REPORT T-3 MARCH, 1972 UILU-ENG 72-2212

CSL COORDINATED SCIENCE LABORATORY

A MANUAL FOR COMPUTER GRAPHICS AND ANIMATION USING THE CSL LINE AND HALF-TONE GRAPHIC SYSTEMS

WILLIAM DOLSON
ROBERT DURRENBERGER

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

A MANUAL FOR COMPUTER GRAPHICS AND ANIMATION

USING THE CSL LINE AND HALF-TONE GRAPHICS SYSTEMS

By

William Dolson

and

Robert Durrenberger

FOREWARD

This Manual is intended as a users guide for the CSL Perspective System in a line or half-tone mode. This system was designed for use on CSL's CDC 1604 and companion CRT display scope. It is assumed that the user has a thorough knowledge of CSL Fortran and is familiar with the 1604.

The original Perspective Program was developed by Professor Ronald Resch, who, while working at CSL was also a member of the faculty of the Department of Architecture at the University of Illinois. The grey scale and movie systems were later developed and made compatible with the Perspective Program by Jack Bouknight and Karl Kelley at CSL.

This manual was produced as part of a CS 290 special course project in cooperation with CSL under the direction of Dr. Robert Chien, of CSL and Karl Kelley, of CAC.

We would like to express our thanks to Dr. C. L. Coates, Dr. Chien, and Mr. W. C. Prothe, who made the writing of this manual possible. We gratefully acknowledge the assistance of Karl Kelley and Jack Bouknight, without whose aid this manual would have proved an impossible task. We also wish to thank Jack Gladin for his help in taming photographic recording problems and devising the printing process for color movies. Our special thanks go to Mrs. Carol Martin for her excellent typing of the manual.

The authors assume sole responsibility for any errors in the text.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER 1. THE CSL PERSPECTIVE SYSTEM AND LINE GRAPHICS

1.1.0 System Concepts

The Perspective System is a FORTRAN system which draws views of objects inputted by the user. The input consists of two distinct parts, initial definition of the objects to be drawn, and manipulations of the objects preparatory to the graphical output. The system will output object veiws in a line or grey-scale mode. This chapter is an explanation of system operation in a line-drawing mode.

1.1.1 Object Definition

Objects are defined in an orthagonal three-space coordinate system by reference to a fixed origin, (0,0,0). Any object to be inputted to the system consists of points, whose three-space coordinates are inputted to the system, and line segments, whose endpoints must be previously defined as points.

To define a 2 x 2 unit square in the x-y plane at z = 2, we must

Define Points

```
Point 1,  x-coordinate, 0., y-coordinate, 0., z-coordiante, 2.
Point 2,  x-coordinate, 0., y-coordinate, 2., z-coordinate, 2.
Point 3,  x-coordinate, 2., y-coordinate, 2., z-coordinate, 2.
Point 4,  x-coordinate, 2., y-coordinate, 0., z-coordinate, 2.
```

and Define Lines

```
Line 1,  from Point 1, to Point 2
Line 2,  from Point 2, to Point 3
Line 3,  from Point 3, to Point 4
Line 4,  from Point 4, to Point 1
```

In a line drawing mode, only line segments are displayed, points are not represented in any way.

In the Perspective System the function of defining points and lines is performed by a set of FORTRAN subroutines. The exact form of the calls to these routines is discussed in Section 1.2.1.

In order to facilitate handling a large number of objects, a collection of points and lines is considered as a "subfigure". For object manipulations it is much easier to be able to specify a group of points and lines to be manipulated rather than manipulating each point and line individually. In the system, points are always defined as being in a subfigure, and lines are defined in a subfigure with reference to points in that subfigure.

Our specification for the square must now read:

```
Define points

     in Subfigure 1, point 1, at x, 0.,y, 0.,z, 2.
     in Subfigure 1, point 2, at x, 0.,y, 2.,z, 2.
     in Subfigure 1, point 3, at x, 2.,y, 2.,z, 2.
     in Subfigure 1, point 4, at x, 2.,y, 0.,z, 2.

Define lines

     in Subfigure 1, line 1, from point 1, to point 2
     in Subfigure 1, line 2, from point 2, to point 3
     in Subfigure 1, line 3, from point 3, to point 4
     in Subfigure 1, line 4, from point 4, to point 1
```

Lines may not be defined with reference to points not in the same subfigure.

1.1.2 Object Manipulation

In addition to defining objects we may also wish to alter them or move them with respect to each other. The system allows the user a wide variety of manipulations; rotation, translation, scaling and combinations of these. All these manipulations are performed by FORTRAN routines in the system and all these routines operate on subfigures. When called, the manipulation is performed on the entire subfigure.

If we wish to rotate our square 90^o around the z axis we would need a command something like

TURN Subfigure 1 around the z axis by 90^o .

The system uses a right handed rotation convention. The exact FORTRAN call to the system will be described later, but this is the general form for all object manipulations.

1.1.3 Graphical Output

The system can define and manipulate individual objects by the use of subfigures. However, for display purposes we would like to be able to exhibit all or at least several objects at once.. This is accomplished by defining a group of subfigures as a "figure".

What we should do is:

Define a figure.

Figure 1 consisting of subfigure 1 through subfigure N.

This is also done via a FORTRAN call to a system subroutine.

Now that we have defined a "figure" we must output it graphically. All objects are defined in the system in a three space. When we display the figure, we must call a routine which performs a perspective transformation from three-space to the two dimensions of the display medium. The display routine computes perspectives with respect to any given "observer" position in the three space of the figure. However, the "observer" must be "looking" from a point in three-space direction parallel to one of the three coordinate axes in either the positive or negative direction. Once the axis and direction are specified, the "observer" may be rotated about the three axes.

One way we could display our square is to:

MOVE the OBSERVER to x = 0., y = 0., z = 10.

We should then call the output routine which performs the perspective transformation and displays the figure. We would:

DRAW figure 1. looking from the OBSERVER position,
parallel to the Z-AXIS in the NEGATIVE direction.

The graphical output device for the Perspective System is the CRT console.

These beginning sections are a basic explanation of the manner in which the system handles graphics. We will now investigate the explicit operation of the system, and the specific FORTRAN code which is required to drive it.

1.2.0 System Operation

The Perspective System employs several unique methods of code checking and routine accessing. The following sections attempt to explain the operation of the system preparatory to discussing a sample program.

1.2.1 Activating the System and Calls to System Routines

The Perspective System is always driven by a user program. The user program must be preceded by a TITLE program. This TITLE Program duplicates the storage allocations given in the Perspective System to allow the user program to access the same storage locations as the system. It also declares equivalences between elements of an array named INP and certain character strings which we shall call "key words". These "key words" consist of the names of all the user called system routines and several error indication flags. A listing of TITLE equivalences can be found in Appendix I. More about key words later.

The user's program itself contains several calls which activate the Perspective System. The routines must be called in this order:

CALL VOCAB - VOCAB zeroes system storage locations, sets values of system constants and sets system storage limits.

CALL XVOCAB(INP) - initializes storage and tests storage allocation parameters. It sets certain constants which are determined by system parameters. XVOCAB then calls VOCAB1 with the address of array INP. VOCAB1 loads INP with integer indexes. When a key word appears in a system call in the users program, it is replaced by an address in INP by the compiler. The contents of that address is the index number of the routine which is represented by the key word. If the key word is not a routine name, it is used as a dummy for error checking calls to system routines.

In Summary: The TITLE Program equates key words to elements of INP. VOCAB then loads INP with integer indices, so that a key word becomes essentially index number of a routine.

CALL CPB1 - CPB1 initializes a series of routines for generating movies if desired and then calls CCP1PL which establishes and packs a buffer for the CRT display.

These three calls are usually followed by a series of statements which initialize or change system constants and set system flags. These statements are dependent on the type of program being run.

Following these statements is the main body of the program. For one-shot line graphics this usually consists of calls to the Perspective System. These calls are divided into two groups: those which are definitions and those which are manipulations. All calls to the System routines are made via a central interpreter. The routines are not called by name as in a normal FORTRAN call.

As an example, suppose we are to call the subroutine DEFPNT which is a routine that defines points. We would <u>not</u> say:

```
CALL DEFPNT (arguments)
```

Instead, we call all routines via a master routine ING. The correct statement is:

```
CALLING(DEFPNT,arguments,END)
```

This brings us back to "key words". DEFPNT is a key word. It has been equivalenced with an INP address which contains an index pointing to the routine DEFPNT. ING stores the INP index corresponding to the key word DEFPNT in an array IPT. ING then starts reading the arguments following the key word. Usually arguments for the called routine (in this case DEFPNT) are nested between additional dummy key words. These dummy key words are used in code checking the calls and are also stored in IPT. When ING encounters an END in its arguments it calls a routine ING1. All calls to system routines via ING must be terminated by an END.

ING1 is an interpreter. It branches to the appropriate routine on the basis of the INP equivalences stored in IPT.

Following the calls to ING which define and manipulate the points, lines and subfigures; figures are defined in additional calls to system routines via ING.

When one or more figures have been defined, the figures are processed by the perspective routine in a final system call and are displayed on the scope. To explain this final operation it is necessary to examine how the subfigure definitions and manipulations are stored in the system.

1.2.2 System Storage Allocations

The user program generally defines a 3-space structure and then manipulates it before any type of graphic output is made. These functions of definition and manipulation are handled very differently by the System.

All three space coordinates are stored in a large array G. G contains the x,y,z coordinates of all points as they are originally defined. (See Appendix II for details on the structure of G.)

When the user program specifies subfigure manipulations, the system stores this information in another array, TMAT. TMAT conceptually consists of a 4 x 4 transformation matrix for each subfigure. When some manipulation is specified, the system does not operate on all the values in G. It merely stores the information in a concise form in the appropriate transformation matrix.

The figures are displayed by calling a system routine DRAW via ING. DRAW multiplies each x,y,z value for each point in the subfigure to be displayed by the transformation matrix for that subfigure. This is considerably faster than multiplying through G for each subfigure manipulation, especially if there are many such manipulations.

After multiplication by the appropriate matrix in TMAT, points undergo the perspective transformation and the resultant two-space values are packed into the scope buffer. The values in G are updated by multiplication by TMAT, that is, if any object manipulations have been specified, the values in G after a DRAW will not be those inputted by the user, but will be those original values operated on by the manipulation via TMAT.

In addition to G and TMAT, there are several other arrays in the System which contain useful information about the status of points, lines, and subfigures. They are listed in Appendix III.

1.2.3 System Storage Limits

The Perspective System has very definite limits on its storage. This is perhaps its greatest disadvantage. The data structure is rigidly organized as follows:

Maximum Number of Subfigures.	40
Maximum Number of Points per Subfigure. . .	30
Maximum Number of Lines per Subfigure . . .	30
Maximum Number of Figures	40

