3-D GRAPHICS IN APL: USER PERSPECTIVE ©

by

Uday G. Gujar

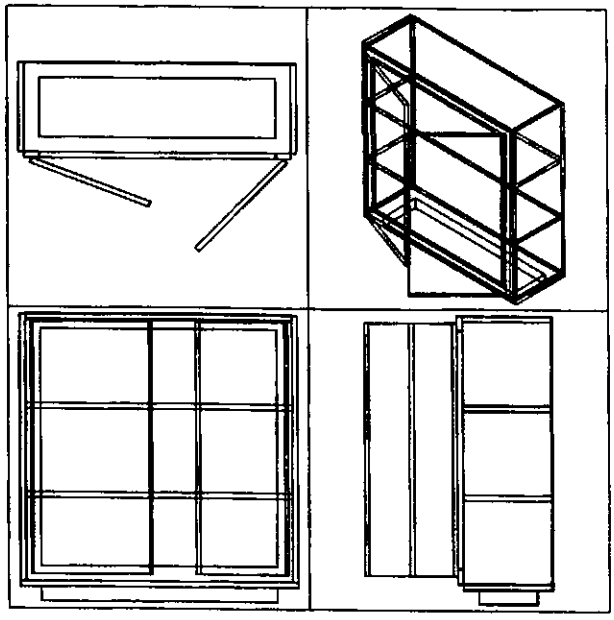
TR84-025, July 1984
Revised November 1988

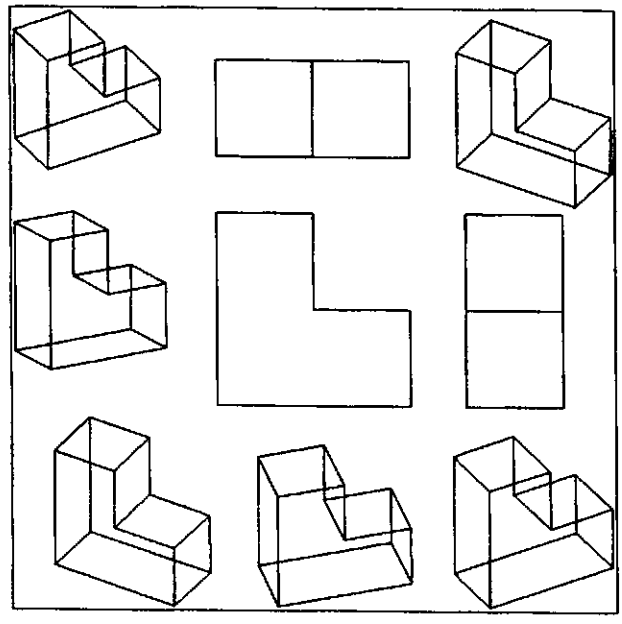
Feedback from users would be appreciated.
Please direct your comments to author

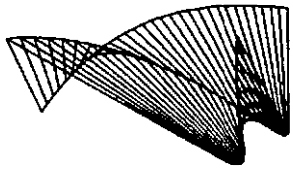
FACULTY OF COMPUTER SCIENCE

UNIVERSITY OF NEW BRUNSWICK

FREDERICTON, N.B. CANADA E3B 5A3



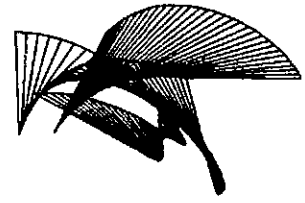


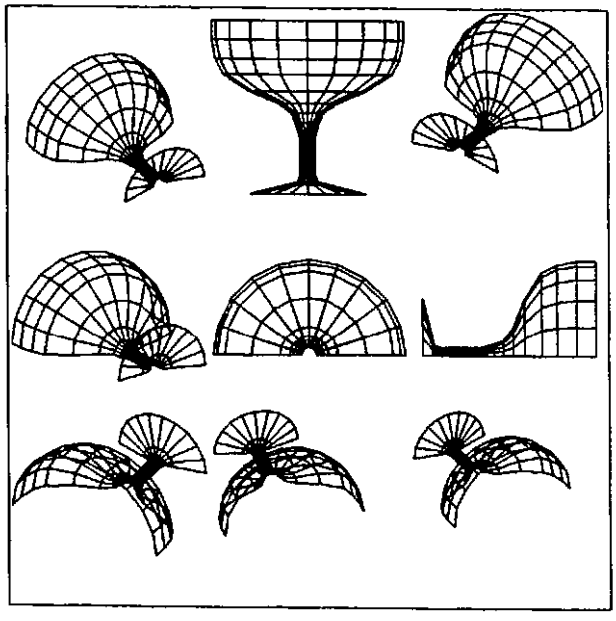


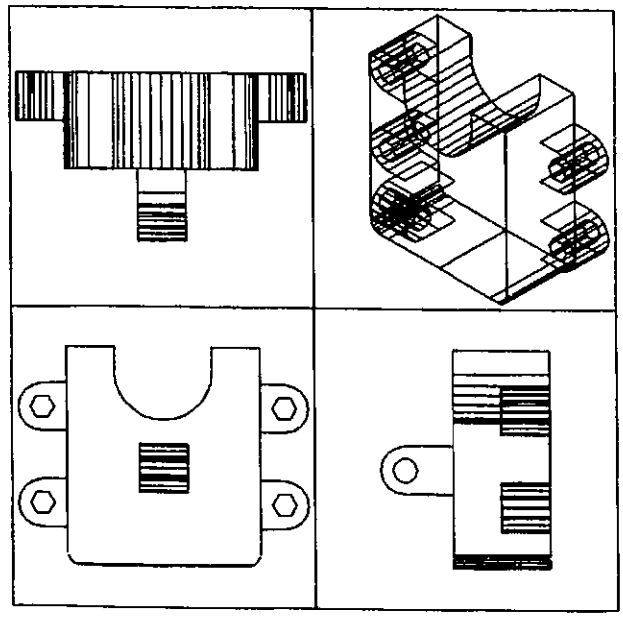
3-D GRAPHICS IN APL: USER PERSPECTIVE

by

UDAY G. GUJAR







3-D GRAPHICS IN APL: USER PERSPECTIVE

Uday G. Gujar
School of Comuter Science
University of New Brunswick
FREDERICTON, N.B. E3B 5A3

The purpose of this document is to explain how a user can generate and display three dimensional objects in APL. We will not deal with how the system is put together. The top-down approach will be used.

1.0 EXAMPLES:

The following APL dialogue illustrates the capabilities of the system. The comments have been added in lower case letters wherever necessary.

```
      )LOAD 22 G3DSA          ← load basic 3-D functions
      )COPY 22 G3DAID         ← copy examples
      )COPY 22 UNBKD          ← copy GRAPHPAK device drivers.

      FOURVIEWS PART9
(null area i.e. default; see Figure 1)

      FOURVIEWS GOB5
(area 5 5; see Figure 2)

      PART3 SHOW ¯9 ¯5 9
(area 3 3; see Figure 3)

      PART3 SHOW 99 99 99
(area 3.75 4; see Figure 4)

      ERASE
      AREA 1.5 1.6 4.5 4.6
      SETPLT ¯3 ¯4 1.5 1
      GOB3 DISPLAY 3 15 10
      AREA 4.5 1.6 7.5 4.6
      SETPLT 0.5 ¯2 11 8
      PART1 DISPLAY ''
      AREA 1.5 4.6 7.5 9.6
      SETPLT ¯5 ¯5 5 5
      NOWPLT FRONT AXLE
      VIEW
        (Figure 5 was generated at this point)
```

```
      FOURVIEWS OBJECTS1
(area 5 5; see Figure 6)

      FOURVIEWS WARP7
(area 3 3; see Figure 7)

      AREA 4.5 1.3 7.5 4.3
      SETPLT ¯2.1 ¯2.4 2.1 3
      NOWPLT ISO 10 REVY ¯2 4 0 , ¯1 2 0 , ¯1 0 0
      VIEW
        (Figure 8 was generated at this point)

      TRI← 5 4 ρ 0 ¯1 0 0 , 1 0 1 0 , 1 1 0 0 , 1 0 ¯1 0 , 1 ¯1 0 0
      TRIT← 5 4 ρ 0 0 0 ¯1
      B←TRI CAT TRIT CAT TRI JOIN TRIT
      FOURVIEWS B
(area 4 4; see Figure 9)

      B←B CAT(B FLIPX 1.5) CAT(B FLIPY ¯1.5) CAT B FLIPZ 0.2
      B←B CAT(B FLIP 1 1.2 1.4) CAT B FLIP- 1 1.2 1.4
      FOURVIEWS B
(area 4 4; see Figure 10)
```

2.0 ORGANIZATION:

There are two workspaces: G3D and G3DAID.

G3D contains all the basic and auxiliary functions; G3DAID contains several examples.

Some of the functions prompt for user input; the user can always give a null input to invoke the predefined input values. In fact, on first attempts it is desirable to give null input to get familiar with the system.

Some of the parameters appear frequently while using various functions. These are defined below:

OBJ - An n by 4 array representing n points along with the control. OBJ[;1] defines the control (0 means move, 1 means draw a line) while OBJ[;2 3 4] give the x,y,z coordines of the points. OBJ1, OBJ2, OBJN are the other names which represent the same data structure.

Thus,

OBJ←4 4 ρ 0 1 1 0 , 1 2 1 0, 1 1.5 1.5 0, 1 1 1 0

would define a triangle joining the points (1,1,0), (2,1,0) and (1.5, 1.5, 0).

ABCLMN - A 3 or 6 element vector defining the position of the eye and the projection plane; if a 3 element vector, it is considered as a 6 element vector with first 3 elements as zeros. First 3 elements represent the x,y,z coordinates of a point ABC while the next 3 elements represent the x,y,z coordinates of a point LMN. The eye is assumed to be situated at LMN and pointing at ABC along the line from LMN to ABC. The projection plane is perpendicular to this line of sight and passes through ABC.

CXY - An n by 3 array representing n points, in the image space, along with the control. First column, i.e. CXY [;1], defines the control (0 means move, 1 means draw a line) while CXY [;2 3] give the x,y coordinates of the two dimensional image as projected by the geometry specified in ABCLMN.

From the user's point of view, there are some global variables of interest. The experienced user's may want to experiment with the settings of these global variables; the novice user may skip this part in the beginning. Note that settings of these variables will affect all the displays that are generated subsequently.

The global variable OBLIQUE contains a character vector which sets the orientation of the projection plane which is first rotated about the X-axis by an angle (±OBLIQUE) [1] followed by about the Y-axis by an angle (±OBLIQUE) [2] thus giving an oblique projection. The default for OBLIQUE is a null which indicates the projection plane remains perpendicular to the line of sight. See Figure 11 for an example.

The global variable CONNECT controls whether the corresponding points on the contour lines forming an object are connected or not. The default is 'ON' which means the corresponding opints on the contour lines are connected; this may be set to 'OFF'. See Figure 12 for an example.

