APPENDIX C

3-Dimensional Graphics Package

Three Dimensional Graphics System

Subroutine Description

Subroutine:

PORT (XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)

Subroutine PORT is used to define the viewport parameters and to calculate a 4 x 4 dimension STMAT used to transform the vectors to the defined viewport cube.

Arguments: XMIN, YMIN, ZMIN defines the 000 cubed vertex.

XMAX, YMAX, ZMAX defines the 111 cubed vertex.

STYLE (I)

This system has the capability of varying the line styles of vectors drawn. If the argument I is positive the greater its value, the shorter the dashes used in drawing the vector in real space. By this I mean, for a particular line style the actual length of the dashes is a constant in three dimensional space. Therefore, it a vector is further away from the view reference point the dashes used to show its projection get smaller. This is analogous to the length of a railroad tie being constant in real space but seeming to decrease in length, as perceived by the eye, the further it is from the observer.

If the argument I is negative, the size of a dash is constant on the projection plane, regardless of the position of the vector in real space. The smaller the argument value, the larger the number of dashes used to draw the vector projection.

After a call to STYLE, setting the style parameter every vector added to an open segment through subroutine LINE will store this line style as an attribute describing this vector. Therefore, every vector entered can have its own unique line style, if a new style is declared between each call to LINE.

If line style is set to 0 a solid line will be used to draw the vectors.

INSTNT (I)

This subroutine sets or clears the immediacy flag, 0 = off, 1 = on. If immediacy is on, any change to the segments of a picture will be effected immediately since this graphics system is to be used with a storage tube or a physical hard copy device. Vectors cannot be erased. Therefore, any operation requiring the removal of a vector must cause the entire picture to be erased and redrawn according to the modification. Therefore, if a segment which was lit is turned off, a call must be made to NEWFRM to create a new frame. Changes requiring the addition of vectors to the picture are also initiated immediately if this flag is set. For instance, if a picture segment previously off is turned on a call is made to NEWSEG which causes all the vectors contained in that segment to be displayed. Similarly, if a vector is added to a segment by a call to LINE, the vector is instantly displayed through a call to NEWLIN.

OPEN (I)

This subroutine opens a display segment such that subsequent calls to LINE will cause the addition of vectors to the open display segment. This subroutine closes any segments left open with a call to CLOSE. IT also sets the visability status of the segment it is opening to condition on. The condition code used to describe the visability of segments is the following:

0 = segment empty

1 = segment dark or off

2 = segment lit or on

VISBLE (I,J)

This subroutine is used to modify the visability status of one of the ten display segments. Argument I refers to the segment number and argument J refers to the visability condition code i.e., 0 = empty, 1 = off, 2 = on. If immediacy is on and a segment previously lit is turned off, a call is made to NEWFRM. If a segment previously off is turned on, a call is directed to NEWSEG.

LINE (RP1, RP2)

This subroutine takes two arguments, RPI and RP2, both dimensioned to be 1 x 3 matrices. These two arguments contain the X, Y and Z coordinates of the end points of a vector to be added to the presently open segment. These vectors are placed in a common area called PAREA as 1 x 4 dimensioned matrices. The fourth position of the first 1 x 4 matrix contains the integer value of the line style attribute to be used in drawing the entered vector. The fourth position of the second 1 x 4 matrix contains an integer from 1 to 10, describing which of the 10, user defined world transformation matrices is to be used in the presentation of this vector. A particular user defined world transform matrix, can be applied or defined to the system through a call to WTRANB. 500 vectors can be entered into each of the 10 segments. If a call is made to LINE and the presently open segment is full, the next higher segment is opened and the present segment is closed. If immediacy is on, a call is made to NEWLIN to display, instantly, each vector entered through line.

CLOSE

This subroutine closes the presently open segment. It opens a FORTRAN data file on disk with a unit number = segment number + 9 i.e., the disk file FOR 10.DAT is associated with segment 1. It stores all the points and attributes associated with the vectors containing in the presently open segment in this disk file. It also resets the line number counter to 1, and the presently open segment to 0 i.e., no segment open.

RITLEF (XP)

This subroutine assigns the world transform conversion matrix for conversion between right and left-handed coordinate systems, to the 4 x 4 matrix (XP).

WTRANB (I, WT)

This subroutine places a user defined world transformation matrix i.e., rotation, translation, scaling, or concatination of any of these, into the ith position of the world transformation common area, WRLTRA.

Ten 4 x 4 matrices can occupy this area simultaneously, and can be referenced through the integer I, where I = 1 - 10. The user can make a call to SETWOR to set the condition code to indicate to subroutine LINE, which user defined world transform matrix is to be applied to the vector, entered through a call to LINE. This capability allows the user to enter the coordinates of an object and place it in the correct position in real space, without knowing the coordinates of the object at that position. For instance, we could plot the motion of Jupiter's moons at monthly intervals with respect to the sun, through proper updating the user defined world transform matrix, used to position each moon. By this, I mean, if the vectors used to draw say, its largest moon, were entered through LINE, while world transformation matrix 1 was declared to be in effect, if we then called set world to 2 and entered the vector describing its second moon, etc., we could place the four mmons in their proper position for January 1st. By updating these four world transformation matrices to their proper positions in February, we could plot their positions using the same data simply by modifying the world transform common area through calls to WTRANB.

User access to these user defined matrices is accomplished through WTRANR.

WTRANR (I, WT)

This subroutine enables the user to access one of the ten user defined world transformation matrices. Argument I indicates which matrix and argument WT is a 4 x 4 matrix which is set equal to the ith user defined matrix in the WTRAL common area.

SETWOR (I)

This subroutine sets the user defined world transform which is to be an attribute of every subsequent vector entered through LINE.

WRLOVR (I)

This subroutine causes a world transform overried indicator to be set. If argument I is an integer from 1 to 10, the ith user defined world transformation matrix is to be applied to every vector output, regardless of the stored world transformation attribute which was in effect at the time the vector was entered through LINE. If the vectors in segment 1 were entered while the eighth user defined world transformation was in effect, the integer 8 appears as an attribute of that vector as the fourth member of the second 1 x 4 matrix, which tells the system to apply the eighth user defined world transformation matrix to that vector. If the user decides he wishes to display that segment or vector with the fifth user defined world transform matrix, he can place a call to WRLOVR with an argument of 5 and then call NEWSEG, NEWLIN, etc.

If the argument I is 0, the system is to use the original world transformation matrix.

STLOVR (I)

This subroutine is analogous to WRLOVR in that, if set to a positive number or a negative number, the defined style is to be applied to every subsequent vector, regardless of the line style which was in effect when the vector was entered. If set to 0, the original line style should be used.

NEWLIN

This subroutine first finds the cross product of the end points of a vector with the overall transformation matrix. It then, provides perspective to these end points, causes a call to clip 3-D, and finally, takes the cross products with the concatinated window and viewport matrix, then initiates a call to plot. If the dash length of the vectors is to be held constant in real space i.e., style≥1, the vector is broken down into a series of vectors before this output process.

SYSMAT

This subroutine causes the concatination of the right to left handed coordinate system transformation matrix, the user defined world transformation matrix, and the system's VTT matrix, for each of the ten user defined world transformation matrices. The produce matrices called the overall world transformation matrix are stored in a common area called system and are utilized by NEWLIN.

REDTRA

This subroutine equates argument WT, a 4 x 4 matrix, with the ith overall world trasnformation matrix found in common area system.

NEWSEG

Subroutine NEWSEG is used to display segment I if the visability of segment I is on. First, it checks which segment is open. If argument I is different from the segment which is now open, it closes the presently open segment, reads in segment I through subroutine SREAD. It then calls NEWLIN according to the number of vectors in the open segment, closes the segment I and reads in the originally open segment. This enables one, while entering the data for segment say, 4 through line to call NEWSEG (1), thereby displaying segment 1, and to be immediately back into the original position ready to enter the next vector into segment 4.

SREAD (I)

This subroutine opens the proper disk data file, reads in the vectors associated with segment I, and updates the line number indicator such that the next vector entered will be added to the end of the vector list.

NEWFRM

This subroutine causes, first, any open segment to be closed, the screen to be cleared, all visability on segments to be displayed through consecutive calls to NEWSEG. The originally open segment is then reopened and its vectors read in from disk.

INIT

This subroutine initialized the entire graphics system and assumes a right handed coordinate system.

OUT4 (Z)

This subroutine writes the contents of the 4 x 4 matrix Z on the screen.

SYSTAT

This subroutine outputs to the screen all relevant information pertaining to the status of the graphics, its matrices, etc.

PROSPT (I)

This subroutine sets the prospective flag 0 = on, 1 = off. This enables the three dimensional graphics output to occur with or without perspective.

TEST (I)

This subroutine causes various stages in the processing of a vector by NEWLIN to be output to the screen, 1 = on, 0 = off.

CLIPON (I)

This subroutine sets the flag which tells NEWLIN whether or not to apply clipping operations. 0 = clipper on, 1 = clipper off.

ROTARO (RX, RY, RZ, X, Y, Z, MAT)

This subroutine will generate a rotation matrix for rotation around an arbitrary point (X,Y,Z) in space where RX, RY, and RZ pertain to the rotation in degrees around the X,Y, and Z axis, repsectively. MAT is the synthesized 4 x 4 matrix.

TRANS (TX, TY, TZ, TMAT)

This subroutine builds a translation matrix TMAT for a translation TX along X, TY along Y, and TZ along the Z axis.

SCALE (SX, SY, SZ, SMAT)

This subroutine builds a scaling matrix, SMAT, wherein the scale factor to be applied to the X,Y, and Z coordinates are SX, SY and SZ, respectively.

ROTX (THETA, XROT)

This subroutine builds a 4 x 4 rotation matrix, XROT, for a rotation theta degrees about the X axis.

ROTY (THETA, YROT)

This subroutine builds a 4 x 4 rotation matrix, YROT for a rotation theta degrees about the Y axis.

ROTZ (THETA, ZROT)

This subroutine builds a 4 x 4 rotation matrix, ZROT, for a rotation theta degrees about the Z axis.

CROS4(AX, BS, PX)

This subroutine generates a 4 x 4 product matrix, PX = to the cross product of AX and BX.

CROS 1 (AX,BX,PX)

This subroutine finds the 1 x 4 product matrix PX which is equal to the cross product of AX and BX.

EQUAS (AX, BX)

This subroutine sets the 4 x 4 matrix, AX = BX.