No dynamic storage allocation is possible. In addition to this, the subfigures and figures must be referenced by the integers 1 through 40, lines and points must be referenced by the integers 1 through 30.

```
@FORTRAN
```

```
      TITLE FIGSYS
      COMMON G(3590),NF(41),NPLN(41),NPLY(1230),LINEQ(2460),NL(41),
     1NPI(200),TMATS(704),IPI(100),IFTAB(100),NSUB,NUMS,NUMTM,NUMP,
     2NUML,IDIM,IDIM1,IDIM2,LDIM,LDIM1,LDIM2,LDIMI,EPS,ATEST,RESULT,
     3MAXS,NSUBS,NOGO,DINCY,VANGLE,XTOY,UP,INGC,ERRPAR,DERAD,RADDE,PI,
     4LBLPTS,LBLLNS,IFNEXT,JDIM,JDIM1,JDIM2,NUMPLY,SHADES(240),
     5NTONE,HTDINCY,DARK,BRIGHT,RASTRSIZ,LSMATRX,LSCOORS(3)
     6,NOSTOP
      INTEGER RASTRSIZ
      DIMENSION INP(100)
      COMMON INP
      EQUIVALENCE (INP(1),ALL),(INP(2),XC),(INP(2),XCOORD),(INP(3),YC),
     1 (INP(3),YCOORD),(INP(4),ZC),(INP(4),ZCOORD),(INP(5),XA),
     2 (INP(5),XAXIS),(INP(5),YZPL),(INP(6),YA),(INP(6),YAXIS),
     3 (INP(6),ZXPL),(INP(7),ZA),(INP(7),ZAXIS),(INP(7),XYPL)
      EQUIVALENCE (INP(8),POS),(INP(9),NEG),(INP(10),ALLCRD),
     1 (INP(10),AC),(INP(11),FOCAL),(INP(12),POINT),(INP(12),POINTS),
     2 (INP(13),LINE),(INP(13),LINES),(INP(14),TMAT),(INP(14),MATRIX)
      EQUIVALENCE (INP(15),BND),(INP(16),ROW),(INP(17),COL) ,
     1 (INP(17),COLUMN),(INP(18),ELMNTS),(INP(19),CONECT),
     2 (INP(20),SUBFLS),(INP(21),SUBFLF),(INP(22),OB),(INP(22),OBSVER)
      EQUIVALENCE (INP(23),OBSMAT),(INP(24),SC),(INP(24),SCREEN),
     1 (INP(25),ALAXIS),(INP(25),ALAX)
      EQUIVALENCE (INP(26),MATMPY),(INP(27),ROTATE),(INP(28),TURN),
     1 (INP(29),READ),(INP(30),WRITE),(INP(31),PUNCHG),(INP(32),COPY),
     2 (INP(33),COPPNT),(INP(34),COPLIN),(INP(35),COPMAT),(INP(36),MOVE)
      EQUIVALENCE (INP(37),MOVETO),(INP(38),ORIGIN),(INP(39),ADD),
     1 (INP(40),STORE),(INP(41),LENGTH),(INP(42),DELETE),
     2 (INP(43),DELPNT),(INP(44),DELLIN),(INP(45),DELMAT)
      EQUIVALENCE (INP(45),DLMAT),(INP(46),MIRROR),(INP(47),FLIP),
     1 (INP(48),DIST),(INP(49),DEFINE),(INP(50),DEFPNT),
     2 (INP(51),DEFLIN),(INP(52),DEFMAT),(INP(53),PERSP)
      EQUIVALENCE (INP(54),TRNSFM),(INP(55),DRAW),(INP(56),SCALE),
     1(INP(57),XCHNGE),(INP(58),EXCHPT),(INP(59),TURNTO),
     2 (INP(60),DEGREE),(INP(61),INTSEC),(INP(62),DEFFIG)
      EQUIVALENCE (INP(63),BLK),(INP(64),TURNOQ),
     1 (INP(65),ZERO),(INP(65),ZRO),(INP(66),TRNDIQ),(INP(67),TRNFLQ),
     2 (INP(68),DELFIG),(INP(69),ADDFIG),(INP(70),FIG),(INP(70),FIGS),
     3 (INP(71),FAN)
      EQUIVALENCE (INP(72),DEFPLN),(INP(73),COPPLN),(INP(74),REFLECT),
     1 (INP(75),DELPLN),(INP(76),PLANES)
      EQUIVALENCE (INP(77),REFCOEF),(INP(77),RCOEF)
      EQUIVALENCE (INP(78),USMAT),(INP(79),ORBIT),(INP(80),LITESORC)
      EQUIVALENCE (INP(81),DRAWTWO)
      REAL LSCOORS
      REAL LSCOOR
      EXTERNAL LSCOOR
      END
@END
```

```
¬=FORTRAN
      PROGRAM SQUARE
      TITLE *
*
*     SQUARE IS A PROGRAM WHICH DEFINES A SQUARE IN THE X-Y PLANE
*     AT Z=2.  THE SQUARE IS TWO BY TWO UNITS.
*
      CALL VOCAB
      CALL XVOCAB(INP)
      CALL OPB1
      UP=YAXIS
*
*     DEFINE THE LINES COMPROMISING THE SQUARE
*
      CALLING(DEFPNT,S(1),P(1),XC,0.,YC,0.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(2),XC,0.,YC,2.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(3),XC,2.,YC,2.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(4),XC,2.,YC,0.,ZC,2.,END)
*
*     DEFINE THE LINES COMPROMISING THE SQUARE
*
      CALLING(DEFLIN,S(1),L(1),P(1),P(2),END)
      CALLING(DEFLIN,S(1),L(2),P(2),P(3),END)
      CALLING(DEFLIN,S(1),L(3),P(3),P(4),END)
      CALLING(DEFLIN,S(1),L(4),P(4),P(1),END)
*
*     NOW DEFINE A FIGURE
*
      CALLING(DEFFIG,S(1),S(2),F(1),END)
*
*     MOVE THE OBSERVER TO A GOOD VANTAGE POINT
*
      CALLING(MOVETO,OB,XC,1.,YC,1.,ZC,5.,END)
*
*     NOW DISPLAY THE SQUARE
*
      CALLING(DRAW,F(1),ZAXIS,OB,NEG,END)
      END
¬=END
¬=ILLAR
         FINIS
¬=END
¬=END
```

1.3.0 A Simple Users Program

The function of FIGSYS has been explained, it is the TITLE program containing storage allocations and INP equivalences.

The PROGRAM statement in the user's program must always be followed by the TITLE* statement so that FIGSYS may be accessible.

Following the comments are the three calls which initiate the Perspective System. These are followed by a system flag; UP = YAXIS. UP indicates which axis is to be chosen as the vertical when the figure is displayed. It can assume three values: XAXIS, YAXIS or ZAXIS. UP is not initialized in the Perspective System and must be specified in the user's program.

We next define the points and lines of subfigure one. These consist of calls to two routines via ING. They are DEFPNT (define point) and DEFLIN (define line). The meaning of these calls is evident by comparison to the defining statements in Section 1.1.1. The arguments for these routines are nested in key words which are interpreted by ING1. The format for these calls to System subroutines may be found in Section 1.5.0.

After the subfigure definition we define a figure by calling routine DEFFIG. This routine defines a figure from subfigure N through subfigure M, where $N \neq M$. Since we have only defined one subfigure, we must use a dummy subfigure number for M. Since all storage is initialized to zero, subfigure 2 contains no points or lines and there is nothing to display. We could just as easily have used any subfigure from 2 to 40 for the terminal M, only subfigure 1 would be displayed because it is the only subfigure which contains any defined lines.

We next MOVE the observer. The routine DRAW displays the output. The observer is looking through a viewing angle of VANGLE degrees along any axis in the $\pm \infty$ direction. VANGLE is initiated to 45°, and with this value we move the observer to a point so that the square fills the field of view when the figure is drawn with the observer looking along the ZAXIS in the negative direction.

We then DRAW the figure. This will cause the square to be traced once on the CRT display.

Again we emphasize that the calls to ING are in a very strict format which must be observed exactly.

1.3.1 Running Line Graphics Programs

Running Perspective System programs is no different from running any other program on the ILLAR system. To run this program on the CDC 1604, we punch it on paper tape, copy it on to magnetic tape and then compile it on magnetic tape in the ILLAR system.

The compiled version is mounted on logical tape 4, a scratch tape is mounted on logical 6, and a Proglib version of the Perspective System is mounted on logical 2[†]. All tapes should be file protected except 6.

In a resident mode type

```
EXITCALL,4,2,name
```

[†] A Proglib version is available on tape D-38.

The name is, of course, your program name. The tapes on 4 and 2 will be loaded. To run your program type

```
GO,name
```

or just

```
name
```

Your program should run. If it does not, error messages will be written on tape 6. Copy these to the printer to obtain a listing and then consult the debugging section of this manual.

1.4.0 ACTIONS

With the graphics program we have presented thus far, the output is very limited. No user interaction is possible.

Jack Bouknight has written an interactive subroutine for the system which calls draw, establishes a looping buffer, and then allows the user to manipulate the observer by typing characters on the console keyboard. This routine is called ACTIONS and is included on the Perspective System tape D-38.

In addition to allowing observer manipulations, ACTIONS contains three arguments which allow the user to transfer control to three points in his program.

ACTIONS also permits other services such as the taking of polarid photographs, and moving the light source for grey-scale graphics.

Jack's write-up of ACTIONS follows.

Operating Instructions for ACTIONS

Calling Sequence

CALL ACTIONS (NFIG, /#1, /#2, /#3, TAXIS, TDIR)

NFIG	-	figure # to be displayed, can be changed by ACTIONS
/#1 2 3	-	statement numbers of user code to be executed from ACTIONS
TAXIS	-	axis of view for calls to DRAW in ACTIONS
TDIR	-	direction of view on TAXIS
CALL REACTION	-	allows return from user operations to resume original control.

Description of Typewriter Operation

Operations are initiated by typing sequences of characters on the console typewriter. A sequence will contain a primary character and possibly a chain of sub-characters. Typing rates are determined by the time required to generate the display by a call to DRAW.

TYPE L	-	conditions the system to move the light source in the current direction
TYPE B	-	conditions the system to move the observer in the current direction
TYPE O	-	conditions the system for orbitting the light source about the current axis
TYPE P	-	conditions the system for panning the observer about the current axis

All four primary characters L, B, O, P, stop motion. Motion begins when - or + are typed.

TYPE +	-	starts motion in the current mode in a positive sense
TYPE -	-	starts motion in the current mode in a negative sense
TYPE (space)	-	causes one move in the current mode of motion unless motion is stopped
TYPE Q	-	queries system for type-out of current parameters of motion

Picture taking is accomplished in three ways.

TYPE .	-	takes Polaroid photo of line drawing
TYPE T	-	outputs data on tape 8 for TONE processing of current image
TYPE SP	-	primary S and subcharacter P cause stereo pair generation. Will request eye separation. Will first present left eye view.

Type . or T

Will then present right eye view.

Type . or T

The remainder of the primary characters are utility operations.

TYPE F	-	change the figure number. Will type NFIG = Respond in XX/format.
TYPE IO	-	primary I, subcharacter O; changes translation increment for observer motion. Will type OB INCR =. Respond in F12.6 format.
TYPE IL	-	primary I, subcharacter L; same as IO except for light source (LS)
TYPE AO	-	primary A, subcharacter O; changes rotation angle for panning observer. Will type OB ANGLE =. Respond in F12.6 format.
TYPE AL	-	primary A, subcharacter L; same as AO except for light source.

```
TYPE X     -    sets X axis and X-direction as current axis and
                current direction for motions.

TYPE Y     -    same as X except Y axis and Y-direction

TYPE Z     -    same as X except Z axis and Z-direction.

TYPE K     -    kill operations and return to program which
                called ACTIONS

TYPE U1    -    initiate user sequence 1
      2                                2
      3                                3

TYPE D     -    define new figure.  Will type list s = ; respond
                in XX/ format.  Will type 2nd s = ; respond in
                XX/ format.  Will type nfig = ; respond in XX/
                format.  Current figure will be set to NFIG.

TYPE
 SCDWX     -    scaling operations

           -    D specifies axis of scale or A for all 3 axes.

           -    W specifies scaling of current figure (F)
                or light source (L)

           -    X specifies scale expansion (U) or contraction (D)

TYPE CS    -    changes scale factor.  Scale factor is

                     1. + SF                   (U)
                     1./(1. + SF)              (D)
                     Will type SCALE FACTOR =
                     Respond with F12.6 format.

TYPE MO    -    move the observer to point XX, YY, ZZ.  Will
                type X, Y, Z =
                Respond with F12.6          (XX)
                             F12.6          (YY)
                             F12.6          (ZZ)

TYPE ML    -    same as MO except for light source.

TYPE SQ    -    conditions system to record each movement as
                frame of movie sequence.

TYPE ES    -    ends recording of movie sequence.
```

TYPE MT	-	start movie sequence.
TYPE MF	-	set up frame count - will ask frame count - respond in XX/ format. Frame count set to 1 when MT typed.
TYPE MS	-	cause frames to be skipped on movie - will ask skip ct. - respond in XX/ format.
TYPE MD	-	complete movie data tape.
TYPE RS	-	change RASTRSIZ - will ask RASTRSIZ = ; respond in XX/format.

Illegal characters will be signalled by a return of/on the typewriter.

TYPE SW	-	sets movie frame window
TYPE RW	-	removes movie frame window
TYPE SL	-	label all pictures. Returns message LABEL, type in up to ten (10) words of BCD message.
TYPE CL	-	clear picture labels

1.4.1 A Sample Program with Actions

We now present the sample program with the calls to DRAW replaced by calls to ACTIONS, with appropriate entry points.

This program is run in the same manner as the other. However, the user may now interact with his creation.

By typing "U1" we will change the scale factor by .5. The scale factor is initialized to 1., so the figure will decrease in size by 1/2 each time we type U1.

If we type "U2" the figure will be rotated about the Z axis by 90 degrees.