3.0 G3D:

This workspace contains the following groups:

```
BASIC3DGRP : 3-D viewing, modelling and transformation
TEXTGRP    : text generation
BASICGRP   : 2-D graphics functions
AUXIGRP    : some useful auxiliary functions
UGGGRP     : Interface to GRAPHPAK
```

From user's point of view, BASIC3DGRP should be of prime importance. All these groups are described in the following subsections.

3.1 BASIC3DGRP:

The functions in this group could be divided into the following categories:

a) viewing functions

b) view aid functions

c) transformation functions

d) modelling functions

e) special views functions and

f) animation functions

3.1.1 Viewing functions:

The following functions are available

```
OBJ SHOW        ABCLMN
    FOURVIEWS   OBJ
    NINEVIEWS   OBJ
    SEVENVIEWS  OBJ
OBJ DISPLAY     ABCLMN
```

SHOW asks for the size of the area and displays the object defined in OBJ as seen from the geometry given in ABCLMN. It automatically scales the image to fit in the desired area. If ABCLMN is specified as null, SHOW assumes the default values for it.

FOURVIEWS asks for the size of the area and displays plan, elevation, side and isometric views of the object defined in OBJ.

NINEVIEWS asks for a three element numeric vector, asks for the size of the area and generates nine different views. The three element vector is used to generate the nine different eye positions.

The size of the area for SHOW, FOURVIEWS and NINEVIEWS can be a four element véctor specifying the x-y coordinates, in inches, of the lower left and upper right corners of the rectangular area; if it is a two element vector, the lower left corner is assumed at (0,0). If the lower left corner is at (0,0) the screen is erased before the views are shown.

SEVENVIEWS always erases the screen, and generates top, bottom, right, left, front, back and isometric views in a predefined area on the screen.

DISPLAY works in exactly the same fashion as SHOW except that it neither erases the screen, nor asks for the size of the area, nor does it

scale the image. All these things can be independently done by the user (see Section 3.2). If ABCLMN is specified as null, it gets a predefined default value.

Two other functions

```
AREASIZE
AREAFORVIEWS N
```

are used by SHOW, FOURVIEWS and NINEVIEWS; these are internal functions.

3.1.2 View Aid Functions:

The functions in this category are:

```
TM   ← VIEWM  ABCLMN
OBJN ← OBJ  NEWP   TM
CXY  ← OBJ  VIEWP  ABCLMN
OBJN ← OBJ  AT XYZ
```

VIEWM is the heart of the system. It accepts the position of the eye and the projection plane in ABCLMN and generates a 4 by 4 generalized homogeneous transformation matrix TM which, when applied to the x,y,z coordinates of a point in space, gives the desired view.

NEWP applies generalized homogeneous transformation matrix TM to the object specified in OBJ to produce the transformed 3-D coordinates (along with controls) in OBJN.

VIEWP uses VIEWM and NEWP to generate an n by 3 matrix of n points along with control codes in CXY. CXY[;1] contains the control (0 means move; 1 means draw a line) while CXY[;2 3] contain the x,y coordinates of the two dimensional image as projected by the geometry specified in ABCLMN.

AT moves the whole object by x,y,z coordinates specified in XYZ. Note that while AT is a convenient function to locate the object at a new position, it would be inefficient to use it in combination with other transformations. Whenever multiple transformations are to be applied to an

object, it is more efficient to combine them into a generalized matrix (see Section 3.1.3) and then apply the combined matrix to the object.

3.1.3 Transformation Functions:

The following functions generate the various elements of the transformation matrix:

```
TM ← RX THETA
TM ← RY THETA
TM ← RZ THETA
TM ← SLOC  XYZ
TM ← SALL S
TM ← TR XYZ
TM ← PRX P
TM ← PRY Q
TM ← PRZ R
TM ← PR PQR
TM ← TM1 THEN TM2
```

RX, RY, RZ generate the rotations about X,Y and Z axes respectively by the specified angle THETA (in radians).

SLOC takes a 3 element vector, XYZ, as an argument which specifies the local scalings on x,y and z coordinates respectively. SALL scales all the axes by the same amount specified in S as a single number. Thus, S > 1 gives magnification, while S < 1 gives compression of the object.

TR generates a translation of the origin to the point specified in XYZ, a 3 element vector representing the x,y,z coordinates.

PRX gives the projection on the x=0 plane as viewed from the (P,0,0) point. PRY gives the projection on the Y=0 plane as viewed from the (0,Q,0) point. PRZ gives the projection on the Z=0 plane as viewed from the (0,0,R) point. PR sets, via PQR, the last elements of the first three rows of the homogeneous matrix; these are the elements which give rise to perspective transformation. First three elements of PQR are used.

THEN combines effects of transformation matrix TM1 followed by TM2.

All these functions generate a 4 by 4 transformation matrix TM. These matrices can be concatenated to produce combined effect such as:

```
TM ← (TR 5 5 5) THEN (RY ÷ 2) THEN PRZ 10
```

or

```
TM ← (TR ABC) THEN (RY -PH) THEN (RX TH) THEN (PRZ R)
```

3.1.4 Modelling Functions

One of the problems in producing 3-D images is the difficulty of generating x,y,z coordinates. The modelling functions are defined to somewhat ease this problem. These are

```
OBJN ← OBJ1     CAT     OBJ2
OBJN ← OBJ      FLIPX   X
OBJN ← OBJ      FLIPY   Y
OBJN ← OBJ      FLIPZ   Z
OBJN ← OBJ      FLIP    XYZ
OBJN ← NFT      REVX    PTS
OBJN ← NFT      REVY    PTS
OBJN ← NFT      REVZ    PTS
OBJN ← OBJ1     JOIN    OBJ2
OBJN ← CONNECT OBJ
OBJ  ← OBJECT
```

The terms OBJ, OBJ1, OBJ2 and OBJN represent the n by 4 arrays defining the object as mentioned in Section 2.0.

CAT catenates the two objects to form a new object. Note that each object is a line drawing, hence the catenation is also a line drawing which contains all the lines from both the objects.

FLIPX gives the mirror image of the object OBJ with mirror placed on and perpendicular to the x axis at a distance X. Similarly, FLIPY and FLIPZ give the mirror images with mirrors placed on and perpendicular to the y and z axes at distances Y and Z respectively . FLIP combines the effects of all the three routines; here XYZ is a 3 element vector, x-coordinates are mirrored as if the mirror is at XYZ[1] on the x-axis,

y-coordinates are mirrored as if the mirror is at XYZ[2] on the y-axis and z-coordinates are mirrored as if the mirror is at XYZ[3].

REVX, REVY and REVZ generate the surfaces of revolutions around x, y and z axes respectively. NFT is a scalar or a three element vector with the following meaning

NFT[1] - number of points to be generated per revolution

NFT[2] - starting angle, in radians, from which revolution starts (default 0)

NFT[3] - ending angle, in radians, where the revolution ends (default 2π).

The array PTS gives the x,y,z coordinates of the points on a curve which is to be revolved around the specified axis. See the functions from GOBGRP (Section 4.1) for various examples.

JOIN and CONNECT connect multiple contours if the global variable CONNECT is 'ON', otherwise they simply return doing nothing. JOIN constructs OBJN such that it has the effect of drawing lines between corresponding points of OBJ1 and OBJ2. Thus, OBJN is composed of OBJ1[1;], OBJ2[1;],OBJ1[2;], OBJ2[2;],...., OBJ1[n;], OBJ2[n;] where n is the number of points in OBJ1 and OBJ2 and all the controls in OBJN for the OBJ1 components are zeroed (i.e. move) and all the controls in OBJN for the OBJ2 components are made one (i.e. line) by JOIN. CONNECT assumes that OBJ contains only the contours (each containing the same number of points) and generates connecting lines to pass through the corresponding points.

OBJECT enables the user to construct an OBJ in an interactive manner. Note that the interaction is not graphical; OBJECT simply prompts for control and x,y,z coordinates; typing HELP gives a simple helpful message.

3.1.5 Special Views Functions:

Some special views are frequently required. The following functions are provided for them.

```
CXY ← TOP OBJ
CXY ← BOTTOM OBJ
CXY ← RIGHT OBJ
CXY ← LEFT OBJ
CXY ← FRONT OBJ
CXY ← BACK OBJ
CXY ← ISO OBJ
```

The first 6 views are self explanatory; as the name suggest, they generate coordinates to produce top, bottom, right, left, front and back views, respectively, of the object specified in OBJ. ISO generates coordinates for an isometric view. See FOURVIEWS, SEVENVIEWS, NINEVIEWS (Section 3.1.1) for examples.

3.1.6 Animation Functions:

Some animation functions are:

```
OBSERVE  OBJ
MOVIE    OBJ
```

OBSERVE is an attempt to generate various views as eye moves from one location to another. As such, it uses a lot of computing resources; B-E-W-A-R-E!! It asks for the initial eye position, then for the incremental changes in x,y and z directions and finally for the number of snap shots. Then the specified number of snap shots are generated changing the eye position by specified increments for each snap shot; the screen is erased before generating a new snap shot. The communication speed limits the movie type of generation.

MOVIE works exactly like OBSERVE, except the snapshots are generated in a strip so that all the snapshots can be viewed at the same time. The

size of each snapshot is determined dynamically and depends upon the number of snapshots specified.

3.2 TEXTGRP:

The functions in this group enable the user to write text in the graphics area. These functions include WRITE, WRTΔLINES and WRTΔMAT.

The syntax of WRITE is:

XYH WRITE TEXT

Both XYH and TEXT may be vectors and/or matrices.

XYH[;1 2] - (X,Y) coordinates of a point where text should start.

XYH[;3] - Desired height of text; default .08".

- If positive, in inches; if negative, in |XYH[;3] Y-user units.

TEXT[I;] - Text that goes with XYH[I;]

The syntax of WRTΔLINES is:

XYHG WRTΔLINES TEXT

XYHG -- vector [X,Y,HEIGHT,GAP]

(X,Y) - point where first line of text is to start; user units.

HEIGHT - if +ve, in inches; if -ve in |XYHG[3] Y units. Default .08" .

G - Gap between lines as a factor of HEIGHT; default 1.7.

TEXT -- Multiline text with lines separated by TEXT[1].

The syntax of WRTΔMAT is:

XYHG WRTΔMAT TEXT

XYHG -- same as in WRTΔLINES

TEXT -- Matrix of text; number of rows gives the number of lines.

3.3 BASICGRP:

The functions in this group handle two dimensional (i.e. x,y) data and as such are used in the image space. The following are the main functions:

```
      AREA    SIZE
      SETPLT  SIZE
      NOWPLT  CXY
      SEE CXY
XY ← LOCATE
```

AREA defines the physical size of the screen, in inches, as given in SIZE; while SETPLT defines extents of the user coordinates in his own units. SIZE is up to a 4 element vector giving the minimum and maximum sizes for x and y directions. SIZE [1 3] are treated as x values and SIZE [2 4] are treated as y values; minimum and maximum may be specified in any order. The possible error of specifying the same values for minimum and maximum is trapped and reported.

