  110 CONTINUE
      IF(VX3.NE.0.)GO TO 130
      TSZ=CL
      TSX=AL
      VT=XO-X1
      IF(TSX.NE.0.)GO TO 120
      VT=YO-Y1
      TSX=BL
  120 CONTINUE
      ZX1=(TSZ/TSX)*VT+Z1
      GO TO 140
  130 CONTINUE
      ZX1=-(VX+VX1*YO+VX2*XO)/VX3
  140 CONTINUE
```

```
      IF(ISILH(J).EQ.0)GO TO 148
      IF(ISILH(J).EQ.ISILH(LE))GO TO 150
  148 CONTINUE
      ZX=-(AC*XO+BC*YO+D)*G
      IF(ABS(ZX-ZX1).LT.EXP)GO TO 150
      IF(ZX1.GT.ZX)GO TO 160
  150 CONTINUE
      M=M+1
      NQ=M+1
C
C     STORES INTERSECTIONS.
C
      XI(NQ)=XO
      YI(NQ)=YO
      ZI(NQ)=ZX1
  160 CONTINUE
  170 CONTINUE
      NGX(1)=M
  190 RETURN
      END
      SUBROUTINE VSRTR(A,LA,IR)
C   IMSL ROUTINE NAME   - VSRTR
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - CDC/SINGLE
C
C   LATEST REVISION     - JANUARY 1, 1978
C
C   PURPOSE             - SORTING OF ARRAYS BY ALGEBRAIC VALUE -
C                           PERMUTATIONS RETURNED
C
C   USAGE               - CALL VSRTR (A,LA,IR)
C
C   ARGUMENTS    A      - ON INPUT, A CONTAINS THE ARRAY TO BE SORTED.
C                         ON OUTPUT, A CONTAINS THE SORTED ARRAY.
C                LA     - INPUT VARIABLE CONTAINING THE NUMBER OF
C                           ELEMENTS IN THE ARRAY TO BE SORTED.
C                IR     - VECTOR OF LENGTH LA.
C                         ON INPUT, IR CONTAINS THE INTEGER VALUES
C                           1,2,...,LA. SEE REMARKS.
C                         ON OUTPUT, IR CONTAINS A RECORD OF THE
C                           PERMUTATIONS MADE ON THE VECTOR A.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C   REMARKS      THE VECTOR IR MUST BE INITIALIZED BEFORE ENTERING
C                VSRTR.  ORDINARILY, IR(1)=1, IR(2)=2, ...,
C                IR(LA)=LA.  FOR WIDER APPLICABILITY, ANY INTEGER
C                THAT IS TO BE ASSOCIATED WITH A(I) FOR I=1,2,...,LA
C                MAY BE ENTERED INTO IR(I).
C
```

```
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE.  NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
      DIMENSION A(1),IR(1)
C                                     SPECIFICATIONS FOR ARGUMENTS
C                                     SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER             IU(21),IL(21),I,M,J,K,IJ,IT,L,ITT
      REAL                T,TT,R
C                                     FIRST EXECUTABLE STATEMENT
      IF (LA.LE.0) RETURN
      M = 1
      I = 1
      J = LA
      R = .375
    5 IF (I.EQ.J) GO TO 45
      IF (R.GT..5898437) GO TO 10
      R = R+3.90625E-2
      GO TO 15
   10 R = R-.21875
   15 K = I
C                                     SELECT A CENTRAL ELEMENT OF THE
C                                     ARRAY AND SAVE IT IN LOCATION T
      IJ = I+(J-I)*R
      T = A(IJ)
      IT = IR(IJ)
C                                     IF FIRST ELEMENT OF ARRAY IS GREATER
C                                     THAN T, INTERCHANGE WITH T
      IF (A(I).LE.T) GO TO 20
      A(IJ) = A(I)
      A(I) = T
      T = A(IJ)
      IR(IJ) = IR(I)
      IR(I) = IT
      IT = IR(IJ)
   20 L = J
C                                     IF LAST ELEMENT OF ARRAY IS LESS THAN
C                                     T, INTERCHANGE WITH T
      IF (A(J).GE.T) GO TO 30
      A(IJ) = A(J)
      A(J) = T
      T = A(IJ)
      IR(IJ) = IR(J)
      IR(J) = IT
      IT = IR(IJ)
C                                     IF FIRST ELEMENT OF ARRAY IS GREATER
C                                     THAN T, INTERCHANGE WITH T
      IF (A(I).LE.T) GO TO 30
      A(IJ) = A(I)
      A(I) = T
      T = A(IJ)
      IR(IJ) = IR(I)
      IR(I) = IT
```

```
      IT = IR(IJ)
      GO TO 30
   25 IF (A(L).EQ.A(K)) GO TO 30
      TT = A(L)
      A(L) = A(K)
      A(K) = TT
      ITT = IR(L)
      IR(L) = IR(K)
      IR(K) = ITT
C                                  FIND AN ELEMENT IN THE SECOND HALF OF
C                                  THE ARRAY WHICH IS SMALLER THAN T
   30 L = L-1
      IF (A(L).GT.T) GO TO 30
C                                  FIND AN ELEMENT IN THE FIRST HALF OF
C                                  THE ARRAY WHICH IS GREATER THAN T
   35 K = K+1
      IF (A(K).LT.T) GO TO 35
C                                  INTERCHANGE THESE ELEMENTS
      IF (K.LE.L) GO TO 25
C                                  SAVE UPPER AND LOWER SUBSCRIPTS OF
C                                  THE ARRAY YET TO BE SORTED
      IF (L-I.LE.J-K) GO TO 40
      IL(M) = I
      IU(M) = L
      I = K
      M = M+1
      GO TO 50
   40 IL(M) = K
      IU(M) = J
      J = L
      M = M+1
      GO TO 50
C                                  BEGIN AGAIN ON ANOTHER PORTION OF
C                                  THE UNSORTED ARRAY
   45 M = M-1
      IF (M.EQ.0) RETURN
      I = IL(M)
      J = IU(M)
   50 IF (J-I.GE.11) GO TO 15
      IF (I.EQ.1) GO TO 5
      I = I-1
   55 I = I+1
      IF (I.EQ.J) GO TO 45
      T = A(I+1)
      IT = IR(I+1)
      IF (A(I).LE.T) GO TO 55
      K = I
   60 A(K+1) = A(K)
      IR(K+1) = IR(K)
      K = K-1
      IF (T.LT.A(K)) GO TO 60
      A(K+1) = T
      IR(K+1) = IT
      GO TO 55
      END
      SUBROUTINE SOLVE(IXR,J,XXX,CCC,II,NNO,NIT,
     1X21,Y21,Z21,IIA,NC,ZM,ZMI,LZ)
```

```
C
C     THIS SUBROUTINE SOLVES FOR THE LINES OF INTERSECTION RESULTING
C     FROM THE INTERSECTIONS OF THE JTH ELEMENT WITH THE OTHER
C     RELEVANT ELEMENTS.
