SANDIA REPORT

SAND87—0872 • UC—32
Unlimited Release
Printed April 1987

Simple, Portable, 3-D Projection Routine

John S. Wagner

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

When printing a copy of any digitized SAND Report, you are required to update the markings to current standards.

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A02
Microfiche copy: A01

Distribution
Category UC-32

SAND87-0872
Unlimited Release
Printed April 1987

Simple, Portable, 3-D Projection Routine

John S. Wagner
Plasma Theory Division
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

A 3-D projection routine is presented for use in computer graphics applications. The routine is simple enough to be considered portable, and easily modified for special problems. There is often the need to draw three-dimensional objects on a two-dimensional plotting surface. For the object to appear realistic, perspective effects must be included that allow near objects to appear larger than distant objects. Several 3-D projection routines are commercially available, but they are proprietary, not portable, and not easily changed by the user. Most are restricted to surfaces that are functions of two variables. This makes them unsuitable for viewing physical objects such as accelerator prototypes or propagating beams. This report develops a very simple algorithm for 3-D projections; the core routine is only 39 FORTRAN lines long. It can be easily modified for special problems. Software dependent calls are confined to simple drivers that can be exchanged when different plotting software packages are used.

INTRODUCTION

Computer generated pictures of three-dimensional objects are becoming increasingly useful aids in computer aided design and simulation. Unfortunately, most computer users are restricted by the graphics software that is commercially available. They assume that 3-D projection algorithms are too complicated for custom tailoring. The purpose of this report is to present a 3-D projection routine that is short, simple and portable. It is not the intention of this report to supply a large comprehensive 3-D graphics package. No bells or whistles are included; they are easy to add and customize individually.

ALGORITHM

The basic problem is to find the projection of a three-dimensional object onto a two-dimensional viewing plane. The viewing plane can be thought of as the translucent plotting surface or a sheet of paper. The 2-D image of the real 3-D object is projected onto the viewing surface. Suppose an observer located at a viewing location $\vec{V}$ (see Fig. 1) looks at the 3-D object through the viewing plane. Consider one "data" vector located on the object $\vec{D}$. The projection of $\vec{D}$ onto the viewing plane, or the data to be plotted, is located at the intersection of the line connecting $\vec{V}$ and $\vec{D}$ with the viewing plane. In parametric form, the line connecting $\vec{V}$ and $\vec{D}$ is given by the set of vectors $\vec{R}$ such that

$$\vec{R} = \vec{V} + \alpha\,(\vec{D} - \vec{V}) \qquad (1)$$

The scalar parameter α determines all points on the line that connect $\vec{V}$ and $\vec{D}$, and in particular α takes on values between [0, 1] when $\vec{R}$ is between $\vec{V}$ and $\vec{D}$.

Now consider the viewing plane. One reference point on the plane is $\vec{P}$, which can be chosen arbitrarily by the viewer as long as it is on the viewing plane. In order to completely specify the plane, one additional point on the viewing plane is needed, along with a normal to the plane. For convenience, choose another point $\vec{P}_x$ which lies in the viewing plane and preferably lies somewhere on the "x-axis" or "bottom"

of the viewing plane. Also for convenience, choose the normal $\hat{c}$ to be parallel to $\vec{V} - \vec{P}$, the line that connects the viewer and the reference point on the viewing plane. Given $\vec{V}$, $\vec{P}$, and $\vec{P}_x$, these definitions allow the computation of the unit vectors $\hat{a}$, $\hat{b}$, and $\hat{c}$, which are located at $\vec{P}$, as in Figure 1.

$$\hat{c} = (\vec{V} - \vec{P})/|\vec{V} - \vec{P}| \quad ,$$

$$\hat{a} = (\vec{P}_x - \vec{P})/|\vec{P}_x - \vec{P}| \quad ,$$

$$\hat{b} = \hat{c} \times \hat{a} \quad . \tag{2}$$

Note that these formulas for $\hat{a}$, $\hat{b}$, and $\hat{c}$ are chosen for convenience and simplicity. Any other set of orthogonal unit vectors on the plane would do as well.