IDENTY (AX)

This subroutine sets the 4 x 4 matrix AX, equal to the Identity matrix.

SETVEW (VRP, VPN, VPD, VUV)

This argument VRP in this subroutine is the view reference point in world coordinates i.e., a 1 x 4 matrix. The first 4 entries of which are the X, Y, and Z location of the eye of the observer. Argument VPN, a 1 x 4 matrix containing the view plane normal vector i.e., a vector which is perpendicular to the plane in space we are projecting the 3-D image on The argument VPD is the view plane distance, a scaler, indicating the distance between the eye of the observer and the plane of projection. The argument VUV is the view up vector, a 1 x 4 matrix describing the vector which points to the direction we wish to be considered vertical.

This subroutine calculates without the use of any FORTRAN Library trigonometric function, the comple WCS to UVW transform (VIT) i.e. T * R1 * R2 * R3 and its inverse.

WINDOW (FD, BD; UMIN, UMAX, VMIN, VMAX)

This subroutine places the window parameters into a common block called WINDO. It also calculates the window matrix WMAT and places that in a common area called WINDOM.

OUT (V1, V2, I)

This subroutine will output two 1 x 4 point matrices to the screen for debugging purposes.

EQUATE (A11, B11)

This subroutine causes the 1 x 4 point matrix A11 to be set equal to B11.

CLIP3D (P1, P2, N)