C
C
C
      DIMENSION XXX(1),CCC(1),NNO(1),
     1ZM(1),ZMI(1),X21(1),Y21(1),Z21(1),IIA(1),IV(2)
      DIMENSION XA(50),YA(50),ZA(50)
      COMMON/DAVE/XCC(800)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      COMMON/DRH/ISILH(1)
      COMMON/INDX/JT,JB
      DATA EXX,ERS/.001,.015/
      C3=XXX(3+JT)
      IF(C3.EQ.0.)GO TO 80
      A3=XXX(1+JT)
      B3=XXX(2+JT)
      D3=XXX(4+JT)
      ZQ=ZM(L2+J)
      DO 70 L=1,II
      K=NNO(L4+L)
C
C     CHECKS TO SEE IF THIS RELEVANT ELEMENT IS TO BE CONSIDERED FOR
C     INTERSECTION
C
      IF((K.LT.0).OR.(K.LT.J))GO TO 70
      IF(ZQ.LT.ZMI(L3+K))GO TO 70
      IF((ISILH(J).EQ.ISILH(K)).AND.(ISILH(J).NE.0))GO TO 70
      JX=L12+(K-1)*5
      MT=0
      A4=XXX(1+JX)
      B4=XXX(2+JX)
      C4=XXX(3+JX)
      D4=XXX(4+JX)
C
C     DETERMINES THE EQUATION OF LINE OF INTERSECTION.
C
      B=A3*C4-A4*C3
      A=B3*C4-B4*C3
      C=D3*C4-D4*C3
      IF((A.EQ.0.).AND.(B.EQ.0.))GO TO 70
      IF(A.NE.0.)GO TO 10
      A=0
      C=C/B
      B=1
      GO TO 20
   10 CONTINUE
      B=B/A
      C=C/A
      A=1
   20 CONTINUE
      IV(1)=J
      IV(2)=K
      S3=2*A+2*B+C
      DO 60 M=1,2
```

```
      I=IV(M)
      JJ=L13+(I-1)*LZ
      IG=5+L12+(I-1)*5
      NK=XXX(IG)*5
      JJ1=JJ+1
      JJ2=JJ+2
      JJ3=JJ+3
      JJ4=JJ+4
      DO 50 JV=1,NK,5
      A1=CCC(JV+JJ)
      B1=CCC(JV+JJ1)
      C1=CCC(JV+JJ2)
C
C     CHECK TO BE SURE LINE OF INTERSECTION IS NOT BOUNDARY LINE
C     OF THE JTH SET.
C
      S2=A1*2.+B1*2.+C1
      IF(ABS(S2-S3).LT.EXX)GO TO 70
C
C
C     DETERMINES THE POINTS OF INTERSECTIONS OF THE LINE OF INTERSECTION
C     WITH OTHER LINES OF RELEVANT ELEMENTS.
C
C
      T8=A1*B-B1*A
      IF(ABS(T8).LE.ERS)GO TO 50
      XO=(C1*A-C*A1)/T8
      IF(A.NE.0.)GO TO 30
      YO=-C1-B1*XO
      GO TO 40
   30 CONTINUE
      YO=-C-B*XO
   40 CONTINUE
      T=XO
      IF(A1.EQ.0.)T=YO
      IF((T-CCC(JV+JJ4))*(T-CCC(JV+JJ3)).GT.0.)GO TO 50
      MT=MT+1
C
C     STORE THE PTS OF INTERSECTIONS.
C
      XA(MT)=XO
      YA(MT)=YO
      ZA(MT)=-(D3+A3*XO+B3*YO)/C3
      ZT=-(D4+A4*XO+B4*YO)/C4
      IF(ABS(ZT-ZA(MT)).GT.EXX)GO TO 70
   50 CONTINUE
   60 CONTINUE
      CALL STAT(MT,NIT,IXR,X21,Y21,Z21,IIA,IV,A,B,
     IC,J,XA,YA,ZA,CCC,XXX,NC,LZ)
   70 CONTINUE
   80 CONTINUE
      NR=5*XXX(5+JT)
      DO 90 IS=1,NR
      XCC(IS)=CCC(IS+JB)
   90 CONTINUE
      XXX(5+JT)=XXX(5+JT)+NIT
      RETURN
```

```
      END
      SUBROUTINE STAT(MT,NIT,IXR,X21,Y21,Z21,
     1IIA,IV,A,B,C,IK,XA,YA,ZA,CCC,XXX,NC,LZ)
C
C
C     THIS SUBROUTINE TAKES THE PTS OF INTERSECTION DETERMINED BY
C     SUBROUTINE SOLVE AND PICKS THE COORDINATES WITH THE MAX AND
C     MIN X COORDINATES PROVIDED THEY LIE ON THE INTERIOR/BOUNDARY
C     OF BOTH ELEMENTS.
C
C
      DIMENSION XXX(1),CCC(1),X21(1),
     1Y21(1),Z21(1),IIA(1),IV(1),XA(1),YA(1),ZA(1)
      COMMON/DAVE/XCC(800)
      COMMON/GO3/L0,L1,L00,L01,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,L13
      DATA EXX/.015/
      NX=0
      IF(NIT.GE.120)GO TO 160
      IF(MT.EQ.0)GO TO 160
      DO 50 JX=1,MT
      EI=0
   10 EI=EI+.1
      IF(EI.GE..5)GO TO 160
      D=EI*XA(JX)-YA(JX)
      DO 40 JO=1,2
      M=IV(JO)-1
      JC=L13+(M)*LZ
      JXC=L12+(M)*5
      NK=XXX(5+JXC)
      I=0
      IB=NK*5
C
C
C     DETERMINE IF THE PROJECTION OF THE POINT OF INTERSECTION
C     BELONGS TO THE INTERIOR OF BOTH PLANES.