For example consider the following two statements:

```
AREA      2   3   4   5
SETPLT   10  200 30  400
```

Here, the user has chosen the 2" square area bounded by lower left corner (2,3) and upper right corner (4,5) - all in inches. The user units are specified by SETPLT as 1"=10 on the x-axis and 1"=100 on the y-axis; SETPLT actually defines the (2",3") point as (10, 200) in user units and (4",5") point as (30, 400) in user units. The same effect would be achieved by

```
AREA      4   3   2   5
SETPLT   30  400 10  200
```

or

```
AREA      2   5   4   3
SETPLT   30  200 10  400
```

NOWPLT generates the line drawing specified in CXY which is considered as a set of triplets; CXY may be an array of any rank. Each triplet is treated as C,X,Y where C represents a control (0 means a move, 1 means a line) and X,Y represent the x-y coordinates of a point to which move is

made or a line is drawn from the current pen position. The current pen position is always remembered. x-y coordinates are specified in the user units.

These three functions are similar to the ones contained in [GUJA76, GUJA82, HARR84].

SEE invokes SETPLT to define the extents of the user units from CXY and then plots these values using NOWPLT. Thus one can use the function SEE to display data given in CXY.

LOCATE is a graphic input function which allows the user to position the cursor in the graphic area on the screen and then reads its position. It returns a two element vector in XY which represent X and Y coordinates, in the current user units, of the cursor position.

The other functions in this group are a housekeeping convenience.

```
R ← GETVALS
    SETVALS  R
    RESET
```

These either get or set some internal global variables; see listings if you are curious.

3.4 AUXIGRP:

The following auxiliary functions are contained in this group:

```
CXY ← N ARC  XYRFT
CXY ← N ARC2  XYRP1P2
TH  ← TANI  ΔXΔY
REP ← QQ  MSG
```

ARC generates N+1 connected points (i.e. N segments) on a circular arc given the centre, radius and starting and ending angles. ARC2 also generates N+1 connected points on a circular arc given the centre, radius and two points. See Figure 13 for the definition of the arguments.

TANI gives the $\tan^{-1}(\Delta y/\Delta x)$ in all the quadrants even when Δx is zero.

QQ prompts the user with the message in MSG and accepts the input on the same line; the characters typed by the user are returned in REP.

3.5 UGGGRP:

The functions in this group are the basic functions which are called directly or indirectly by the other functions. As such the user would not, normally, need to use them.

There are two functions in this group:

```
XY ← LCTRTN
     PLTRTN   CXY
```

LCTRTN is a niladic function which allows the user to position the cursor on the screen and then read its position. XY is a two element vector giving X and Y coordinates, in inches, of the cursor position.

PLTRTN is similar to NOWPLT in that it accepts CXY as a set of triplets, each triplet representing control (0 for move, 1 for line) and X and Y coordinates. The only difference is that X and Y coordinates are in inches.

4.0 G3DAID:

This is an accompanying workspace which contains several examples. This workspace is put together with duel purpose in mind. First, it gives several working models which can be immediately used by the users to grasp the capabilities of the system. Second, the user can look at the functions in this workspace as an aid to forming his own functions.

Following two examples will illustrate the ease of constructing 3-D objects.

```
      ∇R ← OUR
[1]   R ← 0 0 0, 2 2 0, 4 2 0, 4 4 0
[2]   R ← 15  REVY  R  ∇

      ∇R ← OUR1; RR
[1]   R ← 5 4 ρ 0 0 0 0, 1 2 0 0, 1 2 2 0, 1 0 2 0, 1 0 0 0
[2]   RR ← R  FLIPZ  ¯1
[3]   R ← R  CAT RR CAT R JOIN  RR
[4]   R ← R  CAT  R  FLIPX  3 ∇
```

Try these on a terminal.

Various examples are constructed in different groups:

GOBGRP: objects constructed with surface of revolution
PARTGRP: some 3-D objects
FURNGRP: generation of some furniture
TESTGRP: very simple test functions
WARPGRP: generation of warped objects
OBJECTSGRP: Collection of objects

The following subsections describe these groups. A pictorial dictionary of the objects from various groups is given in the Appendix.

4.1 GOBGRP:

```
OBJ ← GOBn   where 1 ≤ n ≤ 20
OBJ ← TIRE
OBJ ← ROCKET
OBJ ← AXLE
OBJ ← TOROID
```

Look at GOB4 or GOB6 to see how a cut glass can be generated using a surface of partial revolution. TOROID is an example where a circle is rotated around an axis to form a donut shape object. TIRE uses similar construction except that a circular arc is used instead of a full circle. GOB9 uses revolution as well as mirror imaging.

4.2 PARTGRP:

The functions in this group generate various three dimensional objects. All of the main functions start with the word PART.

```
OBJ ← PARTn    Where  1 ≤ n ≤ 15
```

These functions basically use CAT, FLIPx (where x = X,Y, or Z) JOIN and in some cases ARC or ARC2. For example, PART3 constructs a desk, PART9 and PART10 form mechanical parts, PART13 constructs a matrix of user specified objects through a user dialogue. Some support functions are also included such as CYL, SPCUBE, PARTnPAR (where n = 3,5,11,12) etc. Close examination of some of these functions should give some insight into constructing new three dimensional objects.

4.3 FURNGRP:

Several cupboards with shelves and doors are constructed in this group. These are:

OBJ ← CUPBOARDn, where $1 \leq n \leq 6$

These are constructed using subfunctions SHELF, BASE, DOORS, FRAME and PLANK; CUPBOARD4 uses UNIT1 and UNIT2 to form a wall unit with two symmetrical units besides a central one.

4.4 TESTGRP:

The functions

OBJ ← TESTn for $1 \leq n \leq 7$

are examples of how one can construct simple objects very easily. For example, TEST2 is a single line function invoking REVX with 4 data points.

4.5 WARPGRP:

The functions

OBJ ← WARPn for $1 \leq n \leq 9$

produce the results that are very interesting. As the name WARP suggests, the surfaces are warped and many a times are in fact not even recognizable;

though they are constructed from simple geometry. For example, WARP7 starts with three disconnected straight lines in space and generates an interesting looking pattern by rotating these by a small angle and translating them by a small amount; this process is repeated thirty times. WARP3 uses random points and therefore will generate a new pattern everytime. Do not attempt to visualize the objects (!), it is (almost) impossible.

4.6 OBJECTSGRP

Two functions are in this group, namely

OBJ ← OBJECTSn for n = 1, 2

These functions use functions from GOBGRP and PARTGRP; therefore these groups have to be copied into the workspace. These functions illustrate how various objects can be combined together. Figure 14 is produced by

FOURVIEWS OBJECTS2

5.0 CONCLUDING REMARKS:

It is hoped that this document will be useful for the users to generate projections of 3-D objects. The user can generate 3-D line drawings using various modelling functions that are supplied. These objects can be manipulated in various ways such as one can scale them, locate them, rotate them and so on. The viewing functions are very simple to use. For example, all the user needs to supply is the X-Y-Z coordinates of the eye position and the screen and the view is generated. Several auxiliary functions are provided to ease the data generation process. A graphic input function is provided to locate the cursor position. Several examples are included.

REFERENCES

FOLE82 Foley, J.D., and Van Dam, A., 'Fundamentals of Interactive Computer Graphics', Addison Wesley, 1982.

GUJA76 Gujar, U.G., 'A Device Independent Computer Plotting System', Proceedings ACM Symposium on Graphics Languages, Miami Beach, Computer Graphics, Vol. 10, No. 1, pp. 85-100, April 1976.

GUJA82 Gujar, U.G., 'Computer Plotting', Computer Centre, University of New Brunswick, Canada, 1972; ninth printing, Aug. 1982.

GUJA84 Gujar, U.G., 'Generation and Manipulation of Three Dimensional Objects', Annual APICS Computer Science Conference, University of New Brunswick, Fredericton, N.B., pp. 64-72, Nov. 1984.

GUJA87 Gujar, U.G., 'A Three Dimensional Wire Frame Graphics System', APL87 Conference Proceedings, Dallas, Texas, pp. 1-16, May 10-14, 1987.

HARR84 Harrigan, K., 'An Interactive Graphics Package in VSPC Fortran', M.Sc.(C.S.) thesis, School of Computer Science, University of New Brunswick, Canada, 1984.

NEWM79 Newman, W.M., and Sproull, R.F., 'Principles of Interactive Computer Graphics', Second Edition, McGraw-Hill, 1979.

ROGE76 Rogers, D.F., and Adams, J.L., 'Mathematical Elements for Computer Graphics', McGraw-Hill, 1976.