Finally, the intersection of the viewing plane with the line that connects the object and viewer can be computed. The intersection occurs when $\vec{R} - \vec{P}$ lies in the viewing plane and is perpendicular to the normal $\hat{c}$, or

$$(\vec{R} - \vec{P}) \cdot \hat{c} = 0 \quad . \tag{3}$$

Substituting for $\vec{R}$ from Equation (1) into (3) yields

$$(\vec{V} + \alpha(\vec{D} - \vec{V}) - \vec{P}) \cdot \hat{c} = 0 \quad , \tag{4}$$

which can be solved for the value of α at intersection,

$$\alpha_i = (\vec{P} - \vec{V}) \cdot \hat{c}/(\vec{D} - \vec{V}) \cdot \hat{c} \quad . \tag{5}$$

Note that this is an easy calculation since the viewing position $\vec{V}$, the object position $\vec{D}$, and the normal $\hat{c}$ from (2) are known and determined by the viewer.

Now that the precise location of the object's projection onto the viewing plane is known, it is easy to solve for the actual plotting coordinates. The point to be plotted is given by

$$\vec{R}_p = \vec{V} + \alpha_i (\vec{D} - \vec{V}) \tag{6}$$

In terms of the x - y coordinate system chosen by the viewer via $\vec{P}$ and $\vec{P}_x$, the plotting coordinates are just

$$x_p = (\vec{P} - \vec{R}) \cdot \hat{a}$$

$$y_p = (\vec{P} - \vec{R}) \cdot \hat{b} \quad . \tag{7}$$

By computing and then plotting the entire set of points (xp, yp) that make up the object, a 3-D projected representation of the object, with appropriate perspective will be drawn. A FORTRAN program containing this algorithm is in Appendix 1, called subroutine "p3d." The only problem that can occur is when the denominator of equation (5) is zero. This happens when the object and the viewer are both in the viewing plane, which should never occur. Nonetheless, the routine protects against this by returning the appropriate coordinates of the point in the viewing plane regardless of the observers inappropriate viewing location.

Appendix 2 contains an auxiliary subroutine "box" which computes the corner points of an imaginary box that contains the object. Knowing the box location and size enables the plot to be scaled in such a way that the viewing plane is 'filled' by the object. It is often convenient to plot the box as well, since this often gives a better feeling of perspective.

Appendix 3 contains subroutine "labl", a short software dependent routine that scales the plot and labels the axis. This routine must be modified if Sandia's plotting package weasel is not used. The modifications for other plotting packages are fairly obvious. Just locate (x_{min}, y_{min}) and (x_{max}, y_{max}) on the lower left and upper right corners of the plotting device respectively. The calls to wcolor are

commented out only to show that color changes are possible, but optional.

Appendix 4 contains the main program that produced Fig. 2. Figure 2 shows four views of the same object, illustrating how perspective effects are exaggerated when the viewer gets very close to the object. The only software dependent calls are to wstart and wend, to open and close the plotting software.

First the viewing location is computed from a length (vmag) and a direction (θ, ϕ). The viewing plane is located along the same radial line only 90% of vmag from the origin. Next the data is generated. Then subroutine box is called to compute and plot a box around the data. Lastly, the object is projected onto the viewing plane with a call to subroutine "p3d" and the point actually plotted with a call to driver. This process is repeated four times for different viewing locations.

Appendix 6 is another main program very similar to Appendix 5. The difference is three physically separate objects are plotted in the same box, and viewed from four different locations. Comparisons of Appendix 5 and Appendix 6 illustrate the flexibility of the algorithm in "p3d" and how complicated 3-D structures can be handled.

Finally, Appendix 6 contains two auxiliary routines, a driver for plotting and a min-max routine. Use of the driver keeps software dependent routines confined to subroutines.

SUMMARY

A simple, portable 3-D projection algorithm is given that is useful in constructing complex 3-D images. The algorithm is portable in the sense that plotting software dependent calls are confined to two subroutines.