C
C
      DO 30 J=1,IB,5
      VE=XA(JX)
      J6=J+JC
      IF(CCC(J6).EQ.0.)VE=YA(JX)
      T=CCC(J6)*YA(JX)+CCC(J6+1)*XA(JX)+CCC(J6+2)
      IF((ABS(T).LT.EXX).AND.((VE-CCC(J6+3))*(VE-CCC(J6+4)).LE.0.))
     1GO TO 40
      R=EI*CCC(J6)+CCC(J6+1)
      IF(R.EQ.0.)GO TO 30
      T=(-CCC(J6+2)+CCC(J6)*D)/R
      IF(T.LT.XA(JX))GO TO 30
      IF(CCC(J6).NE.0.)GO TO 20
      T=EI*T-D
   20 CONTINUE
      IF((T.EQ.CCC(J6+3)).OR.(T.EQ.CCC(J6+4)))GO TO 10
      IF((T-CCC(J6+3))*(T-CCC(J6+4)).GT.0.)GO TO 30
      I=I+1
   30 CONTINUE
      IF(I-(I/2)*2.EQ.0)GO TO 50
   40 CONTINUE
```

```
      NX=NX+1
      XA(NX)=XA(JX)
      YA(NX)=YA(JX)
      ZA(NX)=ZA(JX)
   50 CONTINUE
      IF(NX.EQ.0)GO TO 160
C
C
C
C     THIS CODE FINDS THE MAX/MIN X-COORDINATES(Y-COORDINATES) AND
C     STORES THEM. FURTHERMORE BOTH THE EQUATION OF LINE AND POINTS(2)
C     ARE TREATED LIKE ADDITIONAL EDGES. IN THIS WAY, THE ALGORITHM NEED
C     NOT BE DISTURBED. ESSENTIALLY,THEN,THIS TRICK IS TRANSPARENT TO
C     THE REST OF THE PROGRAM.
C
C
      AMAXX=-(10**6)
      AMINX=-AMAXX
      AMAXY=AMAXX
      AMINY=AMINX
      IS=5+(IK-1)*5+L12
      IS=XXX(IS)
      DO 110 JI=1,NX
      IF(A.EQ.0.)GO TO 80
      IF(XA(JI).GE.AMINX)GO TO 60
      AMINX=XA(JI)
      YI=YA(JI)
      ZI=ZA(JI)
   60 IF(XA(JI).LE.AMAXX)GO TO 70
      AMAXX=XA(JI)
      YII=YA(JI)
      ZII=ZA(JI)
   70 CONTINUE
      GO TO 110
   80 CONTINUE
      IF(YA(JI).GE.AMINY)GO TO 90
      AMINY=YA(JI)
      XI=XA(JI)
      ZI=ZA(JI)
   90 CONTINUE
      IF(YA(JI).LE.AMAXY)GO TO 100
      XII=XA(JI)
      AMAXY=YA(JI)
      ZII=ZA(JI)
  100 CONTINUE
  110 CONTINUE
      NIT=NIT+1
      K=5*(NIT-1+IS)+1
      XCC(K)=A
      XCC(K+1)=B
      XCC(K+2)=C
      IF(A.EQ.0.)GO TO 120
      XCC(K+3)=AMINX
      XCC(K+4)=AMAXX
      AMIN=AMINX
      AMAX=AMAXX
      YE=YII
```

```
      ZE=ZII
      GO TO 130
  120 CONTINUE
      XCC(K+3)=AMINY
      XCC(K+4)=AMAXY
      AMIN=XI
      AMAX=XII
      YI=AMINY
      YE=AMAXY
      ZE=ZII
  130 CONTINUE
      IG=IXR+NIT*3
      I8=IG-2
      X21(I8)=AMIN
      Y21(I8)=YI
      Z21(I8)=ZI
      DO 140 JK=1,2
      IE=IG-JK+1
      X21(IE)=AMAX
      Y21(IE)=YE
      Z21(IE)=ZE
  140 CONTINUE
      DO 150 JK=1,2
      IIA(IG-JK)=0
  150 CONTINUE
      IIA(IG)=1
      TX=(AMAX-AMIN)**2
      TY=(YE-YI)**2
      DX=(TX+TY)**.5
      IF(DX.LT..1)NIT=NIT-1
  160 CONTINUE
      IF(EI.GE..5)NIT=0
      RETURN
      END
      SUBROUTINE VSRT1(A,LA,IR)
      INTEGER            IU(21),IL(21),I,M,J,K,IJ,IT,L,ITT
      INTEGER A(1),IR(1),T,TT
C                                FIRST EXECUTABLE STATEMENT
      IF (LA.LE.0) RETURN
      M = 1
      I = 1
      J = LA
      R = .375
    5 IF (I.EQ.J) GO TO 45
      IF (R.GT..5898437) GO TO 10
      R = R+3.90625E-2
      GO TO 15
   10 R = R-.21875
   15 K = I
C                                SELECT A CENTRAL ELEMENT OF THE
C                                ARRAY AND SAVE IT IN LOCATION T
      IJ = I+(J-I)*R
      T = A(IJ)
      IT = IR(IJ)
C                                IF FIRST ELEMENT OF ARRAY IS GREATER
C                                THAN T, INTERCHANGE WITH T
      IF (A(I).LE.T) GO TO 20
```

```
      A(IJ) = A(I)
      A(I) = T
      T = A(IJ)
      IR(IJ) = IR(I)
      IR(I) = IT
      IT = IR(IJ)
   20 L = J
C                                  IF LAST ELEMENT OF ARRAY IS LESS THAN
C                                  T, INTERCHANGE WITH T
      IF (A(J).GE.T) GO TO 30
      A(IJ) = A(J)
      A(J) = T
      T = A(IJ)
      IR(IJ) = IR(J)
      IR(J) = IT
      IT = IR(IJ)
C                                  IF FIRST ELEMENT OF ARRAY IS GREATER
C                                  THAN T, INTERCHANGE WITH T
      IF (A(I).LE.T) GO TO 30
      A(IJ) = A(I)
      A(I) = T
      T = A(IJ)
      IR(IJ) = IR(I)
      IR(I) = IT
      IT = IR(IJ)
      GO TO 30
   25 IF (A(L).EQ.A(K)) GO TO 30
      TT = A(L)
      A(L) = A(K)
      A(K) = TT
      ITT = IR(L)
      IR(L) = IR(K)
      IR(K) = ITT
C                                  FIND AN ELEMENT IN THE SECOND HALF OF
C                                  THE ARRAY WHICH IS SMALLER THAN T
   30 L = L-1
      IF (A(L).GT.T) GO TO 30
C                                  FIND AN ELEMENT IN THE FIRST HALF OF
C                                  THE ARRAY WHICH IS GREATER THAN T
   35 K = K+1
      IF (A(K).LT.T) GO TO 35
C                                  INTERCHANGE THESE ELEMENTS
      IF (K.LE.L) GO TO 25
C                                  SAVE UPPER AND LOWER SUBSCRIPTS OF
C                                  THE ARRAY YET TO BE SORTED
      IF (L-I.LE.J-K) GO TO 40
      IL(M) = I
