Programmer Document for TDPACK

TDPACK Table of Contents

Point-Transforming Routines:
- TDPRPT - from 3-space to the projection plane
- TDPRPA - from the plane of the reference parallelogram to the projection plane
- TDPRPI - from the projection plane to the plane of the reference parallelogram

Line-Drawing Routines:
- TDLINE - draw the projection of a line in 3-space
- TDLNPA - draw the projection of a line in the plane of the reference parallelogram

Grid-Drawing Routines:
- TDGRDS - draw the grids on all the faces of a box in 3-space
- TDGRID - draw the grid on a particular face of a box in 3-space

Label-Drawing Routines:
- TDLBLS - draw labels for all faces of a box in 3-space
- TDLBLA - draw labels for a particular face of a box in 3-space
- TDPLCH - draw characters in the plane of the reference parallelogram

Surface-Drawing Routines:
- TDDTRI - draw triangles defined by a triangle list
- TDSTRI - add triangles defining a simple surface to a triangle list
- TDITRI - add triangles defining an isosurface to a triangle list
- TDMTRI - add triangles defining a 3D marker to a triangle list
- TDOTRI - order the triangles in a triangle list for proper rendering
- TDSORT - generic sorting routine

Simplified-Interface Routines:
- TDEZ2D - simplified interface for drawing a surface
- TDEZ3D - simplified interface for drawing an isosurface

Internal Parameters

Parameters Accessed Using TDGETI, TDGETR, TDSETI, and TDSETR
- 'CS1' (Character Size 1)
- 'CS2' (Character Size 2)
- 'FOV' (Field Of View)
- 'HND' (Handedness)
- 'LSU', 'LSV', and 'LSW' (Light Source Position)
- 'SET' (Do-SET-Call Flag)
- 'SHD' (Shading Type Flag)
- 'STE' (Stereo Type Flag)
- 'VPB', 'VPL', 'VPR', and 'VPT' (Viewport Specifiers)

Parameters Accessed in Other Ways
- Rendering-Style Arrays

Examples

Example 1
Example 2
Example 3
C Example 1
TDPACK Demo

Introduction to TDPACK

TDPACK (which stands for "Three Dimensional PACKage") is a collection of routines for projecting objects from a 3-dimensional coordinate system having U, V, and W axes to a 2-dimensional projection plane having X and Y axes and/or for drawing the projections of those objects. This can be referred to somewhat loosely as "drawing objects in three dimensions".

The following topics are addressed below:

Basics

Consider Figure 1 (which is actually a crossed-eye stereo pair):