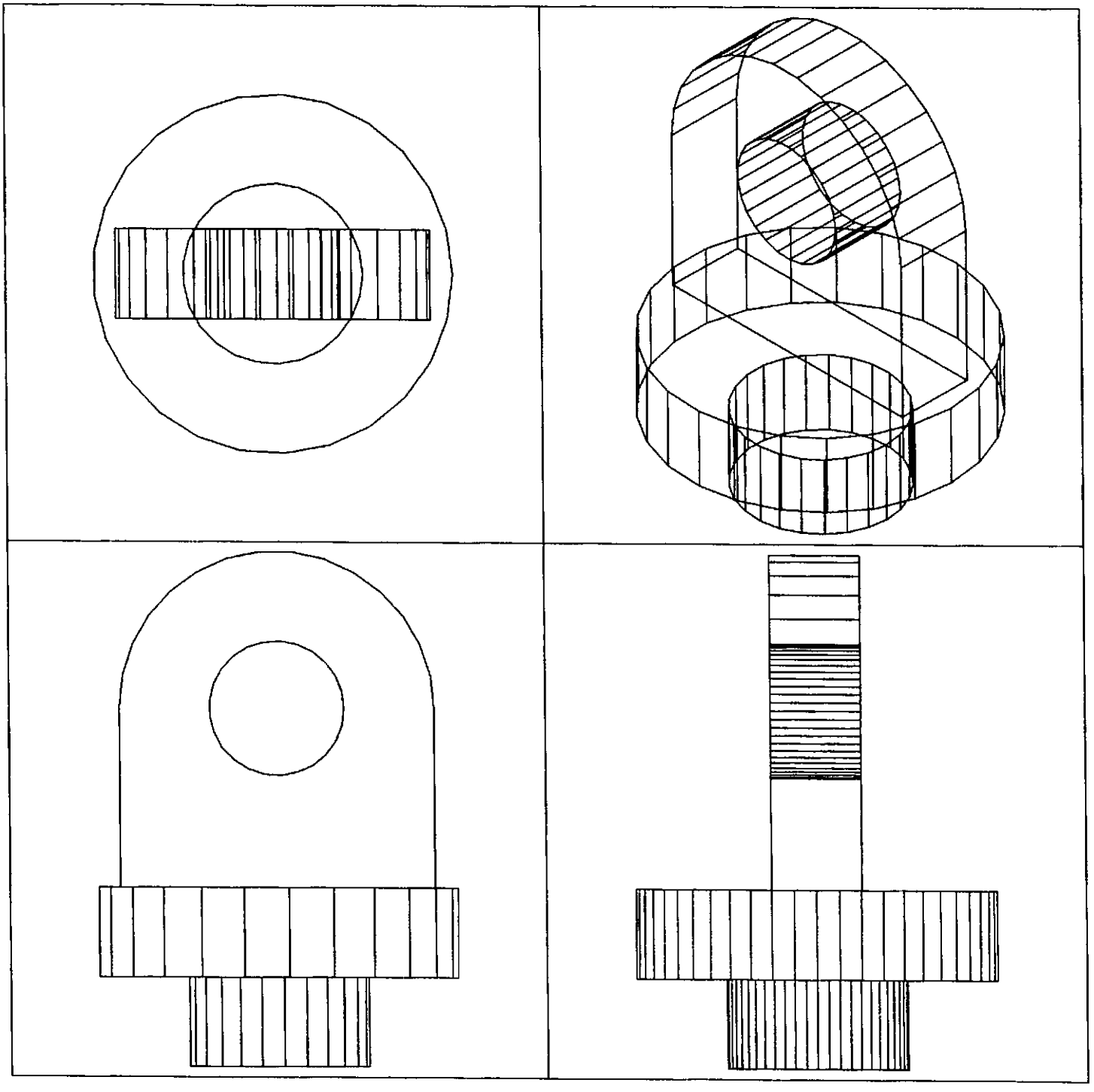


Fig. 1: FOURVIEWS PART9

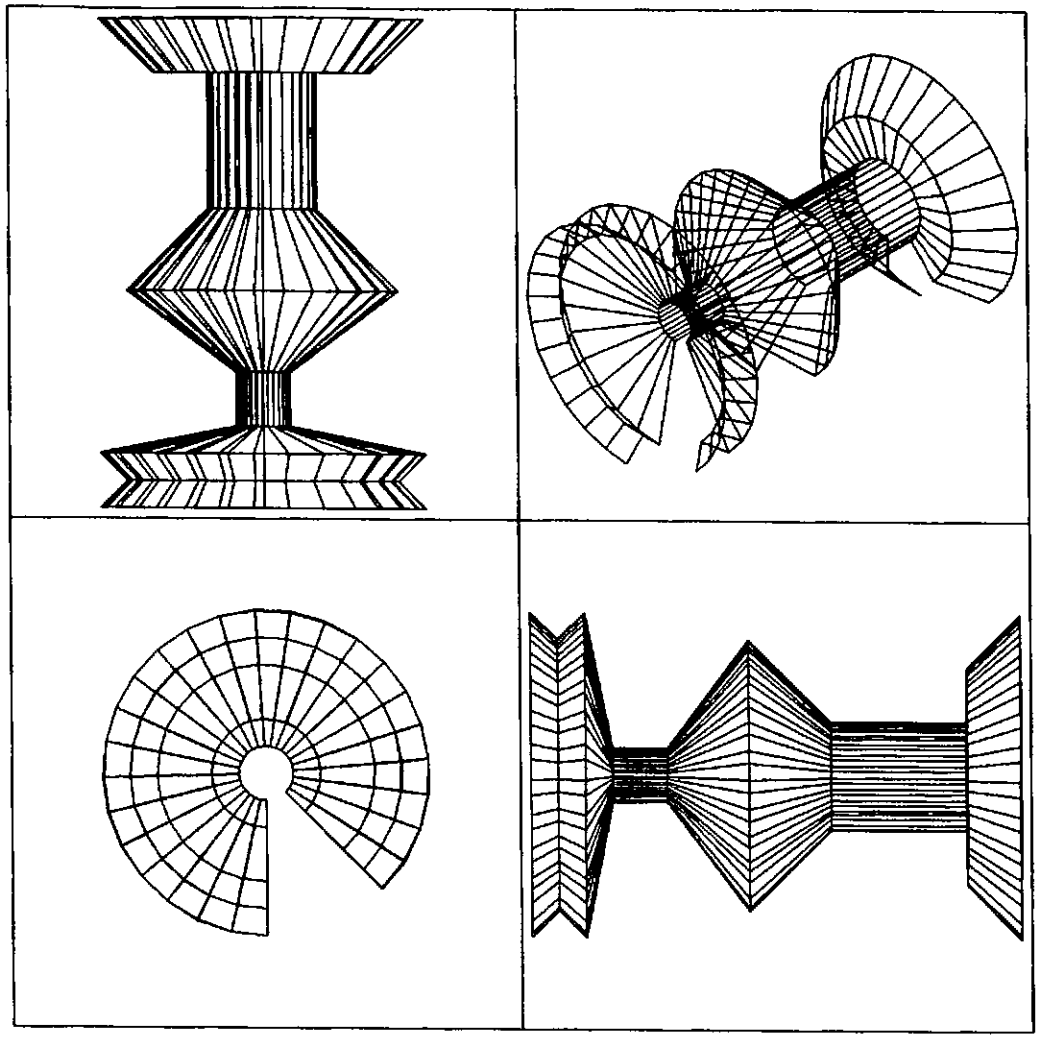


Fig. 2: FOURVIEWS GOB5

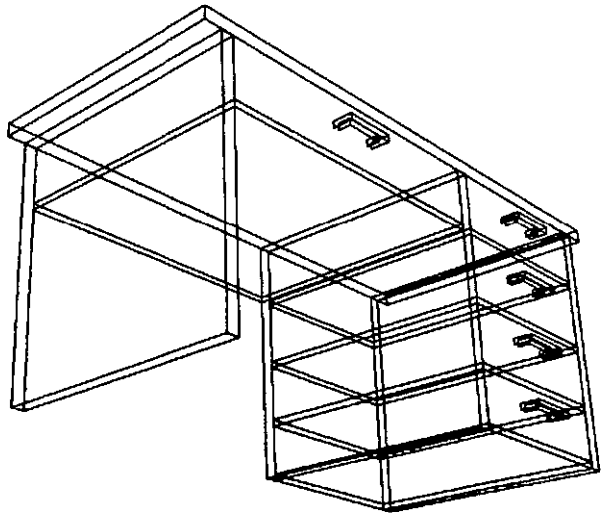


Fig. 3: PART3 SHOW -9 -5 9

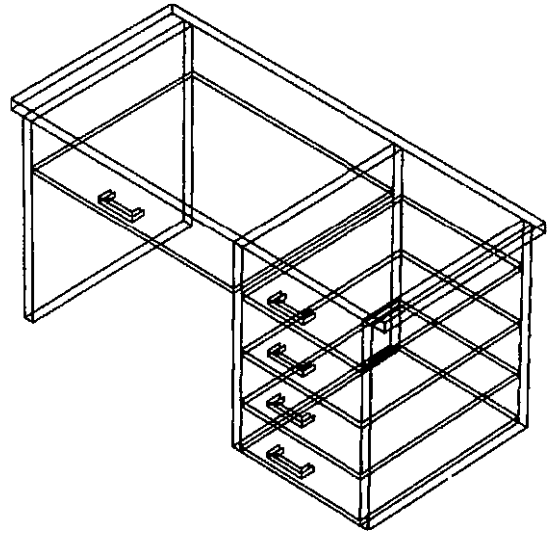


Fig. 4: PART3 SHOW 99 99 99

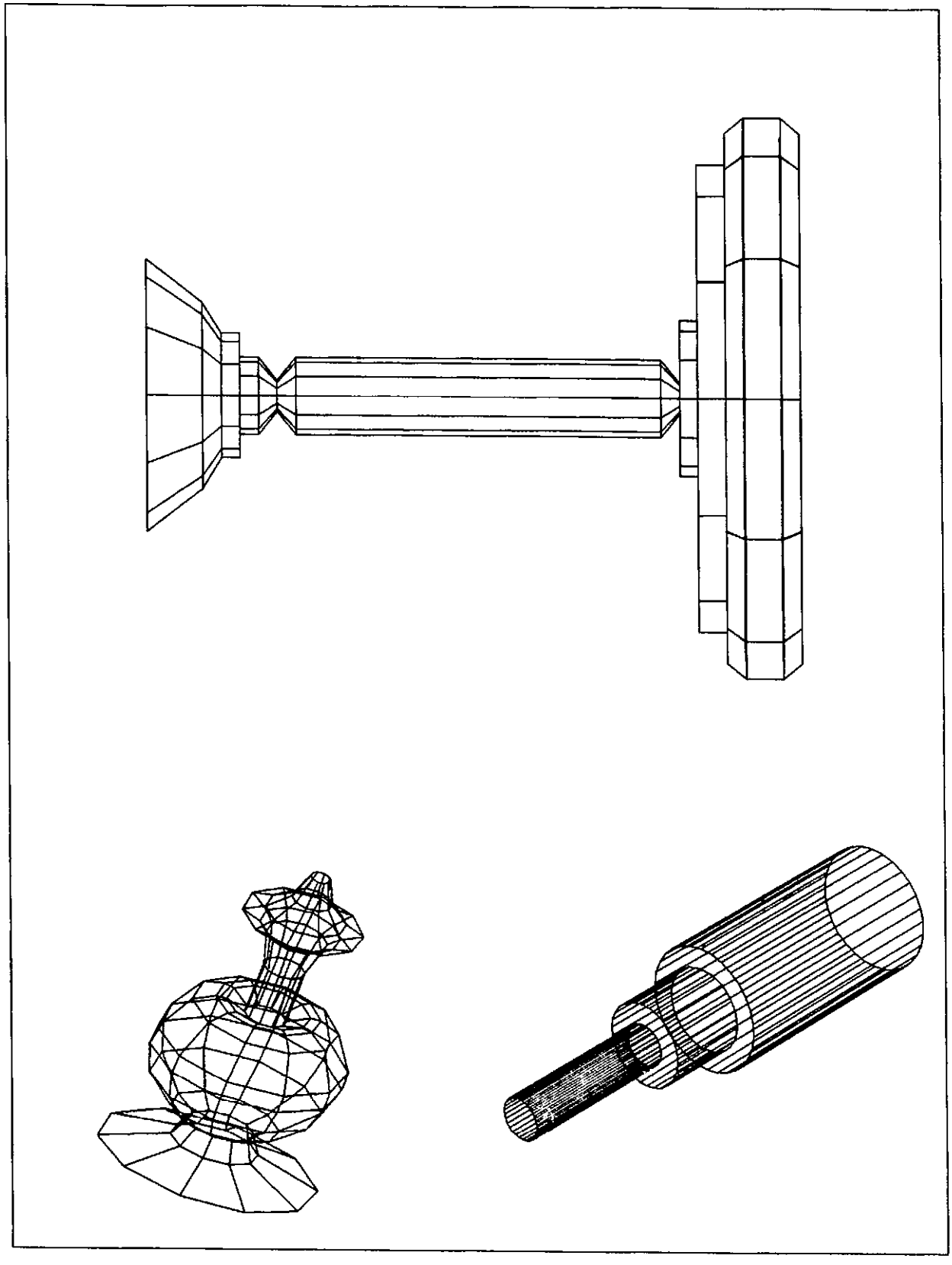


Fig. 5: Use of Some Functions

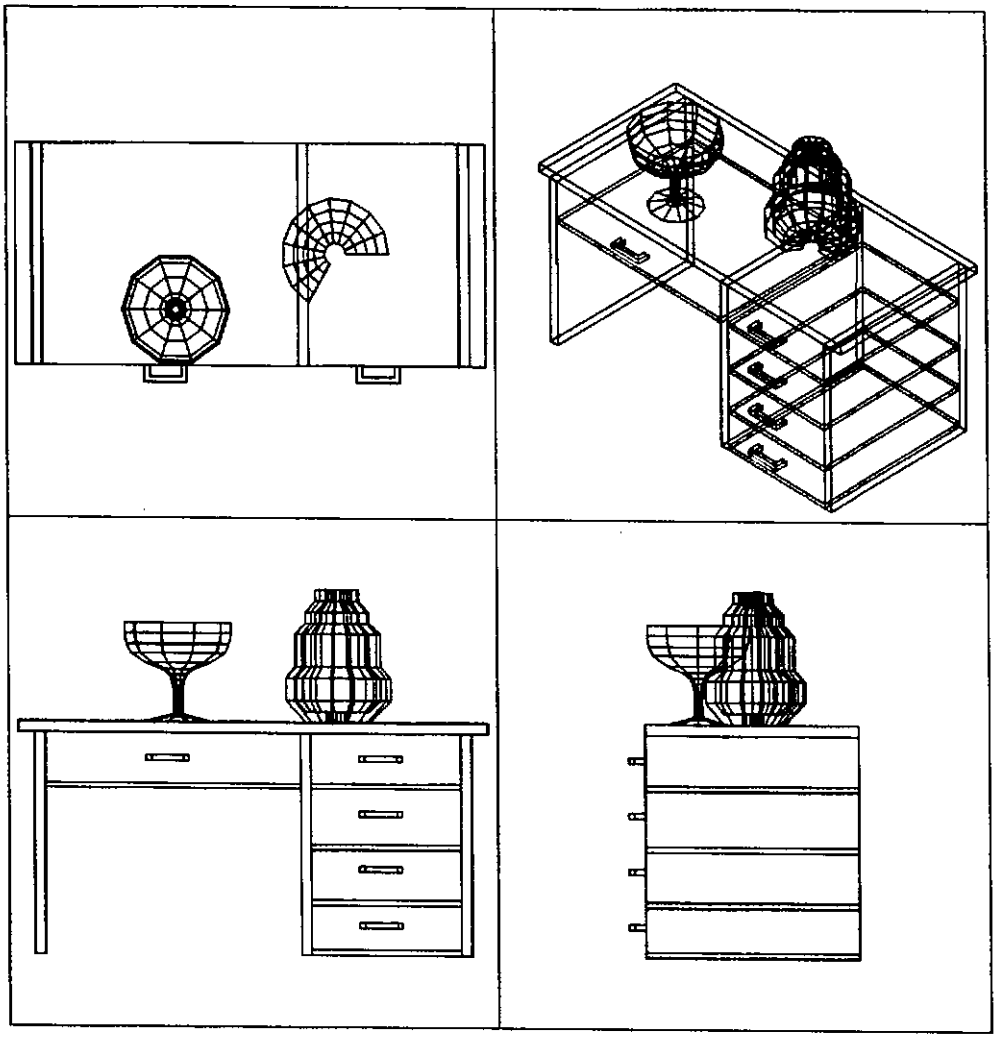


Fig. 6: FOURVIEWS OBJECTS1