      IU(M) = L
      I = K
      M = M+1
      GO TO 50
   40 IL(M) = K
      IU(M) = J
      J = L
      M = M+1
      GO TO 50
C                                  BEGIN AGAIN ON ANOTHER PORTION OF
```

```
C                                       THE UNSORTED ARRAY
   45 M = M-1
      IF (M.EQ.0) RETURN
      I = IL(M)
      J = IU(M)
   50 IF (J-I.GE.11) GO TO 15
      IF (I.EQ.1) GO TO 5
      I = I-1
   55 I = I+1
      IF (I.EQ.J) GO TO 45
      T = A(I+1)
      IT = IR(I+1)
      IF (A(I).LE.T) GO TO 55
      K = I
   60 A(K+1) = A(K)
      IR(K+1) = IR(K)
      K = K-1
      IF (T.LT.A(K)) GO TO 60
      A(K+1) = T
      IR(K+1) = IT
      GO TO 55
      END
```

NASA
Reference
Publication
1085

March 1982

A General Solution to the Hidden-Line Problem

David R. Hedgley, Jr.

NASA

NASA Reference Publication 1085

1982

A General Solution to the Hidden-Line Problem

David R. Hedgley, Jr.
Ames Research Center
Dryden Flight Research Facility
Edwards, California

NASA
National Aeronautics and Space Administration

Scientific and Technical Information Branch

A GENERAL SOLUTION TO THE HIDDEN-LINE PROBLEM

David R. Hedgley, Jr.
Ames Research Center
Dryden Flight Research Facility

INTRODUCTION

The requirement for computer-generated perspective projections of three-dimensional objects by way of line drawings has escalated in recent years. Pictures of objects which show visible and hidden lines are relatively easy to present. Unfortunately, such renderings are often ambiguous and serve as little value to the engineer or scientist. (See fig. 1.)

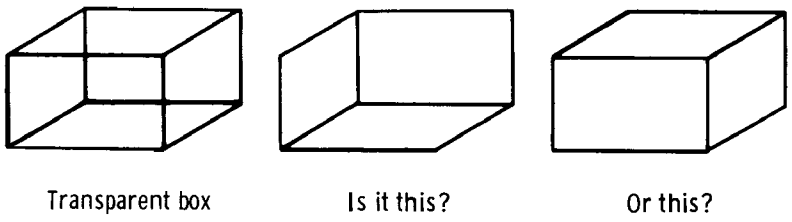


Figure 1. Ambiguous case.

Historically, much literature has appeared addressing this problem (ref. 1). However, prior solutions have exhibited some inherent limitations, one of the most significant being square-law growth, that is, the tendency of the computer execution time to grow as the square of the number of elements. Another significant restriction is the environmental limitations. (See appendix A for a list of typical limitations.)

At NASA Ames Research Center's Dryden Flight Research Facility (DFRF), the need arose to graphically represent aerodynamic stability derivatives as a function of two variables. (See the stability derivative plot in appendix B.) This requirement and the potential for further application served as a motivation for developing a general solution to the hidden-line problem which avoids the undesirable features of prior solutions.

This paper lays the theoretical foundation for the practical implementation of a general hidden-line algorithm. A theorem is presented and proved which does not assume any environmental limitations, unlike prior approaches. Furthermore the theorem allows the determination of the visibility of an entire line segment by choosing only a few points on that line, thus providing a basis for a rapid algorithm.

The author wishes to express his appreciation to Mary Bailer, Victor Pestone, and Paul Redin for their many helpful suggestions and constructive criticisms.

ANALYSIS

This section introduces basic definitions of symbols and terms, develops the equation of a plane, and discusses both the visibility of an arbitrary point and how the points are selected. The criteria of these selections and the sufficiency of the choices are predicated on a theorem which is proved.

Definitions

Let O be any scene or collection of objects which can be represented with straight lines and/or n-sided convex/concave planar polygons (internal boundaries allowed). Also, let each polygon be defined in such a way that every point is a boundary point in the topological sense. For economy of definition, vertices of polygons are sufficient and, similarly, end points for line segments.

With the admissible elements enunciated in the above paragraph, we need only discuss the visibility of a point P and its selection. The argument is then easily extended to the entire scene O.

The Equation of a Plane

The equation of a plane is given by

$$Ax + By + Cz + D = O,$$

where

$$A = b_1c_2 - b_2c_1$$
$$B = c_1a_2 - c_2a_1$$
$$C = a_1b_2 - a_2b_1$$
$$D = -(Ax_i + By_i + Cz_i)$$

where

$$a_i = x_{i+1} - x_i$$
$$b_i = y_{i+1} - y_i \qquad (i = 1,2)$$
$$c_i = z_{i+1} - z_i$$

x_j, y_j, z_j $(j = 1,3)$ are three non-collinear points. A, B, and C are the coefficients of the normal vector to the plane and hence the coefficients of the plane itself.

For consistency with plotter hardware convention, x and y have their standard directions and z is perpendicular to the plane of the paper. The direction is consistent with a right-handed coordinate system.