If we type "U3" the program will exit.

```
@FORTRAN
      TITLE FIGSYS
      COMMON G(3690),NF(41),NPLN(41),NPLY(1230),LINEQ(2460),NL(41),
     1NPT(200),TMATQ(704),IFT(100),IFTAB(100),NSUB,NUMS,NUMTM,NOMP,
     2NUML,IDIM,IDIM1,IDIM2,LDIM,LDIM1,LDIM2,LDIMI,EPS,ATEST,RESULT,
     3MAXS,NSUBS,NOGO,DINCY,VANGLE,XTOY,UP,INGC,ERRPAR,DERAD,RADDE,PI,
     4LBLPTS,LBLLNS,IFNEXT,JDIM,JDIM1,JDIM2,NUMPLY,SHADES(240),
     5NTONE,HTDINCY,DARK,BRIGHT,RASTRSIZ,LSMATRX,LSCOORS(3)
     6,NOSTOP
      INTEGER RASTRSIZ
      DIMENSION INP(100)
      COMMON INP
      EQUIVALENCE (INP(1),ALL),(INP(2),XC),(INP(2),XCOORD),(INP(3),YC),
     1 (INP(3),YCOORD),(INP(4),ZC),(INP(4),ZCOORD),(INP(5),XA),
     2 (INP(5),XAXIS),(INP(5),YZPL),(INP(6),YA),(INP(6),YAXIS),
     3 (INP(6),ZXPL),(INP(7),ZA),(INP(7),ZAXIS),(INP(7),XYPL)
      EQUIVALENCE (INP(8),POS),(INP(9),NEG),(INP(10),ALLCRD),
     1 (INP(10),AC),(INP(11),FOCAL),(INP(12),POINT),(INP(12),POINTS),
     2 (INP(13),LINE),(INP(13),LINES),(INP(14),TMAT),(INP(14),MATRIX)
      EQUIVALENCE (INP(15),END),(INP(16),ROW),(INP(17),COL),
     1 (INP(17),COLUMN),(INP(18),ELMNTS),(INP(19),CONECT),
     2 (INP(20),SUBFLS),(INP(21),SUBFLF),(INP(22),OB),(INP(22),OBSVER)
      EQUIVALENCE (INP(23),OBSMAT),(INP(24),SC),(INP(24),SCREEN),
     1 (INP(25),ALAXIS),(INP(25),ALAX)
      EQUIVALENCE (INP(26),MATMPY),(INP(27),ROTATE),(INP(28),TURN),
     1 (INP(29),READ),(INP(30),WRITE),(INP(31),PUNCHQ),(INP(32),COPY),
     2 (INP(33),COPPNT),(INP(34),COPLIN),(INP(35),COPMAT),(INP(36),MOVE)
      EQUIVALENCE (INP(37),MOVETO),(INP(38),ORIGIN),(INP(39),ADD),
     1 (INP(40),STORE),(INP(41),LENGTH),(INP(42),DELETE),
     2 (INP(43),DELPNT),(INP(44),DELLIN),(INP(45),DELMAT)
      EQUIVALENCE (INP(45),DLMAT),(INP(46),MIRROR),(INP(47),FLIP),
     1 (INP(48),DIST),(INP(49),DEFINE),(INP(50),DEFPNT),
     2 (INP(51),DEFLIN),(INP(52),DEFMAT),(INP(53),PERSP)
      EQUIVALENCE (INP(54),TRNSFM),(INP(55),DRAW),(INP(56),SCALE),
     1(INP(57),XCHNGE),(INP(58),EXCHPT),(INP(59),TURNTO),
     2 (INP(60),DEGREE),(INP(61),INTSEC),(INP(62),DEFFIG)
      EQUIVALENCE (INP(63),BLK),(INP(64),TURNOQ),
     1 (INP(65),ZERO),(INP(65),ZRO),(INP(66),TRNDIQ),(INP(67),TRNFLQ),
     2 (INP(68),DELFIG),(INP(69),ADDFIG),(INP(70),FIG),(INP(70),FIGS),
     3 (INP(71),PAN)
      EQUIVALENCE (INP(72),DEFPLN),(INP(73),COPPLN),(INP(74),REFLECT),
     1 (INP(75),DELPLN),(INP(76),PLANES)
      EQUIVALENCE (INP(77),REFCOEF),(INP(77),RCOEF)
      EQUIVALENCE (INP(78),LSMAT),(INP(79),ORBIT),(INP(80),LITESORC)
      EQUIVALENCE (INP(81),DRAWTWO)
      REAL LSCOORS
      REAL LSCOOR
      EXTERNAL LSCOOR
      END
@END
```

```
*=FORTRAN
      PROGRAM SQUARE
      TITLE *
*
*     SQUARE IS A PROGRAM WHICH DEFINES A SQUARE IN THE X-Y PLANE
*     AT Z=2.  THE SQUARE IS TWO BY TWO UNITS.
*
      CALL VOCAB
      CALL XVOCAB(INP)
      CALL UP81
      UP=YAXIS
*
*     DEFINE THE LINES COMPROMISING THE SQUARE
*
      CALLING(DEFPNT,S(1),P(1),XC,0.,YC,0.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(2),XC,0.,YC,2.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(3),XC,2.,YC,2.,ZC,2.,END)
      CALLING(DEFPNT,S(1),P(4),XC,2.,YC,0.,ZC,2.,END)
*
*     DEFINE THE LINES COMPROMISING THE SQUARE
*
      CALLING(DEFLIN,S(1),L(1),P(1),P(2),END)
      CALLING(DEFLIN,S(1),L(2),P(2),P(3),END)
      CALLING(DEFLIN,S(1),L(3),P(3),P(4),END)
      CALLING(DEFLIN,S(1),L(4),P(4),P(1),END)
*
*     NOW DEFINE A FIGURE
*
      CALLING(DEFFIG,S(1),S(2),F(1),END)
*
*     MOVE THE OBSERVER TO A GOOD VANTAGE POINT
*
      CALLING(MOVETO,OB,XC,1.,YC,1.,ZC,5.,END)
*
*     NOW WE DISPLAY THE FIGURE VIA ACTIONS AND INCLUDE THREE MANIPULATIONS
*     OF THE SQUARE AS ACTIONS ENTRY POINTS.
*
      CALL ACTIONS(1,/701,/702,/703,ZAXIS,NEQ)
      RETURN
 701  CALLING(SCALE,S(1),XC,.5,YC,.5,END)
      CALL REACTION
 702  CALLING(TURN,S(1),ZAXIS,90.,END)
      CALL REACTION
 703  CALLING(TURN,S(1),ZAXIS,-90.,END)
      CALL REACTION
      END
*=END
*=ILLAR
        FINIS
*=END
*=END
```

We may also manipulate the figure with the ACTIONS commands. If we type "bz+" the observer will be moved in the +z direction, a distance equal to the current motion increment. The motion increment is initialized to 10. units. If we type "px+" the observer will be rotated around the XAXIS by the current angle increment. The angle increment is initialized to 10.0 degrees. Actions also allows us to change these increments.

It is well worth becoming proficient in ACTIONS as it is an excellent means of checking complex structures. With ACTIONS one can simply move around and in the structure, examining it from any desired position.

1.5.0 Perspective System Routines and Functions

The Perspective System contains over fifty subroutines which perform object definition and manipulation. This section compromises a complete listing of all user called routines and functions and a description of their use. In all cases the strictest conformity with the calls to ING as given should be observed.

It will be noticed that the order of the arguments in certain calls to ING differs from the order of arguments in the routine name. This is due to the fact that the interpreter, ING1, does special sorting on the arguments and "key words" before it calls the desired routine. Whenever routines are called via ING, the call must be identical to the "CALLING" statement given with each routine in this glossary.

Several routines may be used to manipulate subfigures or the observer. When this is the case, the subfigure argument, S(N), is replaced by the observer argument, OB.

TABLE 1

Perspective System Subroutines

ADD
CALCOMP
CHLINE
COPLIN
COPMAT
COPPLN
COPPNT
COPY
DEFLIN
DEFMAT
DEFPLN
DEFPNT
DEGREE
DELETE
DELLIN
DELPLN
DELPNT
DIST
DRAW
EXCHPT
FLIP
GERROR
IDMAT
ING1
INTSEC
LENGTH
MATMPY
MIRROR
MOVE
MOVETO
MOVPARAM
MOVTOB
PERSP
PLNPAR
PRCFIG
PROJEC
PUNCHE
READ
ROBAOG
ROTATE
SCALE
SHADER
TRNDIS
TRNPLN
TRANSFM
TRNWHI
TURN
TURNOF
TURNTO
UNPACK
VOCAB
WRITE
XROTAT
XTURNOF
XVOCAB

Subroutine ADD(KF1, KS1)

Adds points and lines of subfigure KF1 to subfigure KS1, deletes KF1.

If KF1 is not defined, nothing happens.
If KS1 is not an element of any figure KF1 is copied into KS1 and KF1 is deleted. KS1 is now in the same figure as KF1.

If KS1 is in a figure different from KF1, KF1 is copied into KS1 and KF1 is deleted. KS1 remains in the same figure it occupied before the add.

USER CALL: CALLING(ADD,S(KF1),S(KS1),END)

CALLS:[†] COPY, DELETE, GERROR, TRNSFM, CHLINE

Subroutine CALCMP(X,Y,ALPHA)

Assembly language programs to drive the movie system. Entered through call to CPB1 produces a movie data tape (MDT) or packs the scope buffer and produces the display.

MS... routines are for line drawings, MSV commands for half-tone. SAVESYS allows the system to be restarted at a given frame.

The use of CALCOMP compromises the chapters on line movies and grey-scale movies.

NOT USER CALLED:

by the name CALCOMP, this routine does, however, contain all of the entry points for the tone and line movie calls. Its use is described in Chapter 3.

NO CALLS

[†]All subroutine listings in this section contain a series of "CALLS". These are the external symbols in that routine.

Subroutine CHLINE(KFX,KF1)

The lines of KF1 are added to the lines in KF

If KF1 is Zero-the lines of KF are updated

USER CALL: this routine is not accessed by ING, it may however be useful, so is included in this listing. It is called like a normal FORTRAN routine

CALLS: GERROR

Subroutine CHROMA(ntone,ncolor)

CHROMA monitors line deletions or plane reflection coefficient assignment for color movies. Its use is described in Section 4.2.0.

USER CALL: CALL CHROMA(ntone,ncolor)

CALLS: DELLIN, SHADER

(This is not a part of the original system. It was added by Bill Dolson)

Subroutine COPLIN(KF,KF1)

Copies lines of KF into KF1

USER CALL: CALLING(COPLIN,S(KF),S(KF1),END)

CALLS: GERROR

Subroutine COPMAT(KF,KF1)

Copies TMAT for KF into TMAT of KF1

USER CALL: CALLING(COPMAT,S(KF),S(KF1),END)

CALLS: GERROR

Subroutine COPPLN(KF,KF1)

Copies polygons of subfigure KF into subfigure KF1

USER CALL: CALLING(COPPLN,S(KF),S(KF1),END)

CALLS: GERROR COPLIN

Subroutine COPPNT(KF,KF1)

Copies points of KF into KF1

USER CALLS: CALLING(COPPNT,S(KF),SKF1),END)

CALLS: GERROR

Subroutine COPY(KF,KF1)

Copies points lines and TMAT(transformation matrix) of KF into KF1

USER CALL: CALLING(COPY,S(KF),S(KF1),END)

CALLS: COPPNT COPMAT COPLIN COPPLN

Subroutine DEFBLK(KF,KF1)

This subroutine defines a block as a set of subfigures KF through KF1. If "BLOCK" replaces the argument S(X), in any call to ING, the subfigure specified in DEFBLK will all be processed in one call to ING

USER CALL: CALLING(BLK,S(KF),(SKF1),END)

NO CALLS

Subroutine DEFFIG(KF,KS,IF)

Defines figure IF as either subfigures KF through KS or subfigure KF and KS, depending on the ING calls

USER CALL: for inclusive subfigures KF through KS: CALLING(DEFFIG,S(DK),S(KS),F(IF),END),for just KF and KS:

CALLING(DEFFIG,SUBFLS,S(KF),S(KS),F(IF),SUBFLF,END)

Subroutine DEFLIN(KX,LN,IP,IP1)

Defines points IP and IP1 as the endpoints of line LN in subfigure KX

If LN is ZERO - the next available line number in that subfigure is used

USER CALL: CALLING(DEFLIN,S(KX),L(LN),P(IP),P(IP1),END)

CALLS: GERROR

Subroutine DEFMAT(KF,IROW,ICOLM,T1, T 2,...T16)

Define transformation matrix for subfigure KF
IF IROW and ICOLM are 0;T1,...,T16 give all elements
if IROW is 0;T1,...,T4 give elements of column (ICOLM)
if ICOLM is 0; T1,...,T4 give elements of row (IROW)
if neither is 0; T1 gives element (ICOLM,IROW)

USER CALL: CALL DIRECTLY, CANNOT BE ACCESSED VIA ING

CALLS: GERROR

Subroutine DEGREE(KF,IP,JA)

determines the angle, in degrees, of the vector from the origin to point IP, with that vector projected onto plane JA

JA = 1	YZ plane	(x = 0 plane)
JA = 2	ZX plane	(y = 0 plane)
JA = 3	ZX plane	(z = 0 plane)

the answer in degrees, is put into the global location "RESULT"

USER CALL: CALLING(DEGREE,S(KF),P(IP),JAXIS,END)

CALLS: GERROR

Subroutine DELETE(KF) CALLING(DELETE,S(N),END)

Deletes subfigure KF - by deleting points, lines and TMAT (transformation matrix) of that subfigure

USER CALL: CALLING(DELETE,S(KF),END)

CALLS: GERROR IDMAT

Subroutine DELLIN(KF,LN)

Deletes line LN of subfigure KF. If LN = 0 deletes all lines in KF.

USER CALL: CALLING(DELLIN,S(KF),L(LN),END) if you wish to delete the entire subfigure, the routine must be called directly as CALL DELLIN(KF,Ø)

CALLS: GERROR

Subroutine DEFPLN(KFX,DPLNX,P1,P2,P3,P4,RECOEF)

Defines a plane given P1 through P4 (points) or P1 through P3 in subfigure KFX. Planes may be triangular or quadrilateral.
RCOEF is the reflection coefficient for the plane, if no RCOEF is specified, then it is assumed to be 1.0.