Appendix 1

```
c
c  core 3d routine
c  finds projection of object data on image surface (plot)
c  for a given viewing position
      subroutine p3d (p, px, v, xd, yd, zd, xp, yp, nd)
      dimension p(1), px(1), v(1)
      dimension xd(1), yd(1), zd(1), xp(1), yp(1)
      dimension a(3), b(3), c(3)
      data eps/1.e-32/
c
c  compute unit vectors in image plane
c
      ax = px(1) - p(1)
      ay = px(2) - p(2)
      az = px(3) - p(3)
      am = sqrt(ax*ax + ay*ay + az*az)
      ax = ax/am
      ay = ay/am
      az = az/am
c
      cx = v(1) - p(1)
      cy = v(2) - p(2)
      cz = v(3) - p(3)
      cm = sqrt(cx*cx + cy*cy + cz*cz)
      cx = cx/cm
      cy = cy/cm
      cz = cz/cm
c
      bx = cy*az - cz*ay
      by = cz*ax - cx*az
      bz = cx*ay - cy*ax
c
c  compute image coordinates in image plane
c
      do 20 n = 1, nd
      d = (xd(n) - v(1))*cx + (yd(n) - v(2))*cy + (zd(n) - v(3))*cz
c
c  protect for both object and viewer in projection plane
c
      if (abs(d).lt.eps) then
      xp(n) = (xd(n) - p(1))*ax + (yd(n) - p(2))*ay + (zd(n) - p(3))*az
      yp(n) = (xd(n) - p(1))*bx + (yd(n) - p(2))*by + (zd(n) - p(3))*bz
      go to 20
      end if
      alpha = ((p(1) - v(1))*cx + (p(2) - v(2))*cy + (p(3) - v(3))*cz)/d
      rix = v(1) + alpha*(xd(n) - v(1))
      riy = v(2) + alpha*(yd(n) - v(2))
      riz = v(3) + alpha*(zd(n) - v(3))
      xp(n) = (rix - p(1))*ax + (riy - p(2))*ay + (riz - p(3))*az
      yp(n) = (rix - p(1))*bx + (riy - p(2))*by + (riz - p(3))*bz
20    continue
      return
      end
```

Appendix 2

```
c
c  subroutine scales data and draws box if iopt = 1
c
      subroutine box (p, px, v, xd, yd, zd, xp, yp, nd, iopt)
      dimension xd(1), yd(1), zd(1), xp(1), yp(1)
      dimension p(3), px(3), v(3)
      dimension xb(16), yb(16), zb(16)
      dimension xmsk(16), ymsk(16), zmsk(16)
      data xmsk/-1.,-1.,1.,1.,-1.,-1.,1.,1.,1.,1.,1.,1.,-1.,-1.,-1.,-1./
      data ymsk/-1.,-1.,-1.,-1.,-1.,1.,1.,-1.,-1.,1.,1.,1.,1.,1.,1.,-1./
      data zmsk/-1.,1.,1.,-1.,-1.,-1.,-1.,-1.,1.,1.,-1.,1.,1.,-1.,1.,1./
c
c  find max min of data
c
      call aminmx(xd, 1, nd, 1, xmin, xmax)
      call aminmx(yd, 1, nd, 1, ymin, ymax)
      call aminmx(zd, 1, nd, 1, zmin, zmax)
c
c  find coordinates of box enclosing data
c
      do 10 i = 1, 16
       xb(i) = ifix(xmsk(i) + .1)*xmax - ifix(xmsk(i) - .1)*xmin
       yb(i) = ifix(ymsk(i) + .1)*ymax - ifix(ymsk(i) - .1)*ymin
       zb(i) = ifix(zmsk(i) + .1)*zmax - ifix(zmsk(i) - .1)*zmin
10    continue
c
c  find image of box on image surface (plot)
c
      call p3d(p, px, v, xb, yb, zb, xp, yp, 16)
c
c  make sure plot max and min cause no distortion
c
      call aminmx(xp, 1, 16, 1, xpmin, xpmax)
      call aminmx(yp, 1, 16, 1, ypmin, ypmax)
      dx = xpmax - xpmin
      dy = ypmax - ypmin
      if (dx.gt.dy) then
       ypmax = ypmax + .5*(dx - dy)
       ypmin = ypmin - .5*(dx - dy)
      end if
      if (dy.gt.dx) then
       xpmax = xpmax + .5*(dy - dx)
       xpmin = xpmin - .5*(dy - dx)
      end if
c
c  set scale factors and labels for plot
c
      call labl(xpmin, xpmax, ypmin, ypmax, 'xp', 'yp')
c
c  plot box if iopt = 1
c
      if (iopt.eq.1) call driver (xp, yp, 16)
      return
      end
```

Appendix 3

```
c
c  set scales and label plot
c
      subroutine labl(xmin, xmax, ymin, ymax, nxlab, nylab)
      character*(*) nxlab, nylab
      data nplot/1/
c
      if (abs(xmax - xmin).lt.1.e - 32) return
      if (abs(ymax - ymin).lt.1.e - 32) return
c
c  initialize
c
      call wttype('SOFT')
      call wspace('NDC')
      call wtsize(.03)
      call wspace('USER')
      stepx = (xmax - xmin)/5.
      stepy = (ymax - ymin)/5.
      call wtics(10)
c
c  axis
c
      xl = .4
      yl = .4
      if (nplot.eq.1) then
       xo = .05
       yo = .05
c       call wcolor('BLUE')
      end if
      if (nplot.eq.2) then
       xo = .05
       yo = .55
c       call wcolor('RED')
      end if
      if (nplot.eq.3) then
       xo = .6
       yo = .55
c       call wcolor('GREEN')
      end if
      if (nplot.eq.4) then
       xo = .6
       yo = .05
```