Visibility Criteria

CRITERION I. *Let C be the projection of a planar closed polygon A on the XY-plane such that every point on C is a boundary point. (See fig. 2.) Let P(x, y) be the projection of any point (x, y, z). If P lies on the boundary of C, it is clearly visible with respect to C. And hence (x, y, z) is visible with respect to A. If P lies outside the boundary of C, then a line drawn from P in any direction to infinity will intersect the boundary an even number of times. If P lies inside the boundary of C, then this semi-infinite line drawn from P will intersect the boundary of C an odd number of times. (See ref. 2.)*

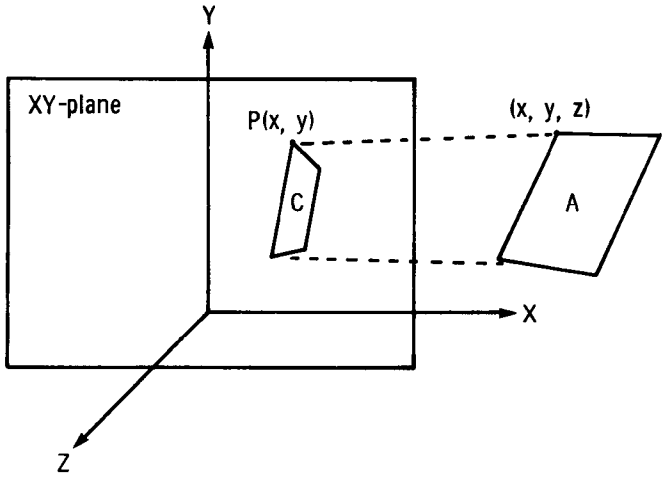


Figure 2. Plane A and its projection, C; point (x, y, z) and its projection, P(x, y).

Figure 3 illustrates this idea. Note that line ℓ_1, drawn from P_1 in the XY-plane in an arbitrary direction to infinity, intersects the edges of the holes and containing

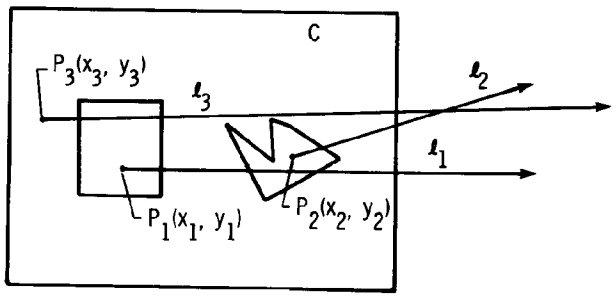


Figure 3. Polygon C with two holes.

plane, C, four times. Thus, the count is even indicating that P_1 is "outside" C and hence (x_1, y_1, z_1) is visible with respect to plane A. This is also true for (x_2, y_2, z_2). However, for $P_3(x_3, y_3)$, the count is odd indicating that the point lies in the interior of C.

If this semi-infinite line should cross a vertex, a line with a different slope should be selected. It is always possible to find such a line since the number of combinations of vertices is finite, whereas the number of slopes is infinite. Moreover, if C should have an arbitrary number of "holes" whose boundaries have the character of the external boundary of C, then no generality is lost. For if P lies inside C and outside all "holes," then the count is odd with respect to C as a polygon and even with respect to all holes. Therefore the total count remains odd. The remaining cases are argued similarly.

With the preceding observations in combination with the equation of a plane, it is now possible to determine the visibility of (x, y, z) with respect to plane A if P is "inside" C.

CRITERION II. $z \geq -(A_o x + B_o y + D_o)/C_o$ *implies* (x, y, z) *is visible with respect to* A.

CRITERION III. $z < -(A_o x + B_o y + D_o)/C_o$ *implies* (x, y, z) *is invisible with respect to* A.

Whether or not P is inside, outside, or on the boundary of C can be determined from Criterion I. Clearly then, if (x, y, z) is visible with respect to all A_i, it is visible. Note that if A is a line or if $C_o = 0$, then Criteria II and III do not apply. In this case, the projection is a line and hence every point is visible with respect to it.

Point Selection Criterion

Having a method for visibility determination is not by itself sufficient for a practical implementation. That is, hidden-line algorithms typically take huge amounts of computer time for even moderately simple scenes. The worst case is brute force

where every line is quantified into many points and each point is tested. Hence, economy of points and efficiency of implementation are very important. The following discussion presents an ordered approach to improve computational efficiency. We first consider this important definition.

DEFINITION. Let S be a sequence of distinct triplets (x_i, y_i, z_i) belonging to a line determined by (x_1, y_1, z_1) and (x_2, y_2, z_2). Define S to be "ordered" if given any two triplets, say (x_k, y_k, z_k) and (x_n, y_n, z_n). Then $n > k$ implies that $(x_n - x_1)^2 + (y_n - y_1)^2 \geqslant (x_k - x_1)^2 + (y_k - y_1)^2$. Define a primed symbol, such as ℓ', to mean the projection of that element, ℓ, onto the XY-plane. (Fig. 4 represents a typical sequence S.)

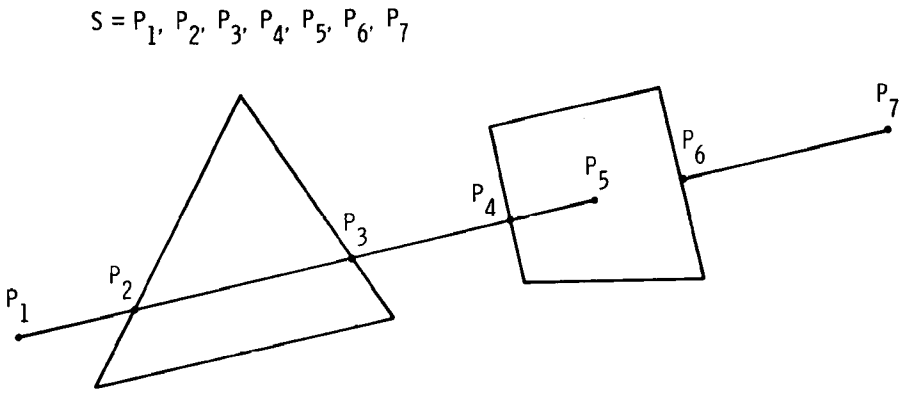


Figure 4. Spatial representations of sequence S and points that comprise set $\{P_i\}$. Note that here $\{P_i\} = P_1, P_5, P_6, P_7$, which follows from the definition of a valid intersection on ℓ_o along with its end points.