USER CALL: CALLING(DEFPLN,S(KFX,PLANE(KPLNX,P(P1),P(P2),P(P3),P(P4), P(P1),END)

CALLING(DEFPLN,S(KFX),PLANE(KPLNX),P(P1),P(P2),P(P3),P(P1), END)

CALLS: GERROR DEFLIN

Subroutine DEFPNT(KFX,IP,XC,YC,ZC)

Defines the given coordinates X, Y and Z as Point IP in subfigure KF
Points must be defined in sequential (numerical) order - otherwise a nonfatal error is given. By setting ERRPAR = 1 this can be silenced.

USER CALL: CALLING(DEFPNT,S(KFX),P(IP),X,XC,Y,YC,Z,ZC,END)

CALLS: GERROR TRNSFM

Subroutine DELPLN(KF,KPLN)

deletes PLANE KPLN from subfigure KF. If KPLN = 0, deletes all planes from KF

USER CALL: CALLING(DELPLN,S(KF),PLANE(KPLN),END)

CALLS: GERROR DEFLIN

Subroutine DELPNT(KF,IP)

Deletes point IP of subfigure KF
IF IP is ZERO - Deletes all points of KF

Also - If KF is less than or equal to NUMS, it deletes lines and calls IDMAT

USER CALL: CALLING(DELPNT,S(KF),P(IP),END)

CALLS: GERROR IDMAT CHLINE

Subroutine DIST(KF,IP,KF1,IP1)

Puts in "RESULT" (global) the distance from point IP of subfigure KF to point IP1 of subfigure KF1.

USER CALL: CALLING(DIST,S(KF),P(IP),S(KF1),P(IP1),END)

NO CALLS

Subroutine DRAW(NSFX,IX,JAX,OBX,DIRCX)

NSFX is number of the subfigure to draw.
Subfigure numbers are in array IPT DRAW draws looking down JAXIS in "POS or NEG" direction
IX = 1 compute observer position
IX = 2 use observer position OBX
IX = 3 use observer at origin but move subfigure by TMAT(OBX) (where OBX is an integer) before drawing

Observer looks through angle of VANGLE (degrees)
If DIRCX LE. 0 observer looking $\rightarrow - \infty$
G 0 observer looking $\rightarrow - \infty$

NOGO $\neq$ 0 do not finish drawing, subsequent calls include more for current drawing

NTONE = 0 - line drawing

NTONE $\neq$ 0 - output tone data on tape 8

USER CALL: CALLING(DRAW,F(NSFX),JAXIS,$\text{OB}_{\text{NEG}}^{\text{POS}}$,END)

CALLS: GERROR CCP1PL TAPBINOT WRDBINOT DRAWFADE

MATMPY TRNSFM ENDBINOT BKSP

Subroutine EXCHPT(KFX,IP,IP1)

exchanges coordinates of point
IP and IP1. in subfigure KFX

USER CALL: CALLING(EXCHPT, S(KFX),P(IP),P(IP1),END)

CALLS: GERROR

Subroutine FLIP(KJ,JX,JY,JZ)

Reflects subfigure about axis (axes).
Does reflect about axis.
If corresponding J's are $\neq 0$.
Moves subfigure, does not produce new subfigure.

USER CALL: CALLING(FLIP,S(KF),XAXIS,YAXIS,ZAXIS,END)

In the ING call, the presence of the keyword for an axis causes a flip about that axis. One, two, or three of the keywords maybe present.

CALL: GERROR

Subroutine GERROR(X,NUMER,IPARAX)

Called from system.
Produces error messages on tape 6 if subroutines detect any incorrect workings. Errors are fatal or nonfatal. If fatal, execution is terminated. If nonfatal, execution continues, but your output will probably be messed-up. Always print tape six (6) after a graphics run. If you figure doesn't look like it should, GERROR can probably find the bugs.

NOT USER CALLED

CALLS SYSERR

Subroutine IDMAT(KF)

Replaces TMAT(Transformation matrix) for KF by the identity matrix.

USER CALL: CALLING(IDMAT,(KF),END)

CALLS: GERROR

Subroutine ING1

Interpreter and system tester checks all subroutines and initializes graphics system. ING1 is the core of the entire system. It analyzes the the contents of array IPT. Branches to the correct system routine, and calls GERROR if calls to ING have errors.

NOT USER CALLED

CALLS: UNPACK and all Perspective Routines

Subroutine INTSEC(KFX,LN,KF1X,LN1,KF2,IP)

Finds the intersection of lines.
If line segments KFX,LN and KF1X,LN1, intersect, Set RESULT to one.
If the line segments do not intersect, set RESULT to zero.
If the segments do not intersect, but extenstions of the segments do intersect, then set RESULT to minus one.
If IP is non-zero, and conditions are such that RESULT is one then insert the coordinates of the intersection point in point of IP of subfigure KF2

USER CALL: CALLING(INTSEC,S(KFX),L(LN),S(KF1),L(LN1),S(KF2),P(IP), END)

CALLS: DEFPNT

Subroutine LENGTH(KF,LN)

Computes length of line LN of subfigure KF.
Calls LINEX function to get points and then uses DIST subroutine to get length.

USER CALL: CALLING(LENGTH,S(KF),L(LN),END)

CALLS: DIST LINEX

Subroutine MATMPY(KFA,KFB,KFC)

Multiplies transformation matrix KFA by matrix KFB and puts the result in matrix KFC.

USER CALL: CALLING(MATMPY,S(KFA),S(KFB),S(KFC),END)

CALLS: COPMAT GERROR

Subroutine MIRROR(KF,JX,JY,JZ)

adds mirror image of subfigure KF about
axis to subfigure via FLIP
Mirrors only axes specified by nonzero JX,JY,JZ

USER CALL: CALLING(MIRROR,S(KF),XAXIS,YAXIS,ZAXIS,END)

CALL: GERROR COPY FLIP ADD

Subroutine MOVE(KF,XP,YP,ZP)

Moves subfigure KF or observer OB a specified distance in three coordinates. Note: XP,YP,ZP are distances, not a point in space.

```
                          OB
USER CALL:  CALLING(MOVE,S (KF),XC,XP,YC,YP,ZC,AP,END)
                         LSMAT
```

The MOVE may be specified along on, two, or three axis depending which key words are present.

CALLS: GERROR

Subroutine MOVETO(KF,IPX,JX,JY,JZ,KFk,IP1,XP,YP,ZP)

When JX,JY or JZ > 0 move subfigure KF along axis corresponding to JX,JY or JZ so that point IPX has same axis value as point IP1.
When JX,JY or JZ < 0 move subfigure KF along axis corresponding to JX,JY or JZ so that point IPX has same J axis value as XP,YP, or ZP, depending on which J being considered.
J = 0 movement along respective coordinate axis suppressed.

```
                          OB
USER CALL:  CALLING(MOVETO,S(KF),XC,XP,YC,YP,ZC,ZP,END)
                            LSMAT
CALLS:      GERROR    MOVE
```

Subroutine MOVPARM

A set of MV type routines for movie production.
Similar to the movie section of CALCOMP.
The use of this routine is discussed Chapter 3.

CALLS: Not User called as MOVPARAM.
Entries to some movie routines handled here

NO CALLS

Subroutine MOVTOB(KF,JX,JY,JZ,KF1X,IP1X,XPYP,ZP)

positions "observer" to a specified point

```
JX = 0  no x axis motion
   = 1  move to x coordinate of point  IP1X of S(KF1X)
   = 1  move to X coordinate of XP  and similarily for Y and Z
```

```
USER CALL:  CALLING(MOVETO,OB     ,XC,XP,YC,YP,ZC,ZP,END)
                           OBSMAT
CALLS:      MOVE
```

Subroutine PERSP(KF,JAXIS,YO,FOC)

Computes a perspective of a single subfigure YO - observer coordinate.
FOC - FOCAL LENGTH - directed distance from observer to point - if observer is at +x looking at (0,0,0), FOC is -x.

USER CALL: CALLING(PERSP,S(KF),JAXIS,YO,FOC,END)

CALLS: GERROR

Subroutine PLNPAR(KF,IP1,IP2,IP3,A,B,C,D)

Computes coefficients of normal form of plane given by points IP1, IP2,IP3, of subfigure KF

USER CALL: must be called directly, not accessed by ING - This is not intended to be a User accessed routine, however, it may be called if required.

This routine is called by other routines within the system, particularly by DRAW and DEFPLN

NO CALLS

Subroutine PRCFIG(IWHAT,NFIGX,NSFX,KF1X,NFIG1)

IWHAT = 1	delete old figure (NFIGX) and add new figure in IPT
IWHAT = 2	delete fighre NFIGX
IWHAT = 3	merge figure NFIGX into figure (NFIG1) delete NFIGX. If NFIGX = NFIG1X, then there is no deletion of NFIGX.

USER CALL: This routine is not called by the user, it is used for internal bookkeeping.

CALLS: GERROR

Subroutine PROJEC(KKK,III,KK1,II1,KK2,II2,XO,YO,ZO)

Computes coordinates of projection line (KK1,II1)(KKK,III) onto line (KK1,II1)(KK2,II2)

USER CALL: Must be called directly, not accessed by ING. This routine is not intended for user use, it may, however, be helpful.

CALLS: GERROR

```
Subroutine  PUNCHE(KF,NLORPX,NTYPE)

          Writes on tape 5 in Format Compatible with subroutine Read.
          NTYPE = 1 - (writes)  Points from Subfigure (KF)
                  2 - (writes)  Lines from Subfigure (KF)
                  3 - (writes)  TMAT(KF)
                  4 - (writes)  End Card
                  5 - No command
                  6 - (writes)  Polygons of Subfigure KF
          If Nlorp is zero - punch all lines or points in subfigure
          If Nlorp is not zero - punch only line or point NLORP of the subfigure.
                                                        ,LINE
          USER CALLS:     CALLING(PUNCHQ,S(KF),P(IP),POINT,END)
                                                        ,TMAT
                                                        ,PLANE

                          To write an end data file the routine must be called
                          as    CALL PUNCHE(1,1,4)

                          To write data with NLORP = 0 - the routine must be
                          called directly - You must always call Punche(1,1,4)
                          when finished writing with PUNCHE if you intend to
                          read the tape with READ - READ reads until it
                          encounters the end data file written by PUNCHE.

          CALLS:          GERROR  TRNSFM
```

Subroutine READ

Reads object definitions from tape 7 in card form

for points

col 1-5	7-21	22-36	37-51	52-66
POINT	Subfigure # N	starting point. If zero delete subfigure N		
blank	XC	YC	ZC	

for lines

LINE	Subfigure # N	starting line - if zero, delete all lines in subfigure N first		
blank	point 1	point 2		

for planes

PLANE	Subfigure # N	starting plane - if zero, delete all planes in subfigure N		
blank	point 1	point 2	point 3	point 4

for matrices

MATRIX	Subfigure # N	if "L" set to identity matrix		
blank	row 1 col 1	row 1 col 2	row 1 col 3	row 1 col 4
blank	row 2 col 1	row 2 col 2	row 2 col 3	row 2 col 4
blank	row 3 col 1	row 3 col 2	row 3 col 3	row 3 col 4
blank	row 4 col 1	row 4 col 2	row 4 col 3	row 4 col 4

Read will read until it encounters and end card col 1 - 5
end - - it will then return

NOTE: IF planes are read, the reflection coefficients must be initialized to 1.0 in the users program via SHADER. READ neglects to do this.

USER CALL: CALLING(READ,END,END)

CALLS: GERROR, IDMAT, TRNSFM, DEFLIN

Subroutine ROBAOG(KF,LA,DE)

Rotates observer about origin where origin is (0,0,0) transformed by TMAT(KF).

The rotation is by the angle of DE(degrees) about axis LA

LA = 1 - XAXIS is rotation axis
LA = 2 - YAXIS is rotation axis
LA = 3 - ZAXIS is rotation axis

USER CALL: CALLING(PAN,OB,ZAXIS,DE,END)

CALLS: DEFPNT XTRNOF

Subroutine ROTATE(KF,IPX,LA,JC,QUAD,BPX,KF1,IP1)

Rotates subfigure KF about the LA axis so that point IPX in KF has the same JC coordinate axis value as point IPI in subfigure KF1. QUAD and BPX are the Sine and Cosine of the rotation

USER CALL: CALLING(ROTATE,S(KF),P(IPX),LAXIS,JAXIS,S(KF1), P(IP1), END)

CALLS: GERROR XROTAT

Subroutine SCALE(KF,JK,JY,JZ,X),YP,ZP)

Changes the scale factor for a coordinate axis. If the "J" for that coordinate axis is not 0. The new scale factors are XP,YP,ZP. The scale factor is initialized to 0.

When using ING - the user may specify that all coordinates are to be scaled, ING will fill in all J's and parameters.

USER CALL: CALLING(SCALE,S(KF),XC,X),YC,YP,ZC,ZP,END)

CALLS: GERROR

Subroutine SHADER(KSFX,KPLNX,VALUE)

Sets the value of the reflection coefficient for a given polygon. The range of coefficients is 0.0 to 1.0

USER CALL: CALLING(REFLECT,S(KSFX),PLANE(KPLNX),VALUE,END)

CALLS: GERROR

Subroutine TIEOFF(NTONE)

Tieoff is used to divide long tone movies into 50 frame segments. It has no effect on line movies. When ntone = 1, it seals off the MDT currently being written. It then types the segment number completed, asks the user to mount a new tape, and when the user indicates, starts another segment.