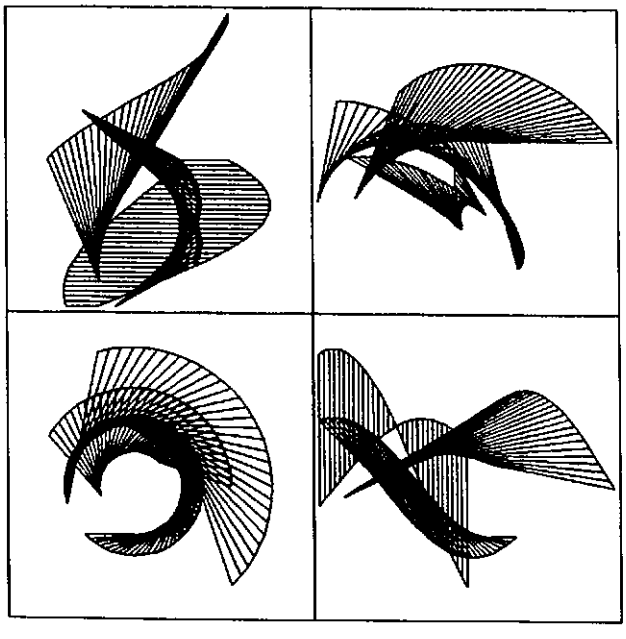


Fig. 7: FOURVIEWS WARP7

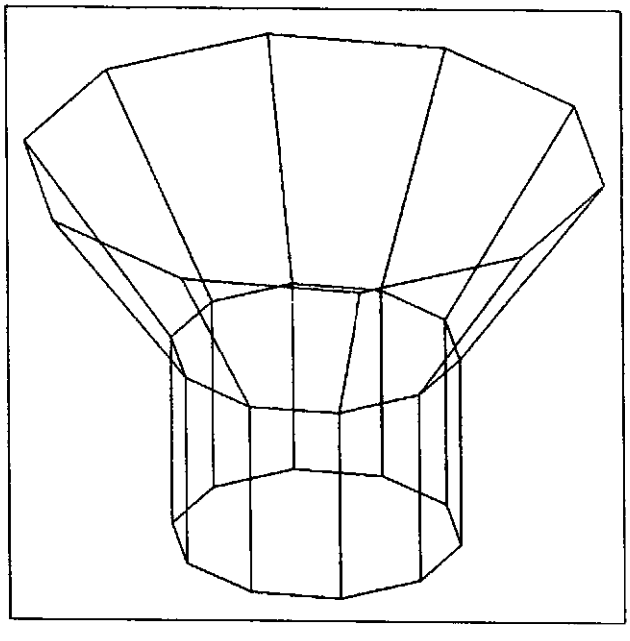


Fig. 8: An Isometric View

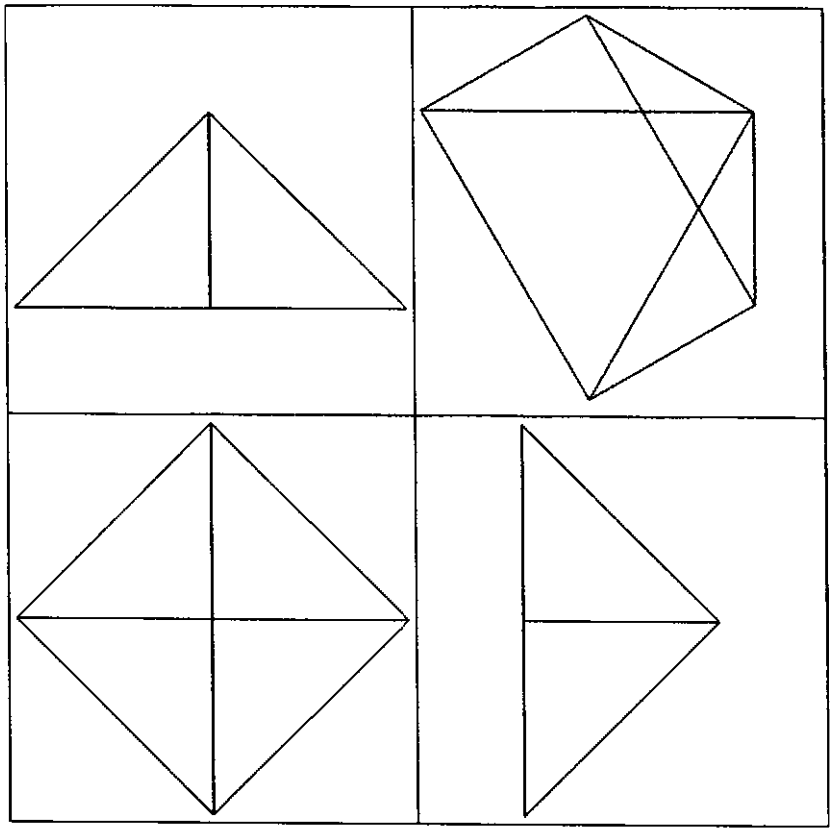


Fig. 9: Use of Modelling Functions

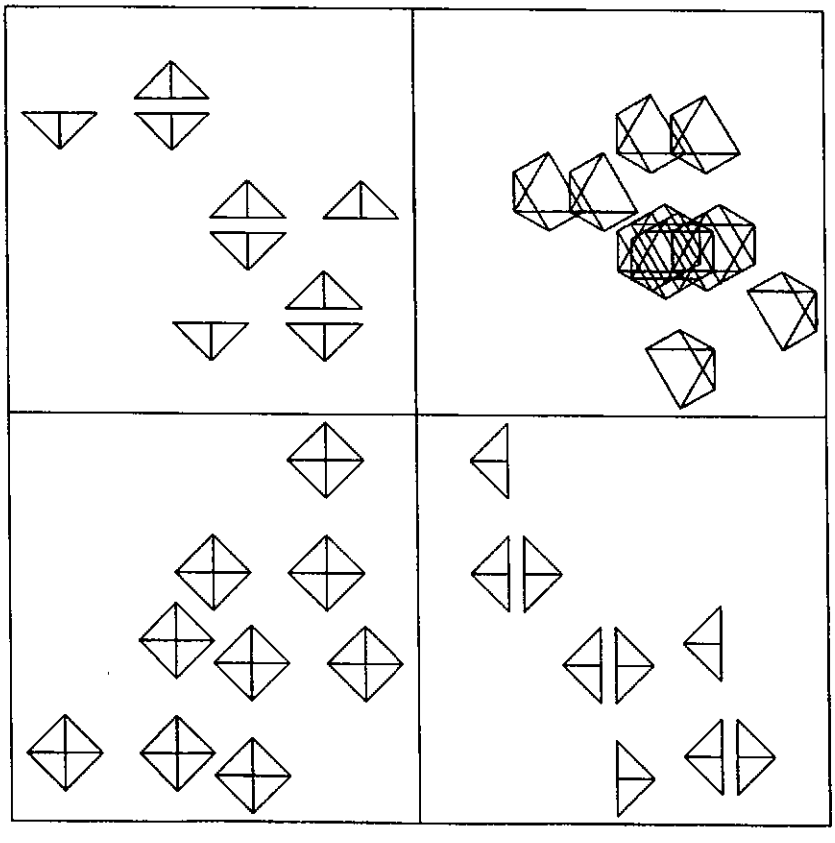


Fig. 10: More Modelling Functions

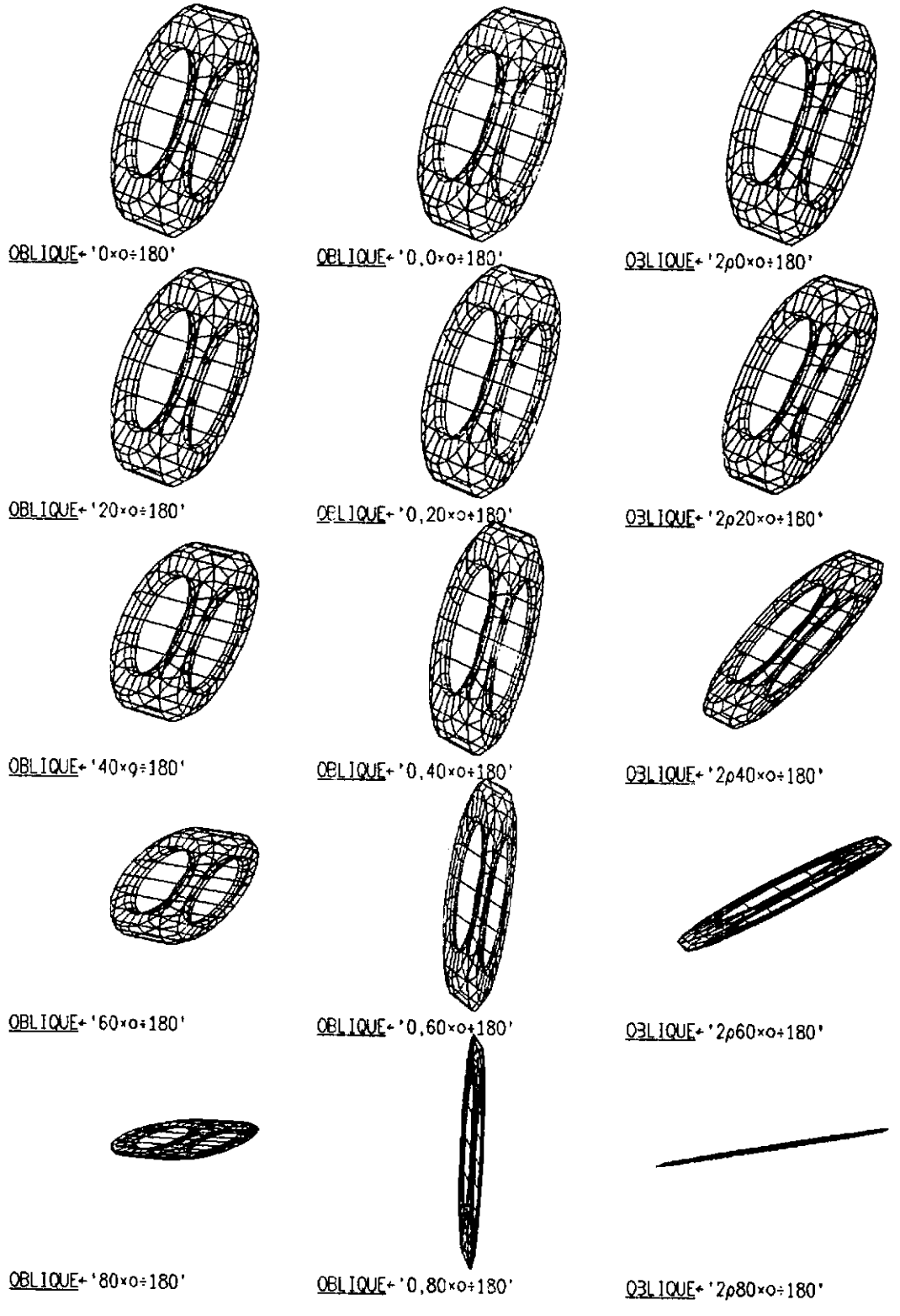
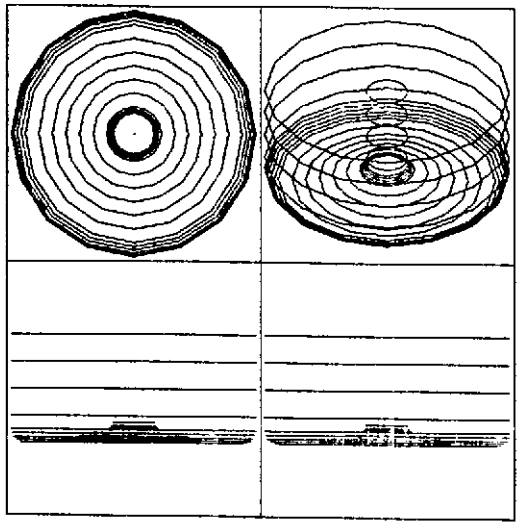
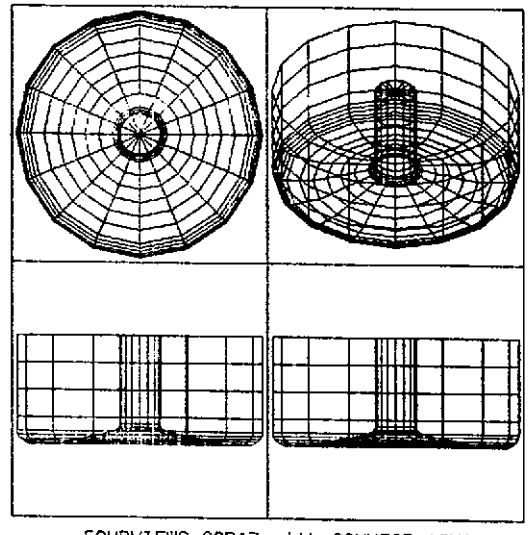