This subroutine clips the line segment between the 1 x 4 point matrices P1 and P2, to the defined window volume in the UVW system by using the midpoint subdivision technique utilized in subroutine CLIP of the two dimensional graphics system.

```
$
      SUBROUTINE TRANS(TX,TY,TZ,TMAT)
C *   THIS SUBROUTINE BUILDS A TRANSLATION MATRIX FOR 3D
      DIMENSION TMAT(4,4)
      CALL IDENTY(TMAT)
      TMAT(4,1)=TX
      TMAT(4,2)=TY
      TMAT(4,3)=TZ
      RETURN
      END
      SUBROUTINE SCALE (SX,SY,SZ,SMAT)
C * THIS SUBROUTINE BUILDS A SCALING MATRIX (3D)
      DIMENSION SMAT(4,4)
      CALL IDENTY (SMAT)
      SMAT(1,1)=SX
      SMAT(2,2)=SY
      SMAT(3,3)=SZ
      RETURN
      END
      SUBROUTINE ROTX(THETA,XROT)
      DIMENSION XROT(4,4)
C *   THIS SUBROUTINE BUILDS A 3D ROTATION MATRIX FOR ROTATION
C     ABOUT THE X AXIS (THETA IN DEGREES)
      THETA=(THETA/180.)*3.141592654
      CALL IDENTY(XROT)
      COSA=COS(THETA)
      SINA=SIN(THETA)
      XROT(2,2)=COSA
      XROT(3,2)=SINA
      XROT(2,3)=-1.*SINA
      XROT(3,3)=COSA
      RETURN
      END
      SUBROUTINE ROTY(THETA,YROT)
      DIMENSION YROT(4,4)
C * THIS SUBROUTINE BUILDS A 3D ROTATION MATRIX FOR ROTATION ABOUT
C     THE Y AXIS (THETA IS IN DEGREES CLOCKWISE WHEN VIEWING
C     THE ORIGIN FROM THE +Y AXIS)
      CALL IDENTY(YROT)
      THETA=(THETA/180)*3.141592654
      COSA=COS(THETA)
      SINA=SIN(THETA)
      YROT(1,1)=COSA
      YROT(1,3)=SINA
      YROT(3,1)=-1.*SINA
      YROT(3,3)=COSA
      RETURN
      END
      SUBROUTINE ROTZ(THETA,ZROT)
C *   THIS SUBROUTINE BUILDS A 3D ROTATION MATRIX FOR ROTATION
C     CLOCKWISE ABOUT THE Z AXIS AS VIEWED LOOKING AT THE ORIGIN
*
```

```
$
C       FROM THE +Z AXIS
        DIMENSION ZROT(4,4)
        CALL IDENTY(ZROT)
        THETA=(THETA/180.)*3.141592654
        COSA=COS(THETA)
        SINA=SIN(THETA)
        ZROT(1,1)=COSA
        ZROT(2,1)=SINA
        ZROT(1,2)=-1.*SINA
        ZROT(2,2)=COSA
        RETURN
        END
        SUBROUTINE CROS4 (AX,BX,PX)
        DIMENSION AX(4,4),BX(4,4),PX(4,4)
C       PX=AX * BX
        DO 20 I=1,4
        PX(I,4)=0.
        DO 10 J=1,3
        PX(I,J)=AX(I,1)*BX(1,J)+AX(I,2)*BX(2,J)+AX(I,3)*BX(3,J)
10      CONTINUE
20      CONTINUE
        PX(4,1)=PX(4,1)+BX(4,1)
        PX(4,2)=PX(4,2)+BX(4,2)
        PX(4,3)=PX(4,3)+BX(4,3)
        PX(4,4)=1.
        RETURN
        END
        SUBROUTINE CROS1 (AX,BX,PX)
        DIMENSION AX(1,4),BX(4,4),PX(1,4)
C PX=AX * BX
        DO 10 J=1,3
        PX(1,J)=AX(1,1)*BX(1,J)+AX(1,2)*BX(2,J)+AX(1,3)*BX(3,J)+
        1 AX(1,4)*BX(4,J)
10      CONTINUE
        PX(1,4)=1.
        RETURN
        END
        SUBROUTINE EQUAS (AX,BX)
        DIMENSION AX(4,4),BX(4,4)
C AX IS SET TO EQUAL BX
        DO 20 I=1,4
        DO 10 J=1,4
        AX(I,J)=BX(I,J)
10      CONTINUE
20      CONTINUE
        RETURN
        END
        SUBROUTINE IDENTY (AX)
        DIMENSION AX(4,4)
C AX IS SET TO EQUAL IDENTY MATRIX
*
```

```
      DO 20 I=1,4
      DO 10 J=1,4
      AX(I,J)=0.
      IF (I.EQ.J) AX(I,J)=1.
      AX(4,4)=1.
10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE SETVEW(VRP,VPN,VPD,VUV)
      COMMON/SYSCHA/ICHA
      COMMON /VEWMAT/VTT
      COMMON /VEWINV/VTTI
      COMMON /VEWPLD/DIST
      COMMON/LSTYLE/ISTYLE
      COMMON/NOW/IIK
      COMMON /HARD/A1,A2,A3,A4
      DIMENSION A1(1,4),A2(1,4),A4(1,4)
      DIMENSION VRP(1,4),VPN(1,4),VUV(1,4),W(1,4),RT(4,4),UV(1,4)
      DIMENSION T(4,4),R1(4,4),R2(4,4),R3(4,4),R1I(4,4),VORIG(1,4)
      DIMENSION TI(4,4),R2I(4,4),R3I(4,4),VTT(4,4),VTTI(4,4)
      ICHA=1
      CALL EQUAP(A1,VRP)
      CALL EQUAP(A2,VPN)
      CALL EQUAP(A4,VUV)
      A3=VPD
      DIST=VPD
C VRP= THE VIEW REFERENCE POINT IN WORLD COORDINATES
C     (THE X,Y,Z LOCATION OF THE EYE OF THE OBSERVER)
C VPN= THE VIEW PLANE NORMAL IS A VECTOR WHICH IS PERPENDICULAR
C     TO THE PLANE IN SPACE WE ARE PROJECTING THE 3D IMAGE ON.
C VPD= THE VIEW PLANE DISTANCE IS THE DISTANCE BETWEEN THE
C     EYE OF THE OBSERVER (VRP) AND THE VIEWING PLANE
C VUV= THE VIEW UP VECTOR IS A VECTOR IN WORLD COORDINATES
C     POINTING TOWARD THE DIRRECTION WE WISH TO BE CONSIDERED VERTICLE
C W=   A UNIT VECTOR IN THE UVW COORDINATE SYSTEM CORRESPONDING TO
C     THE Z DIRRECTION IN THE WORLD COORDINATE SYSTEM
C T=   A TRANSLATION MATRIX (4X4) WHICH WILL TRANSLATE THE POINTS
C     FROM A WCS ORIGIN TO A UVW ORIGIN
C R1=  A ROTATION MATRIX (4X4) FOR ROTATION ABOUT THE X AXIS
C R1I= THE INVERSE OF R1
C R2=  A ROTATION MATRIX (4X4) FOR ROTATION ABOUT THE Y AXIS
C R2I= THE INVERSE IF R2
C R3=  A ROTATION MATRIX (4X4) FOR ROTATION ABOUT THE Z AXIS
C R3I= THE INVERSE OF MATRIX R3
C VTT= THE COMPLETE WCS TO UVW TRANSFORM IE. TXR1XR2XR3
C VTTI= THE INVERSE OF VTT IE R3IXR2IXR1IXTI
C V=   A VECTOR POINTING VERTICLE IN UVW CREATED BY THE PROJECTION
C     OF THE VUV VECTOR ONTO THE VIEWING PLANE
C VORIG=THE ORIGIN OF THE UVW SYSTEM IN WORLD COORDINATES (1X4)
*
```

```
$
C CREATE UNIT VECTOR W
      VLENTH=SQRT(VPN(1,1)**2+VPN(1,2)**2+VPN(1,3)**2)
      W(1,1)=VPN(1,1)/VLENTH
      W(1,2)=VPN(1,2)/VLENTH
      W(1,3)=VPN(1,3)/VLENTH
      W(1,4)=1.
C CREATE ORIGIN OF UVW SYSTEM IN WORLD COORDINATES (VORIG)
      VORIG(1,1)=VPD*W(1,1)+VRP(1,1)
      VORIG(1,2)=VPD*W(1,2)+VRP(1,2)
      VORIG(1,3)=VPD*W(1,3)+VRP(1,3)
      VORIG(1,4)=1.
C CREATE THE TRANSFORMATION MATRIX (T)
      CALL IDENTY (T)
      T(4,1)= -1.*VORIG(1,1)
      T(4,2)= -1.*VORIG(1,2)
      T(4,3)= -1.*VORIG(1,3)
C CREATE THE ROTATION MATRIX R1 (ROTATION ABOUT THE X AXIS)
      V=SQRT(W(1,2)**2+W(1,3)**2)
      IF(V.EQ.0)GO TO 77
      COSA=W(1,3)/V
      SINA=W(1,2)/V
      CALL IDENTY (R1)
      R1(2,2)=COSA
      R1(2,3)=SINA
      R1(3,2)=-1.*SINA
      R1(3,3)=COSA
      GO TO 78
77    CONTINUE
      CALL IDENTY(R1)
78    CONTINUE
C CREATE ROTATION MATRIX R2 (ROTATION ABOUT THE Y AXIS)
      SINB=W(1,1)
      COSB=V
      CALL IDENTY (R2)
      R2(1,1)=COSB
      R2(1,3)=SINB
      R2(3,1)=-1.*SINB
      R2(3,3)=COSB
C CONCATENATE R1 AND R2 SUCH THAT RT=R1XR2
      CALL CROS4(R1,R2,RT)
C TRANSFORM THE VIEW UP VECTOR TO ALIGN ITS Z AXIS WITH THE VIEW
C       PLANE NORMAL VECTOR (W AXIS IN UVW)
      CALL CROS1(VUV,RT,UV)
C CREATE THE ROTATION MATRIX R3 (ROTATION ABOUT THE Z AXIS)
      UV(1,3)=0.
      Q=SQRT(UV(1,1)**2+UV(1,2)**2)
      IF(Q.EQ.0)GO TO 55
      SINC=-1.*UV(1,1)/Q
      COSC=UV(1,2)/Q
      CALL IDENTY (R3)
*
```

```
$
        R3(1,1)=COSC
        R3(1,2)=-1.*SINC
        R3(2,1)=SINC
        R3(2,2)=COSC
        GO TO 56
55      CONTINUE
        CALL IDENTY(R3)
56      CONTINUE
C CREATE THE VTT MATRIX BY CONCATENATION OF T,R1,R2,ANDR3
        CALL CROS4(RT,R3,VTT)
        CALL EQUAS(RT,VTT)
        CALL CROS4(T,RT,VTT)
C CREATE INVERSE OF T MATRIX
        CALL IDENTY (TI)
        TI(4,1)=VORIG(1,1)
        TI(4,2)=VORIG(1,2)
        TI(4,3)=VORIG(1,3)
C CREATE INVERSE OF R1 MATRIX
        CALL EQUAS (R1I,R1)
        R1I(2,3)=R1I(3,2)
        R1I(3,2)=SINA
C CREATE INVERSE OF R2 MATRIX (R2I)
        CALL EQUAS (R2I,R2)
        R2I(1,3)=R2I(3,1)
        R2I(3,1)=SINB
C CREATE INVERSE OF R3 MATRIX (R3I)
        CALL EQUAS (R3I,R3)
        R3I(2,1)=R3I(1,2)
        R3I(1,2)=SINC
C CONCATENATE INVERSE MATRIXS SUCH THAT VTTI=R3IXR2IXR1IXTI