USER CALL: CALL TIEOFF(NTONE)

CALLS: MVSTART, MVDONE

This routine is not part of the original Perspective System. It was added by Bill Dolson.

Subroutine TRNDIS(KF,IP,KF1,IP1,KF2,IP2,DIST,KF4,IP4,IWAYQ)

Computes SIN and COS for rotation of point IP in subfigure KF about the line whose end points are IP1 (in subfigure KF1) and IP2 (in subfigure KF2). Such that IP is a distance DIST from point IP4 (in subfigure KF4). Calls XTRNOF which does the rotation on subfigure KF. Looking from point IP2 (in subfigure KF2) toward point (KFI,IP1) rotation is done CCW to first point if IWAY = 1 or CW to first point if IWAY = - 1. If IP negative, returns with IPL IPL4 = SA IWAY = CA.

USER CALL: CALLING(TRNDIQ,S(KF),P(IP),S(KF1),P(IP1),S(KF2),P(IP2), DIST,S(KF4),P(IP4),IWAY,END)

CALLS: PROJEC GERROR XTRNOF

Subroutine TRNPLN(KF,IP,KF1,IP1,KF2,IP2, KPLN,IPLN1,IPLN2,IPLN3,IWAYQ)

Computes sin and cos for rotation of point (KF,IP) about line (KF1,IP1),(KF2,IP2) until it intersects plane given by points IPLN1, IPLN2, IPLN 3, of subfigure KPLN.

Call XTURNOF which does rotation of KF looking from (KF2,IP2) toward (KF1,IP1). The rotation is CCW if IWAY = 1, CW if IWAY = 1. If IP is negative, it returns with

IPLN1 = SA
IPLN2 = CA

USER CALL: CALLING (TRNPLQ,S(KF),P(IP),S(KF1),P(IP1),S(KF2),P(IP2), S(KPLN),P(IPLN1),P(IPLN2),P(IPLN3),IWAY,END)

CALLS: PLNPAR PROJEC GERROR TRNWHI XTRNOF

Subroutine TRNSFM(KFX,IX,MAT,KF1X)

If IX is zero, transform KFX using TMAT(KF)IX, put result into KFIX if KF.LE.NUMS

IF IX is one, transform KF using TMAT(MAT) put result into KF1X if KF,KF1 are both less than or equal to MAXS?

USER CALL: CALLING(TRNSFM,S(KFX),END

When called via ING, only the first operation can be performed with KFX and KF1X the same subfigure.

CALLS: GERROR COPPNT

Subroutine TRNWHI(XO,ZO,X1,Z1,IWAY,X,Z)

XO, ZO and X1, Z1, are points in XZ plane on circle with center at origin. If IWAY = 1, X, Z is set to 1st point CCW from the Z axis, if IWAY = -1, X. Z is set CW.

CCW - Counter clockwise   CW - clockwise

USER CALL: Must be called directly, not accessed by ING.

NO CALLS

Subroutine TURN(KF,LA,DE)

Calculates the sign and cosine of DE and calls XROTAT with the result.

USER CALL: CALLING(TURN,S(KF),JAXIS,DE,END)

CALLS: XROTAT

Subroutine TURNOF(KF,KF1,IP1, KF2,IP2,DE)

Computes Sine and Cos for rotation of subfigure KF about line (KF1,IP1),(KF2,IP2) calls XTRNOF which does the rotation looking from point (KF2,IP2) toward (KF1,IP1) the rotation is done CCW by DE degrees.

USER CALL: CALLING(TURNOQ,S(KF),S(KF1),P(IP1),S(KF2),P(IP2),DE,END)

CALLS: XTRNOF

Subroutine TURNTO(KF,IPX,LA,DE)

Turns the subfigure KF till the point IP is at an angle DE(degrees) in the plane given by LA and IPX.

USER CALL: CALLING(TURNTO,S(KF),P(IPX),LAXIS,DE,END)

CALLS: GERROR DEGREE TURN

Subroutine UNPACK(KR,J1,J2,J3,J4,J5,J6,J7,J8)

Defines integer indices which ING uses to pack IPT.

NOT USER CALLED

NO CALLS

Subroutine VOCAB(DSTEP,BSTEP, NPRECT,NDURATN)

Contains the entry points VOCAB and DRAWFADE. VOCAB sets storage allocation parameters and calls XVOCAB.
Sets display parameters for 1024 scope.

USER CALL: CALL VOCAB

CALLS: DRAWLABL CPB1LBL

Subroutine WRITE(KF,NLORPX,NTRYPEX)

Writes point, line, subfigure, TMAT's on tape 6. The format is more compressed than that of PUNCHE, however, it is not readable by READ. Instead, its purpose is to output information to the user by a printout.. (A dump mechanism)†

If NLORP = 0 then all points or lines in subfigure KF are written or the subfigures in all figures are written otherwise; only point, line, or figure NLORP is written.

```
IF NTYPE = 1  points are written
           2  lines are written
           3  then TMAT(NLORP) is written
           4  then the points which are connected to point(s)(NLORP)
                are written
           5  the subfigures of figure NLORP are written
           6  the points defining polygons of subfigure NLORP are
                written
```

```
                                        POINTS
USER CALL:  CALLING (WRITE,S(KF),LINES ,END)
                                        MATRIX
```

CALLS: GERROR TRNSFM *TAPBCDOT *WRDBCDOT *ENDBCDOT

Subroutine XROTAT(KF,LA,SN,CN)

Rotates Subfigures KF about axis LA by DE where SN = sin DE CN = cos DE. This routine is usually called by TURN

USER CALL: must be called directly, not accessed by ING

CALLS: GERROR

†Output, 6, p, before the "go" will give regular print output of this.

Subroutine XTRNOF(KF,KF1,IP1,KF2,IP2,SA,CA)

Produces a CCW rotation of subfigure KF about the line (KF1,IP1), (KF2,IP2) by angle = $\sin^{-1}$(SA) = $\cos^{-1}$(CA) called TURNOF.

USER CALL: must be called directly, not user called.

CALLS:	GERROR	MATMPY	MATMPY

Subroutine XVOCAB(INP)

Initializes (zeroes) storage and tests the storage allocation parameters.

USER CALL: CALL XVOCAB(INP)

CALLS	GERROR	IDMAT	VOCAB1	*CCPCFFTN

TABLE II

PERSPECTIVE SYSTEM FUNCTIONS

F	NPOLY
GX	P
L	PLANE
LINEX	S
LSCOOR	SHADE

These functions require no special accessing and are treated strictly as FORTRAN functions.

Function F(I)

Generates a figure number for interpretation by ING.

NO CALLS

Function GX(IP,IA,KF)

Maps† point IP-in subfigure KF with coordinate of IA out of array G. If a transformation matrix exists for KF, it specifies GX with respect to that matrix.

```
GX = value of (KF,IP)  in G
IA = 1  GX is X coordinate value
IA = 2  GX is Y coordinate value
IA = 3  GX is Z coordinate value
```

NO CALLS

Function L(I)

Generates a line number for intpretation by ING.

CALLS: GERROR

Function LINEX(KF,LN,IP)

Computes mapping function for line array, similar to GX.

NO CALLS

Function LSCOOR(IP)

Generates light source coordinates

CALLS: IDMAT

†See Appendix II for mapping function details.

Function NPOLY(IP,IPLN,KSF)

Mapping function for NPLY, similar to GX

NO CALLS

Function P(I)

Generates point number for interpretation by ING.

CALLS: GERROR

Function PLANE(I)

Generates polygon number for ING.

CALLS: GERROR

Function S(I)

Generates subfigure number for interpretation by ING.

CALLS: GERROR

Function SHADE(KSFX,KPLNX)

Computes mapping function for reflection coefficient array SHADES, similar to GX.

NO CALLS.

TABLE III

PERSPECTIVE SYSTEM ROUTINE INDEX

This table lists the user-called Perspective System routines by function. It is a good place to look when you have a problem to solve, you may save yourself considerable effort.

Subfigure Combinations

ADD	adds one subfigure to another, deletes the first
COPY	copies points, lines, TMAT for KF into KS
DELETE	deletes subfigure
DEFBLK	defines a group of subfigures as a "BLOCK". These subfigures may then be manipulated in a single call.

Point Combinations

COPPNT	copies points of one subfigure into another
DEFPNT	defines points
DELPNT	deletes points
EXCHPT	exchanges coordinates of 2 points

Line Combinations

CHLINE	adds lines of one subfigure to another, deletes lines in first subfigure
COPLIN	copies lines of one subfigure into another
DEFLIN	defines lines
DELLIN	deletes line

Plane Combinations

COPPLN	copies planes of one subfigure into another
DEFPLN	defines plane
DELPLN	deletes plane

Subfigure Manipulations

FLIP	reflects subfigure about specified axis-movement, no new subfigure
MIRROR	adds mirror image of one subfigure to another
MOVE	moves subfigure specified distance in three axes
MOVETO	moves subfigure to specified point
ROTATE	rotates a subfigure until a point has a given value
TRNDIS	computes rotation sin and cosine for rotating subfigure KF about given line - rotates by reference to a point in KF
TRNPLN	same as TRNDIS - rotates by reference to plane in a subfigure
TURN	turns a subfigure about axis LA by DE degrees - calls XROTAT
TURNOF	turns a subfigure about given line by DE degrees - calls XTRNDF
TURNTO	turns a subfigure till given point in that subfigure is at DE degrees to plane given by that point and a given axis
XROTAT	turns a subfigure about an axis until coordinates of a point are at CN,SN or by degrees where XROTAT is called by TURN
XTURNOF	turns a subfigure by degrees when called from TURNOF, otherwise equivalent to XROTAT where CN,SN, are in plane L to line

Observer Manipulations

MOVTOB	moves observer to a specified point
ROBAOG	rotates observer about origin transformed by TMAT(KF)

INPUT -OUTPUT

PUNCHE	write graphics data on tape 6 in format compatible with READ
READS	reads cards from tape 7 written by PUNCHE
WRITE	writes data on tape 6 this format is not compatible with READ
DRAW	outputs figures to display terminal for line graphics of tape 8 for grey-scale graphics.

TMATRIX Manipulations

COPMAT	copies TMAT for one subfigure into TMAT for another
DEFMAT	defines TMAT for a subfigure
IDMAT	deplaces TMAT for a subfigure by identity matrix
MATMPY	multiplies TMAT for a subfigure by TMAT for another subfigure and puts result in TMAT for a third subfigure
SCALE	changes scale factor for a subfigure
TRNSFM	transforms a subfigure by any TMAT and puts the result in G

Parameter Determinations

DEGREE	determines angle of $\overline{(O,P)}$ projected onto axis plane
DIST	distance from between two points
INTSEC	determines whether lines intersect, can make a point in a subfigure the intersection coordinates
LENGTH	computes length of a line in a subfigure
PROJEC	computes coordinates of projection of one line onto another
TURN	calculates sin + cosine of DE for XROTAT
TURNOF	calculates sin + cos for rotation about line calls XTRNOF

1.6.0 Debugging

The Perspective System incorporates a code checking subroutine GERROR, which informs the user of certain errors in the user program. GERROR writes error messages on a tape placed on logical unit 6. Copy the tape out to printer for a print-out of messages. The error messages are written in a formated FORTRAN. They are self-explanatory.

The errors are fatal and non-fatal. Fatal errors cause immediate termination of the user program. Non-fatal errors do not force an exit. By setting "ERRPAR = 1" in the user program GERROR writes out only fatal errors. Leaving the ERRPAR flag unspecified tells GERROR to write both fatal and non-fatal errors.

Two main types of errors will be found. First, improper calls to ING are commonly made. Consult the Subroutine Glossary and be certain you use the correct CALLING statement. Second, redefining the same point, line or plane (overwriting) is wrong. This is a non-fatal error and can be done if necessary but should usually be avoided.

Note the following. User program subroutines which call ING must contain the statement "TITLE *" placed directly after the subroutine identification. All variables in the user program and its subroutines should be of the explicit type. No integer variables (I,J,K,L,M,N) may be made REAL, no real variables may be made INTEGER.

Also, take care not to accidentally use any FIGSYS array or varible names as varibles in your program. This would cause your program to blow up instantly.

On rare occasions a CALLING statement cannot be properly compiled and thus ING cannot access the requested subroutine. In such a case, consult the Subroutine Glossary for the direct call to the subroutine, and call it directly as a normal FORTRAN routine call.

When GERROR lists errors, it prints with each error the value of a system varible INGC. INGC is initialized to zero and is indexed by one each time ING is called. It therefore indicates where in a program an error has been detected. In very long programs it is easy to lose track of INGC. To aid in debugging it is recommended that INGC be set to a known unique value (10,000, 20,000,etc) at various points in long programs.

Nonfatal errors, Execution continues

Points skipped in subfigure ---

Points overwritten in subfigure ---

Lines overwritten in Subfigure ---

Subfigure --- overwritten

Lines skipped in subfigure ---

Less than four rows of TMAT were read, the rest of the rows of an Identity Matrix.