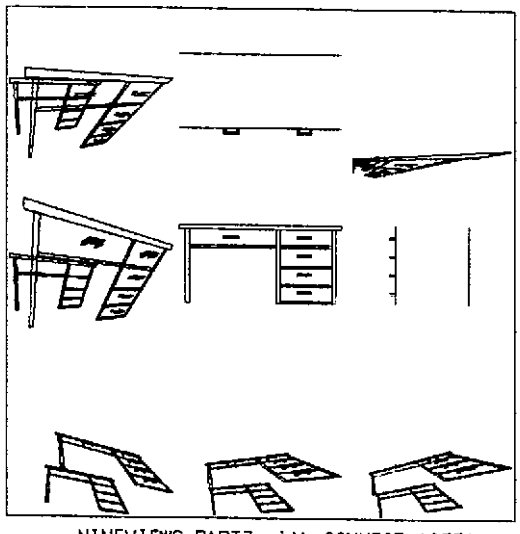
Fig. 11: Oblique views obtained by GOB16 DISPLAY 10 20 20



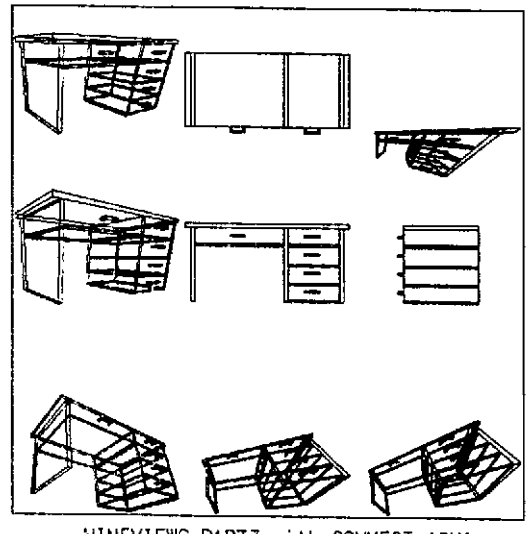
FOURVIEWS GOB17 with CONNECT+'OFF'



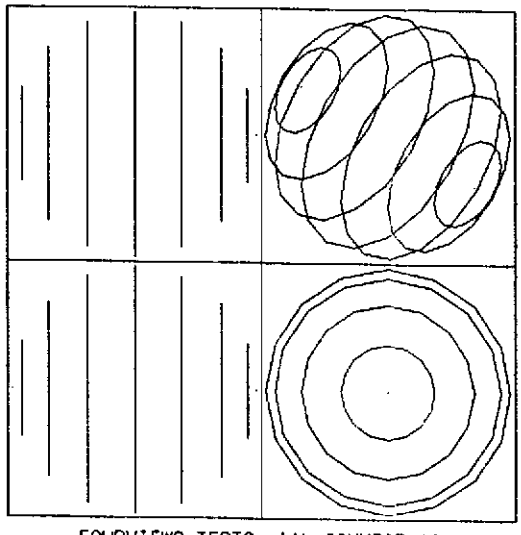
FOURVIEWS GOB17 with CONNECT+'ON'



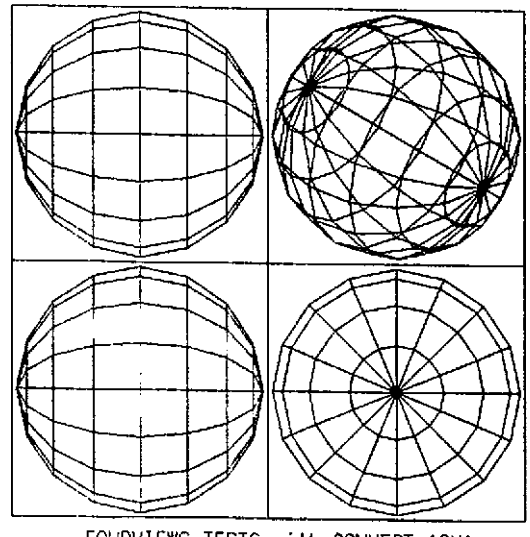
NINEVIEWS PART3 with CONNECT+'OFF'



NINEVIEWS PART3 with CONNECT+'ON'



FOURVIEWS TEST6 with CONNECT+'OFF'



FOURVIEWS TEST6 with CONNECT+'ON'