        CALL CROS4(R3I,R2I,RT)
        CALL CROS4(RT,R1I,VTTI)
        CALL EQUAS (RI,VTTI)
        CALL CROS4(RI,TI,VTTI)
        IF(IIK.EQ.1)CALL NEWFRM
        RETURN
        END
        SUBROUTINE EQUAP(S,SS)
        DIMENSION S(1,4),SS(1,4)
        DO 10 I=1,4
        S(1,I)=SS(1,I)
10      CONTINUE
        RETURN
        END
        SUBROUTINE PORT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)
        COMMON/VPORT/A,B,C,D,E,F
        COMMON/NOW/J/SYSCHA/ICHA/PORTM/STMAT
        DIMENSION STMAT(4,4),TMAT(4,4),SMAT(4,4)
C *     THIS SUBROUTINE PLACES THE VIEWPORT PARAMETERS IN A COMMON
C       AREA CALLED VPORT AND CAUSES A CALL TO NEWFRM IF IMMEDIACY
*
```

```
$
C       IS ON
        ICHA=1
        A=XMIN
        B=XMAX
        C=YMIN
        D=YMAX
        E=ZMIN
        F=ZMAX
        SX=(XMAX-XMIN)
        SY=(YMAX-YMIN)
        SZ=(ZMAX-ZMIN)
        TX=XMIN
        TY=YMIN
        TZ=ZMIN
        CALL SCALE(SX,SY,SZ,SMAT)
        CALL TRANS(TX,TY,TZ,TMAT)
        CALL CROS4(SMAT,TMAT,STMAT)
        IF(J.GT.0)CALL NEWFRM
        RETURN
        END
        SUBROUTINE STYLE(I)
        COMMON /LSTYLE/J
C * THIS SUBROUNINE PLACES THE STYLE PARAMETER (AN INTEGER .GE. 10)
C       IN A COMMON AREA CALLED LSTYLE
        J=I
        RETURN
        END
        SUBROUTINE INSTNT(I)
        COMMON/NOW/J
C * THIS SUBROUTINE PLACES THE IMMEDIACY VALUE 0=OFF; 1=ON IN A
C       COMMON AREA CALLED NOW.
        J=I
        RETURN
        END
        SUBROUTINE OPEN (I)
        COMMON/SOPEN/J/SEGMNT/SEGSTS/LINNUM/L/UI/KK
C       SOPEN...INTEGER VALUE OF OPEN SEGMENT
C       SEGMNT..A TEN ELEMENT INTEGER ARRAY INDICATING VISIBILITY STATUS
C       OF EACH SEGMENT
C       LINNUM...AN INTEGER SHOWING NEXT LINE TO BE FILLED
C * THIS SUBROUTINE OPENS A SEGMENT AND STORES THE INTEGER SEGMENT
C       NUMBER (INTEGER 1 TO 10) IN A COMMON AREA CALLED OPEN
C       IT ALSO RESTORES THE LINE NUMBER COUNTER TO THE FIRST LINE (1)
C       IE. LINNUM =1;AND CLOSES ANY SEGMENT LEFT OPEN
C       IT THEN UPDATES THE SEGMNT COMMON AREA TO TURN THE VISIBILITY
C       OF PRESENT SEGMENT ON (0=EMPTY;1=OFF;2=ON)
        DIMENSION SEGSTS(10)
        INTEGER SEGSTS
        KK=I
        IF(J.LT.1)GO TO 50
*
```

```
$
      CALL CLOSE
      J=I
      SEGSTS (I)=2
      RETURN
50    CONTINUE
      J=I
      L=1
      SEGSTS(I)=2
      RETURN
      END
      SUBROUTINE VISBLE (I,J)
C * THIS SUBROUTINE SETS THE VISIBILITY STATUS OF SEGMENTS BY CHANGING
C     THE ARRAY SEGSTS FOUND IN THE SEGMNT COMMON AREA
C     IF A FILLED SEGMENT IS TURNED ON WHICH WAS PREVIOUSLY
C     WAS OFF A CALL IS MADE TO NEWSEG...IF IMMEDIACIACY IS ON
C     IF A SEGMENT IS TURNED OFF AND IMMEDIACY IS ON
C     A CALL IS MADE TO NEWFRM
      COMMON/SEGMNT/SEGSTS/NOW/LL
      DIMENSION SEGSTS (10)
      INTEGER SEGSTS
      JK=SEGSTS(I)
      SEGSTS(I)=J
      IF(LL.EQ.0)GO TO 50
      IF(JK.EQ.1.AND.J.EQ.2)CALL NEWSEG(I)
      IF(JK.EQ.2.AND.J.EQ.1)CALL NEWFRM
50    CONTINUE
      RETURN
      END
      SUBROUTINE LINE(RP1,RP2)
      COMMON/LSTYLE/L/NOW/J/LINNUM/K/SOPEN/OP/PAREA/RPOINT
*     COMMON/WORLD/WTRANS/FORCE/IFORC
C * THIS SUBROUTINE PLACES POINTS RPI AND RP2 IN A COMMON AREA
C     CALLED PAREA WHICH WILL HOLD 500 VECTORS (1000 POINTS)
C     IT ALSO PLACES IN THIS COMMON AREA THE INTEGER VALUES
C     OF THE LINE STYLE TO BE USED FOR THIS VECTOR AND THE NUMBER
C     OF A SPECIAL USER DEFINED WORLD TRANSFORM TO BE APPLIED TO
C     THIS VECTOR (#1-10)
      DIMENSION RPOINT (1000,4),RP1(1,3),RP2(1,3)
      INTEGER OP,WTRANS
      IF(IFORC.EQ.1)CALL FORCFT(RP1,RP2)
      RPOINT(K,1)=RP1(1,1)
      RPOINT(K,2)=RP1(1,2)
      RPOINT(K,3)=RP1(1,3)
      RPOINT(K,4)=L
      RPOINT(K+1,1)=RP2(1,1)
      RPOINT(K+1,2)=RP2(1,2)
      RPOINT(K+1,3)=RP2(1,3)
      RPOINT(K+1,4)=WTRANS
      K=K+2
      IF(K+2.GT.1000)CALL OPEN(OP+1)
*
```

```
$
      IF(J.EQ.1)CALL NEWLIN
      RETURN
      END
      SUBROUTINE CLOSE
      COMMON /SEGMNT/SEGSTS/LINNUM/LINE/SOPEN/IH/PAREA/RPOINT
      DIMENSION SEGSTS(10),RPOINT(1000,4),RP1(1,4)
      INTEGER PP,SEGSTS
      IF(IH.EQ.0)RETURN
      IF (LINE.GT.1)GO TO 50
      SEGSTS(IH)=0
      IH=0
      RETURN
50    CONTINUE
      PP=IH+9
      IF(PP.EQ.0)RETURN
      OPEN (UNIT=PP,DEVICE='DSK')
      DO 100 I=1,LINE-2,2
      IB=RPOINT(I,4)
      IC=RPOINT(I+1,4)
      WRITE (PP,20)RPOINT(I,1),RPOINT(I,2),RPOINT(I,3),IB
      N=I+1
      WRITE (PP,20)RPOINT(N,1),RPOINT(N,2),RPOINT(N,3),IC
100   CONTINUE
      IH=0
      LINE=1
20    FORMAT (G,G,G,I)
      IF(PP.EQ.0)RETURN
      CLOSE (UNIT=PP,DEVICE='DSK')
      RETURN
      END
      SUBROUTINE RITLEF (XP)
C * THIS SUBROUTINE BUILDS THE WORLD COORDINATE SYSTEM CONVERSION
C     MATRIX FOR CONVERSION BETWEEN RIGHT TO LEFT HAND COORDINATE
C     SYSTEMS OR VISE VERSA
      DIMENSION XP(4,4)
      CALL IDENTY(XP)
      XP(3,3)=-1.
      RETURN
      END
      SUBROUTINE WTRANB (I,WT)
      COMMON /WRLTRA/WMATRX/SYSCHA/ICHA
C * THIS SUBROUTINE PLACES THE WORLD TRANSFORM WT IN THE
C     (I)TH POSITION IN THE WORLD TRANSFORM COMMON AREA
      DIMENSION WT(4,4),WMATRX(10,4,4)
      ICHA=1
      DO 50 J=1,4
      DO 40 L=1,4
      WMATRX (I,J,L)=WT(J,L)
40    CONTINUE
50    CONTINUE
*
```

```
$
      RETURN
      END
      SUBROUTINE WTRANR (I,WT)
      COMMON/WRLTRA/WMATRX
      DIMENSION WMATRX(10,4,4),WT(4,4)
C * THIS SUBROUTINE READS THE WRLTRA COMMON AREA AND EQUATES
C     THE MATRIX WT TO THE (I)TH ENTRY IN THIS COMMON AREA
      DO 50 J=1,4
      DO 40 L=1,4
      WT(J,L)=WMATRX(I,J,L)
40    CONTINUE
50    CONTINUE
      RETURN
      END
      SUBROUTINE SETWOR (I)
      COMMON/WORLD/J
C * THIS SUBROUTINE SETS THE PRESENT WORLD TRANSFORM WHICH IS TO
C     BE IN EFFECT...AN INTEGER 1-10 STORED IN THE COMMON AREA
C     WORLD
      J=I
      RETURN
      END
      SUBROUTINE WRLOVR (I)
      COMMON /IGNOR/J/NOW/L
C * THIS SUBROUTINE WILL SET A WORLD TRANSFORM OVER RIDE WHICH WILL
C     CAUSE THE (I)TH WORLD COORDINATE SYSTEM TRANSFORMATION
C     MATRIX TO BE APPLIED TO EACH VECTOR REGARDLESS OF THE
C     WORLD TRANSFORM MATRIX WHICH WAS IN EFFECT AT THE TIME
C     THE VECTOR WAS ENTERED THROUGH LINE....IF SET TO 0 THE
C     SYSTEM WILL USE THE ORIGIONAL TRANSFORM...IF SET TO AN
C     INTEGER 1-10, THE (1-10)TH WORLD TRANSFORM WILL BE APPLIED
C     THIS WILL ALLOW THE PROGRAMMER TO ENTER ONE SET OF DATA
C     IN THE SEGMENTS AND THEN SHOW FOR INSTANCE THE EARTH REVOLVING
C     ABOUT THE SUN ECT. USER DEFINED WCS TRANSFORMS ARE ENTERED
C     THROUGH SUBROUTINE WTRANB (I,WT) WHERE I IS AN INTEGER 1-10
C     IF IMMEDIACY IS ON A CALL WILL BE MADE TO NEWFRM
      IF(L.EQ.1)CALL NEWFRM
      J=I
      RETURN
      END
      SUBROUTINE STLOVR (I)
      COMMON /IGNOR2/J/NOW/L
C * THIS SUBROUTINE SETS THE STYLE OVER RIDE WHICH WILL
C     CAUSE EACH VECTOR TO BE DRAWN WITH THE STYLE NUMBER
C     GIVEN AS ARGUMENT (I) REGARDLESS OF THE STYLE IN EFFECT
C     AT THE TIME THE VECTOR WAS ENTERED THROUGH SUBROUTINE LINE
C     IF (I) =0 THE ORIGIONAL STYLE APPLIES, IF AN INTEGER GE 10
C     THE STYLE (I) WILL BE USED FOR ALL SEEGMENTS...IF
C     IMMEDIACY IS ON A CALL IS PLACED TO NEWFRM
      J=I
*
```

```
$
      IF (L.EQ.1)CALL NEWFRM