Header Card causes deletion of points in subfigure ---

Header Card causes deletion of lines in subfigure ---

Header card causes deletion of entire subfigure---

Line has been defined with uninitialized point number

Exceed max polygons for subfigure ---

--- Lines and Planes of subfigure ---

Planes overwritten in subfigure ---

Fatal Errors, Execution Terminates

Array size for G exceeded

Array size for LINE exceeded

Attempts to access undefined line in subfigure ---

Type Number must be 1,2,3, or 4, not ---

Attempts to access undefined point in subfigure ---

Illegal transformation matrix

Attempts improper access of subfigure ---

Improper valve for array parameter

Illegal header card

Missing header card

Attempted access of nonexistant transformation matrix

--- is Illegal error routine parameter

Error in parameter list

No subroutine specified

Too many arguments for called subroutine

Too few arguments for called subroutine

Incorrect number of points, lines, or subfigures for Called Subroutine

No end in parameter list

Transform on subfigure sets scaling parameter to zero

Attempted rotation about wrong axis

Attempted rotation to a point which is not within axis of rotation

Points defining an axis are coincident

Radius of rotation is zero

Fatal errors(continued)

Storage overflow when defining figure---

Figure number --- is out of range

Attempts to access undefined polygon

No subfigure specified

No plane specified

Array size for NPOLY exceeded

CHAPTER II. GREY SCALE GRAPHICS

2.1.0 Using the Tone System

The use of the system thus far described allows the user to output figures in a line draw mode on the CRT. A system is available which accepts object definitions from the Perspective system and produces data on a mag tape which after processing can be displayed in a raster-scanned grey-scale mode on the CRT. In the grey-scale mode, the object is represented by "planes", which are specified by the Perspective System call DEFPLN.

To run a perspective program in grey-scale mode, we must set several additional flags and structure our object definitions in a different manner than those for line graphics. We first define points. We do not, however, define lines. In the grey-scale mode, lines and points are not represented. Instead we define planes. Planes are defined by reference to points in a manner similar to lines. To define a plane we would insert the following call:

```
CALLING(DEFPLN,S(1),PLANE(1),P(1),P(2),P(3),P(4),P(1),END)
```

This defines plane one in subfigure one. The vertices of the plane are points one through four. When point one is repeated it signals the termination of the plane definition. Planes may be quadrilateral or triangular. For triangular planes the call would be:

```
CALLING(DEFPLN,S(1),PLANE(1),P(1),P(2),P(3),P(1),END)
```

When a plane is defined, the data for the points compromising the plane is stored in array NPLY, and lines are defined between the vertices to form the boundary of the plane. The storage limit for planes is 6 planes per subfigure.

After planes are defined, the Perspective System data is outputted on tape 8. This is accomplished by setting a flag, NTONE, equal to 1. When DRAW is called with NTONE = 1, the lines in the figure are not displayed on the scope, instead, the plane data is written on tape 8. This is called a "Tone Draw Tape". After the data is written on 8 it must be processed by two systems, the TONE system and the PIXSCANR system.

The TONE system analyzes the draw output tape and by means of the LINESCAN[†] algorithm determines which planes are visible, which portions of the planes are visible, and what intersections, if any, exist between planes. This data is outputted to another tape, the PIXSCANR tape. This tape contains the basic information needed to construct the raster-scan for the final graphical output. The PIXSCANR system reads this tape and assembles the rast-scan information in a scope buffer in core. The scope is then activated and the picture is scanned.

In addition to determining the plane visibility, the TONE system computes the effect of a variable light source position on the plane display. If the planes were all given the same grey-scale level, the output would be practically unitelligible. Instead, the brightness of the planes is assigned as a function of the angle between the surface of the polygon and a given light source postion. The values for the brightness of the plane surfaces are quantized into the 256 intensity levels of the scope.

[†]See reference list at end of Manual.

The position of the light sources is given by calling MOVE or MOVETO and specifying LSMAT (light source matrix) instead of a subfigure.

```
CALLING(MOVE,LSMAT,XC,value,YC,value,ZC,value,END)
```

The light source postion is initialized to (0,0,0)

For most photographic films, the range of intensities which the scope can produce greatly exceeds the dynamic range of the film. In this case the intensity range must be narrowed to accomodate the film response. This is done by setting two system variables; DARK and BRIGHT. Both values range from .0 to 1. DARK is the minimum grey scale intensity, and BRIGHT is the maximum intensity. For polaroid film, DARK = .2 and BRIGHT = .9 gives good linearity over all grey scale intensities. DARK is initialized to 0. and BRIGHT to 1., however, no film has this wide a latitude, so they should always be reset to more conservative values.

A final system parameter to set is RASTRSIZ (integer) RASTRSIZ specifies the fineness of the raster scan which creates the tone display. The display will scan 2^{RASTRSIZ} by a 2^{RASTRSIZ} raster. It is initialized to 9. Ten is a suggest value. Values any higher than ten will result in a display which is too bright to photograph.

In summary, for grey-scale graphics, the user's program must take the following form.

```
PROGRAM NAME
TITLE*
CALL VOCAB
CALL VOCAB(INP)
CALL CPB1
UP = nAXIS
DARK = value
BRIGHT = value
RASTRSIZ = 10
CALLING(MOVE,LSMAT,XC,value,YC,value,ZC,value,END)
object point defintions
object plane definitions
object manipulations
CALLING(DEFFIG,S(N),S(M),F(I),END)
NTONE = 1
CALLING(DRAW,F(I),nAXIS,OB,direction,END)
END
```

This program would cause one tone draw to be written on tape 8 and the program will exit. In practice, it is useful to transfer control to ACTIONS and then have one user entry point do a tone draw on 8. In this case, the final statements of the program would be

```
      CALLING(DEFFIG,S(N),S(M),F(I),END)
      CALL ACTIONS(I,|701,|702,|703,nAXIS,direction)
      Go to 704
701.  NTONE = 1
      CALLING(DRAW,F(I),nAXIS,OB,direction,END)
      NTONE = 0
      CALL REACTION
```

```
702  User code
     CALL REACTION
703  User code
704  END
```

While in ACTIONS, the user may move the observer and also vary the light source position. Any number of Tone Draws may be made consecutively on Tape 8. In Actions, the figure will be displayed on the scope in a line draw mode. The lines displayed are those defined by a call to DEFPLN. No additional lines need be or should be defined in a tone mode program.

When your program is finished, you should have a series of Tone Draws on tape 8. The next section will describe how to process them for a finished picture.

2.2.0 Processing Tone Draw Tapes

If your grey scale program has worked correctly, you should have one or more tone draws on tape 8. Rewind tape 8 and autoload the NON-SHADOW TONE system (tape A21). If you have a series of pictures to process, raise jump switch 1.

The tone draw tape is on unit N.

A scratch tape should be on unit M when the TONE system is autoloaded, type

```
TONE,N,M carriage return
```

The TONE system now will begin to process each tone draw. For a maximum of 240 planes, (6 planes in each of 40 subfigures), computation time is about 5 minutes per picture. The computation time is proportional to the number of visible edges in the picture.

When TONE returns the message

"readtape endfile"

it will have computed the PIXSCANR sequence for all the pictures on tape N. Rewind tape M and autoload the PIXSCANR system (tape A23).

When the system is autoloaded, type

PIXPROCS,M

The PIXSCANR system will read in one picture and start scanning it on the scope. The time of scan is depenedent upon the resolution needed. The scan takes about 10 seconds.

On the first scan, the shutter is opened and a photograph taken. Use F4.0 as a starting point for the polaroid camera. If the picture is not satisfactory, open or close the F-stop as necessary and take another photgraph.

In the PIXSCANR system, three typewriter commands exist:

"X" - Make another photograph
space - Go to next picture on tape M
"." - exit

If you find that two planes are not adequately distinguished from one another, as in the case of two parallel planes which are assigned the same intensity by Tone but which you wish to differentiate, it is possible to change the intensity of either plane. In the Perspective System, each plane has an associated "reflection coefficient", which is used as a starting point for the intensity calculation in TONE. All reflection coefficients are intialized to 1.0. They may, however, range from .0 to 1.0. The reflection coefficient may be changed by calling a Perspective System routine SHADER in the user's program. The call is:

```
CALLING(REFLECT,S(N),PLANE(I),reflection coefficient,END)
```

If the reflection coefficient for a plane is reduced, it will appear darker than others to which it is parallel.

2.2.1 Debugging

On rare occasions the TONE system can blow up unexplainably or produce strange horizontal "glitches". This is usually due to some type of plane condition which produces round-off-error. Usually a slight repositioning of the observer is enough to produce a toneable draw tape. If the trouble persists, check your program. If you have read in plane definitions via READ, check to make sure you have initialized the reflection coefficients for the planes read in, READ will not do this, it must be done by calling SHADER.

TONE prints one error message, it is:

```
READTAPE SEQUENCE ERROR
```

It always indicates a tape read error. Rewind the tone draw tape and autoload the system again. If the error continues try another tape unit.

TONE and PIXSCANR are exceptionally trouble free. You should not encounter any problems with them. The only serious error you can get occurs when you give TONE a quadrilateral plane defintion where the four points do not in acutality lie in a plane. If the system does not blow up, you will probably get garbage from PIXSCANR. Should this happen, recheck your plane definitions. The first three points of a plane definition determine the plane. If those three happen to be colinear the plane is totally ignored.

2.3.0 Shadow Tone Graphics

Karl Kelley has written a special version of the TONE system which not only does a grey-scale display of planes, but also computes shadows falling on these planes. The system is called SHADOW-TONE and is on tape A22. The user program and tone draw processing are identical to those used in NON-SHADOW TONE with the exception that tape A22 is substituted for tape A21 in the tone draw processing steps. Processing time is about 1.5 times greater than for NON-SHADOW Tone.

The user is, however, cautioned. Due to storage limitations imposed by the 1604, SHADOW-TONE is limited to 200 planes. There is no guarantee that this number will be successfully processed by SHADOW-TONE. The storage limit for shadows is readily exceeded with only a few tens of planes. The light source should be positioned so that the number of shadows is at a minimum. This is easier said than done for many types of figures. If your figure cannot be processed successfully by SHADOW-TONE, be content with NON-SHADOW-TONE. Mr. Kelley recommends that SHADOW-TONE only be used for simple figures where the planes are well separated.

CHAPTER III. ANIMATION

3.1.0 Line Graphics Animations

One of the recording devices which can be attached to the CRT display console is a MITCHELL 16mm. animation camera. Jack Bouknight has written an assembly language system which controls the operation of the movie camera and the writing of special data tapes (known as Movie Data Tapes or MDTs) which enable the user to write records on tape representing movie frames. This MDT can later be read by the movie system and recorded on film with the movie camera. The MOVIE system is system 19 in the 1604 library. System 19 is on the FORTRAN auxiliary master and the perspective system tape D-38.

The Perspective System subroutines CALCOMP and MOVPARAM allow the user to generate an MDT in a simple manner by means of calls inserted in this program. For line movies there are six calls which generate movie data tapes.

```
CALL MSSTART (tape unit, 77B, 8HBLKWHITE)
```

This call initiates the movie system. It is the first call in a program which generates a line MDT. The arguments are the logical tape unit on which the tape is to be written, 77B which is a movie system flag, and 8HBLKWHITE, which is an MDT label.

```
CALL MSTAKE(NFRMS,NEXPL)
```

This call causes the next call to DRAW to be recorded as NFRMS, frames of a movie, where each frame is exposed NEXPL times. For lines NEXPL = 8.

```
CALL MSSKIP (NFRMS)
```

This call causes NFRMS frames of film to be skipped when the MDT is recorded on film.

CALL MSDONE

This is the last call in a movie sequence. It causes an MDT to be completed by writing a special MDTEND record.

The remaining two calls are used when you wish to monitor the movie generation with ACTIONS.

CALL MDTFDBK

CALL MDTHOLD1

MDTHOLD1 turns off the movie system for one call to DRAW. If you call MSTAKE, CALL MDFHOLD1, CALL MDTFDBK and CALL ACTIONS -- REACTION, MSTAKE tells the movie system to record the next DRAW as an MDT frame. However, MDTHOLD1 allows the movie system to skip the first draw in ACTIONS which activates the display. You are now in ACTIONS and the display CRT is activated. MDTFDBK constrains the movie system while in ACTIONS so that a frame is only recorded when the space bar is pressed. You may therefore move the observer in ACTIONS and record your choice of views as movie frames. The MDTFDBK flag may be cleared by calling MDTNFDBK.

The structure of a Line Movie Program is as follows:

```
Activate Perspective System
Define Points
Define Lines
Define Figures
CALL MSSTART (arguments)
CALL MSTAKE (arguments)
CALLING(DRAW,F(I),nAXIS,OB,direction,END)
CALL MSDONE
END
```

The MSTAKE and DRAW calls are usually nested in a DO loop with an incremented call to some Perspective System manipulation routine. MSSTART and MSDONE are only called once. MSTAKE and DRAW must be called for every frame. When the MSSTART call is encountered, the MOVIE System will type "TYPE ID LABEL". You may then type in up to 8 BCD characters and carriage return. This becomes a label for the MDT.

If your program works you should get a good movie data tape on the unit selected in the MSSTART call. Line movie data tapes compute very rapidly and are fairly compact. As a rule of thumb, for a maximum of 1200 lines, one full tape can hold about 1000 frames.