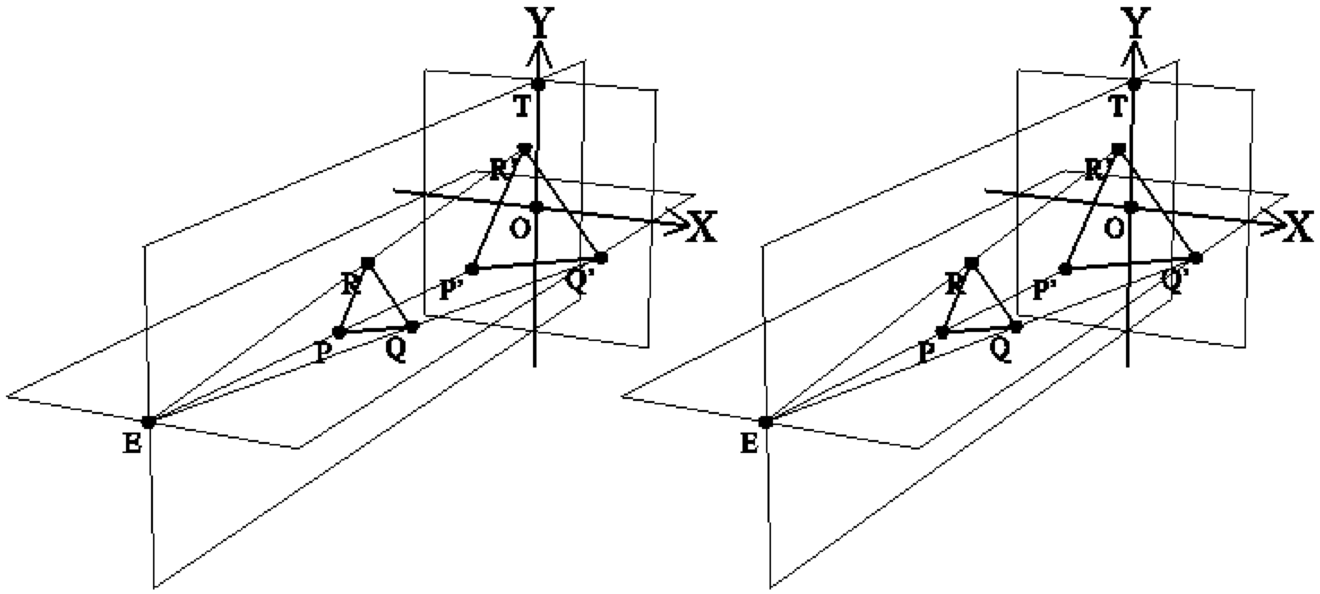


Figure 1. The Projection Scheme

The user of TDPACK is expected to specify the positions of three points in the UVW system: 1) the point E where the eye is; 2) a point O toward which the eye is looking; and 3) a third point T, which may be thought of as "above" the point O. The line of sight is the line from E to O. The projection plane passes through O and is perpendicular to the line of sight. The plane passing through E, O, and T is the plane of bilateral symmetry of the viewer; its intersection with the projection plane is the Y axis. The X axis is perpendicular to the Y axis and passes through O, which is thus the origin of the XY coordinate system. All of these quantities are defined by calling the TDPACK routine TDINIT.

Points

The basic object projected is a point in 3-space. The projection of such a point, P, is the point P' where the extension of the straight line from E to P intersects the projection plane. Figure 1 shows three such points - P, Q, and R - and their projections - P', Q', and R'. The principal point-projection routine in TDPACK is called TDPRPT.

Lines

The basic object drawn is the projection of the straight line segment joining two points in 3-space. Figure 1 shows three such line segments - PQ, QR, and RP - and their projections - P'Q', Q'R', and R'P'. The principal line-drawing routine in TDPACK is called TDLINE.