      RETURN
      END
      SUBROUTINE NEWLIN
      COMMON/VLENTH/VL/CLPER/ICLIP/SPECTI/SPECT/ISTOUT/TEST
      COMMON /READ/LSTWT/LINNUM/NUMBER/PAREA/X/IGNOR/IGN1/IGNOR2/IGN2
      COMMON /NEWLN/WT/WINVEW/WINVEU/VEWPLD/DIST/SYSCHA/ICHA
      DIMENSION X(1000,4),XI(1,4),WT(4,4),XP1(1,4),XP2(1,4)
      DIMENSION TP1(1,4),TP2(1,4),WINVEU(4,4)
      INTEGER TRANNO,TEST,SPECT
      IF(ICHA.EQ.1)CALL SYSMAT
      LNUM=NUMBER-2
      DO 10 I=1,4
      XP1(1,I)=X(LNUM,I)
      LN=LNUM+1
      XP2(1,I)=X(LN,I)
10    CONTINUE
      ISTYLE=XP1(1,4)
      XP1(1,4)=1.
      IF(IGN2.GT.0)ISTYLE=IGN2
C *   ISTYLE IS THE LINE STYLE TO BE USED
      TRANNO=XP2(1,4)
      XP2(1,4)=1.
      IF(IGN1.GT.0)TRANNO=IGN1
C *   TRANNO IS THE NUMBER OF THE USER DEFINED WORLD TRANSFORM
C     MATRIX TO BE USED
      IF(LSTWT.EQ.TRANNO)GO TO 50
      CALL REDTRA(TRANNO,WT)
      LSTWT=TRANNO
50    CONTINUE
      IF (ISTYLE.EQ.0.OR.ISTYLE.LT.0)GO TO 100
      STYLE=ISTYLE
      ISTYLE=0
      CALL EQUAP(TP1,XP1)
      CALL EQUAP(TP2,XP2)
      DX1=TP2(1,1)-TP1(1,1)
      DY1=TP2(1,2)-TP1(1,2)
      DZ1=TP2(1,3)-TP1(1,3)
      V=SQRT(DX1**2+DY1**2+DZ1**2)
      NUM=INT(V/VL)*STYLE/5
      NUM=(NUM/2)*2+1
      RNUM=NUM
      DX1=DX1/RNUM
      DY1=DY1/RNUM
      DZ1=DZ1/RNUM
      DO 200 J=0,NUM-1,2
      J2=J+1
      XP1(1,1)=TP1(1,1)+J*DX1
      XP1(1,2)=TP1(1,2)+J*DY1
      XP1(1,3)=TP1(1,3)+J*DZ1
*
```

```
$
      XP2(1,1)=TP1(1,1)+J2*DX1
      XP2(1,2)=TP1(1,2)+J2*DY1
      XP2(1,3)=TP1(1,3)+J2*DZ1
100   CONTINUE
      IF(TEST.EQ.1)CALL OUT(XP1,XP2,1)
      CALL CROS1(XP1,VT,XI)
      CALL EQUAP(XP1,XI)
      CALL CROS1(XP2,VT,XI)
      CALL EQUAP(XP2,XI)
      IF(TEST.EQ.1)CALL OUT(XP1,XP2,2)
      IF(SPECT.EQ.1)GO TO 31
      A=(1./(DIST+XP1(1,3)))*DIST
      B=(1./(DIST+XP2(1,3)))*DIST
      XP1(1,1)=XP1(1,1)*A
      XP1(1,2)=XP1(1,2)*A
      XP2(1,1)=XP2(1,1)*B
      XP2(1,2)=XP2(1,2)*B
31    CONTINUE
      IF(TEST.EQ.1)CALL OUT(XP1,XP2,3)
      IF (ICLIP.EQ.1)GO TO 404
      CALL CLIP3D(XP1,XP2,IY)
404   CONTINUE
      IF(TEST.EQ.1)CALL OUT(XP1,XP2,4)
      IF (IY.EQ.0)RETURN
      CALL CROS1(XP1,WINVEU,XI)
      CALL EQUAP(XP1,XI)
      CALL CROS1(XP2,WINVEU,XI)
      CALL EQUAP(XP2,XI)
      IF(TEST.EQ.1)CALL OUT(XP1,XP2,IY)
      CALL PLOT (XP1,XP2,ISTYLE)
200   CONTINUE
      RETURN
      END
      SUBROUTINE SYSMAT
      COMMON/VEWMAT/VTT/SYSTEM/TRANS/SYSCHA/IJ/WINVEW/WINVEU
      COMMON/PORTM/TPORT/WINDOW/TWIND/COORD/CORDIN
      COMMON/HARD/A1,A2,A3,A4/WINDO/B1,B2,B3,B4,B5,B6
      DIMENSION A1(1,4),A2(1,4),A4(1,4)
      DIMENSION VTT(4,4),TRANS(10,4,4),T(4,4),T2(4,4),TPORT(4,4)
      DIMENSION WINVEU(4,4),TWIND(4,4),CORDIN(4,4)
      CALL SETVEW(A1,A2,A3,A4)
      CALL WINDOW(B1,B2,B3,B4,B5,B6)
      DO 10 I=1,10
      LC=I
      CALL WTRANR(LC,T)
      CALL CROS4(CORDIN,T,T2)
      CALL EQUAS(T,T2)
      CALL CROS4(T,VTT,T2)
      DO 30 IL=1,4
      DO 20 IK=1,4
```

```
$
        TRANS(I,IL,IK)=T2(IL,IK)
20      CONTINUE
30      CONTINUE
10      CONTINUE
        IJ=0
        CALL CROS4(TWIND,TPORT,WINVEU)
        RETURN
        END
        SUBROUTINE REDTRA(I,WT)
        COMMON/SYSTEM/TRANS
        DIMENSION TRANS(10,4,4),WT(4,4)
C *     THIS SUBROUTINE READS THE (I)TH WORLD TRANSFORM
C       FROM THE COMMON AREA SYSTEM AND SETS WT EQUAL TO IT
        DO 10 IL=1,4
        DO 20 IK=1,4
        WT(IL,IK)=TRANS(I,IL,IK)
20      CONTINUE
10      CONTINUE
        RETURN
        END
        SUBROUTINE NEWSEG(I)
        COMMON /SOPEN/OP/LINNUM/L/PAREA/X/SEGMNT/SEGSTS/NEWW/IL,IK
        DIMENSION X(1000,4),SEGSTS(10)
        INTEGER OP,SEGSTS
        IF (SEGSTS(I).NE.2)RETURN
        IK=OP
        IL=I
        IF(OP.EQ.I)GO TO 100
        CALL CLOSE
        CALL SREAD (I)
100     CONTINUE
        IJ=L
        DO 50 L=3,L,2
        CALL NEWLIN
50      CONTINUE
        L=IJ
        IF(IK.EQ.0)RETURN
        IF(IL.NE.IK)CALL CLOSE
        IF(IL.NE.IK)CALL SREAD(IK)
        RETURN
        END
        SUBROUTINE SREAD (I)
        COMMON/SOPEN/OP/LINNUM/L/PAREA/X
        DIMENSION X(1000,4)
        INTEGER PP
        OP=I
        PP=OP+9
        IF(PP.EQ.0)RETURN
        OPEN(UNIT=PP,DEVICE='DSK')
        DO 10 II=1,1000
```

```
      READ(PP,25,END=26)X(II,1),X(II,2),X(II,3),ILIM
      X(II,4)=ILIM
      LINE=II
10    CONTINUE
25    FORMAT(G,G,G,G)
26    CONTINUE
      L=LINE+1
      IF(PP.EQ.0)RETURN
      CLOSE(UNIT=PP,DEVICE='DSK')
      RETURN
      END
      SUBROUTINE NEWFRM
      COMMON/SYSCHA/ICHA/SEGMNT/SEGSTS/SOPEN/OP/SAVE/IS
      COMMON/AXON/ION
      DIMENSION SEGSTS(10)
      INTEGER SEGSTS
      IS=OP
      IF(ION.EQ.1)CALL AXIS
      CALL ERASE
      CALL CLOSE
      IF (ICHA.NE.0)CALL SYSMAT
      DO 10 I=1,10
      IJJJ=I
      IF(SEGSTS(IJJJ).NE.2)GO TO 5
      CALL NEWSEG(IJJJ)
5     CONTINUE
10    CONTINUE
      CALL SREAD(IS)
      RETURN
      END
      SUBROUTINE INIT
      COMMON/FIT/XMX,YMX,ZMX,XMN,YMN,ZMN
      COMMON /SYSCHA/ICHA
      DIMENSION VRP(1,4),VPN(1,4),VUY(1,4),X(4,4)
      XMX=-1E30
      YMX=-1E30
      ZMX=-1E30
      XMN=1E30
      YMN=1E30
      ZMN=1E30
      ICHA=1
C *   THIS SUBROUTINE INITIALIZES THIS THREE DIMENTIONAL
C     GRAPHICS SYSTEM
      CALL INITT(30)
      CALL TERM (0,4096)
      CALL SWINDO (0,4096,0,2731)
      CALL DWINDO (0.,1.2,0.,1.2)
      VRP(1,1)=0.
      VRP(1,2)=0.
      VRP(1,3)=10.
```

```
$
      VRP(1,4)=1.
      VPN(1,1)=0.
      VPN(1,2)=0.
      VPN(1,3)=-5.
      VPN(1,4)=1.
      DIST=10.
      VUV(1,1)=0.
      VUV(1,2)=1.
      VUV(1,3)=0.
      VUV(1,4)=1.
      CALL SETVEW(VRP,VPN,DIST,VUV)
      CALL WINDOW(-1.,1.,-1.,1.,-1.,1.)
      CALL PORT(0.,1.,0.,1.,0.,1.)
      CALL SETWOR(1)
      CALL RIGHT
      CALL IDENTY(X)
      DO 50 I=1,10
      IJJJ=I
      CALL WTRANB (IJJJ,X)
50    CONTINUE
      CALL INSTNT(0)
      CALL STYLE (0)
      CALL WRLOVR (0)
      CALL STLOVR (0)
      CALL OPEN (1)
      RETURN
      END
      SUBROUTINE OUT4(Z)
C * THIS SUBROUTINE OUTPUTS A 4X4 MATRIX TO THE SCREEN
      DIMENSION Z(4,4)
      DO 10 I=1,4
      WRITE(5,20)Z(I,1),Z(I,2),Z(I,3),Z(I,4)
10    CONTINUE
20    FORMAT(G,G,G,G)
      RETURN
      END
      SUBROUTINE PLOT(XP1,XP2,ISTYLE)
      DIMENSION XP1(1,4),XP2(1,4)
      IF(XP1(1,1).GT.1.)XP1(1,1)=1.
      IF(XP1(1,2).GT.1.)XP1(1,2)=1.
      IF(XP2(1,1).GT.1.)XP2(1,1)=1.
      IF(XP2(1,2).GT.1.)XP2(1,2)=1.
      IF(XP1(1,1).LT.0.)XP1(1,1)=0.
      IF(XP1(1,2).LT.0.)XP1(1,2)=0.
      IF(XP2(1,1).LT.0.)XP2(1,1)=0.
      IF(XP2(1,1).LT.0.)XP2(1,1)=0.
      IF(XP2(1,2).LT.0.)XP2(1,2)=0.
      IF(XP1(1,1).EQ.XP2(1,1).AND.XP1(1,2).EQ.XP2(1,2))RETURN
      IF(ISTYLE.LT.0)CALL PLODSH(XP1,XP2,ISTYLE)
      IF(ISTYLE.LT.0)RETURN
*
```

```
$
      CALL MOVEA(XP1(1,1),XP1(1,2))
      CALL DRAWA(XP2(1,1),XP2(1,2))
      CALL DRAWA(XP1(1,1),XP1(1,2))
      RETURN
      END
      SUBROUTINE SYSTAT
C * THIS SUBROUTINE REPORTS THE SYSTEM STATUS
      COMMON/VPORT/A,B,C,D,E,F
      COMMON/NOW/J/SYSCHA/ICHA/PORTM/PORTMT
      COMMON/WINDO/FD,BD,UMIN,UMAX,VMIN,VMAX/WINDOM/WINDMT
      COMMON/HARD/VRP,VPN,VPD,VUV/VEWMAT/VTT/LINNUM/LINE/SEGMNT/SEGSTS
      COMMON/SOPEN/INOPEN/PAREA/TPOINT/LSTYLE/LSTYLE
      COMMON/WORLD/INDWLD/COORD/CORDIN/IGNOR/WLDOVR
      COMMON/IGNOR2/ISTY