THEOREM. *Let ℓ_o be any line segment which belongs to O, any scene. Also, let S be an ordered sequence $\{P_i\} = \{x_i, y_i, z_i\}$ belonging to ℓ_o, which includes its end points along with a subset of the interior points of ℓ_o. Let these points of ℓ_o be such that their projections are a subset of all of the intersections, if any, of ℓ_o' with the interior points of other lines, say $\ell_1', \ell_2', \ldots, \ell_n'$, as well as all of the intersections of ℓ_o with the boundaries or interiors of planes $A_j, \ldots, A_k$, if any. Further, let this subset have the property that if (x_o, y_o, z_o) belongs to ℓ_o whose projection is the intersection ℓ_o' with ℓ_n', and (x_o, y_o, z_n) belongs to ℓ_n, then $z_n > z_o$. (Fig. 4 illustrates a valid selection of points that comprise the set $\{P_i\}$.)*

Then if P_i and P_{i+1} are both visible, the visibility of every interior point of

$[P_i, P_{i+1}]$ *has the same character. Moreover, if* P_i *or* P_{i+1} *is invisible, then every interior point of* $[P_i, P_{i+1}]$ *is also invisible.*

PROOF. Let both P_i and P_{i+1} be visible. Also, let P_m be any interior point of $[P_i, P_{i+1}]$ and be visible. If P_s is any other interior point, then it must also be visible. Assume the contrary whereby P_s is assumed to be invisible. Clearly then, P_s is hidden by some polygon, say A_1, which implies that P_s' lies in the interior of A_1' and satisfies Criterion III. Since P_m is visible, it is visible with respect to every polygon and in particular with respect to A_1. Therefore, P_m' either lies outside or on the boundary of A_1', or P_m' lies in the interior of A_1' and P_m satisfies Criterion II. (See fig. 5.)

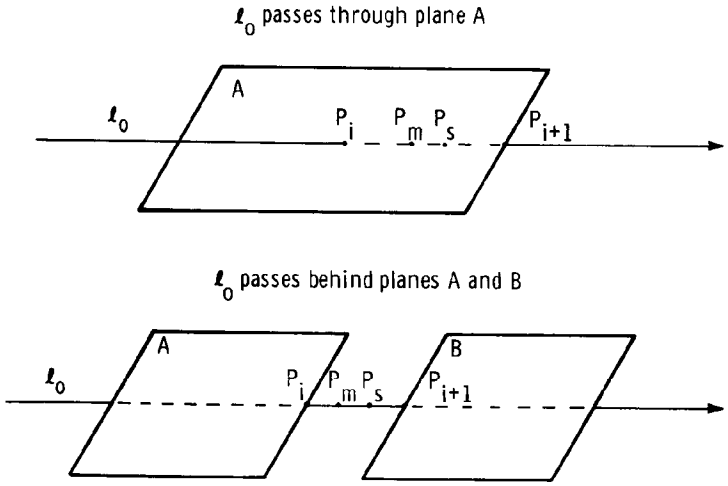


Figure 5. Visibility examples.

Suppose P_m' lies outside the boundary of A_1'. Since A_1' is closed, an edge of A_1' intersects ℓ_o' between P_m' and P_s'. That is, there is an intersection of projections between P_i' and P_{i+1}'. This implies that the z-coordinate of the point on the edge of A_1 whose projection is this intersection must be less than or equal to the z-coordinate of the point on ℓ_o whose projection is this same intersection. This observation follows from the definition of S.

Thus, there is a point on ℓ_o, say $P_t(x_t, y_t, z_t)$, such that P_t' lies in the interior of A_1' and P_t satisfies Criterion II, or P_t lies on the boundary of A_1 since P_s' lies in the interior of A_1' and P_s satisfies Criterion III. Hence, ℓ_o must intersect A_1 inside

the boundary between P_i and P_{i+1}. But this is a contradiction which follows from the construction of S.

Now if P'_m lies on the boundary of A'_1, then the z-coordinate of the point on ℓ_o whose projection is the intersection of ℓ'_o with the projection of an edge of A_1 must be greater than or equal to the z-coordinate of a point on the edge of A_1 whose projection is this intersection. Clearly, this again implies there is an intersection of ℓ_o with the plane A_1 in its interior or boundary between P_i and P_{i+1}, which is impossible from the definition of S.

The same contradiction follows if P'_m lies in the interior of A'_1 and P_m satisfies Criterion II.

On the other hand, if P_m is invisible, the proof that every interior point of $[P_i, P_{i+1}]$ is also invisible follows immediately by assuming there exists an interior point, say P_s, that is visible. By letting P_m be the known invisible point in $[P_i, P_{i+1}]$, the above argument may be employed and the same contradiction reached.

Let us now define P_i or P_{i+1} to be invisible. Since, say, P_i is invisible, it lies in the interior of, say, A_1. Since P'_i is an interior point of A'_1 and P_i satisfies Criterion III, there exists another point P_s such that P'_s lies in the interior of A'_1, and P_s satisfies Criterion III. Now if we assume there exists another interior point, say P_m belonging to $[P_i, P_{i+1}]$, that is visible, again the above argument may be used and the same contradiction reached. Q.E.D.

IMPLEMENTATION

An algorithm based on the theory presented in this paper has been implemented on a CDC-6500 computer at DFRF. This approach represents a significant improvement over existing algorithms in that it minimizes the number of points interrogated for visibility without assuming any environmental limitations. Although the theorem provides a formal basis for assuring generality and rapid execution, it does not address the nuisance of square-law growth. That will now be discussed.