3.2.0 Grey-Scale Graphics Animations

To produce a grey scale movie data tape, the process is a bit more complicated. As with one-shot tone graphics, an intermediate tone draw tape must be made. This is then processed by TONE and PIXSCANR to produce the final tone MDT.

To generate the intermediate tone draw MDT, calls similar to those for line movies are used. They are inserted into a grey-scale graphics program. The calls are:

CALL MVSTART - no arguments otherwise identical to MSSTART. The tone draw tape is always written on lgical tape unit 8.

CALL MVTAKE(NFRMS,NEXPL) - identical to MSTAKE for tone, NEXPL = 1

CALL MVDONE - identical to MSDONE

MDTFDBK,MDTNFDBK, and MDTHOLD1 are the same for line or tone movies.

These calls are used in the same manner as their line counterparts.

The intermediate tone draw tape is escessively bulky. For a maximum of 240 planes, one full tape can hold about 50 frames. It is strongly recommended that the user's tone draw tape not contain more than 50 frames. If a longer animation is required then the tape should be sealed off after 50 frames. Longer animations which require more than one tape can later be recombined into a single sequence in the form of movie data tapes after processing by TONE and PIXSCANR. The Perspective System Routing TIEOFF will automatically divide movies into 30 frame segments.

If your program has run correctly, you should now have one or more tone draw tapes of 50 frames. These must now be processed by NON-SHADOW-TONE and PIXSCANR. This is done with a group of programs called the SURFMOVI system. This is a "ping-pong" or "leap frog" routine. Once started, it runs continually until the entire tone draw tape has been processed into an MDT, alternately loading TONE and PIXSCANR for each frame to be processed.

To SURFMOVI a tone draw tape, mount the folliwng tapes:

Unit 1	- FORTRAN Master
Unit 2	- Scratch Tape
Unit 3	- Scratch Tape
Unit 4	- SURFMOVI System (Tape A24)
Unit 5	- FORTRAN Auxiliary Master
Unit 6	- Tone Draw Tape (from movie program)
Unit 7	- Scratch Tape
Unit 8	- Output Tape - Final MDT

To run the system put jump switch 2 up and type

```
GO,SURFMOVI
```

the program now requests,

```
"TYPE ID LABEL"
```

type up to eight BCD characters, carriage return.

The system will run automatically until finished. The system types two segment numbers for each frame. This is a great help in predicting run time and keeping track of SURFMOVI processing. Computation time is the same as that for individual tone pictures.

While running Surfmovi you may encounter two error messages.

Readtape Sequence Error

Readtape asks for more records than on Writetape. When either occurs it always indicates a read error on the part of the tape units, and the system should be re-started at the beginning. This is the primary reason for the 50 frame maximum per tape. Should the system fail, there is no way to restart it at the frame where the error was made. With an average computation time of at least a minute, little damage is done if the system must be re-started after a tape error on frame 49. If the error were on frame 490, the situation would be much more serious!

There is a SHADOW TONE processing system for tone movies. It is called SHADMOVI and is on tape A25. It is run just like SURFMOVI. However, the same restrictions apply to movies as to one-shot graphics. SHADMOVI is only recommended for fairly simple figures.

When SUFMOVI or SHADMOVI are finished, they will have produced a tone MDT similar to the line MDT. These MDT's are now ready to be viewed and shot on movie film.

3.3.0 Movie Data Tape Service Programs

The MDT's you now have may be ready to shoot or, more likely, may need some editing before being recorded on film. The MOVIE System has several service routines which allow the user to view his MDT on the scope, edit it, combine it with others, or obtain a listing of its contents on the printer.

The following routines are on the Auxiliary Master. It should be mounted on Tape N.

SCANMOVI - will display each frame of an MDT on the scope. To activate it type "CALL,N,SCANMOVI", where N is the number of the tape unit which contains the MDT. The first frame will then be displayed. To view the next frame touch the display scope with the light pen. To take a polaroid photograph type a period. SCANMOVI will ask for an exposure count. Type 1 and carriage return. A polaroid is then taken of the current frame.

LISTMOT - is called and run just like SCANMOVI. It will ask for a movie record type. You type "BLKWHITE". LISTMDT then prints the sequence number, frame number and exposure count for each frame on the MDT.

COMBMDT - combines several MDTs in to one MDT. This is used to combine the 50 frames segments from long tone movies. This program is called like SCANMOVI. To run it the user types "COMBMDT". The routine then asks for a string of tape unit numbers. The first number will indicate the output tape where the combined results will be written. The other numbers indicate locations of MDTs to be combined. They are combined in the order inputted.

COPYMDT - copies one MDT onto another tape. It is called like SCANMOVI. To run COPYMDT type "COPYMDT,N,M". N is the tape unit where the original MDT is. M is where the copy is to be produced.

CHANGMDT - is used to edit MDTs. Edits are read from paper tape in the following form.

SEGMENT(s), first segment #, second segment #, task, value.

SEGMENT means one frame. SEGMENTS means a sequence of frames. If SEMENTS, then only the first segments is specified. The numbers are taken from the LISTMDT printout.

There are 8 tasks, they are:

DELETE - deletes specified segment

FRAMES - changes frame count for given segment to "value"

EXPOSE - changes exposure count for given segment to "value"

REPEAT - repeats given segment "value" times

FADEIN - produces "value" frame fade-in of given segment

FADEOUT - produces "value" frame fade-out of given segment;

EXTEND - extends given segment, "value" frames by changing framecount on the last frame

SKIP - skips given segment, the data tape is ended by typing FINIS at the end.

These routines enable the user to manipulate MDTs to get the exact sequence of frames desired. He need never edit a piece of film.

3.4.0 Shooting Movie Data Tapes

Now that you have a good MDT, we are ready to record it on film. For line or tone graphics, the recommeded film is Kodak 4-X Negative 16 mm movie film, double perforated. For lines, recommended exposure is 8 exposures at f8.0. For tone movie recommended settings are 1 exposure at f4.0 with DARK = .2, BRIGHT = 1. RASTRSIZ = 10. The Mitchell animation camera has 400 feet magazines which can take loads on cores or reels.

The first thing to do is set up the animation camera.

1) Turn off the Polaroid camera power supply.

2) Remove the Polaroid Camera.

3) Remove the two large brass nuts from the front of the movie camera mount beneath the camera.

4) Wheel the movie camera into position on the scope, making sure the camera mount is snug against the CRT console attache points.

5) Screw the three floor jacks on the camera mount all the way down.

6) Tighten the two draw bars on the floor mounts of the CRT-camera mount.

7) Replace two large brass nuts and tighten securely.

8) Carefully attach bellows to Polaroid camera mount.

9) Connect 8 prong plug from movie camera to side of CRT console.

10) Check for a green light in the hole on the left side of the console. If it is on, press the reset button beside the light. It should turn off.

11) Plug in 110V line from movie camera to lower back of console.

12) Turn movie camera power on at small power supply on camera mount.

Note: Always turn off power supply before removing any plugs.

If the power supply light does not come on

a) Make sure the camera is racked all the way to the left.

b) Push the film reset button above the footage counter.

c) check the power supply fuse.

13) Load the film magazine.

14) Push the reset button above the footage counter.

15) Remove the lense cap and dark slide.

Now we are ready to shoot the MDT. This is handled by a movie system routine named DOMOVIE. Mount the FORTRAN Auxiliary Library on tape 5 and type

```
CALL,5,DOMOVIE
```

and then

```
GO,DOMOVIE
```

DOMOVIE will then type

"INPUT FOOTAGE AND FRAME COUNT"

you type the number of feet of film in the magazine followed by the number of frames shot. For a 100 foot magazine with no frames shot, type

100,0 carriage return

DOMOVIE then types

"ADJUST CAMERA SYSTEM AND PICTURE"

and displays a focus pattern.

Press the button on the lever on the back of the camera and rack the camera all the way to your right. The focus pattern is now visible in the viewer. Open the black door on the side of the lens box and open the f-stop all the way to f2.0. Push the knob below the viewer all the way forward and focus the camera.

Now allign the cross hair in the viewer on the center of the focus pattern. Use the adjustment knobs of the camera mount.

Check to see that the switch in back of the camera scope is in the "P" position.

If the corners of the test pattern do not intersect correctly adjust the camera scope line length knob in back of the scope. Do not judge this by the display scope, look at the pattern on the camera scope.

Adjust the f-stop to that desired for the movie run. Rack the camera back over the left with the lever. Make sure the camera locks full left. Camera power should come back on. Close the door on the lens box after setting the f-stop. A carriage return signals the program to continue.

DOMOVIE then types:

"INPUT AVAILABLE TAPES BY LTN"

Type the logical number of the unit or units to be used for reading MDTs. Type a carriage return after each number. Type a zero to signal the end of the list.

DOMOVIE will type:

"INPUT STATUS OF FILM LOAD"

Type "NEW" for a new reel just loaded,

"OLDCUT" for a partially exposed reel that has been removed from the camera, and

"OLD" for a partially exposed reel that has not been removed from the camera.

DOMOVIE then types:

"INPUT REEL NUMBERS"

Type the name of the reel(s) which contain MDTs to be shot. Carriage return after each reel number. Type "END" to end the reel list.

DOMOVIE will then direct a tape to be mounted on a logical number tape unit. When this is complied with, press the space bar to continue.

DOMOVIE finally asks

"INPUT JOB NAME"

Type up to 16 BCD characters to be used as a label on the film. A recommended label is film type, number of exposures, and f-stop.

DOMOVIE then begins to shoot the movie data tape one frame at a time. Line frames take about 3 seconds a piece to expose. Tone frames may take as much as 30 seconds.

When DOMOVIE is finished shooting the MDT it will type several lines of data about the run and the type.

"DECIDE NEXT ACTION--NEWJOB, CUT, EXIT"

Type "NEWJOB" to shoot another MDT immediately.

Type "CUT" to shoot 5 feet of tails and then exit.

Type "EXIT" to leave the system immediately.

If you are going to remove the film magazine you should always type "CUT".

When finished, remove the magazine and label it as shown.

_________________________FILM TYPE

_________________________FEET SHOT

_________________________(your name)

_________________________(production name)

____

(Indicate if a print is desired.)

Return the magazine to CSL Photo Shop.

It is suggested that a work print positive be made for all footage. Negative deteriorates rapidly when projected. Use your print for viewing and preserve the negative. Additional prints are easily made from the negative. If you plan to do so the negative should <u>never</u> be projected or even run through a viewer.

A final but important note. If you are shooting film which is expected to be edited together, it is important to shoot it at one sitting. The movie camera can never be exactly repositioned when it is moved, so identical frames shot at different times will not register perfectly.

CHAPTER IV. COLOR GRAPHICS

4.1.0 Techniques for Color Graphics

The usual recording material for graphics produced on the 1604 is black and white film, either polaroid or movie. Occasionally, it is useful to be able to record the graphical output in color. This capability can add another dimension to the already powerful impact of graphic material.

A color recording method has been designed for use with the Perspective System. Its implementation is, however, somewhat time consuming and expensive but for certain applications it can be more than justified.

Color recording films essentially consist of three separate emulsions, each sensitive to one of the three additive primaries, RED, BLUE, and GREEN. By differential exposure of one or more of these emulsions the entire visible spectrum may be simulated. To produce color graphics, three exposures for each frame must be made, one for each primary. Each exposure contains only those lines or planes which are to appear on the corresponding layer of the color emulsion.

Thus for example, if we wish to color a line in green, we would shoot three black and white positives. The line would be absent in the first two exposures corresponding to the red and blue emulsions in the color film, and the line would only appear in the third exposure, corresponding to the green layer of the color emulsion.

If these three positives were printed on the same frame of color film, each through the correct filter, our line would appear as green on a black background. If we wished to reproduce a line as white, it would appear on all three exposures. If we wished it to appear red, it would only appear on the first exposure. If blue, only on the second.

The three black and white positives we must produce and then record through filters onto color film are called, logically enough, color separation positives.

In the case of color line graphics, in addition to the three primaries and white we may also include the three complementaries, YELLOW, MAGENTA (a pinkish-purple), and CYAN(a deep blue-green). For a line to appear yellow it must appear on the red and green positives. For it to appear MAGENTA it must appear on the red and blue negatives, and for CYAN, the blue and green positives..

For line graphics, we therefore have the capability of recording six colors and white.

For toned graphics, the situation becomes much more complex. Lines may only be maipulated on our positive in a binary sense, they may be present or absent. Planes can be manipulated in this manner, but another possibility exists. By assigning a different reflection coefficient to plane in each color separation negative we can produce a wide range of colors. As an illustration, if the reflection coefficient of a plane is 1.0 in the red color separation positive, .5 in the blue, and .5 in the green, when the positives are shot on a color film the plane will appear as a subdued purplish red. By manipulating reflection coefficients in this manner we may reproduce the entire spectrum in a fairly accurate manner.

The user need not worry about the process of deleting lines or assigning reflection coefficients. Bill Dolson has written a routine which is in the

Perspective System and allows the user to assign a color to a line or plane via paper tape. The routine then decides which planes or lines to modify for each color separation negative. The use of this routine is described in the next section.