      DIMENSION WINDMT(4,4),PORTMT(4,4),VRP(1,4),VPN(1,4),VUV(1,4)
      DIMENSION VTT(4,4),SEGSTS(10),TPOINT(1000,4)
      DIMENSION WS(4,4),CORDIN(4,4)
      INTEGER WLDOVR,SEGSTS
      WRITE(5,10)
10    FORMAT('        WINDO PARAMETERS:')
      WRITE(5,20)FD
20    FORMAT('                 FRONT DISTANCE= ',G)
      WRITE(5,25)BD
25    FORMAT('                 BACK DISTANCE= ',G)
      WRITE(5,30)UMIN
30    FORMAT('                 UMIN= ',G)
      WRITE(5,35)UMAX
35    FORMAT('                 UMAX= ',G)
      WRITE(5,40)VMIN
40    FORMAT('                 VMIN= ',G)
*     WRITE(5,45)VMAX
45    FORMAT('                 VMAX= ',G)
      WRITE(5,50)
50    FORMAT('        VIEWPORT PARAMETERS:')
      WRITE(5,55)A
55    FORMAT('                 XMIN= ',G)
      WRITE(5,60)B
60    FORMAT('                 XMAX= ',G)
      WRITE(5,65)C
65    FORMAT('                 YMIN= ',G)
      WRITE(5,70)D
70    FORMAT('                 YMAX= ',G)
      WRITE(5,75)E
75    FORMAT('                 ZMIN= ',G)
      WRITE(5,80)F
80    FORMAT('                 ZMAX= ',G)
      WRITE(5,85)
85    FORMAT('        SETVIEW PARAMETERS:')
      WRITE(5,90)VRP
90    FORMAT('                 VIEW REFERENCE POINT= ',G)
*
```

```
$
        WRITE(5,95)VPN
95      FORMAT('                VIEW PLANE NORMAL VECTOR= ',G)
        WRITE(5,100)VPD
100     FORMAT('                VIEW PLAND DISTANCE= ',G)
        WRITE(5,105)VUV
105     FORMAT('                VIEW UP VECTOR= ',G)
        WRITE(5,106)
106     FORMAT('          VTT MATRIX 4X4:')
        CALL OUT4(VTT)
        WRITE (5,108)
108     FORMAT('          WINDOW MATRIX')
        CALL OUT4(WINDMT)
        WRITE(5,110)
110     FORMAT('          VIEWPORT MATRIX')
        CALL OUT4(PORTMT)
        WRITE(5,125)J
125     FORMAT('          IMMEDIACY INDICATOR 0=OFF;1=ON ',G)
        WRITE(5,130)
130     FORMAT('          SEGMENT STATUS 2=ON,1=OFF,0=EMPTY')
        DO 140 K=1,10
        WRITE(5,145)K,SEGSTS(K)
140     CONTINUE
145     FORMAT(G,G)
        WRITE(5,150)LINE
150     FORMAT('          NEXT LINE TO BE FILLED= ',G)
        WRITE(5,155)IMOPEN
155     FORMAT('          PRESENTLY OPEN SEGMENT INDICATOR 0=ALL OFF ',G)
        WRITE(5,160)ISTY
160     FORMAT('          STYLE OVER RIDE NUMBER 0=OFF ',G)
        WRITE(5,165)WLDOVR
165     FORMAT('          WORLD MATRIX OVER RIDE INDICATOR 0=OFF',G)
        WRITE(5,170)ICHA
170     FORMAT('          WORLD MATRIX UPDATE INDICATOR 0=OK,1=UPDATE ',G)
        WRITE(5,175)INDWLD
175     FORMAT('          WORLD TRANSFORM INDICATOR (INPUT) ',G)
        WRITE(5,180)LSTYLE
180     FORMAT('          LINE STYLE INDICATOR (INPUT) ',G)
        IF(CORDIN(3,3).LT.0)GO TO 200
        WRITE (5,190)
190     FORMAT('          SYSTEM NOW ASSUMING LEFT HANDED CS INPUT')
        GO TO 210
200     CONTINUE
        WRITE(5,205)
205     FORMAT('          SYSTEM NOW ASSUMING RIGHT HANDED CS INPUT')
210     CONTINUE
        WRITE (5,215)
215     FORMAT ('         PRESENT STATUS OF LEFT TO RIGHT MAT')
        CALL OUT4(CORDIN)
        WRITE (5,220)
220     FORMAT('          USER DEFINED WORLD TRANSFORMS')
*
```

```
$
      DO 230 I=1,10
      K=I
      CALL REDUSE (K,WS)
      WRITE (5,350)I
350   FORMAT(G)
      CALL OUT4(WS)
230   CONTINUE
      WRITE(5,360)
360   FORMAT('        COMPLETE SYSTEM TRANSFORMS')
      DO 390 I=1,10
      WRITE(5,370)I
370   FORMAT(G)
      K=I
      CALL REDTRA(K,WS)
      CALL OUT4(WS)
390   CONTINUE
      WRITE(5,260)
260   FORMAT('        LIST OF VECTORS ENTERED IN NOW OPEN SEGMENT')
      WRITE (5,265)
265   FORMAT ('       LINE NO. X1  Y1  Z1   LINE STYLE')
      WRITE (5,270)
270   FORMAT ('       LINE NO. X2  Y2  Z2  WORLD TRANSFORM NUMBER')
      DO 300 I=1,LINE-1
*     WRITE(5,29)I,TPOINT(I,1),TPOINT(I,2),TPOINT(I,3),TPOINT(I,4)
29    FORMAT(I4,G,G,G,G)
300   CONTINUE
      RETURN
      END
      SUBROUTINE RIGHT
      COMMON/SYSCHA/ICHA/COORD/CORDIN
      DIMENSION CORDIN(4,4)
      ICHA=1
      CALL RITLEF(CORDIN)
      RETURN
      END
      SUBROUTINE LEFT
      COMMON /SYSCHA/ICHA/COORD/CORDIN
      DIMENSION CORDIN(4,4)
      ICHA=1
      CALL IDENTY(CORDIN)
      RETURN
      END
      SUBROUTINE REDUSE (I,WT)
      COMMON/WRLTRA/X
      DIMENSION X(10,4,4),WT(4,4)
      DO 10 K=1,4
      DO 20 J=1,4
      WT(K,J)=X(I,K,J)
20    CONTINUE
10    CONTINUE
*
```

```
$
      COMMON/WRLTRA/X
      DIMENSION X(10,4,4),WT(4,4)
      DO 10 K=1,4
      DO 20 J=1,4
      WT(K,J)=X(I,K,J)
20    CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE SETHLP
      DIMENSION SP(1,4),TP(1,4),VRP(1,4),VPN(1,4),VUV(1,4)
      WRITE(5,10)
10    FORMAT('         WHERE IS YOUR EYE (X,Y,Z)?')
      READ(5,20)VRP(1,1),VRP(1,2),VRP(1,3)
      VRP(1,4)=1.
      WRITE(5,30)
20    FORMAT(G,G,G)
30    FORMAT('         MOVING ALONG YOUR LINE OF SIGHT WHAT IS THE
      1 POINT AT WHICH YOU INTERSECT THE SCREEN OF PROJECTION(X,Y,Z)')
      READ(5,20)SP(1,1),SP(1,2),SP(1,3)
      WRITE(5,40)
40    FORMAT('         STANDING ON POINT JUST ENTERED WHAT IS
      1 A POINT YOU WISH TO BE DIRRECTLY OVER YOUR HEAD?')
      READ(5,20)TP(1,1),TP(1,2),TP(1,3)
      VPN(1,1)=SP(1,1)-VRP(1,1)
      VPN(1,2)=SP(1,2)-VRP(1,2)
      VPN(1,3)=SP(1,3)-VRP(1,3)
      VPD=SQRT(VPN(1,1)**2+VPN(1,2)**2+VPN(1,3)**2)
      VPN(1,4)=1.
      VUV(1,1)=TP(1,1)-SP(1,1)
*     VUV(1,2)=TP(1,2)-SP(1,2)
      VUV(1,3)=TP(1,3)-SP(1,3)
      VUV(1,4)=1.
      CALL INIT
      CALL SETVEW (VRP,VPN,VPD,VUV)
      CALL SYSMAT
      RETURN
      END
      SUBROUTINE PROSPT(I)
      COMMON/SPECTI/J
      J=I
      RETURN
C     THIS SUBROUTINE TURNS THE PROSPECTIVE ON(0),OFF(1)
      END
      SUBROUTINE TEST(I)
      COMMON/TSTOUT/J
      J=I
      RETURN
C     THIS SUBROUTINE CAUSES VECTOR STAGES IN NEWLIN TO BE OUTPUT
C     1=ON...0=OFF
*
```

```
$
      END
      SUBROUTINE CLIPON (I)
      COMMON/CLPER/ICLIP
C THIS SUBROUTINE TURNS ON THE CLIPPER (I=0) OR OFF(I=1)
      J=I
      RETURN
      END
      SUBROUTINE ROTARO(RX,RY,RZ,X,Y,Z,MAT)
      DIMENSION MAT(4,4),X1(4,4),VROTX(4,4),VROTY(4,4),VROTZ(4,4)
      DIMENSION TEMP(4,4)
C     THIS SUBROUTINE WILL GENERATE A ROTATION MATRIX FOR ROTATION
C     ABOUT A POINT X,Y,Z ...ROTATION ORDER X,Y,Z AXIS 1,2,3
      CALL ROTX(RX,VROTX)
      CALL ROTY(RY,VROTY)
      CALL ROTZ(RZ,VROTZ)
      CALL TRANS(-1*X,-1*Y,-1*Z,X1)
      CALL CROS4(X1,VROTX,TEMP)
      CALL CROS4(TEMP,VROTY,X1)
      CALL CROS4(X1,VROTZ,TEMP)
      CALL TRANS(X,Y,Z,X1)
      CALL CROS4(TEMP,X1,MAT)
      RETURN
      END
      SUBROUTINE PLODSH(XP1,XP2,ISTYLE)
      DIMENSION XP1(1,4),XP2(1,4)
      NSTYLE=-1*ISTYLE+100
      VECLEN=SQRT((XP2(1,1)-XP1(1,1))**2+(XP2(1,2)-XP1(1,2))**2)
      NUMVEC=(VECLEN/4.*NSTYLE)
      NUMVEC=NUMVEC/2
      NUMVEC=NUMVEC*2+1
      DX=(XP2(1,1)-XP1(1,1))/NUMVEC
      DY=(XP2(1,2)-XP1(1,2))/NUMVEC
      DO 100 I=1,NUMVEC,2
      J=I-1
      X1=XP1(1,1)+J*DX
      Y1=XP1(1,2)+J*DY
      X2=XP1(1,1)+I*DX
      Y2=XP1(1,2)+I*DY
      CALL MOVEA(X1,Y1)
      CALL DRAWA(X2,Y2)
      CALL DRAWA(X1,Y1)
100   CONTINUE
      RETURN
      END
      SUBROUTINE FORCFT(RP1,RP2)
      COMMON/FIT/XMX,YMX,ZMX,XMN,YMN,ZMN
      DIMENSION RP1(1,3),RP2(1,3)
      IF(RP1(1,1).GT.XMX)XMX=RP1(1,1)