Initially, an $m \times n$ grid in the XY-plane is constructed whose size is $\log_2 N$ + constant, where again N is the number of elements. If an element A_j is entirely contained in a grid block, B_i, an index i, which represents the grid block, is placed in E_j. If, however, some part of the element belongs to the boundary boxes of four or less blocks and is not a proper subset of any block, then i will be

$$i = \sum_{s=1}^{k} L_s * \text{base}^{\,s-1}$$

where

$1 < k \leqslant 4$

L_s = the grid block number involved

base = $\log_2 N$ + constant + 1

This value of i is also placed in E_j. Thus, E_j will contain up to four of the blocks involved for the element A_j. Additionally, if A_j is inside the boundary box of a grid block but not properly contained in it, then j is stored in the matrix, $M_{k,i}$, where k is the k*th* element with this property and i is the grid block number.

If A_j belongs to more than four grid blocks, E_j is zero.

The indices found in E_j are sorted only once. With this arrangement, it is now possible to assign a unique address, C_{E_j}, to each sequence of like indices ignoring those values in E_j which represent more than one block involvement (i.e., $i > \log_2 N$ + constant + 1) (Schcme 1).

Thus, given an element A_j whose E_j is not zero, its relevant elements will be the elements corresponding to C_{E_j} and the M_{k,E_j} matrix. The total number of relevant elements, TN, as they relate to A_j will be

$$TN = \sum_{i=1}^{T} C_{L_i} + M_{k,L_i}$$

where

C_{L_i}, M_{k,L_i} = addresses of relevant elements

$$T = [\log_{\text{base}} E_j] + 1$$

Here, L_i are the packed block numbers derived from E_j.

If E_j is zero, then the entire collection represents the relevant elements of A_j. Note that although each C_{L_i} is mutually exclusive, this is not true for the matrix,

M_{k,L_i}. An element belonging to M_{k,L_p} may also belong to M_{R,L_w}. Thus allowances are made to eliminate redundancies.

A second scheme (Scheme 2) is adopted as follows. The minimum x- and y-coordinate values along with the maximum z-coordinate value of each element are sorted once using a method with $N \log N$ growth. This results in three lists, each arranged in ascending order. Then, given an element A, the location of its maximum x and y and minimum z in each list in turn is found logarithmically, if possible. Thus, only that part of each list such that (1) $x_{max} \leqslant x_i$, (2) $y_{max} \leqslant y_j$, and (3) $z_{min} \geqslant z_k$ is retained. The minimum $(i, j, n-k)$ and its corresponding elements become the smallest relevant set with respect to A with this scheme.

With the number of relevant elements known from both methods, the minimum count from both schemes is chosen along with the appropriate corresponding elements.

The final relevant set of elements resulting from comparisons of elements within this relevant set will be smaller yet. However, its computational efficiency is not salient since the growth pattern is ultimately predicated on the TN value. Clearly, if only the members of this class are tested against A_i, then square-law growth is avoided.

Rigorous testing at DFRF verified that the algorithm enjoys almost linear growth. It should be noted that this algorithm was benchmarked against the Watkins algorithm and Loutrel algorithm (refs. 1 and 2) and was found to be superior to both in terms of speed. This superiority increased with the complexity of the scene.

The computer program developed for the testing has about 1700 statements. The memory required for data is about $69N$ decimal words, where N is the number of elements.

Line drawings presented in appendix B illustrate the generality of the algorithm. The steps of the algorithm are presented in appendix C.

The program can be obtained from the Computer Software Management and Information Center (COSMIC), 112 Barrow Hall, the University of Georgia, Athens, Georgia 30602.

CONCLUDING REMARKS

This paper addresses a classical problem in computer graphics and presents the theoretical basis for a practical hidden-line algorithm that surmounts all of the limitations of previous solutions. Furthermore, the efficiency of the algorithm does not suffer because of its generality. To the author's knowledge, this is the most robust approach known and represents the first completely general solution to this most popular and important problem.

Dryden Flight Research Facility
Ames Research Center
National Aeronautics and Space Administration
Edwards, California, November 16, 1981

APPENDIX A

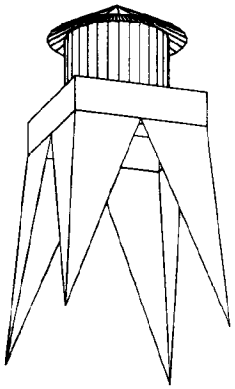
LIMITATIONS OF OTHER SOLUTIONS

The following is a list of limitations of which at least one applies to all of the known hidden-line (calligraphic) solutions to date.

Requires more than one pass.
Does not handle *n*-sided polygons.
Does not accept both concave and convex polygons as well as line segments.
Does not tolerate faces with internal boundaries (concave or convex).
Requires more topological information than the vertices.
Possesses square-law growth.
Does not accept penetrating polygons complete with lines of intersection.
Execution time grows linearly with the scale factor.
Execution time is excessive for even moderately simple structures (500 lines).
Makes mistakes frequently.
Does not handle several adjacent faces which are transparent and opaque.
Assumes a certain orientation of vertices (i.e., clockwise or counterclockwise).
Requires that polygons not be larger than a certain size.
Does not tolerate situations where an edge belongs to an arbitrary number of faces.
Has limited orientations.
Requires that every polygon belong to a polyhedron.