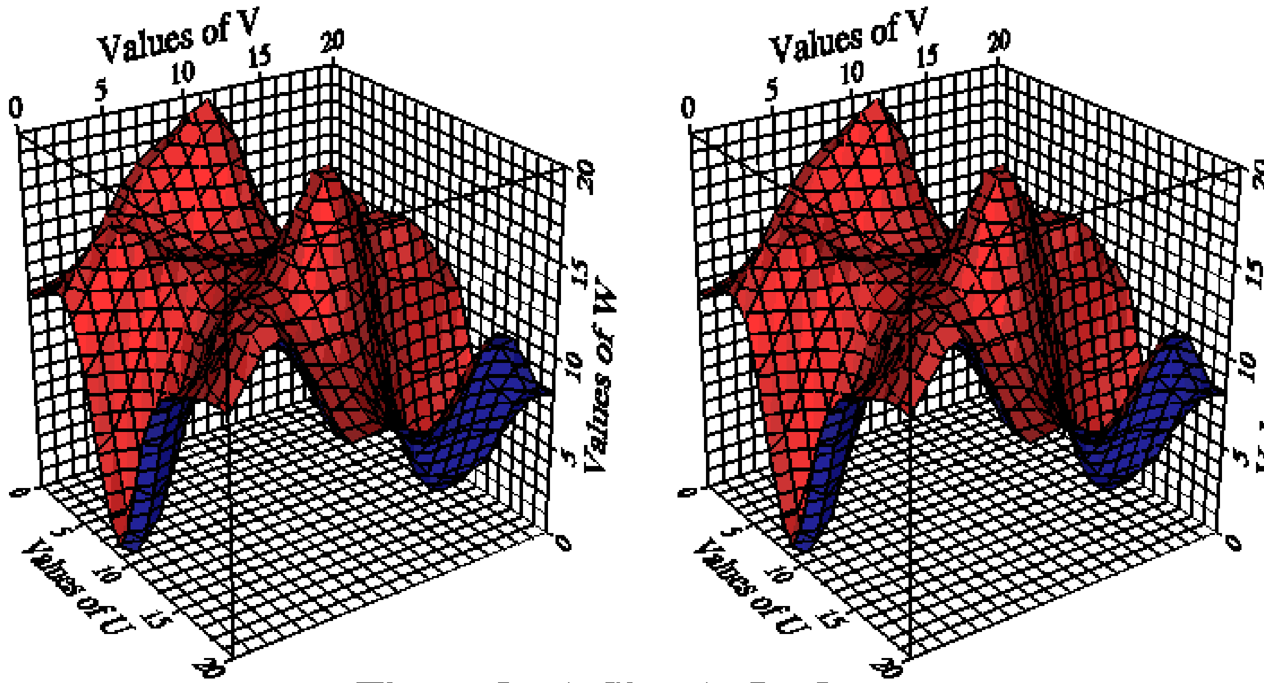
Perimeters, Ticks, Grids, And Labels

A perimeter, with or without tick marks or grid lines, can be drawn for each of the six faces of a box in 3-space. Numeric and informational labels can be drawn on selected edges of such a box. These more complicated objects are either formed from line segments or perhaps (in the case of character strings) from filled polygons. The TDPACK routines involved in this are TDPARA, which allows one to define a "reference parallelogram" within which a planar object is to be positioned, TDPRPA and TDPRPI, which map points from the reference parallelogram to the projection plane and vice-versa, TDLNPA, which draws lines in the reference parallelogram, TDGRDS and TDGRID, which draw grids, TDLBLS and TDLBLA, which draw labels, and TDPLCH, which just draws character strings.

Surfaces

TDPACK is also capable of rendering both simple surfaces and isosurfaces. The routines involved in this are TDSTRI, which generates triangles representing a simple surface, TDITRI, which generates triangles representing an isosurface, TDMTRI, which generates triangles representing a marker at a specified point, TDOTRI, which orders a collection of triangles for correct rendering, and TDDTRI. which draws a collection of triangles.

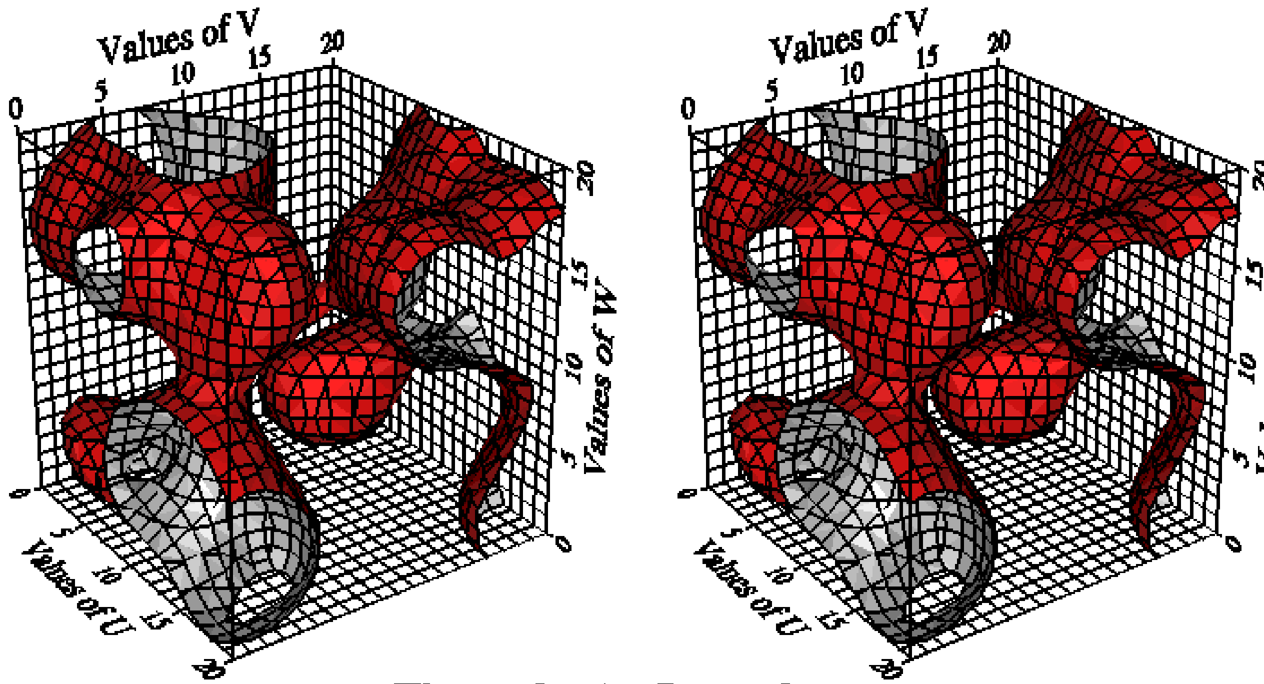
Figure 2 depicts a simple surface, as defined by a function w(u,v):