      IF(RP2(1,1).GT.XMX)XMX=RP2(1,1)
      IF(RP1(1,1).LT.XMN)XMN=RP1(1,1)
```

```
$
      IF(RP2(1,1).LT.XMN)XMN=RP2(1,1)
      IF(RP1(1,2).GT.YMX)YMX=RP1(1,2)
      IF(RP2(1,2).GT.YMX)YMX=RP2(1,2)
      IF(RP1(1,2).LT.YMN)YMN=RP1(1,2)
      IF(RP2(1,2).LT.YMN)YMN=RP2(1,2)
      IF(RP1(1,3).GT.ZMX)ZMX=RP1(1,3)
      IF(RP2(1,3).GT.ZMX)ZMX=RP2(1,3)
      IF(RP1(1,3).LT.ZMN)ZMN=RP1(1,3)
      IF(RP2(1,3).LT.ZMN)ZMN=RP2(1,3)
      RETURN
      END
      SUBROUTINE MAKFIT(I)
      COMMON/FORCE/IFORC
      IFORC=I
      RETURN
      END
      SUBROUTINE AXIS
      COMMON/FIT/XMX,YMX,ZMX,XMN,YMN,ZMN/FORCE/IFORC/AXON/ION
      DIMENSION X1(1,3),X2(1,3)
      CALL MAKFIT(1)
      IF(ION.EQ.1)GO TO 100
      ION=1
      RETURN
100   CONTINUE
      DX=XMX-XMN
      DY=YMX-YMN
      DZ=ZMX-ZMN
      DXLOG=ALOG10(DX)
      DYLOG=ALOG10(DY)
      DZLOG=ALOG10(DZ)
      RX=EXP(INT(DXLOG)-1.)
      RY=EXP(INT(DYLOG)-1.)
      RZ=EXP(INT(DZLOG)-1.)
      DO 200 I=1,200
      XTEST=XTEST-RX
      IF(XTEST.LT.XMN)GO TO 201
200   CONTINUE
201   CONTINUE
      DO 250 I=1,200
      YTEST=YTEST-RY
      IF(YTEST.LT.YMN)GO TO 251
250   CONTINUE
251   CONTINUE
      DO 300 I=1,200
      ZTEST=ZTEST-RZ
      IF(YTEST.LT.ZMN)GO TO 301
300   CONTINUE
301   CONTINUE
      XDIST=RX*(.5)
      YDIST=RY*(.5)
*
```

```
$
        ZDIST=RZ*(.5)
        DO 400 I=0,100
        XNOW=XTEST+I*RX
        YNOW=YTEST+I*RY
        ZNOW=ZTEST+I*RZ
        X1(1,1)=XNOW
        X1(1,2)=0.
        X1(1,3)=-ZDIST
        X2(1,1)=XNOW
        X2(1,2)=0.
        X2(1,3)=ZDIST
        CALL LINE(X1,X2)
        X1(1,2)=-YDIST
        X1(1,3)=0.
        X2(1,2)=YDIST
        X2(1,3)=0.
        CALL LINE(X1,X2)
        X1(1,1)=-XDIST
        X1(1,2)=YNOW
        X1(1,3)=0.
        X2(1,1)=XDIST
        X2(1,2)=YNOW
        X2(1,3)=0.
        CALL LINE(X1,X2)
        X1(1,1)=0.
        X1(1,3)=-ZDIST
        X2(1,1)=0.
        X2(1,3)=ZDIST
        CALL LINE(X1,X2)
        X1(1,1)=0.
        X1(1,2)=-YDIST
        X1(1,3)=ZNOW
        X2(1,1)=0.
        X2(1,2)=YDIST
        X2(1,3)=ZNOW
        CALL LINE(X1,X2)
        X1(1,1)=-XDIST
        X1(1,2)=0.
        X2(1,1)=XDIST
        X2(1,2)=0.
        CALL LINE(X1,X2)
400     CONTINUE
        RETURN
        END
        SUBROUTINE WINDOW (A,B,C,D,E,F)
C * THIS SUBROUTINE PLACES WINDOW PARAMETERS INTO COMMON BLOCK
        COMMON/WINDO/FD,BD,UMIN,UMAX,VMIN,VMAX /NOW/III/WINDOM/WMAT
        COMMON /VLENTH/VL/SYSCHA/ICHA
        DIMENSION WMAT(4,4),SMAT(4,4),TMAT(4,4)
        ICHA=1
```

```
$.
        FD=A
        BD=B
        UMIN=C
        UMAX=D
        VMIN=E
        VMAX=F
        TX=-1.*UMIN
*       [illegible].*VMIN
        TZ=-1.*FD
        SX=1./(UMAX-UMIN)
        SY=1./(VMAX-VMIN)
        SZ=1./(BD-FD)
        VL=SQRT(SX**2+SY**2+SZ**2)
        CALL TRANS(TX,TY,TZ,TMAT)
        CALL SCALE(SX,SY,SZ,SMAT)
        CALL CROS4 (TMAT,SMAT,WMAT)
        IF(III.EQ.1)CALL NEWFRM
        RETURN
        END
        SUBROUTINE OUT (V1,V2,I)
        DIMENSION V1(1,4),V2(1,4)
        WRITE (5,28)I
        WRITE (5,28)V1
        WRITE (5,28)
        WRITE (5,28)V2
        WRITE (5,28)
        WRITE (5,28)
28      FORMAT (G)
        RETURN
        END
        SUBROUTINE EQUATE (A11,B11)
C * A11(1,4) IS SET EQUAL TO B11(1,4)
        DIMENSION A11(1,4),B11(1,4)
        DO 20 I=1,4
        A11(1,I)=B11(1,I)
20      CONTINUE
        RETURN
        END
        SUBROUTINE CLIP3D (P1,P2,N)
        COMMON/WINDO/FD,BD,UMIN,UMAX,VMIN,VMAX
        DIMENSION P1(1,4),P2(1,4),P1N(1,4),P2N(1,4)
        REAL P1,P2,P1N,P2N
C * THIS SUBROUTINE CLIPS A LINE SEGMENT BETWEEN P1 AND P2 TO THE
C       THE DEFINED WINDOW VOLUME IN THE UVW SYSTEM
C * N=1 FOR SEGMENT VISIBLE; N=0 IF INVISIBLE
        N=0
        CALL EQUATE (P1N,P1)
        CALL EQUATE  (P2N,P2)
        DW=P2(1,3)-P1(1,3)
        DX=P2(1,1)-P1(1,1)
*
```

```
$
        DY=P2(1,2)-P1(1,2)
        IF(P1(1,3).LT.FD.AND.P2(1,3).LT.FD)RETURN
        IF(P1(1,3).GT.BD.AND.P2(1,3).GT.BD)RETURN
        IF (DW.EQ.0)GO TO 300
        RDW=1./DW
        N=1
        RDWDX=RDW*DX
        RDWDY=RDW*DY
        IF (P1(1,3).GT.FD )GO TO 50
        P1N(1,3)=FD
        P1N(1,1)=((FD-P1(1,3))*RDWDX)+P1(1,1)
        P1N(1,2)=((FD-P1(1,3))*RDWDY)+P1(1,2)
50      CONTINUE
        IF (P2(1,3).GT.FD)GO TO 100
        P2N(1,3)=FD
        P2N(1,1)=((FD-P1(1,3))*RDWDX)+P1(1,1)
        P2N(1,2)=((FD-P1(1,3))*RDWDY)+P1(1,2)
100     CONTINUE
        IF(P1(1,3).LT.BD)GO TO 150
        P1N(1,3)=BD
        P1N(1,1)=((BD-P1(1,3))*RDWDX)+P1(1,1)
        P1N(1,2)=((BD-P1(1,3))*RDWDY)+P1(1,2)
150     CONTINUE
        IF(P2(1,3).LT.BD)GO TO 200
        P2(1,3)=BD
        P2N(1,3)=BD
        P2N(1,1)=((BD-P1(1,3))*RDWDX)+P1(1,1)
        P2N(1,2)=((BD-P1(1,3))*RDWDY)+P1(1,2)
200     CONTINUE
300     CONTINUE
        CALL CLIP(P1N,P2N,N)
        IF (N.EQ.0)RETURN
        IF(ABS(DY).LT.1.0E-10)GO TO 210
        P1N(1,3)=(((P1N(1,2)-P1(1,2))/DY)*DW)+P1(1,3)
        P2N(1,3)=(((P2N(1,2)-P1(1,2))/DY)*DW)+P1(1,3)
        GO TO 220
210     CONTINUE
        IF(ABS(DX).LT.1.0E-10)GO TO 220
        P1N(1,3)=(((P1N(1,1)-P1(1,1))/DX)*DW)+P1(1,3)
        P2N(1,3)=(((P2N(1,1)-P1(1,1))/DX)*DW)+P1(1,3)
220     CONTINUE
        CALL EQUATE(P1,P1N)
        CALL EQUATE(P2,P2N)
        RETURN
        END

        SUBROUTINE CLIP (TP1,TP2,N)
C * THIS SUBROUTINE FINDS PORTION OF LINE VECTOR EXISTING
C * WITHIN DEFINED WINDOW
        DIMENSION TP1(1,3),TP2(1,3),IA(3,4),ITEST(4),TP(2,3),S1(1,3),
*
```

```
$
         1 S2(1,3)
         COMMON/WINDO/FD,BD,XWINDL,XWINDH,YWINDL,YWINDH
         N=1
         DO 5 I=1,3
         TP(1,I)=TP1(1,I)
         TP(2,I)=TP2(1,I)
5        CONTINUE
20       CONTINUE
         DO 8 I=1,2
         IA(I,1)=0
         IF(TP(I,1).GT.XWINDH)IA(I,1)=1
         IA(I,3)=0
         IF(TP(I,2).GT.YWINDH)IA(I,3)=1
         IA(I,2)=0
         IF(TP(I,1).LT.XWINDL)IA(I,2)=1
         IA(I,4)=0
         IF(TP(I,2).LT.YWINDL)IA(I,4)=1
8        CONTINUE
         DO 10 I=1,4
         IA(3,I)=0
         IF(IA(1,I).EQ.1.AND.IA(2,I).EQ.1)IA(3,I)=1
10       CONTINUE
         ITEST(2)=MAX0(IA(1,1),IA(1,2),IA(1,3),IA(1,4))
         ITEST(3)=MAX0(IA(2,1),IA(2,2),IA(2,3),IA(2,4))
         ITEST(1)=MAX0(ITEST(2),ITEST(3))
         ITEST(4)=MAX0(IA(3,1),IA(3,2),IA(3,3),IA(3,4))
         IF (ITEST(1).EQ.0)GO TO 30
         N=0
         IF (ITEST(4).EQ.1)RETURN
         N=1
         ITEST(4)=0
C * IF ITEST(2)=0 POINT TP1 INSIDE
C * IF ITEST(3)=0 POINT TP2 INSIDE
C * IF ITEST(1)=0 ENTIRE VECTOR INSIDE WINDOW
C * IF ITEST(4)=1 ENTIRE VECTOR OUTSIDE WINDOW
         GO TO 40
30       CONTINUE
         RETURN
40       CONTINUE
         IF(ITEST(2).EQ.0.OR.ITEST(3).EQ.0)GO TO 200
         CALL EQUAL(S1,TP2)
         CALL EQUAL(S2,TP1)
50       CONTINUE
C * THE FOLLOWING FIVE LINES WILL FIND A POINT INSIDE BY
C * TAKING MIDPOINTS PROGRESIVELY
         CALL MIDPNT(TP1,S1)
         CALL MIDPNT(TP2,S2)
         CALL INSIDE(S1,L1)
         CALL INSIDE(S2,L2)
         IF(L1.EQ.1.AND.L2.EQ.1)GO TO 50
*
```

```
$
      IF(L1.EQ.0.AND.L2.EQ.1)CALL EQUAL (L2,L1)
      IF(L1.EQ.1.AND.L2.EQ.0)CALL EQUAL(L1,L2)
C * NOW S1 IS A POINT INSIDE WINDOW AND ON LINE
      IF(ITEST(2).EQ.1) CALL INTSEC(TP1,S1)
      IF(ITEST(3).EQ.1) CALL INTSEC(TP2,S1)