Appendix 3, cont.

```
c         call wcolor('CYAN')
        end if
        call wstyle('NONUMBERS')
        call wstyle('NOTICS')
        call wstyle('FLIPPED')
        call wxaxis(xo, yo + yl, xl, xmin, xmax, stepx, nxlab)
        call wyaxis(xo + xl, yo, yl, ymin, ymax, stepy, nylab)
        call wstyle('NOFLIPPED')
        call wxaxis(xo, yo, xl, xmin, xmax, stepx, nxlab)
        call wyaxis(xo, yo, yl, ymin, ymax, stepy, nylab)
c
        nplot = nplot + 1
        if (nplot.gt.5) then
         nplot = 1
         call wnewpg
        end if
c
        return
        end.
```

Appendix 4

```
c
c  main program to demonstrate 3-d projection plotting
c
      parameter (nd = 2000)
      dimension p(3), px(3), v(3)
      dimension xd(nd), yd(nd), zd(nd)
      dimension xp(nd), yp(nd)
c
c  open plotting software
c
      call wstart(1., 0)
c
c  define viewing location, image plane location
c  and point on x-axis of image plane coordinate system
c
      dr = .5
      do 100 nr = 1, 4
       vmag = 1. + dr*float(nr)
       theta = -30.*(2.*3.1415926535/360.)
       phi = 65.*(2.*3.1415926535/360.)
       v(1) = vmag*cos(theta)*sin(phi)
       v(2) = vmag*sin(theta)*sin(phi)
       v(3) = vmag*cos(phi)
       pmag = .9*vmag
       p(1) = pmag*cos(theta)*sin(phi)
       p(2) = pmag*sin(theta)*sin(phi)
       p(3) = pmag*cos(phi)
       px(1) = pmag*cos(theta + .001)*sin(phi)
       px(2) = pmag*sin(theta + .001)*sin(phi)
       px(3) = pmag*cos(phi)
c
c  generate data for object
c
      rd = nd - 1
      do 10 n = 1, nd
       rn = n - 1
       xd(n) = cos(2.*3.1415825535*rn/rd)
       yd(n) = sin(2.*3.1415825535*rn/rd)
       zd(n) = .2*cos(80.*3.415825535*rn/rd)
10    continue
c
c  draw box around object
c
      call box(p, px, v, xd, yd, zd, xp, yp, nd)
      call driver(xp, yp, nd)
c
c  draw object
c
c     call p3d(p, px, v, xd, yd, zd, xp, yp, nd)
      call driver (xp, yp, nd)
```

Appendix 4, cont.

```
c
c  go to next frame
c
c       call wnewpg
100     continue
c
c  close plotting software
c
        call wend
        stop
        end
```