R L

Figure 2. A Simple Surface

Figure 3 depicts an isosurface, as defined by an equation f(u,v,w)=C:



R L

Figure 3. An Isosurface

A surface is drawn by generating a collection of 3-space triangles whose union represents the surface and then rendering the projections of those triangles using the so-called "painter's algorithm": the triangles are processed in order of decreasing distance from the eye, so that projections of nearer ones are drawn on top of projections of further ones. Each triangle is generally first filled with a user-selected color (if a wire-frame image is desired, this step may be skipped); the fill color can be a specific one chosen by the user or one selected from a range of user-specified colors according to the angle between the line of sight and the plane of the triangle (which results in a shading of the surface). After the triangle is filled, line segments representing its boundary and/or its intersection with planes of constant U, V, and/or W may be drawn in a contrasting color.

The two sides of a given surface (the "top" and "bottom" of a simple surface or the "higher" and "lower" sides of an isosurface) can be shown differently (by making the depiction of each triangle depend on which side of it is visible).

A possibility for the future is a routine making it possible to render images of the Earth (as represented by the EZMAP or World Data Bank II datasets) in a similar manner.

Note that the painter's algorithm for rendering surfaces is quite simple and straightforward to implement, but can lead to rather large metafiles, so it is not recommended to generate large numbers of images in batch mode.

Stereo

Stereo views can easily be done using TDPACK: one just executes all of the object-drawing calls twice, the first time preceded by an initialization call for the left eye and the second time by an initialization call for the right eye. The final argument of TDINIT makes this easy to do.

To see the stereo effect, one must present the stereo pair in such a way that each eye sees the image intended for it. The simplest scheme (but the hardest for most people to use) is to just put the two images side by side. If the left eye's image is to the left and the right eye's image is to the right, the eyes must be aimed at a point beyond the projection plane; because the two images cannot be allowed to overlap and because few people can actually make their eyes diverge very much, the distance between like features of the two views must be less than the distance between the eyes of the viewer (about 2.5 inches) and the size of the resulting image is thus rather limited. If, on the other hand, the left eye's image is put on the right and the right eye's image is put on the left, then the eyes must be aimed at a point between the viewer and the projection plane (i.e., more or less crossed); this is a learned talent and seems to be difficult for most people to do without strain, but the images can then be larger. Figures 1, 2, and 3 are all of this type.

Old-fashioned stereoscopes used non-overlapping images and a set of lenses and/or prisms, allowing the eyes of the user to relax and focus "at infinity" while still allowing the use of quite large images.

One can draw large overlapping images in complementary colors and view them through glasses with a different filter for each eye, each of which transmits only one of the colors in the drawing, but this scheme is quite clumsy and has the disadvantage of not allowing for the full use of color in the images.

Modern stereo devices allow two large images to overlie one another, but still allow only one of the two to be seen by each eye. A typical device uses a stereo monitor that projects the two images in rapidly alternating time slices. The viewer wears glasses that are synchronized with the screen so as to present an image to only one eye at a time and to present each eye with only the images meant for it. This is very easy to use, causes no eyestrain, and gives excellent stereo views.

Internal Parameters

TDPACK has many "internal parameters" - variables in TDPACK common blocks - each of which affects some aspect of the behavior of the package. A typical internal parameter has a three-character mnemonic name and a default integer or real value that results in useful behavior; the value of the parameter can be retrieved by calling either TDGETI or TDGETR and changed by calling either TDSETI or TDSETR. There are some internal parameter arrays that define rendering styles; these are accessed by calling one of the routines TDGTRS (to retrieve a definition) or TDSTRS (to change the definition).

A complete list of the internal parameters may be found here.

The Reference Parallelogram

The positions of some objects (for example, character strings and perimeters) are specified relative to a "reference parallelogram" in 3-space. At any given time, such a parallelogram is defined by internal parameters of TDPACK; there is a routine (TDPARA) whose only function is to allow redefinition of those internal parameters. For example, when one is drawing a grid on a particular face of the object box, one first defines the reference parallelogram to lie in one corner of that face and then draws the grid using calls defining points with reference to that parallelogram.