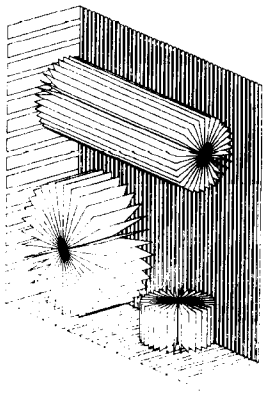
APPENDIX B

COMPUTER-GENERATED LINE DRAWINGS



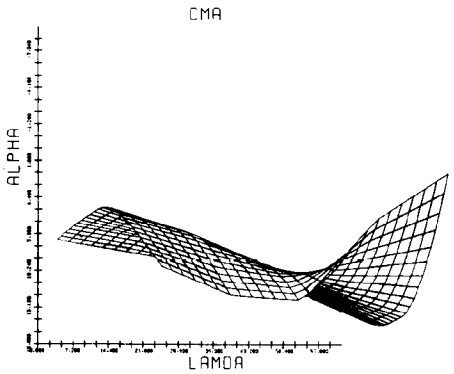
Water tower

73 polygons
Execution time = 10 sec



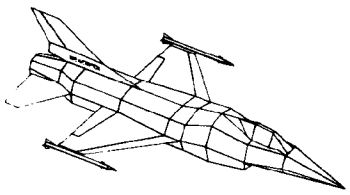
Array of planes

171 polygons
Execution time = 60 sec



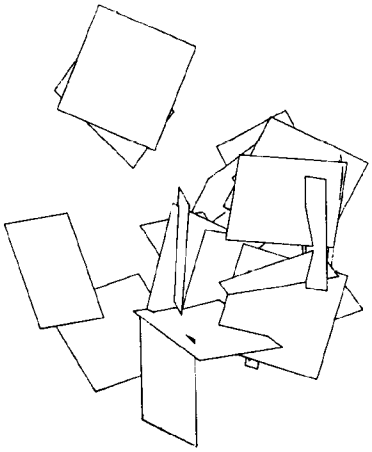
Stability derivative plot

625 polygons
Execution time = 60 sec



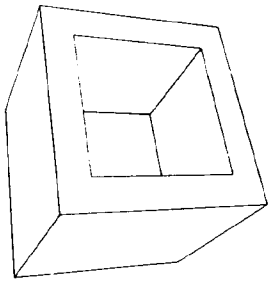
Modified F-16

159 polygons
Execution time = 12.5 sec



Intersecting planes

20 polygons
Execution time = 1.8 sec



Box with a hole

6 polygons,
Execution time = .35 sec

APPENDIX C

STEPS OF THE ALGORITHM

The sequence of the programmed algorithm is presented in the following discussion. All elements have been input in the form of (x, y, z) triplets (end points of line segments or vertices of polygons) to the hidden-line program.

Step 1. Performs the Eulerian transformations on the (x, y, z) triplets for each element. Stores all of the computed information. All further computations will be on the transformed triplets.

Step 2. Determines the equation of each plane and its edges, if applicable. All information is stored as in Step 1 and is kept in locations which correspond to that element number.

Step 3. Computes minimum and maximum x and y in the projection plane for the entire scene.

Step 4. Constructs a grid whose divisions are equal to $\log_2 N$ + constant, where N is the total number of elements. The area of the grid is predicated on the minimum and maximum values of Step 3. The divisions are formed with lines parallel to the x- and y-axes.

Step 5. Determines which elements are properly contained in a grid block and records which block. Also determines which elements are in the boundary boxes of the block but not contained in the block. Stores this information in arrays whose indices are the grid block numbers involved (Scheme 1).

Step 6. Sorts the minimum x and y and maximum z of each element.

Step 7. Begins main loop for point visibility test of each element.

Step 8. Using Scheme 2, selects revelant elements with respect to each element in turn. (Relevant elements as they relate to a given element are those elements which could possibly hide some portions of the given element.)

Step 9. Retrieves alternate set of relevant elements from Scheme 1 determined from Step 5. Chooses minimum count and corresponding elements from both schemes.

Step 10. Reduces the relevant set to a smaller subset by performing boundary box tests on the x-, y-, and z-dimensions of the given element as that element relates to its relevant set. Also performs strict overlap tests. This final set is used in the remaining calculations.

Step 11. Determines the equations of the lines of intersections, if any, between the given element and its relevant set. These equations are added to the stack of edges which bound the given element. This augmented count is used only for this particular element. The count is decremented to its original value when the algorithm proceeds to the next element.

Step 12. For each element A_j, finds the points on each edge of A_j that intersect all of the edges of the relevant elements. The actual intersection points chosen are dictated by the theorem presented.

Step 13. Sorts the intersections from left to right per line segment.

Step 14. Determines the visibility of each intersection point along with the end points of each line segment. Initially, only every other intersection point need be examined. This follows from the theorem.

The visibility criteria are described previously in the report.

The process from Step 7 to Step 14 is repeated N times.

REFERENCES

1. Newman, William M.; and Sproull, Robert F.: Principles of Interactive Computer Graphics. New York, McGraw Hill, 1973, pp. 421-454.

2. Sutherland, Ivan E.; Sproull, Robert F.; and Schumacker, Robert A.: A Characterization of Ten Hidden-Surface Algorithms. Computing Surveys, vol. 6, no. 1, Mar. 1974, pp. 1-55.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA RP-1085		

4. Title and Subtitle	5. Report Date
A GENERAL SOLUTION TO THE HIDDEN-LINE PROBLEM	March 1982
	6. Performing Organization Code 505-35-24
7. Author(s)	8. Performing Organization Report No.
David R. Hedgley, Jr.	H-1162
	10. Work Unit No.
9. Performing Organization Name and Address Ames Research Center Dryden Flight Research Facility P.O. Box 273 Edwards, California 93523	11. Contract or Grant No.
	13. Type of Report and Period Covered Reference Publication
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Information for obtaining the computer program is included.

16. Abstract

The requirement for computer-generated perspective projections of three-dimensional objects has escalated. A general solution has been developed at NASA Ames Research Center's Dryden Flight Research Facility. The theoretical solution to this problem is presented. The method is very efficient as it minimizes the selection of points and comparison of line segments and hence avoids the devastation of square-law growth.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer graphics Hidden-line problem Projection Planar polygon	Unclassified-Unlimited STAR category 59

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	17	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1982

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2000	Technical Memorandum

4. TITLE AND SUBTITLE

User's Guide for SKETCH

5. FUNDING NUMBERS

WU 710550 4M 18E 8RR1 005

6. AUTHOR(S)

David R. Hedgley, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Dryden Flight Research Center
P.O. Box 273
Edwards, California 93523-0273

8. PERFORMING ORGANIZATION REPORT NUMBER

H-2411

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA/TM-2000-210388

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified—Unlimited
Subject Category 59
This report is available at http://www.dfrc.nasa.gov/DTRS/

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A user's guide for the computer program SKETCH is presented on this disk. SKETCH solves a popular problem in computer graphics-the removal of hidden lines from images of solid objects. Examples and illustrations are included in the guide. Also included is the SKETCH program, so a user can incorporate the information into a particular software system.

14. SUBJECT TERMS

Eulerian angles, Hidden line, ICORE, SKETCH

15. NUMBER OF PAGES

1 CD-Rom

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited