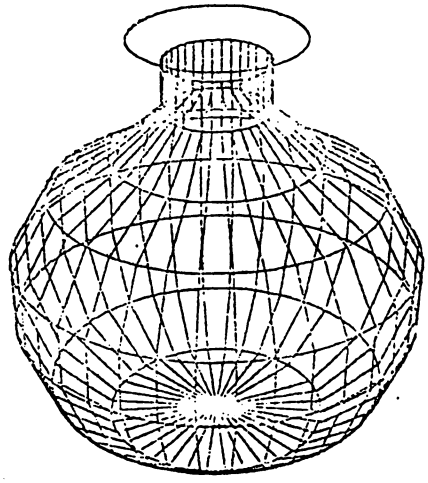
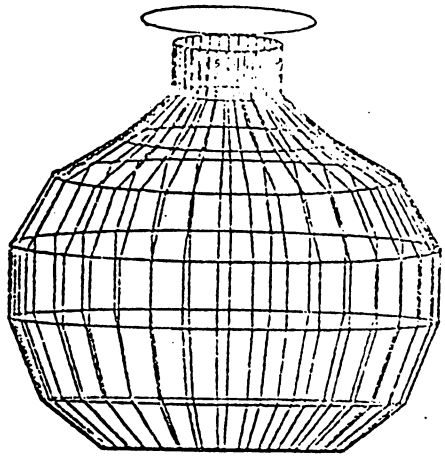
GPGS-F, A PORTABLE DEVICE INDEPENTANT GRAPHIC SYSTEM.

Authors : Fredrik Major, Ditlef Martens
The Ship Research Institute of Norway
(NSFI)





The Ship Research Institute of Norway (NSFI) has implemented GPGS-F on the HP-3000 series computer and is responsible for HP-versions of the system.

GPGS-F is a system near to the standards for graphic systems proposed by standardisation organizations in several countries and is choosen as standard graphic system by the Norwegian Co-operation in Computer Graphics (NORSIGD). This organization is responsible for the basic development of GPGS-F which is carried out on a Univac 1100 series computer by the Computing Centre at the University of Trondheim (RUNIT).

The system is implemented on various diffirent computers and has device drivers for many of the common graphic devices. In this paper GPGS-F will be described from a functional point of view, aiming to give potential users of the system an overview of it's capabilities.

1. HISTORY

The GPGS system was originally designed by Rekencentrum, Delft University of Technology, The Netherlands and Science Faculty, Catholic University Nijmegen, The Netherlands in 1972. A version of the system written in standard Fortran has been developed by NORSIGD. The first Fortran based version was released in 1975 and named GPGS-F. GPGS-F has been under continous development since the first version was released. This work has been guided by annual user meetings. The result of these meetings has been new features and minor changes to the system.

2. DEVICE CONTROL

GPGS-F provides device independent programming with choice of graphic device at run-time.

Figure 1 shows how the device independancy is obtained. The device independant part produces the same picture code for all devices, and the device driver(s) translates this code to the bit pattern required for the actual device. The device independant code is put on such a level that advanced graphic devices may be used in an efficient way. Examples of such functions are character, circle and marker generation. GPGS-F will also be able to take advantage for the functions of more advanced refresh display such as hardware scaling, rotation and depth quing. The graphic devices currently supported in the HP-3000 version og GPGS-F are:

Tektronix 4010/4014
Tektronix 4662
Calcamp plotters
HP 7221 plotter
HP 2648 graphic screen

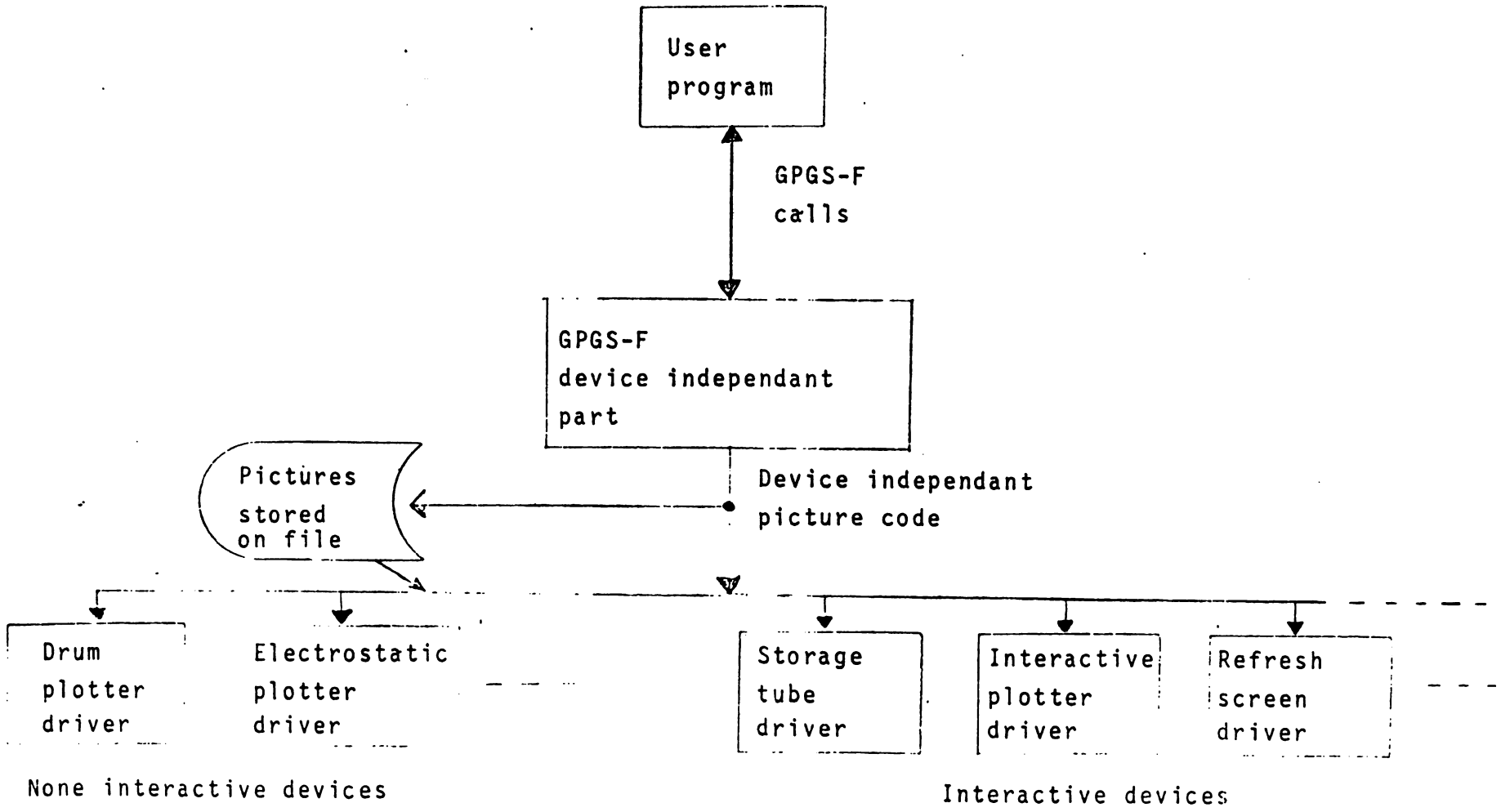


Fig. 1. MAIN STRUCTURE OF GPGS-F.

3. GRAPHIC ELEMENTS.

By single calls to the basic routines of GPGS-F, the following graphic elements may be generated:

- straight lines
- set of straight lines
- circles/circle areas
- text
- markers
- functions

All graphic elements may be hardware generated by the graphic device.

The elements are defined in a user defined coordinate system (2-or 3-D Kartesian) and are fully transformable.

4. COORDINATE SYSTEMS

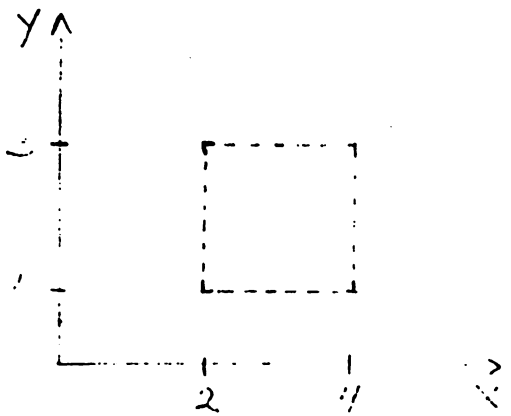
The GPGS-F user defines the graphic elements related to a userdefined coordinate system (2- or 3-D kartesian).

For the graphic device's drawing surface, a standardized coordinate system (2- or 3-D kartesion) is used. In this system the length of the shortest side of the device is set to be 1.0.

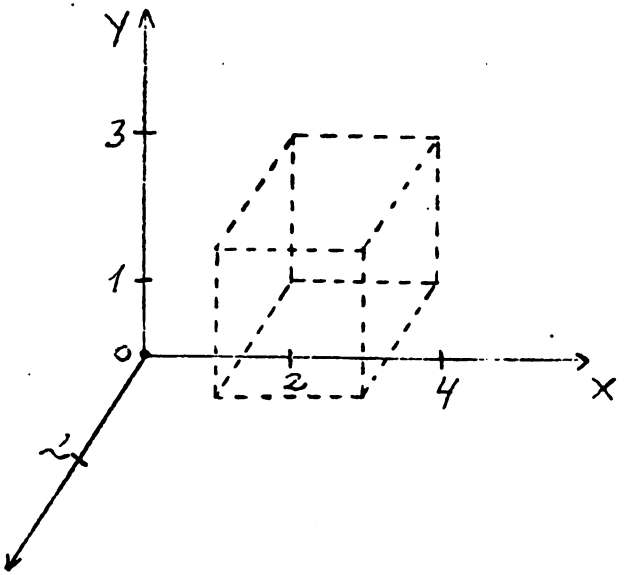
When the user defines a drawing in user coordinates, the system will map this drawing into the device's coordinates. The user defines the part of the user coordinate system which is to be mapped into the device's drawing surface by specifying a window. Examples of 2D and 3D windows are shown in fig. 2.

The part of the device's surface where the data inside the window are to be shown is defined by setting a viewport. Example of viewport and mapping from user coordinates into device coordinates are shown in fig. 3.

2-D WINDOW:



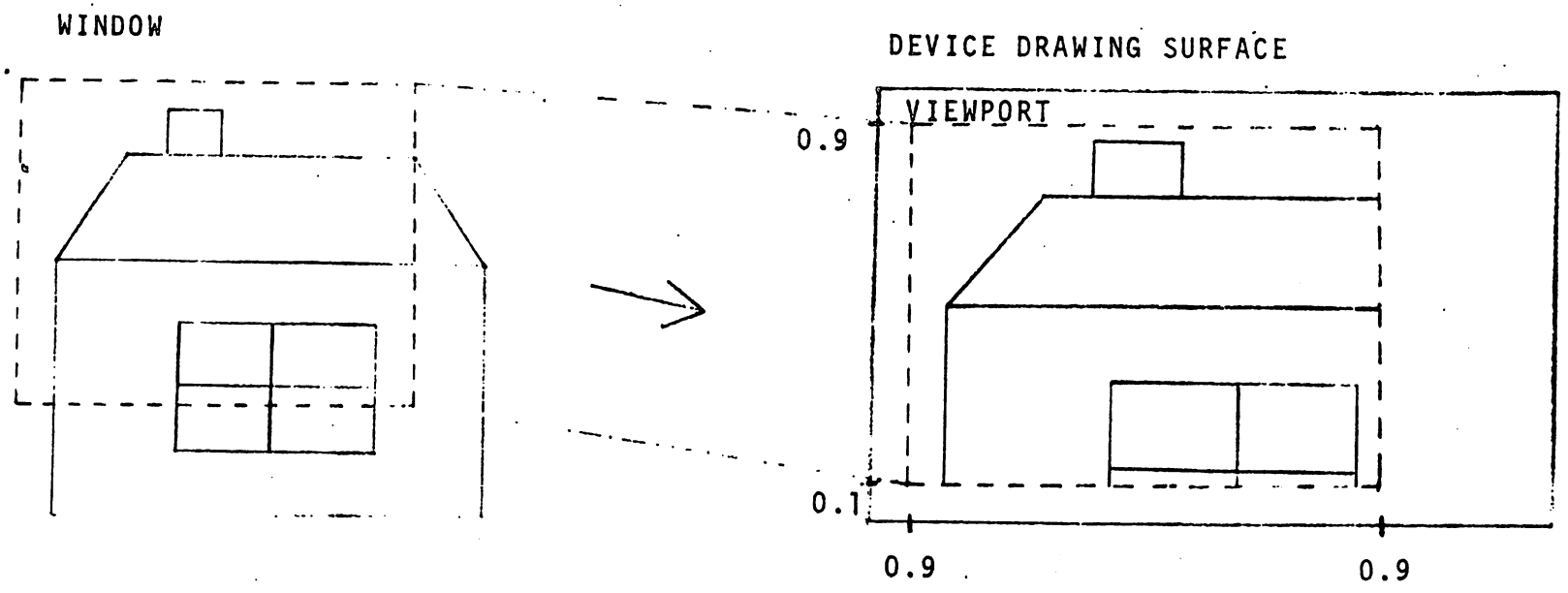
3-D WINDOW:




```
DIMENSION W (4)
DATA W / 2.0, 4.0, 1.0, 3.0/
CALL WINDW (W)
```

```
DIMENSION W(6)
DATA W/ 2.0, 4.0, 1.0, 3.0, 0.0, 2.0/
CALL WINDW3 (W)
```

FIG. 2 EXAMPLES OF 2D AND 3D WINDOWS



Defined picture in
user's coordinates

Definition of viewport:

```
DIMENSION V(4)
DATA V / 0.1, 0.9, 0.1, 0.9 /

CALL VPORT (V)
```

Fig. 3 VIEWPORT AND MAPPING FROM USER TO DEVICE COORDINATES

When a 3-D picture is shown on a 2-D drawing surface a projection is made. By default the picture will be projected parallelly into the X/Y plane part of the window box and then mapped into the viewport. Later in this paper other projections will be discussed.

5. TRANSFORMATIONS

GPGS-F system transformations change the coordinates a user passes to the system before they are viewed through the window. The change is carried out as a matrix multiplication of the input coordinate elements and a transformation matrix. Let (x,y,z) be the input points and T the current transformation matrix (homogenous), then the transformed point (x', y', z') will be:

$$[x', y', z', 1] = T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

As the GPGS-F user can access the transformation matrix, any type of transformation that can be contained within the 4 x 4 square matrix can be performed.
Some transformation types are standard, and special routines provides automatic modification of the transformation matrix.

A list of the available transformations without user manipulation is given below:

- translation
- scaling
- rotation
- shearing

The effect of these transformation types are shown in fig. 4 based on examples in 1 .

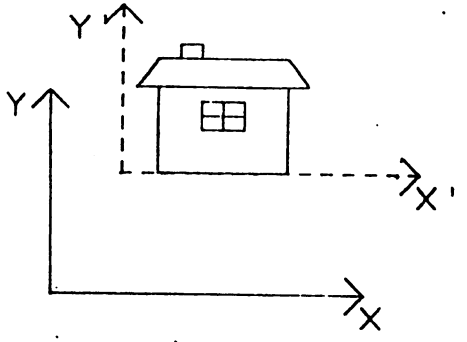
TRANSFORMATIONS

XLAT(Xdisp,Ydisp)

XLAT3(Xdisp,Ydisp,Zdisp)

Example:

```
CALL XLAT(20.0,40.0)
CALL HOUSE
```



ROTATE

ROTA(Angle,laxis)

ROTAD(Dangle,laxis)

Angle=radians
Dangle=degrees

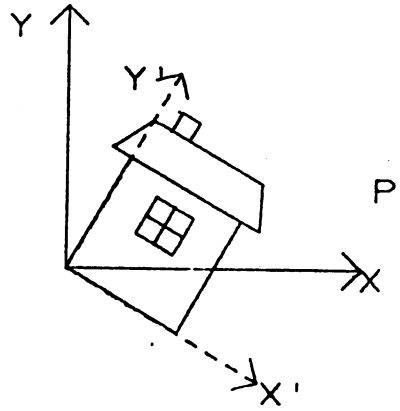
laxis=1 X-axis
laxis=2 Y-axis
laxis=3 Z-axis

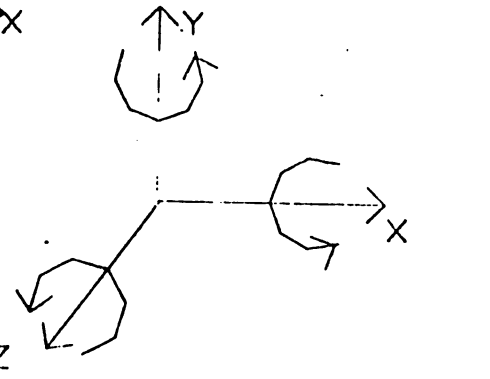
Example:

```
CALL ROTA(-0.524,3)
CALL HOUSE
```

or

```
CALL ROTAD(-30.0,3)
CALL HOUSE
```





SCALE

SCAL(Scale,laxis)

laxis=0 X,Y,Z axis
=1 X axis
=2 Y axis
=3 Z axis

Example:

```
CALL SCAL(2.0,1)
CALL HOUSE
```

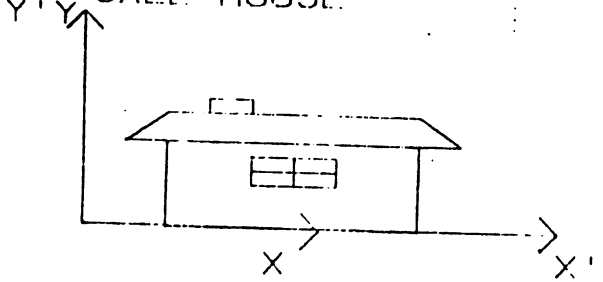
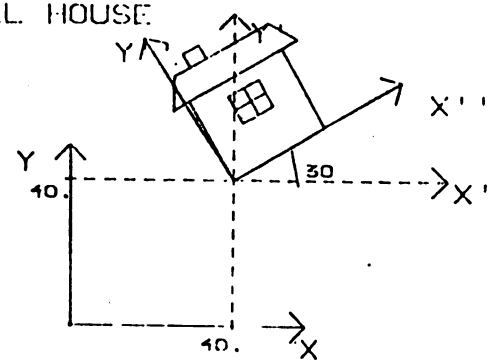


FIG. 4. EXAMPLE OF BASIC TRANSFORMATIONS

Combining Basic Transformations

```
CALL XLAT(40.,40.)
CALL ROTAD(30.,3)
CALL HOUSE
```




```
CALL ROTAD(30.,3)
CALL XLAT(40.,40.)
CALL HOUSE
```

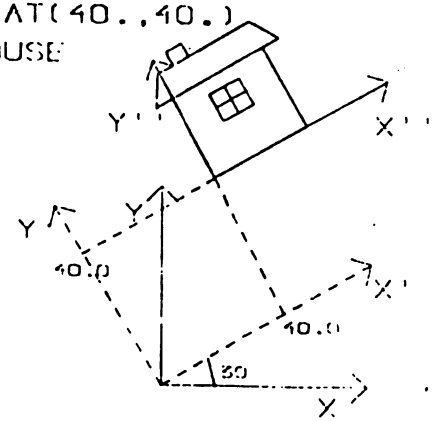


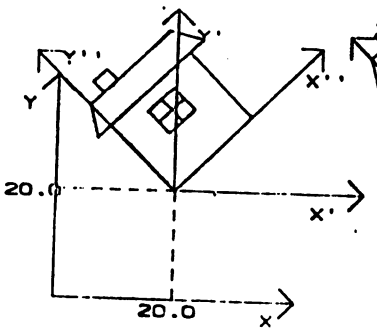
FIG. 5 COMBINING TRANSFORMATIONS

PICTURE MODE - SPACE MODE

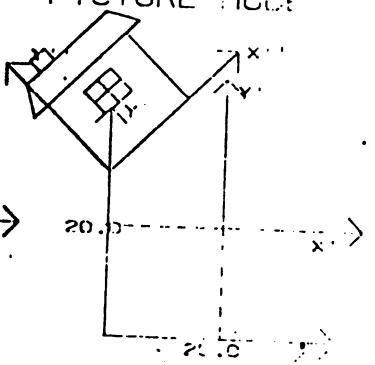
```
CALL MODTRN(I)
  I=0 activate space mode
                    =(default)
  I=1 activate picture mode
```

```
CALL XLAT(20.,20.)
CALL ROTAD(15.,3)
CALL HOUSE
```

SPACE MODE



PICTURE MODE



Transformations relative:

to current coordinate system

to original coordinate system

Transformations may be combined, and are relative to either current or original coordinate system depending on the transformation mode. (See fig. 5, from [2])

The current transformation may be stacked in up to 10 levels for later restore. This is useful for example in a drawing subroutine with internal transformations.

6. PROJECTIONS

When a 3-D drawing is viewed on a 2-D device, the drawing will be projected parallelly into the X-Y plane if nothing else is specified.

Routines for spesification of the projection are provided by GPGS-F.

Two types of projections are provided, parallell and perspective. The projection is set up by means of a single routine call specifying the eye point. Examples of projections are shown in appendix A.

7. PICTURE SEGMENTS

Picture segments are a collection of graphic elements forming a picture or picture part that may be separately identified for example for lightpen picking.

Picture segments may be stored in primary or on secondary storage by means of a pseudo driver for later use.

Picture segments fetched from such store may be retransformed. Figure 6 shows how this facility is linked to GPGS-F.

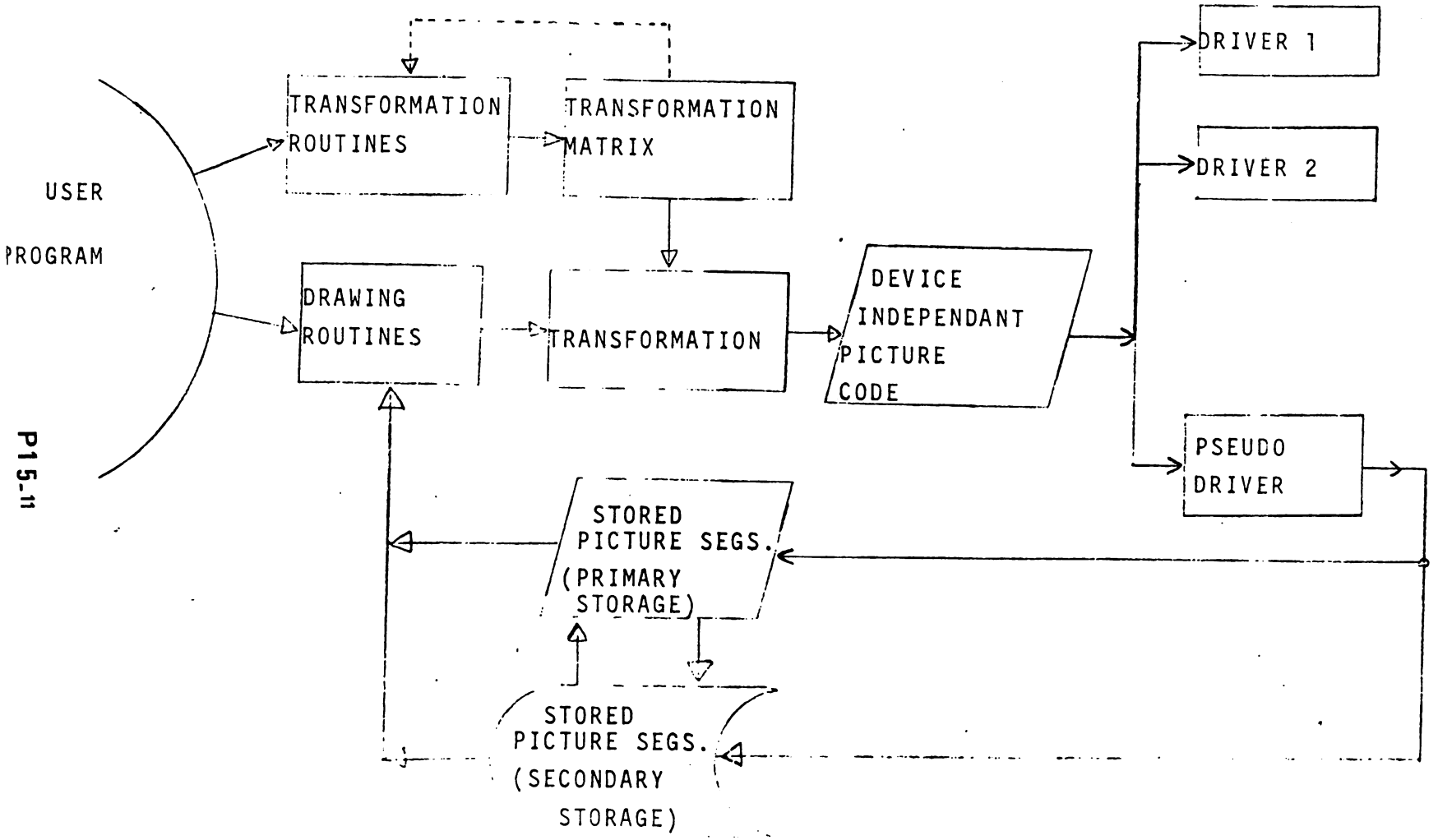


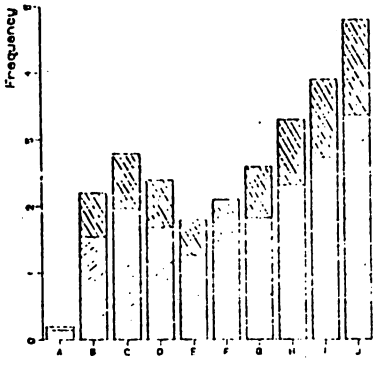
FIG. 6 STRUCTURE FOR STORING OF PICTURE SEGMENTS

8. INTERACTIONS

For access of interactive facilities of the graphic devices GPGS-F uses reference to logical tools. In this way device independany is ensured.

An example showing how this works is that digitizing equipment for an interactive plotter, crosshair for a storage tube and graphic curser for a raster graphics terminal will all be considered as the same tool from GPGS-F.

It goes up and down in life



9. GRAPH AND HISTOGRAM PLOTTING

Also available is a subroutine package for general graph and histogram plotting called GRAPHISTO 3 . Output of computed results is an often used task and having the results plotted gives a much better view of the data than having them printed in columns on a printer.

This package is designed to remove the programmer from doing all the drawing details like finding range of datas, drawing axes et.

It uses GPGS-F basic routines for drawing and will therefore give the user easy access to all the graphic output devices available through GPGS-F. Other facilities like changing pen colour to get different colours on different curves, plotting in true scale, and so on can be used together with GRAPHISTO.

GRAPHISTO provides 4 so-called 'chart' routines that will in answer to a single call draw a complete diagram with annotated axis, texts on axis and datapoints.

The types of plots provided through these 4 routines are:

- Histogram with labels under each bar and linear or logaritmic axis in x and y.
- Table of lines with straight lines between plots.
- Smooth curve through specified points.
- Pie chart.

These 4 routines use the basic GRAPHISTO routines as axis drawing, range computation, 'nice' value computation and curve plotting. The lower level routines are also available to the user and offers possibilities for sophisticated non-standard plots like multiple axis, marking special data points etc. Appended to this chapter are some plots to demonstrate the use and possibilities of GRAPHISTO.

Available facilities are:

Chart plotting:

- Simple and smooth (cubic spline) curves in diff ·ent linestyles.
- Histograms. They may be plotted with or without hatching of bars in any angle.
- Pie charts with texts and percentage of total pie.

Axis drawing

- Near to plot on either side of plot.
- Through any data or page value.
- Several parallell with different units.

Axis annotation:

- On either side of axis
- Numeric or text labels
- Any character size and angle
- Upper and lower case
- Extra tick marks
- Title

Grid:

- Along x- or y-axis
- Linear or logaritimic
- Any line type or '+' at grid crossings

Dataplotting:

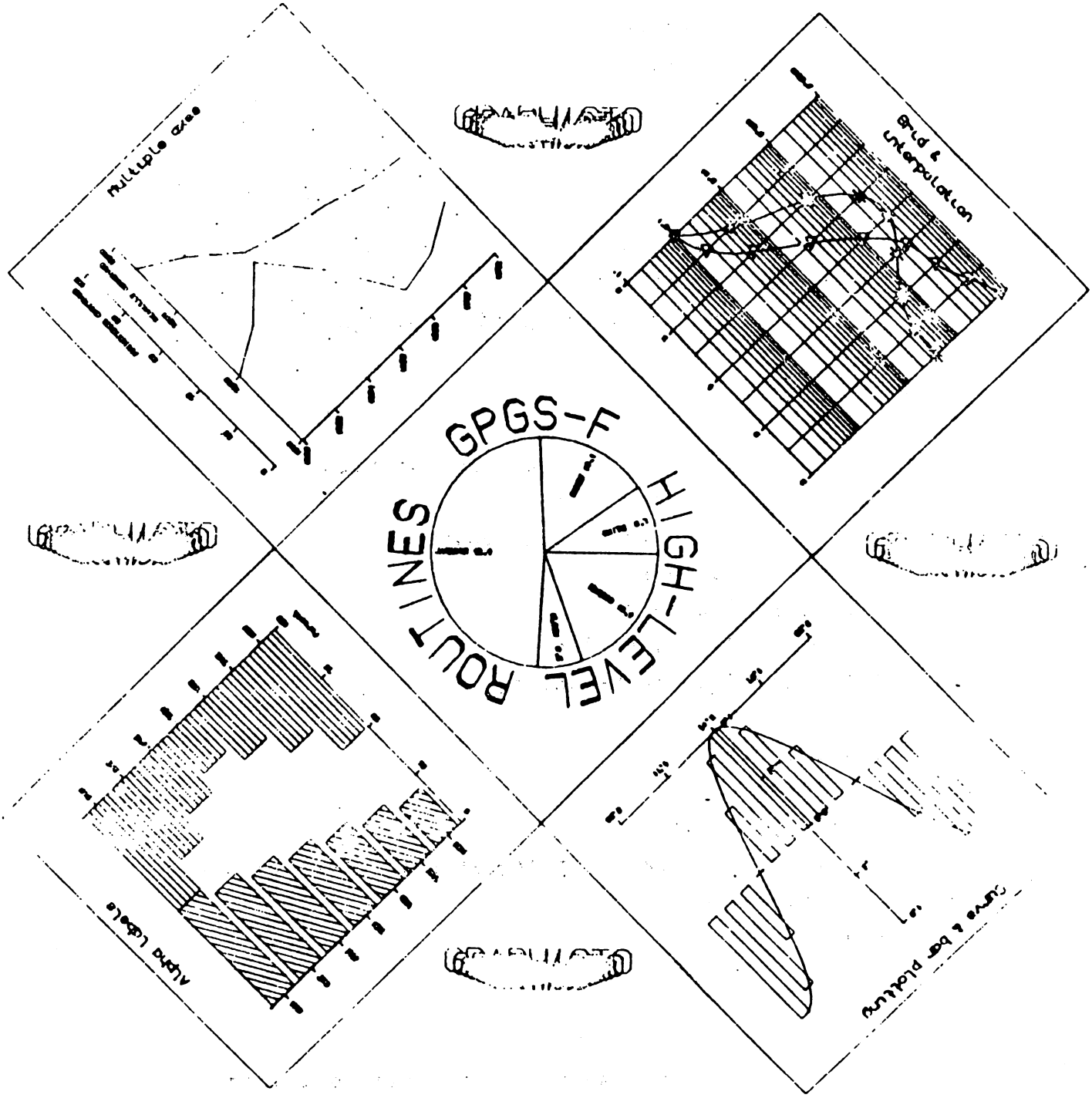
- Table of values in x- and y or functions
- Simple connected points
- Interpolated curve through points
- Markers at points
- 'Undefined' points
- Automatic data indexing
- Automatic data incrementing

Page layout:

- Centered heading
- Positioning of dataplotting area in users window
- Bar and curve legends
- Frame

Miscellaneous:

- 'Best-fit' range computation of data contained in array or function.
- 'Best-fit' label format computation.
- Conversation between coordinates in users window and in plotting coordinates.


GPGS-F
HIGH-LEVEL ROUTINES

Error Reporting:

- The routines in the GRAPHISTO system uses the error system of GPGS-F. This includes parameter checking and print of routine number, where error occured. Also the number of calls made to GRAPHISTO routines and wrong parameters are printed. This makes it easier for the user to find place and reason of error.

10. 3-DIMENSIONAL SURFACE PLOTTING

Newly developed by the graphic group at RUNIT is a package called SURRENDER 4 (SURface RENDERing) that is presented to the user as a means for visualizing bivariate surfaces on some graphic device. This package uses GPGS-F basic routines for drawing and GRAPHISTO routines for axes drawing and curve smoothing.

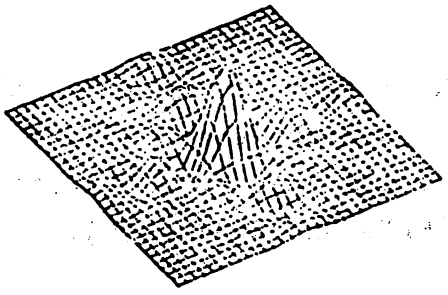
Base for all SURRENDER plotting is a rectangular x-y grid (matrix) with z-values in each node. A surface with M grid points in x-direction and N grid in y-direction will be stored in a Fortran DIMENSION ARRAY (N,M).

This grid may be rendered as a 3-D perspective plot of the grid (Isolines for x and y) with hidden lines removed or as a 2-D contour map (Isolines for z). Also other usefull facilities like drawing axes, marking points etc. are available and will be further explained later.

The package is built up much the same way as GRAPHISTO with some routines that makes a complete plot in one call, and others to add features for a more sophisticated plot.

Example of minimum effort plot:

```
      DIMENSION IWORK(200),VP(4),ZMAT(31,31),IOPT(3)
      DATA IDEV/8/,VP/0.0,0.3,0.0,0.3/
      DATA IBLUE/20/,IRED/60/
C
C  COMPUTE THE FUNCTION 'SIN(X)*SIN(Y)/(X*Y)'
C
      DO 1000 IY=-15,15
         Y=FLOAT(IY)
         SINYY=1.0
      IF (IY.NE.0) SINYY=SIN(Y)/Y
      DO 1000 IX=-15,15
         X=FLOAT(IX)
         SINXX=1.0
      IF (IX.NE.0) SINXX=SIN(X)/X
         ZMAT(IX+16,IY+16)=10.0*SINXX*SINYY
 1000 CONTINUE
C
C  MAKE MINIMUM EFFORT PLOT
C
         CALL NITDEV(IDEV)
         CALL BGNPIC(1)
         CALL PLOMA3(ZMAT,31,31,-15.,15.,-15.,15.,IWORK,200)
         CALL ENDPIC
```

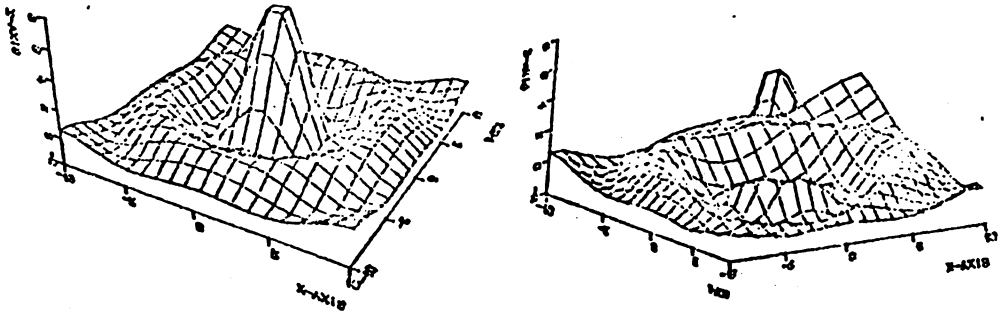


By default the hidden lines will be removed. To do this the system uses a working array to be supplied by user. As the needed size of this array is dependant upon the number of points to plot, this method gives no restrictions about number of points.

Setting focal point and eye position. The surface may be seen from any point in space and some routines are used to set this point either using cartesian or spherical coordinate system. The viewing may be either axonometric or perspeetive.

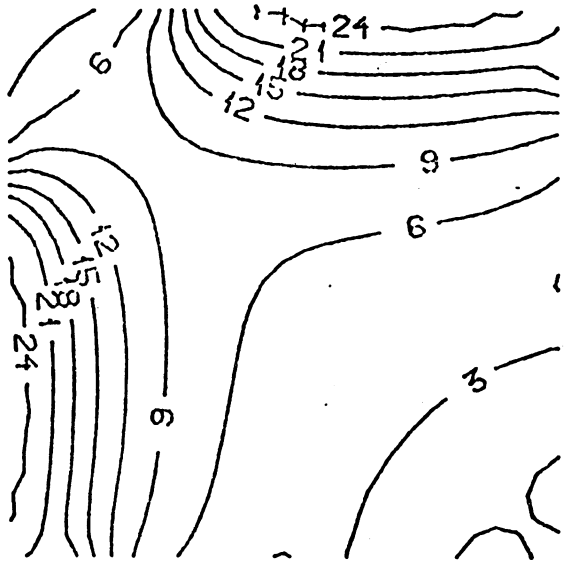
Adding axes to the plot.

Axes may be added by a call to a single routine giving standard axis annotation or by several calls to GRAPHISTO Axes routines for special labels and format.



Contour plots.

Contour plots consists of isolines in the z-direction. The range and number of contour lines may be given, and they may be added to the perspective plot or plotted as a separate 2-dimensional plot.



Default contour plot.

The SURRENDER package is not yet available on HP-3000, but this can easily be done on user requests.

11. USING GPGS-F ON HP-3000.

The HP-3000 version of GPGS-F is kept as near up to the standard version as possible and only a few routines has been changed from the users point of view. As seen from the foregoing text the system contains of many logical parts like the basic routines, a device driver, a plotting package and so on. 2 USL-files are used to contain the system and these are segmented into a number of segments each containing a logical part of GPGS-F.

One USL-file contains the basic routines and the most common used drivers. This is copied to users own USL with FCOPY.

The other contains the rest of GPGS-F, and wanted segments are copied using the SEGMENTER. This is done to save disc space for the user and preparation time.

A RL-file must be referenced at preparation time. This file contains dymmy entry-points for drivers and buffer-filesystem when not in use.

A complete set up for running GPGS-F on HP-3000:

```
: COMMENT copy basic part of GPGS-F
: RUN FCOPY.PUB.SYS
  FROM = GPGS.GPGS; TO = USERUSL; NEW EXIT
: COMMENT compile your program
: FORTRAN USERSOOR, USERUSL
: PREP USERUSL, USERPROG; RL = GPGSRL.GPGS
: RUN USERPROG
```

Aproximate segment sizes are as follows (decimal)

Basic drawing routines:	8400 (6 segments)
Incore buffer system:	2000
GRAPHISTO	9100 (3 segments)

Drivers:

Pseudo and file	400 each
Tektronix screens	2500 each
" plotter 4662	3100
Hewlet Packard 7221	3900
" 2648	3500

12. ERROR HANDELING

Inproper or inconsistent use of GPGS-F is possible and requires informative error messages.

GPGS-F provides the following informations in an error message:

1.	Error number	- indicates the type of error which occured
2.	GPGS subroutine no.	- identifics the routine that was called causing the error
3.	Parameter no.	- indicates which parameter, if any, caused the error
4.	Severity code	- indicates which action GPGS-F will take
5.	Call no	- indicates number of calls to GPGS since the program started
6.	Argument value	- value of the parameter in error, if any.

A GPGS-F user can instead of using the standard error handeling code his own error routine and in this way prevent error termination. This is done by dynamical change of eventual arguments with bad values.

PROGRAM USED TO CREATE FIGURES ON TITLE PAGE

```
      PROGRAM ROTBODY
C
C     MAIN PROGRAM FOR GENERATION AND PERSPECTIVE VIEWING
C     OF A ROTATION BODY
C
      DIMENSION LIY(100),BUFF(100,2),EYE(3)
C
C     SELECT TYPE OF GRAPPIC DEVICE (ENTER DEVICE CODE)
C
      DISPLAY 'DEVICE NO.'
      ACCEPT ID
C
C     INITIATE GPGS-F WITH SELECTED DEVICE
C
      CALL INITDEV (ID)
C
C     GENERATE THE CONTOUR TO BE ROTATED CREATING THE ROTATION BODY
C
      CALL BGNPIC(1)
      CALL GENPIC(BUFF,LIY)
      CALL ENDPIC
      IPIC = 1
   10 CONTINUE
C
C     SELECT EYEPOINT FOR PERSPECTIVE VIEW
C
      DISPLAY 'EYEPOINT'
      ACCEPT EYE
C
C     STACK CURRENT TRANSFORMATION FOR LATER RESTORE
C
      CALL BGNTRN
      IF (EYE(1).GE.9999.) GO TO 100
      IF (ABS(EYE(1)+EYE(2)+EYE(3)).GT.0.01)
     1   CALL PERS (EYE(1),EYE(2),EYE(3))
C
C     CLEAR DRAWING SURFACE
C
      CALL CLRDEV(ID,0)
C
C     INITIATE NEW PICTURE SEGMENT
C
      IPIC = IPIC + 1
      CALL BGNPIC(IPIC)
C
C     DRAW HORIZONTAL CONTOUR LINES
C
      CALL HORCNT(BUFF,LIY)
C
C     DRAW VERTICAL CONTOUR LINES
C
      CALL VERCNT(BUFF,LIY)
      CALL ENDPIC
C
C     POP LATEST STACKED TRANSFORMATION
C
      CALL ENDTRN
      GO TO 10
  100 CONTINUE
C
C     END  -  TERMINATE GPGS-F
C
      CALL RLSDEV(ID)
      STOP
      END
```

```
C
C
      SUBROUTINE GENPIC (BUFF,LTY)
C
C     SUBROUTINE TO GENERATE A CONTOUR CONSISTING OF
C     UP TO 99 STRAIGHT LINES (VISIBLE OR INVISIBLE)
C
      DIMENSION LTY(100),BUFF(100,2)
      DATA IL/2HL /,IP/2HP /,IS/2HS /
C
C     THE FIRST INPUT POINT IS USED AS A POINT ON THE VERTICAL
C     ROTATION AXIS
C
      CALL CURWIN (ICH,DX,DY)
C
C     THE COORDINATE SYSTEM IS TRANSLATED TO ENABLE DRAWING
C     RELATIVE TO AN ORIGIN IN THE FIRST POINT ON THE CONTOUR
C
      CALL XLAT (DX,DY)
C
C     MARK FIRST POINT WITH A CROSS (MARKER NO 4)
C
      CALL LINE (0.,0.,0)
      CALL MARKER (4)
      BUFF(1,1) = 0.
      BUFF(1,2) = 0.
      LTY(1) = 0
C
C     GENERATE THE OTHER POINTS ON THE CONTOUR,
C       END OF CONTOUR IS INDICATED BY ENTERING 'S' WITH THE POINT,
C       BLANK LINE TO POINT WITH 'P'
C
      DO 10 I = 2,100
      CALL CURWIN (ICH,X,Y)
      IF (ICH.EQ.IS) GO TO 20
      LINTYP = 1
      IF (ICH.EQ.IP) LINTYP = 0
      X = X - DX
      Y = Y - DY
C
C     VERIFY INPUT BY DRAWING
C
      CALL LINE (X,Y,LINTYP)
      BUFF(I,1) = X
      BUFF(I,2) = Y
      LTY(I) = LINTYP
   10 CONTINUE
      RETURN
   20 CONTINUE
C
C     LINETYPE -1 INDICATES END OF DATA ON CONTOUR
C
      LTY(I) = -1
      RETURN
      END %
```

```
      SUBROUTINE GETWIN (ICHAR,XPOS,YPOS)
C
C     GETWIN RETURNS A COORDINATE POINT AND A CHARACTER
C     INPUT BY MEANS OF THE GRAPHIC DEVICE'S POSITION INPUT TOOL
C
C     PARAMETERS:
C          OUTPUT:  ICHAR - TYPED CHAR (A1 FORM.)
C                   XPOS  - X COORDINATE OF INPUT POINT
C                   YPOS  - Y COORDINATE OF INPUT POINT
C
      DIMENSION IED(2),FAR(2)
      DIMENSION W(6),V(6)
      DATA IED /201,-1/
C
C     FETCH CURRENT DEVICE NO., WINDOW AND VIEWPORT FROM GPGS-P
C
      CALL DATDNO (ID)
      CALL DATWIN (W)
      CALL DATVP (V)
C
C     SET UP POSITIONING TOOL FOR ACTIVE DEVICE
C     AND WAIT FOR INPUT FROM TERMINAL USER
C
      I = INWAIT (-1.,IED,ICHAR,1,FAR,2)
C
C     CONVERT FROM DEVICE TO USER COORDINATES
C
      XPOS = W(1) + FAR(1)*(W(2)-W(1))/(V(2)-V(1))
      YPOS = W(3) + FAR(2)*(W(4)-W(3))/(V(4)-V(3))
      IF (ID.NE.11.AND.ID.NE.12) RETURN
C
C     IF DEVICE IS INTERACTIVE PLOTTER, NO CHARACTER CAN BE ENTERED
C     FROM THE PLOTTER DIRECTLY -
C     CHARACTER IS THE READ FROM THE ATTACED TERMINAL
C
      CALL WRITOL (2,0,0,0.,0)
    1 FORMAT (" ENTER CHARACTER:")
      WRITE (6,1)
    2 FORMAT (A1)
      READ (5,2) ICHAR
      RETURN
      END
```

```
C
C
      SUBROUTINE VERCNT (BUFF,LTY)
C
C     SUBROUTINE TO DRAW THE VERTICAL CONTOUR
C     IN 35 POSITIONS ROTATED AROUND THE ROTATION AXIS
C
      DIMENSION LTY(100),BUFF(100,2)
C
C     STACK CURRENT TRANSFORMATION
C
      CALL BGNTRN
      DO 100 J = 1,35
C
C     DRAW CONTOUR IN CURRENT COORDINATE SYSTEM
C
      DO 10 I = 1,100
      LINTYP = LTY(I)
      IF (LINTYP.LT.0) GO TO 20
      CALL LINE (BUFF(I,1),BUFF(I,2),LINTYP)
   10 CONTINUE
   20 CONTINUE
C
C     ROTATE COORDINATE SYSTEM 10 DEGS. AROUND Y-AXIS (HOR. AXIS)
C
      CALL ROTAD (10.,2)
  100 CONTINUE
C
C     POP LATEST STACKED TRANSFORMATION
C
      CALL ENDTRN
      RETURN
      END
C
C
      SUBROUTINE HORCNT (BUFF,LTY)
C
C     SUBROUTINE TO DRAW HORIZONTAL CONTOUR LINES (CIRCLES)
C
      DIMENSION LTY(100),BUFF(100,2)
C
C     LOOP OVER ALL POINTS ON THE VERTICAL CONTOUR
C
      DO 100 I = 1,100
      IF (LTY(I).LT.0) RETURN
      X = BUFF(I,1)
      Y = BUFF(I,2)
      IF (ABS(X).LT.0.001) GO TO 100
C
C     DRAW HORIZONTAL CIRCLES THROUGH A POINT ON THE VERTICAL
C     CONTOUR LINE WITH CENTRE IN THE ROTATION AXIS
C
      CALL LINE (X,Y,0)
      CALL CIRD3 (0.,Y,0.,X,Y,X,360.,1)
  100 CONTINUE
      RETURN
      END
```

REFERENCES.

1) M. Zachrisen : GPGS-F User's Guide, 3rd edition TAPIR Forlag 1978.

2) M. Zachrisen : GPGS-F Cource Manual, RUNIT 1977.

3) M. Zachrisen : GRAPHISTO, a subroutine package for general graph and histogram plotting with GPGS-F. RUNIT report STF14 A78019.

4) M. Zachrisen : PRELIMINARY description of SURRENDER - a subroutine packet for rendering bivariate surfaces.