Appendix 5

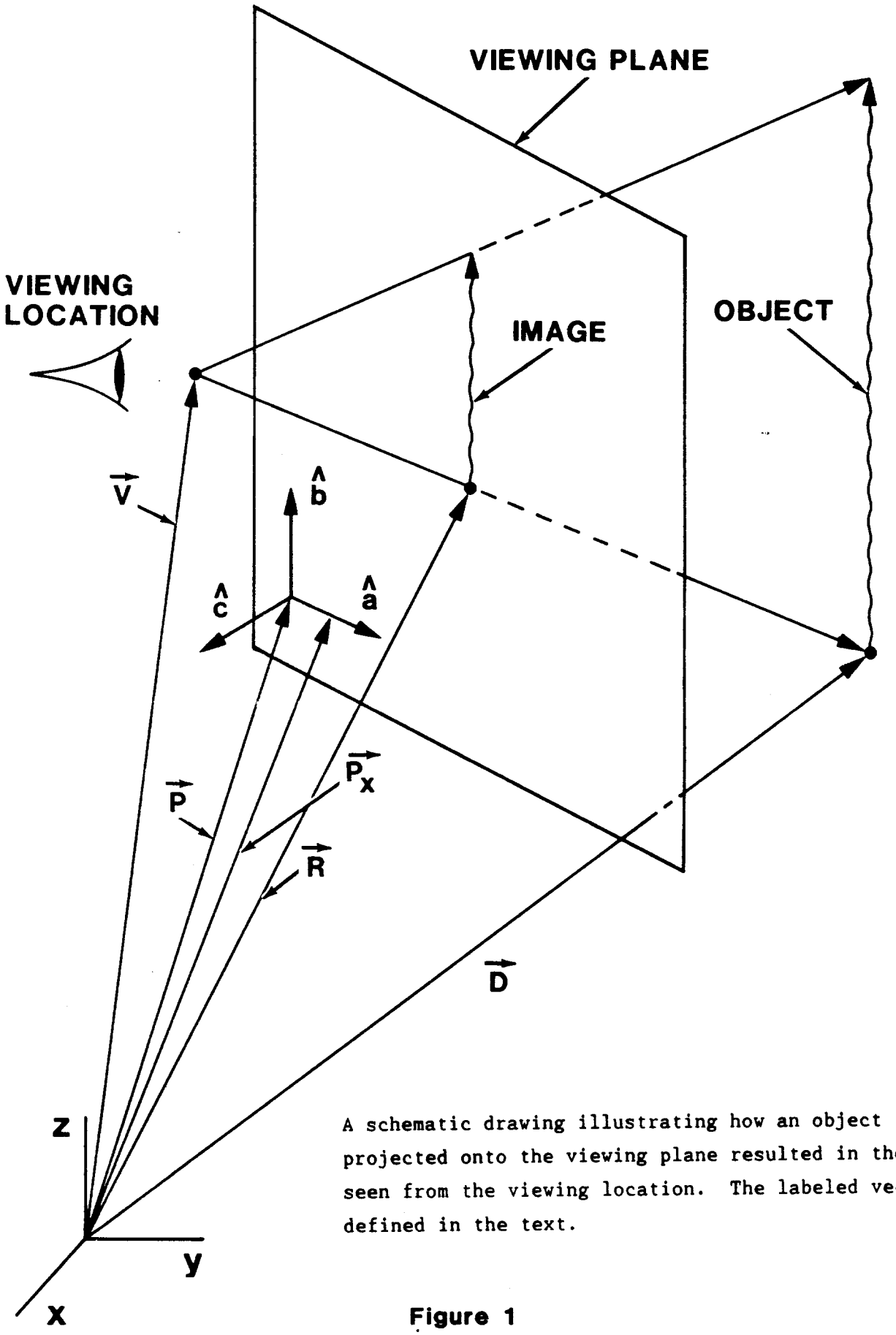
```
c
c  main program to demonstrate 3-d projection plotting
c  three objects in a box
c
      parameter (nd1 = 2000, nd2 = 2000, nd3 = 2000, nd = nd1 + nd2 + nd3)
      dimension p(3), px(3), v(3)
      dimension xd1(nd1), yd1(nd1), zd1(nd1)
      dimension xd2(nd2), yd2(nd2), zd2(nd2)
      dimension xd3(nd3), yd3(nd3), zd3(nd3)
      dimension xd(nd), yd(nd), zd(nd)
      dimension xp(nd), yp(nd)
c
c  open plotting software
c
      call wstart(1., 0)
c
c  define viewing location, image plane location
c  and point on x-axis of image plane coordinate system
c
      vmag = 12.
      do 100 nt = 1, 4
       theta = -30.*float(nt - 1)*(2.*3.1415926535/360.)
       phi = 60.*(2.*3.1415926535/360.)
       v(1) = vmag*cos(theta)*sin(phi)
       v(2) = vmag*sin(theta)*sin(phi)
       v(3) = vmag*cos(phi)
       pmag = .9*vmag
       p(1) = pmag*cos(theta)*sin(phi)
       p(2) = pmag*sin(theta)*sin(phi)
       p(3) = pmag*cos(phi)
       px(1) = pmag*cos(theta + .001)*sin(phi)
       px(2) = pmag*sin(theta + .001)*sin(phi)
       px(3) = pmag*cos(phi)
c
c  generate data for object #1
c
      do 10 n = 1, nd1
       rn = n
       rd = nd1
       xd1(n) = (rn - .5*rd)**2/(.5*rd)**2 - 3.
       yd1(n) = (rn/rd)**2*sin(.1*rn)
       zd1(n) = (rn/rd)**2*cos(.1*rn)
       xd(n) = xd1(n)
       yd(n) = yd1(n)
       zd(n) = zd1(n)
10    continue
c
c  generate data for object #2
c
```

Appendix 5, cont.

```
      do 20 n = 1, nd2
       rn = n
       rd = nd2
       xd2(n) = (rn - .5*rd)**2/(.5*rd)**2.
       yd2(n) = (rn/rd)**2*sin(.1*rn)
       zd2(n) = (rn/rd)**2*cos(.1*rn)
       xd(n + nd1) = xd2(n)
       yd(n + nd1) = yd2(n)
       zd(n + nd1) = zd2(n)
20    continue
c
c  generate data for object #3
c
      do 30 n = 1, nd1
       rn = n
       rd = nd3
       xd3(n) = (rn - .5*rd)**2/(.5*rd)**2 + 3.
       yd3(n) = (rn/rd)**2*sin(.1*rn)
       zd3(n) = (rn/rd)**2*cos(.1*rn)
       xd(n + nd1 + nd2) = xd3(n)
       yd(n + nd1 + nd2) = yd3(n)
       zd(n + nd1 + nd2) = zd3(n)
30    continue
c
c  draw box around all objects
c
      call box(p, px, v, xd, yd, zd, xp, yp, nd, 1)
c
c  draw object #1
c
      call p3d(p, px, v, xd1, yd1, zd1, xp, yp, nd1)
      call driver(xp, yp, nd1)
c
c  draw object #2
c
      call p3d(p, px, v, xd2, yd2, zd2, xp, yp, nd2)
      call driver (xp, yp, nd2)
c
c  draw object #3
c
      call p3d(p, px, v, xd3, yd3, zd3, xp, yp, nd3)
      call driver(xp, yp nd3)
c
c  go to next frame
c
c        call wnewpg
100   continue
c
c  close plotting software
c
      call wend
      stop
      end
```

Appendix 6

```
c
c  simple plotting driver
c
      subroutine driver(xp, yp, np)
      dimension xp(1), yp(1)
      if (np.eq.0) return
c  dissplla    call curve (xp, yp, np,0)
      call wdraw(xp, yp, np,0)
      return
      end
c
c  max min finder
c
      subroutine aminmx(a, ns, nf, ni, amin, amax)
      dimension a(1)
      amin = a(1)
      amax = amin
      do 10 i = ns, nf, ni
      if (a(i).lt.amin)amin = a(i)
      if (a(i).gt.amax)amax = a(i)
10    continue
      return
      end
```



A schematic drawing illustrating how an object is projected onto the viewing plane resulted in the image seen from the viewing location. The labeled vectors are defined in the text.

Figure 1

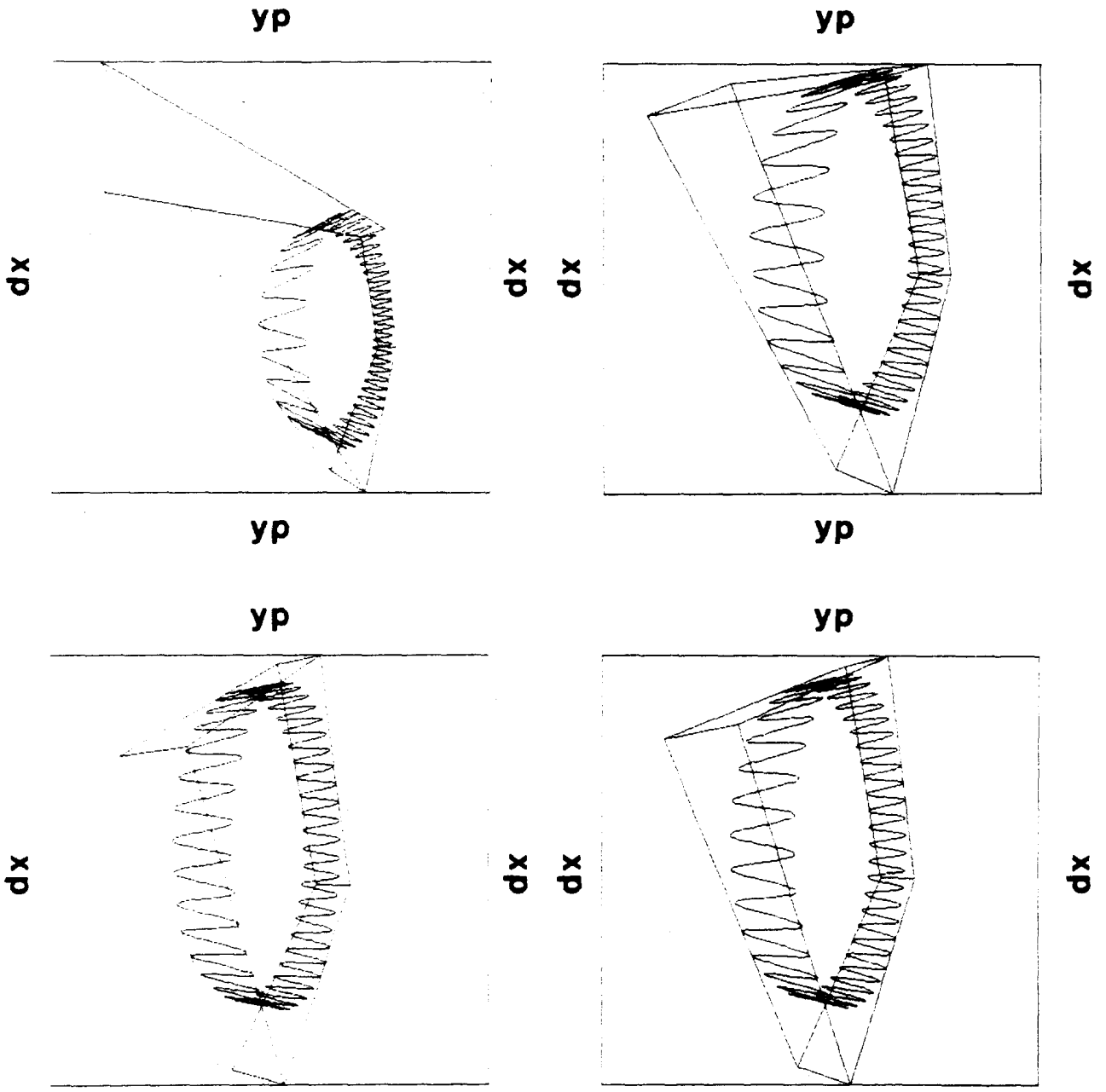


Figure 2

Plots of one object from the same angle along different radii illustrating perspective distortion effects. The program which made these plots is in Appendix 4.