      RETURN
200   CONTINUE
      IF(ITEST(2).EQ.0)CALL INTSEC(TP2,TP1)
      N=1
      IF(ITEST(3).EQ.0)CALL INTSEC(TP1,TP2)
      RETURN
      END
      SUBROUTINE INSIDE (XP,NZZ)
      DIMENSION XP(1,3)
C * THIS SUBROUTINE DETERMINES IF A POINT IS INSIDE WINDOW
      COMMON/WINDO/FD,BD,XWINDL,XWINDH,YWINDL,YWINDH
      X=XP(1,1)
      Y=XP(1,2)
      IA1=0
      IF(X.GT.XWINDH)IA1=1
      IA2=0
      IF(X.LT.XWINDL)IA2=1
      IA3=0
      IF(Y.GT.YWINDH)IA3=1
      IA4=0
      IF(Y.LT.YWINDL)IA4=1
      NZZ=MAX0(IA1,IA2,IA3,IA4)
      RETURN
      END
      SUBROUTINE MIDPNT (ZP1,ZP2)
C * THIS SUBROUTINE FINDS THE MIDPOINT OF LINE SEGMENT
C * ZP1 —ZP2 AND RETURNS IT AS ZP2
      DIMENSION ZP1(1,3),ZP2(1,3)
      ZP2(1,1)=(ZP1(1,1)+ZP2(1,1))/2
      ZP2(1,2)=(ZP1(1,2)+ZP2(1,2))/2
      RETURN
      END
      SUBROUTINE INTSEC(TPA,TPB)
C * THIS SUBROUTINE FINDS A POINT TPA SUCH THAT TPA—TPB
C * IS ENTIRELY INSIDE WINDOW DIMENSIONS...TPB MUST BE
C * INSIDE WINDOW
      DIMENSION TPA(1,3),TPB(1,3),TMIDL(1,3),TMIDH(1,3),TMID(1,3)
      CALL EQUAL(TMIDH,TPA)
      CALL EQUAL(TMID,TPB)
      CALL EQUAL (TMIDL,TPB)
      CALL MIDPNT(TMIDL,TMID)
      CALL INSIDE(TMID,N)
10    CONTINUE
      IF(N.EQ.0)GO TO 20
      CALL EQUAL(TMIDH,TMID)
*
```

```
$
      CALL EQUAL(TMID,TMIDL)
      CALL MIDPNT(TMIDH,TMID)
      GO TO 30
20    CONTINUE
      CALL EQUAL(TMIDL,TMID)
      CALL EQUAL(TMID,TMIDH)
      CALL MIDPNT(TMIDL,TMID)
30    CONTINUE
      CALL INSIDE(TMID,N)
      A=ABS(TMIDH(1,1)-TMID(1,1))
      B=ABS(TMIDH(1,2)-TMID(1,2))
      IF(A.GT.0.0001.OR.B.GT.0.0001)GO TO 10
      CALL EQUAL(TPA,TMID)
      RETURN
      END
      SUBROUTINE EQUAL(XX1,XX2)
      DIMENSION XX1(1,3),XX2(1,3)
C * POINT XX1 GETS POINT XX2
      XX1(1,1)=XX2(1,1)
      XX1(1,2)=XX2(1,2)
*     XX1(1,3)=XX2(1,3)
      RETURN
      END
      SUBROUTINE BEZIER(J,N,U,B)
      DIMENSION FACLOG(100)
      DATA(FACLOG(I),I=1,10)/0.,.6931472,1.791759,3.178054,
     1 4.787492,6.579251,8.525161,10.60460,12.80183,15.10441/
      DATA(FACLOG(I),I=11,20)/17.50231,19.98721,22.55216,25.19122,
     1 27.89927,30.67186,33.50507,36.39545,39.33988,42.33562/
      DATA(FACLOG(I),I=21,30)/45.38014,48.47118,51.60668,54.78473,
     1 58.00361,61.26170,64.55754,67.88974,71.25704,74.65824/
      DATA(FACLOG(I),I=31,40)/78.09223,81.55796,85.05447,88.58083,
     1 92.13618,95.71970,99.33062,102.9682,106.6318,110.3208/
      DATA(FACLOG(I),I=41,50)/114.0342,117.7719,121.5331,125.3173,
     1 129.1239,132.9526,136.8027,140.6739,144.5657,148.4778/
      DATA(FACLOG(I),I=51,60)/152.4096,156.3608,160.3311,164.3201,
     1 168.3274,172.3528,176.3959,180.4563,184.5338,188.6282/
      DATA(FACLOG(I),I=61,70)/192.7391,196.8662,201.0093,205.1682,
     1 209.3426,213.5322,217.7369,221.9564,226.1906,230.4390/
      DATA(FACLOG(I),I=71,80)/234.7017,238.9784,243.2689,247.5729,
     1 251.8904,256.2211,260.5649,264.9217,269.2911,273.6731/
      DATA(FACLOG(I),I=81,90)/278.0676,282.4743,286.8931,291.3240,
     1 295.7666,300.2210,304.6869,309.1642,313.6528,318.1526/
      DATA(FACLOG(I),I=91,100)/322.6635,327.1853,331.7179,336.2612,
     1 340.8151,345.3794,349.9541,354.5391,359.1342,363.7394/
      IF(U.EQ.0..AND.J.EQ.0.)B=1
      IF(U.EQ.0..AND.J.NE.0.)B=0.
      IF(U.EQ.0)GO TO 50
      B=EXP((FACLOG(N)-(FACLOG(J)+FACLOG(N-J)))+(J*ALOG(U))+(N-J)*
     1 ALOG(1-U))
*
```

```
C
50      CONTINUE
        RETURN
        END
        SUBROUTINE GRID
C * THIS SUBROUTINE FILLS A FORTRAN DATA FILE WITH THE BEZIER GRID
        WRITE(5,50)
50      FORMAT('     WHAT ARE THE DIMENSIONS OF YOUR BEZIER GRID?')
        WRITE(5,52)
52      FORMAT('     INPUT N+1,M+1')
        READ(5,54)N,M
54      FORMAT(G,G)
        OPEN(UNIT=04,DEVICE='DSK')
        WRITE(04,56)N
        WRITE(04,56)M
56      FORMAT(G)
        WRITE(5,58)N,M
58      FORMAT('    INPUT YOUR 'G' BY 'G' MATRIX ROW BY ROW ENTERING
        1 X,Y,Z FOR EACH POINT')
        DO 100 I=1,N
        DO 200 J=1,M
        READ(5,150)X,Y,Z
150     FORMAT(G,G,G)
        WRITE (04,150)X,Y,Z
200     CONTINUE
100     CONTINUE
        RETURN
        END
        SUBROUTINE SURFAC
C * THIS SUBROUTINE GENERATES THE DATA TO DRAW A BEZIER SURFACE
C * IF MORE THAN 30 X 30 GRID POINTS ARE DESIRED CHANGE DIMENSIONS
        DIMENSION P(15,15,3),XP1(1,3),XP2(1,3),POINTS(45,45,3)
        OPEN(UNIT=04,DEVICE='DSK')
        READ(04,10)N
        READ(04,10)M
10      FORMAT(G)
        DO 30 I=1,N
        DO 20 J=1,M
40      FORMAT(G,G,G)
        READ(04,40)POINTS(I,J,1),POINTS(I,J,2),POINTS(I,J,3)
20      CONTINUE
30      CONTINUE
        DO 70 IU=1,15
        U=IU/15.
        DO 80 IV=1,15
        V=IV/15.
        DO 50 I=1,N
        DO 60 J=1,M
        CALL BEZIER(I-1,N-1,U,B1)
        CALL BEZIER(J-1,M-1,V,B2)
        P1=B1*B2
```

```
$
      P(IU,IV,1)=P(IU,IV,1)+POINTS(I,J,1)*P1
      P(IU,IV,2)=P(IU,IV,2)+POINTS(I,J,2)*P1
      P(IU,IV,3)=P(IU,IV,3)+POINTS(I,J,3)*P1
60    CONTINUE
50    CONTINUE
80    CONTINUE
70    CONTINUE
      DO 100 IU=1,15
      DO 200 IV=1,14
      XP1(1,1)=P(IU,IV,1)
      XP1(1,2)=P(IU,IV,2)
      XP1(1,3)=P(IU,IV,3)
      XP2(1,1)=P(IU,IV+1,1)
      XP2(1,2)=P(IU,IV+1,2)
      XP2(1,3)=P(IU,IV+1,3)
      CALL LINE(XP1,XP2)
200   CONTINUE
100   CONTINUE
      DO 300 IV=1,15
      DO 400 IU=1,14
      XP1(1,1)=P(IU,IV,1)
      XP1(1,2)=P(IU,IV,2)
      XP1(1,3)=P(IU,IV,3)
      XP2(1,1)=P(IU+1,IV,1)
      XP2(1,2)=P(IU+1,IV,2)
      XP2(1,3)=P(IU+1,IV,3)
      CALL LINE(XP1,XP2)
400   CONTINUE
300   CONTINUE
      RETURN
      END
      SUBROUTINE CUBE(I)
      DIMENSION XP1(1,3),XP2(1,3)
      XP1(1,1)=0
      XP1(1,2)=0
      XP1(1,3)=0
      XP2(1,1)=0
      XP2(1,2)=0
      XP2(1,3)=I
      CALL LINE(XP1,XP2)
      XP1(1,1)=0
      XP1(1,2)=0
      XP1(1,3)=I
      XP2(1,1)=I
      XP2(1,2)=0
      XP2(1,3)=I
      CALL LINE(XP1,XP2)
      XP1(1,1)=I
      XP1(1,2)=0
      XP1(1,3)=I
*
```

```
$
      XP2(1,1)=I
      XP2(1,2)=0
      XP2(1,3)=0
      CALL LINE(XP1,XP2)
      XP1(1,1)=I
      XP1(1,2)=0
      XP1(1,3)=0
      XP2(1,1)=0
      XP2(1,2)=0
      XP2(1,3)=0
      CALL LINE(XP1,XP2)
      XP1(1,1)=0
      XP1(1,2)=I
      XP1(1,3)=0
      XP2(1,1)=0
      XP2(1,2)=0
      XP2(1,3)=0
      CALL LINE(XP1,XP2)
      XP1(1,1)=0
      XP1(1,2)=I
      XP1(1,3)=0
      XP2(1,1)=0
      XP2(1,2)=I
      XP2(1,3)=I
      CALL LINE(XP1,XP2)
      XP1(1,1)=0
      XP1(1,2)=I
      XP1(1,3)=I
      XP2(1,1)=I
      XP2(1,2)=I
      XP2(1,3)=I
      CALL LINE(XP1,XP2)
      XP1(1,1)=I
      XP1(1,2)=I
      XP1(1,3)=0
      XP2(1,1)=I
      XP2(1,2)=I
      XP2(1,3)=I
      CALL LINE(XP1,XP2)
      XP1(1,1)=I
      XP1(1,2)=I
      XP1(1,3)=0
      XP2(1,1)=0
      XP2(1,2)=I
      XP2(1,3)=0
      CALL LINE(XP1,XP2)
      XP1(1,1)=I
      XP1(1,2)=I
      XP1(1,3)=0
      XP2(1,1)=I
```

```
$
        XP2(1,2)=0
        XP2(1,3)=0
        CALL LINE(XP1,XP2)
        XP1(1,1)=I
        XP1(1,2)=I
        XP1(1,3)=I
        XP2(1,1)=I
        XP2(1,2)=0
        XP2(1,3)=I
        CALL LINE(XP1,XP2)
        XP1(1,1)=0
        XP1(1,2)=I
        XP1(1,3)=I
        XP2(1,1)=0
        XP2(1,2)=0
        XP2(1,3)=I
        CALL LINE(XP1,XP2)
        RETURN
        END
@*
```