Fig. 12: Effects of CONNECT

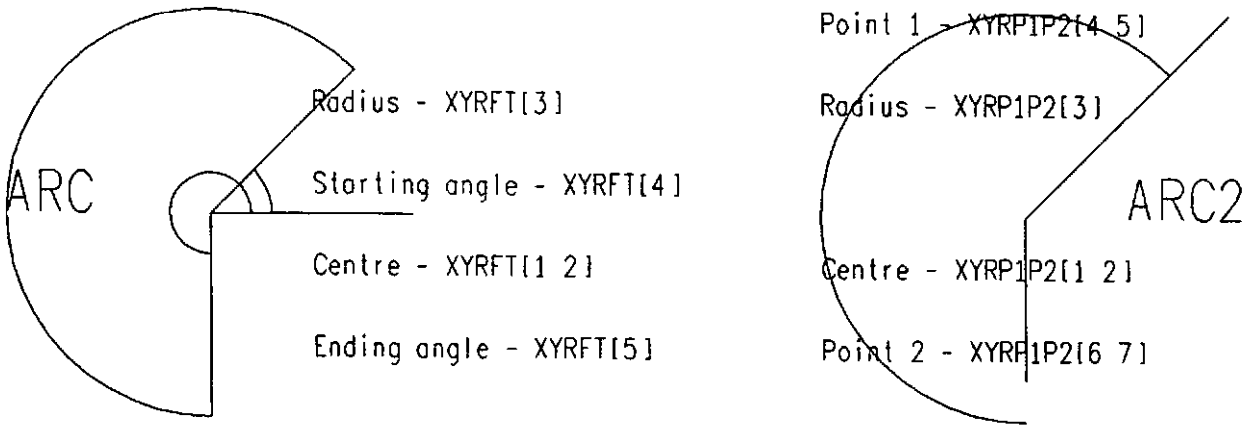


Fig. 13: Parameters of ARC and ARC2

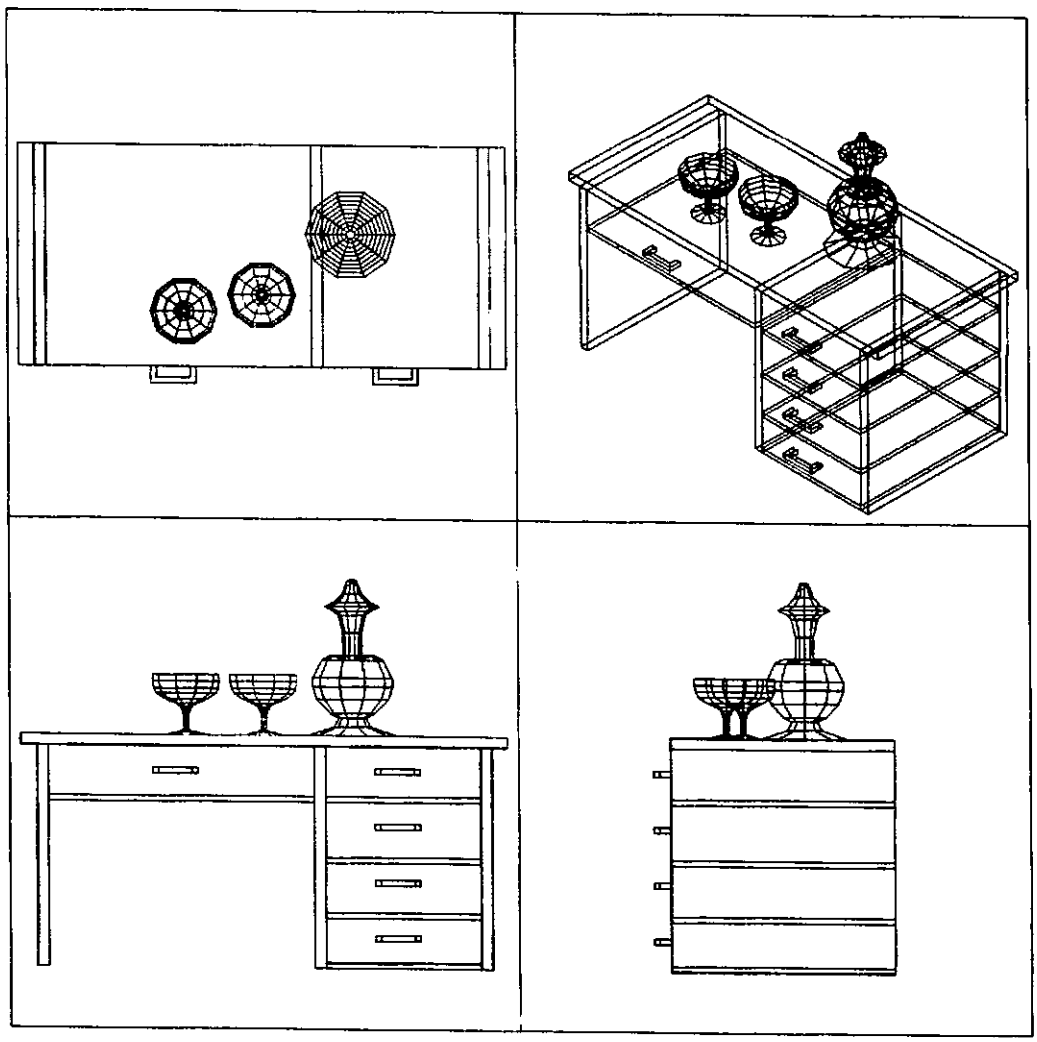
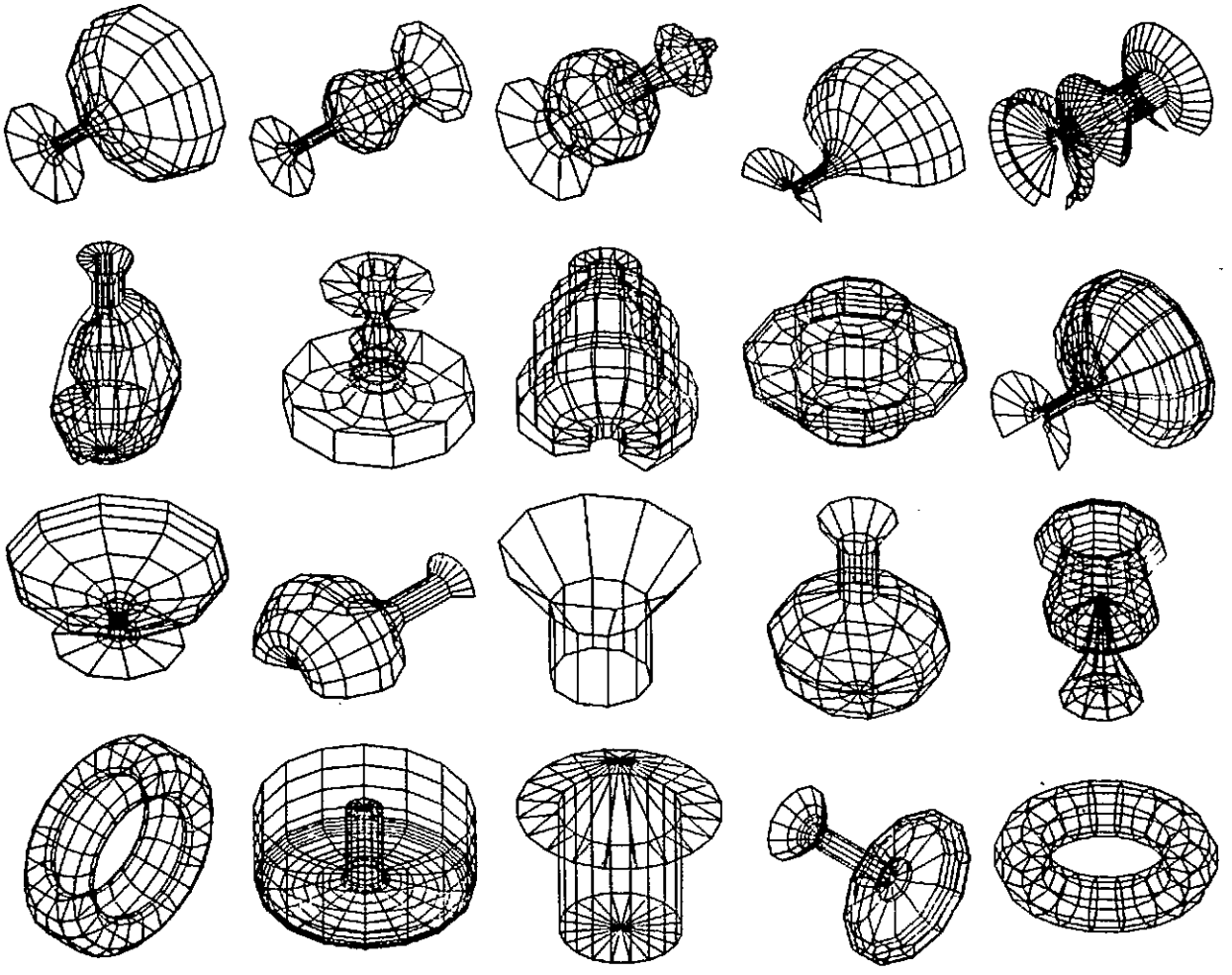