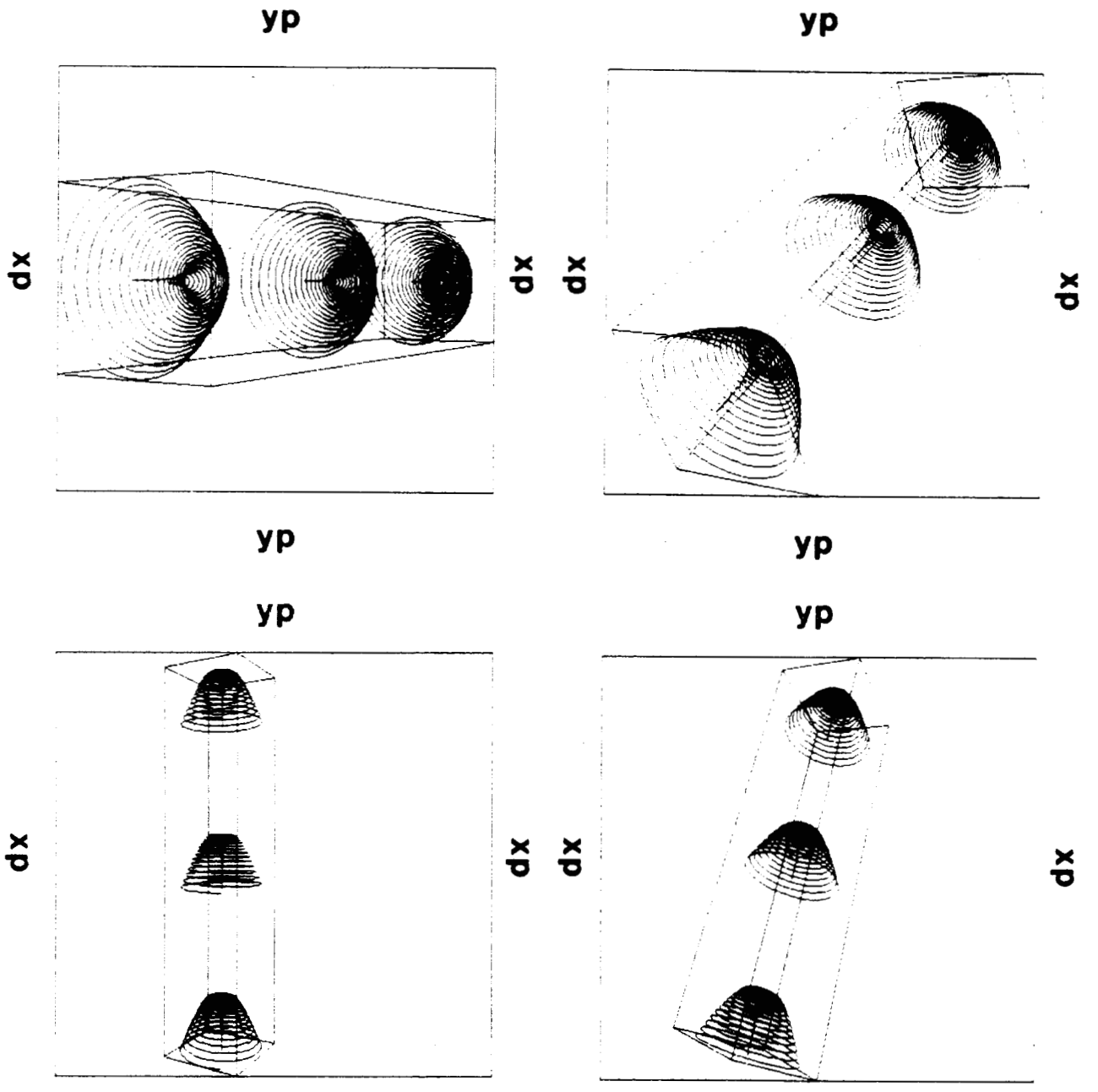


Figure 3

Plots of three objects from four different viewing angles. These plots were created by the program in Appendix 5.

DISTRIBUTION:
Unlimited Release

Air Force Weapons Laboratory
Kirtland Air Force Base
Albuquerque, NM 87117
Attn: Raymond Lemke

JAYCOR
205 S. Whiting Street
Alexandria, VA 22304
Attn: R. Hubbard

Lawrence Berkeley Laboratory
Building 47
1-Cyclotron Road
Berkeley, CA 94720
Attn: E. P. Lee

Lawrence Livermore National Laboratory (5)
P. O. Box 808
Livermore, CA 94550
Attn: W. Barleta
F. W. Chambers
W. H. Fawley
W. Sharp
Simon Yu

Los Alamos National Laboratory
P. O. Box 1663
Los Alamos, NM 87545
Attn: T. Starke

Mission Research Corporation (8)
1720 Randolph Road, SE
Albuquerque, NM 87106
Attn: M. Campbell
B. B. Godfrey
T. Hughes
G. Kiuttu
D. Mitrovich
R. Richter-Sands
N. Roderick
L. Wright

National Research Council
Ottawa, Canada KIAOR8
Attn: Francois Brunel

DISTRIBUTION (continued):

Naval Research Laboratory (5)
Department of the Navy
Washington, D. C. 20375
Attn: R. Fernsler
B. Hui
G. Joyce
M. Lampe
S. Slinker

Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20910
Attn: H. Uhm

Pulse Sciences, Inc.
14796 Wicks Blvd.
San Leandro, CA 94577
Attn: S. D. Putnam

Science Applications International Corporation (3)
4615 Hawkins, NE
Albuquerque, NM 87109
Attn: M. Hayworth
W. Reinstra
R. Richardson

Science Applications International Corporation (3)
5150 El Camino Real, Suite C-31
Los Altos, CA 94022
Attn: R. L. Feinstein
R. E. Johnston
D. Keeley

Southwest Research Institute
Drawer 28510
6220 Culebra Road
San Antonio, TX 78284
Attn: C. S. Lin

University of Alaska
Geophysical Institute
Fairbanks, AK 99701
Attn: S. I. Akasofu

U. C. L. A. (3)
Physics Department
Los Angeles, CA 90024
Attn: J. Dawson
V. Decyk
R. Sydora

DISTRIBUTION (continued):

University of Texas at Austin (2)
Institute for Fusion Studies
Austin, TX 78712
Attn: J. N. Leboeuf
F. Tajima

1000	Vice President
1200	J. P. VanDevender
1230	J. E. Powell
1231	J. R. Lee
1240	K. R. Prestwich
1241	J. R. Freeman
1241	R. S. Coats
1241	B. M. Marder
1241	K. J. O'Brien
1241	C. L. Olson
1241	J. W. Poukey
1241	D. B. Seidel
1241	J. S. Wagner (15)
1244	R. A. Gerber
1245	J. J. Ramirez
1248	M. T. Buttram
1250	T. H. Martin
1260	D. L. Cook
1261	M. J. Clauser
1265	J. P. Quintenz
1270	R. B. Miller
1272	D. E. Hasti
1272	C. A. Frost
1272	G. T. Leifeste
1272	M. G. Mazarakis
3141	S. A. Landenberger (5)
3151	W. L. Garner (3)
3154-3	C. H. Dalin for DOE/OSTI (32)
8024	P. W. Dean (1)