4.2.0 Making Color Movies with CHROMA

Subroutine CHROMA allows the user to painlessly generate the three color separation positives for producing color movies. If color stills are desired, they may be printed from color movie frames. Negatives are shot on 4-X Negative film with the same exposure and f-stop used for black and white movies. These negatives must then be printed to obtain the color separation positive.

CHROMA is called immediately after the MSSTART or MVSTART movie calls. The call is:

```
CALL CHROMA(ntone, ncolor)
```

The NTONE argument tells chroma whether a line or tone movie is being produced. Again, NTONE = 0 for line movies, NTONE = 1 for tone movies. The argument NCOLOR tells CHROMA which color separation negative is being produced, 1 = RED, 2 = BLUE, 3 = GREEN. CHROMA expects to read a paper tape with color assignments. In a line mode it reads in 2I2,I1 format. The first integer is a subfigure number. The second integer is a line number. The third integer is a color number, which is the color assigned to that line where:

1 =	RED
2 =	BLUE
3 =	GREEN
4 =	CYAN
5 =	YELLOW
6 =	MAGENTA
7 =	WHITE

If the line number is zero, the entire subfigure is colored. An all zero record signals the end of data. All lines not assigned a color will appear white. Note: <u>All lines to be colored must be inputted in reversed numerical order</u>. There is no constraint on the order of subfigures.

In a tone mode CHROMA reads data from paper tape in 2I2,I3 format. The first integer is a subfigure number, the second is a plane number. The third integer is the color number for that plane. This color number is taken from the color bar chart in the Perspective System listing in the machine room. Copies of this chart may be obtained from Jan Kremers. There are 216 colors available. It should be remembered that the colors as represented on the chart are at their maximum saturation and brightness. If the plane to be colored is not normal to a line from the light source to the centroid of the plane, the colors will substantially decrease in brightness, just as the intensity of planes decreased in black and white grey scale photographs. To insure visibility of colored planes, it is wise to choose a conservatively bright color. There is no constraint on the order of the data read by CHROMA in a greyscale mode. Again, an all zero record signals the end of data.

Your movie program should be run three times, onece with NCOLOR = 1, once with NCOLOR = 2, and once with NCOLOR = 3. Each run will produce the MDT's for one color separation negative.

When you have finished shooting all three negatives, personally return the film and magazine to Jack Gladdin in the CSL Photo Shop. Describe exactly what is contained in the footage and which color separation negative is where. It is essential that you give correct instructions for printing. An innocent misunderstanding could easily cost several hundred dollars.

Also, there is an absolute minimum of 20 feet of color separation footage per color. You will be charged for multiples of 100 feet of film regardless of the length of your footage, so save your MDT's until you have 100 feet of film for each color.

The details of the printing technique for color separation negatives is discussed in Appendix V.

APPENDIX I

TITLE EQUIVALENCES

INP(1) = ALL	INP(21) = SUBFLF
INP(2) = XC = XCOORD	INP(22) = OB = OBSVER
INP(3) = YC = YCOORD	INP(23) = OBSMAT
INP(4) = ZC = ZCOORD	INP(24) = SC = SCREEN
INP(5) = XA = XAXIS = YZPL	INP(25) = ALAXIS = ALAX
INP(6) = YA = YAXIS = ZXPL	INP(26) = MATMPY
INP(7) = ZA = ZAXIS = XYPL	INP(27) = ROTATE
INP(8) = POS	INP(28) = TURN
INP(9) = NEG	INP(29) = READ
INP(10) = ALLCRD = AC	INP(30) = WRITE
INP(11) = FOCAL	INP(31) = PUNCHQ
INP(12) = POINT = POINTS	INP(32) = COPY
INP(13) = LINE = LINES	INP(33) = COPPNT
INP(14) = TMAT = MATRIX	INP(34) = COPLIN
INP(15) = END	INP(35) = COPMAT
INP(16) = ROW	INP(36) = MOVE
INP(17) = COL = COLMN	INP(37) = MOVETO
INP(18) = ELMNTS	INP(38) = ORIGIN
INP(19) = CONECT	INP(39) = ADD
INP(20) = SUBFLS	INP(40) = STORE

```
INP(41) = LENGTH
INP(42) = DELETE
INP(43) = DELPNT
INP(44) = DELLIN
INP(45) = DELMAT = IDMAT
INP(46) = MIRROR
INP(47) = FLIP
INP(48) = DIST
INP(49) = DEFINE
INP(50) = DEFPNT
INP(51) = DEFLIN
INP(52) = DEFMAT
INP(53) = PERSP
INP(54) = TRNSFM
INP(55) = DRAW
INP(56) = SCALE
INP(57) = XCHNGE
INP(58) = EXCHPT
INP(59) = TURNTO
INP(60) = DEGREE
INP(61) = INTSEC
INP(62) = DEFFIG
INP(63) = BLK
INP(64) = TURNOQ
INP(65) = ZERO = ZRO
INP(66) = TRNDIQ
INP(67) = TRNPLQ
INP(68) = DELFIG
INP(69) = ADDFIG
INP(70) = FIG = FIGS
INP(71) = PAN
INP(72) = DEFPLN
INP(73) = COPPLN
INP(74) = REFLECT
INP(75) = DELPLN
INP(76) = PLANES
INP(77) = REFCOEF = RCOEF
INP(78) = LSMAT
INP(79) = ORBIT
INP(80) = LITESORC
INP(81) = DRAWTWO
```

APPENDIX II

MAPPING IN THE ARRAY G

G is defined as a one dimensional array, usually stored in common.

In this array the Resch System stores the X,Y,Z coordinates for defined points and the point numbers of defined lines.

The three-space coordinates for a point are stored in such a way that the address of the coordinates are essentially the subfigure number and point number.

This function in the present system is performed by the subroutine GX.

In a normal fortran array of the type DIMENSION A(I,J,K).

The mapping function is given by

(1) $A(i,j,k) = A(i + (j-1) \cdot I + (k-1) \cdot I\text{-}J)$

which is conceptually equivalent to stacking the storage words into a three-dimensional array or cube of dimension n,m,J.

Applying some algebra to (1)

$A(i,j,k) = A(i + j * I - I + k * I * J - I * J)$

$= A(i + j * I + K + I * J - I - I * J)$

$= A(i + (j + k + J) * I - (1 + J) * I)$

$= A(i + I * (j + k * j) - I * (J + 1))$

In array G I = IDIM1 = max no pts/subfigure

J = IDIM2 = max no space coordinates per point

$I\text{-}(J+1)$ = IDIM = array size constant

K = NUMS = maximum number of subfigures

We now address G as

(3) G(i,j,k) = G(i + IDIM1 * (j * k * IDIM2) - IDIM)

where G(i,j,k) = G(point #, coordinate, sufigure #). To find the three space coordinates of point IP in subfigure KF we then need to compute

G(IP,1,KF), G(IP,2,KF), G(IP,3,KF)

That is, evaluate .3 for values of j = 1,2,3 since only j varies.

G(IP,3,KF) = G(G(IP,2,KF) + IDIM1)

= G(G(IP,1,KK) + LD1MD + 1DIM1)

if we again consider the cube analagy, we now have

n = IDIM1 points per subfigure

m = maximum # subfigures = NUMS

J = IDIM2 = 3 coordinates per subfigure

The three coordinates of the first point in first subfigure are at G(1), G(n + 1), G(n + 2).

The arrays LINE, TMAT, NPLY, and SHADES are also mapped in this manner, with the appropriate values of I,J,K. See Appendix IV for these constants.

APPENDIX III

PERSPECTIVE SYSTEM ARRAY NAMES

†G - x,y,z coordinates for points

INP - array of key words for processing of system calls by VOCAB1

IPT - array of storage locations of INP key words after processing by VOCAB1

†LINE - numbers of points in lines

LSCOORS(Real) - light source coordinates

NF - NF(I) = number of figure which subfigure I is in

NL - NL(I) = number of lines in subfigure I

NPLN - NPLN(I) = number of planes in subfigure I

†NPLY - number of points in polygons

NPT - NPT(I) = number of points in subfigure I

†SHADES - reflection coefficients of planes in polygons

†TMAT - contains transformation matrices for subfigures

† Conceptual 3-dimensional array, must be mapped as in Appendix II.

APPENDIX IV

IMPORTANT PERSPECTIVE SYSTEM VARIABLE NAMES

ATEST - test value for first TMAT for a given subfigure

ATEST = 2476.13 set by XVOCAB

BRIGHT - specifies grey scale maximum intensity

DARK - specifies grey scale minimum intesity

DERAD - radian/degree VOCAB sets to 0.01745329251994

DINCY = HTDINCY = 4094

DIRCX - when less than or equal to zero, viewer looks toward - ∞.

- when greater than zero, viewer looks toward + ∞

EPS - VOCAB sets to 0.0001

ERRPAR - if equal to 1 will only print fatal errors; otherwise is equal to zero

HTDINCY = 2 * * RASTRSIZ

IDIM = IDIM * (IDIM2 + 1) - array size constant for mapping into G

IDIM1 - maximum number of points per subfigure equals 30(= NUMP)

IDIM2 - maximum number of space coordinates needed to define a point (= 3)

INGC - ZERO set by XVOCAB

JDIM = JDIM1 * (JDIM2+1) - array size constant for polygons

JDIM1 = 6 - polygons per subfigure

JDIM2 = NUMPLY = maximum number of points to define a polygon = 4

K - subfigure number

KF - subfigure number

KS - subfigure number

LDIM = (LDIM1 * (LDIM2 + 1)) - array size constant for LINE

LDIM1 = MAXS

LDIM2 = NUML maximum number points to define a line = 2

LDIMI = LDIM1 * LDIM2

MAXS - maximum subfigure storage = LDIM1 = 41

NOGO - flg for draw - if 1, draw again, VOCAB sets to ZERO

- if non zero, draw will not finish

NSUB = NUMS = 40

NSUBS = 1 set by XVOCAB

NUML - maximum number of lines in any subfigure (= LDIM2 = 30)

NUMP - maximum number of points in any subfigure (= IDM1 = 30)

NUMPLY - maximum number of polygons per subfigure (= JDIM2 = 6)

NUMS - maximum number of subfigures = 40

NUMTM - maximum number of transformation matrices = 44

NTONE = 0 set by XVOCAB

PI = π = VOCAB sets to 3.141592653589793

RADDE - degrees/radian VOCAB sets to 57.2957795

RASTRSIZ(INTEGER) - the power of two indicating resolution to be used for grey-scales, RASTRSIZ = 9 grey-scale system to work on (2^9 = 512) 512 x 512. Recommend 10.

RCOEF - reflection coefficient for planes

RESULT - a system "register" used as a place to put results of system routines, e.g., DIST.

VANGLE - specifies viewing angle of observer in degrees, VOCAB sets to 45.$^\circ$, range of 0. to 360. degrees.

XTOY - X to Y ratio, VOCAB sets to 1.0

APPENDIX V

PROCESSING COLOR SEPARATION NEGATIVES

Jack Gladin has agreed to let us include his notes on the very delicate task of printing a color positive from the three black and white negatives produced by CHROMA.

Normally, these three negatives are first printed into black and white print stock. This is a normal printing operation. CHROMA and the entire grey-scale system assume that the final photographic record is a positive.

These three positives are then printed on Kodak Ektachrome Print Film with a normal pack.

The RED positive is printed through a 25A filter and .4 Neutral Density filter at light 18.

The GREEN positive is printed through a 58 filter and a .2 Neutral Density filter at light 11.

The BLUE positive is printed through a 47B filter and NO Neutral Density Filter at light 14.

In rare cases, a color print may be required directly from the camera negative. All color and tone values will be reversed, but for certain applications this is acceptable.

The three negatives are printed on Ektachrome Print Film with a normal pack.

The RED negative is printed through a #25A filter and a .40 Neutral Density filter at light 8.

The GREEN negative is printed through a #58 filter and a 120 Neutral Density filter at light 8.

The BLUE negative is printed through a #47B filter and no Neutral Density filter at light 22.

These procedures give very good color rendition and saturation. The following data illustrate the accuracy of this method.

A control strip and a color print were analyzed with a densitometer.

	Overall Density	Red Density	GREEN Density	BLUE Density
Control Strip	.87	.87	.85	.84
Print	1.54	1.53	1.50	1.59

This is the best possible compromise between the three colors. It leaves blues a little weak and reds too intnse, but this can be compensated for by choosing appropriate colors from the CHROMA color chart.

ADDITIONAL REFERENCES

For those who are interested in further investigation of CSL's CRT display, look to:

Jack Stifle "A Cathode Ray Tube Display," CSL Report R-357, June, 1967. (revised and reprinted, March 1970).

If anyone should desire a knowledge of the internal workings of the tone and shadow-tone systems the following two publications may be of interest.

W. J. Bouknight, "An Improved Procedure for Generation of Half-tone Computer Graphics Presentations," CSL Report R-432, September, 1969.

Karl C. Kelly "A Computer Graphics Program for the Generation of Half-tone Images with Shadows," CSL Report R-444, November, 1969.

Source listings of the Perspective System and the Movie System are available in the computer room. The Perspective System is System 37. The Movie System is System 19.