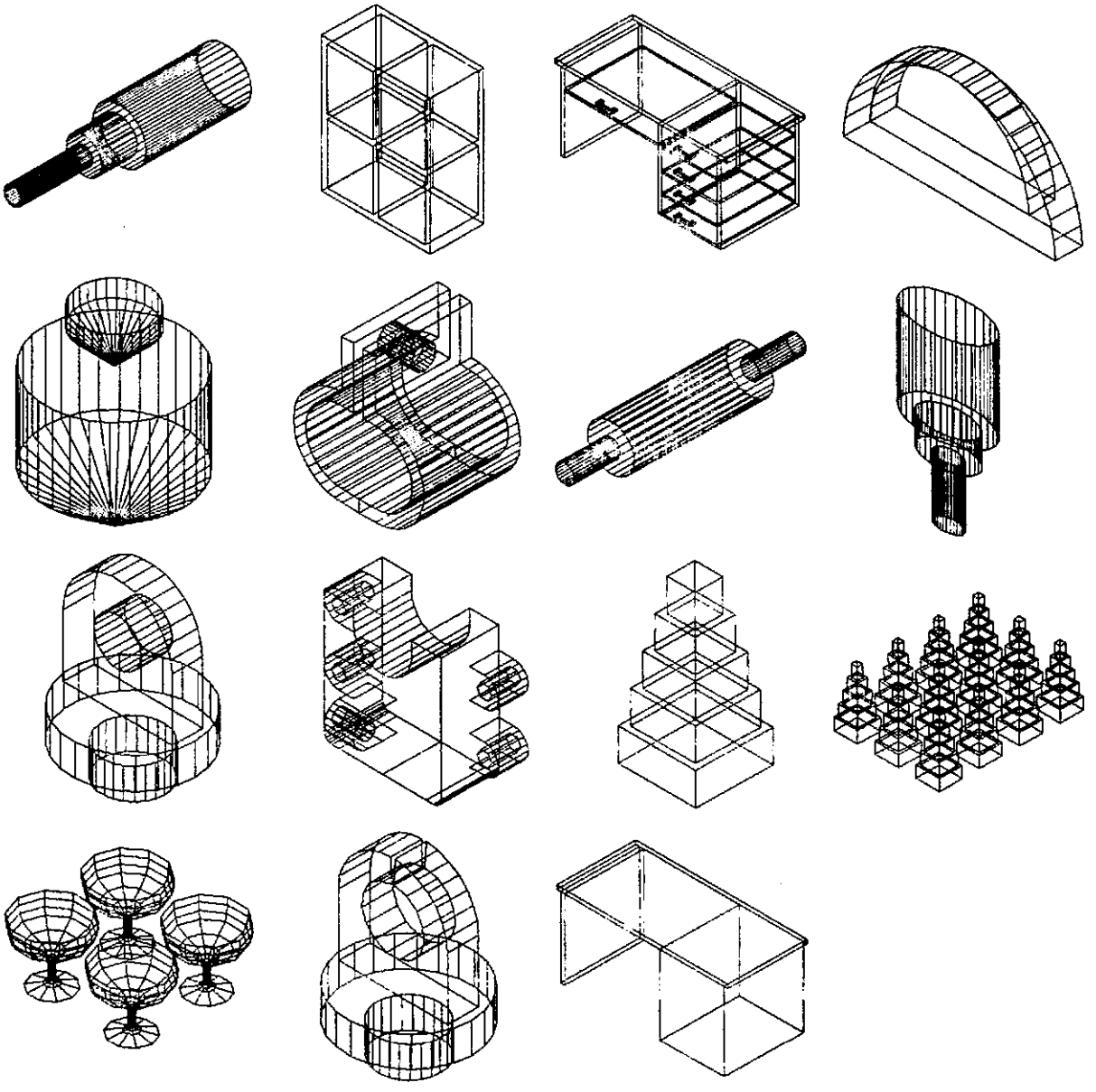
Fig. 14: FOURVIEWS OBJECTS2

APPENDIX
A Pictorial Dictionary

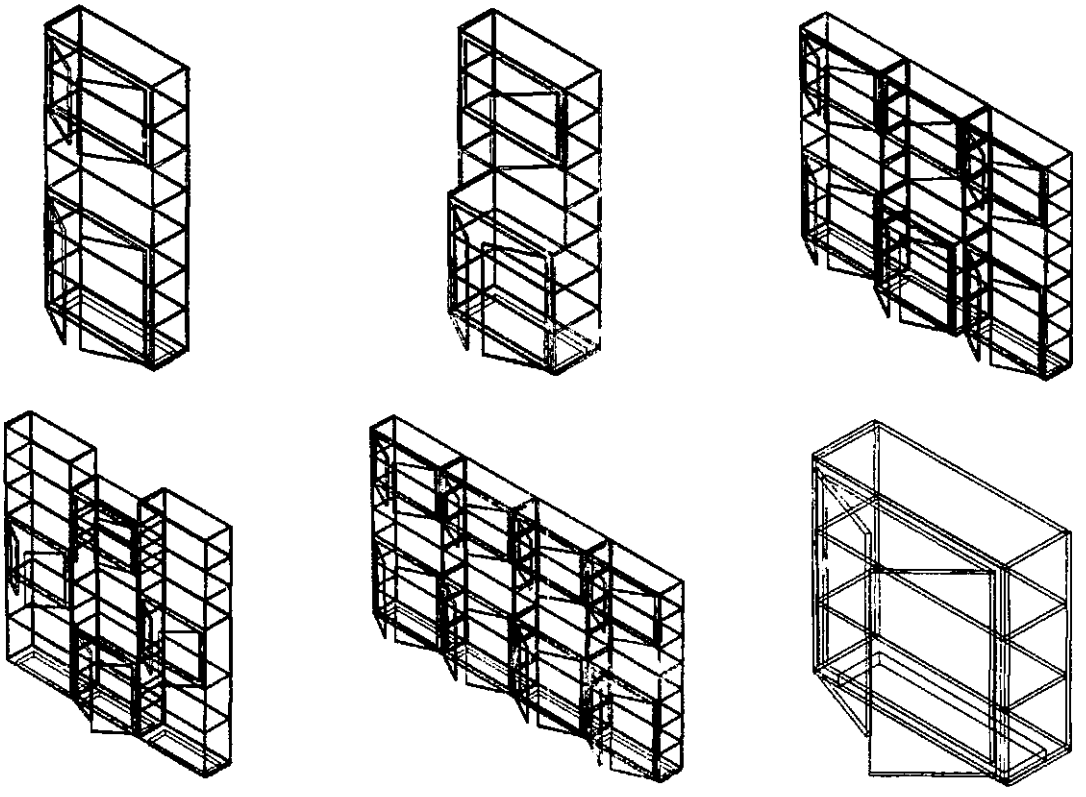
GOB$_n$ (FOR $n \in 20$)



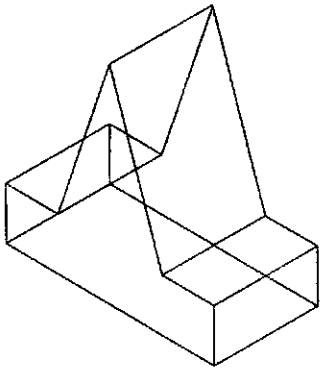
PART$_n$ (FOR $n \in 15$)

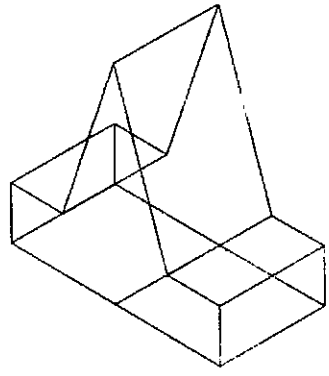


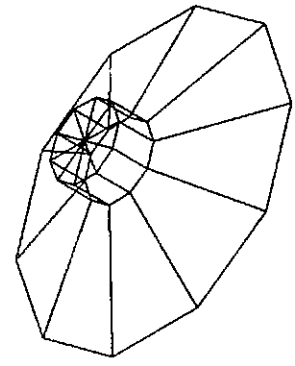
CUPBOARDn (FOR ١٤6)

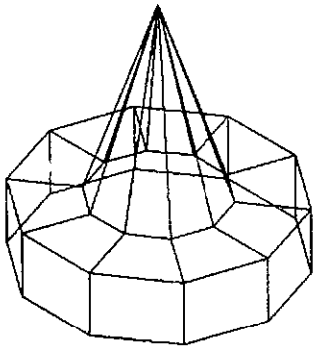


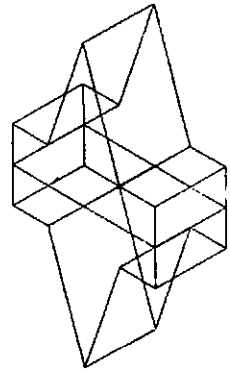
TESTn (FOR $n \epsilon 7$)

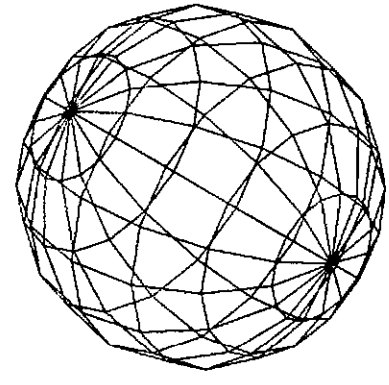


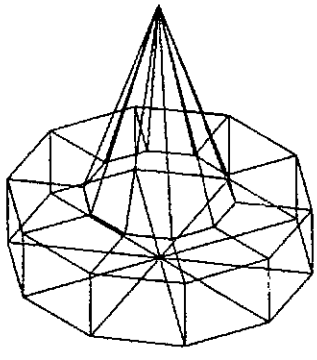




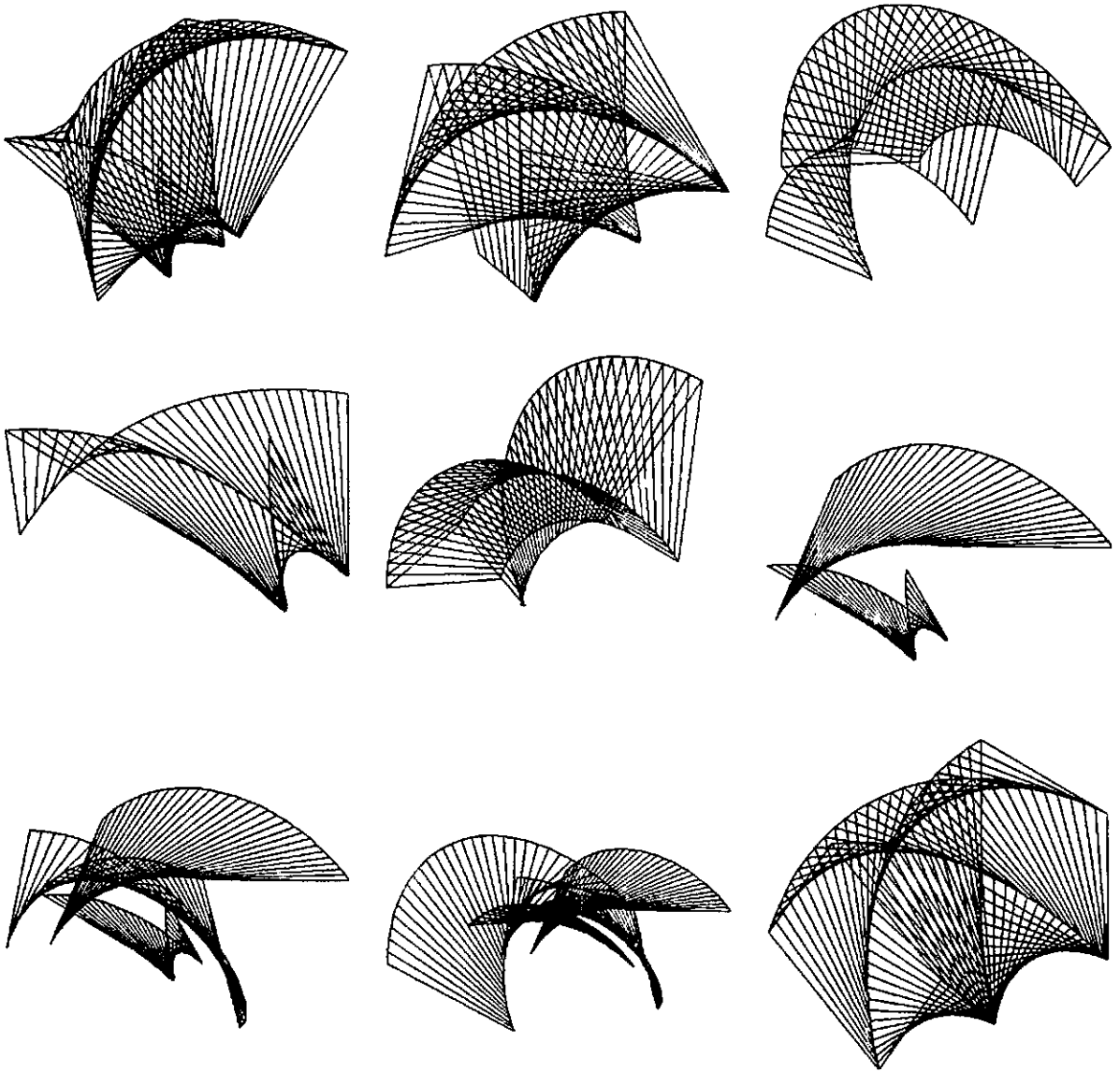








WARPn (FOR $n \in 9$)



OBJECTS$_n$ (FOR $n \in 2$)