TDPACK Routines

The package TDPACK contains the following routines:

- Initialization Routines:
  - TDINIT - define eye position, line of sight, up direction, and stereo flag
  - TDPARA - define the reference parallelogram
  - TDCLRS - define colors for TDPACK
- Parameter-Access Routines:
  - TDGETI - get the integer value of an internal parameter
  - TDGETR - get the real value of an internal parameter
  - TDGTRS - get the definition of a specified rendering style
  - TDSETI - set the integer value of an internal parameter
  - TDSETR - set the real value of an internal parameter
  - TDSTRS - define a specified rendering style
- Point-Transforming Routines:
  - TDPRPT - from 3-space to the projection plane
  - TDPRPA - from the plane of the reference parallelogram to the projection plane
  - TDPRPI - from the projection plane to the plane of the reference parallelogram
- Line-Drawing Routines:

TDLINE - draw the projection of a line in 3-space
TDLNPA - draw the projection of a line in the plane of the reference parallelogram
Grid-Drawing Routines:
TDGRDS - draw the grids on all the faces of a box in 3-space
TDGRID - draw the grid on a particular face of a box in 3-space
Label-Drawing Routines:
TDLBLS - draw labels for all faces of a box in 3-space
TDLBLA - draw labels for a particular face of a box in 3-space
TDPLCH - draw characters in the plane of the reference parallelogram
Surface-Drawing Routines:
TDDTRI - draw triangles defined by a triangle list
TDSTRI - add triangles defining a simple surface to a triangle list
TDITRI - add triangles defining an isosurface to a triangle list
TDMTRI - add triangles defining a 3D marker to a triangle list
TDOTRI - order the triangles in a triangle list for proper rendering
TDSORT - generic sorting routine
Simplified-Interface Routines:
TDEZ2D - simplified interface for drawing a surface
TDEZ3D - simplified interface for drawing an isosurface

Initialization Routines

TDINIT

SUBROUTINE TDINIT (UMID, VMID, WMID, UORI, VORI, WORI, UTHI, VTHI, WTHI, OTEP)

```
#include <ncarg/ncargC.h>
void c_tdinit(float umid, float vmid, float wmid, float uori, float vori,
              float wori, float uthi, float vthi, float wthi, float otep)
```

This initialization routine is called to define the position of the eye, the position of the point looked at, which way is up, and whether or not a stereo view is to be done. TDINIT precomputes some quantities in TDPACK common blocks that will subsequently be used in projecting points from 3-space (U, V, and W coordinates) to 2-space (X and Y coordinates).

By default (that is to say, if the internal parameter 'SET' has its default value), TDINIT also calls the SPPS routine SET to define the mapping from the "user" coordinate system (the X/Y system) to the "fractional" coordinate system (in GKS terms, NDC space). This is done in such a way as to show in the projection plane a field of view of 'FOV' degrees. The viewport to be used in the fractional coordinate system is that defined by the values of the internal parameters 'VPL', 'VPR', 'VPB', and 'VPT'. In some situations, it may be desirable, after calling TDINIT, to call the SPPS routine GETSET to retrieve the arguments with which TDINIT called SET and then recall SET with a different set of arguments.

Normally, the creation of an image with TDPACK starts with a call to TDINIT (perhaps preceded by calls to TDSETI, TDSETR, and/or TDSTRS to reset internal parameters of the package) and continues with calls to draw objects. (Of course, if all of the arguments in a call to TDINIT have the same values as in the last call and the SET call that was done as a result is still in effect, then it's not necessary to repeat the call to TDINIT.)

For stereo views, one calls TDINIT with a negative OTEP, executes the object-drawing calls, calls TDINIT again with a positive OTEP, and then repeats all of the object-drawing calls. The exact way in which stereo views are drawn is also affected (slightly) by the value of the internal parameter 'STE'.

To understand how to specify the values of the TDPACK arguments, it may be helpful to examine this figure.

Arguments

UMID, **VMID**, and **WMID** are input reals defining the position of the eye, (if a single view is being drawn), or of a point midway between the two eyes (if a stereo view is being drawn).

UORI, **VORI**, and **WORI** are input reals defining the position of the point the eye is looking at. That point defines the origin of the XY projection plane.

UTHI, **VTHI**, and **WTHI** are input reals defining the position of the third point needed to completely specify the orientation of the projection plane.

OTEP ("Offset To Eye Position") is an input real which is non-zero if and only if a stereo view is to be drawn. Make the value negative to draw a left-eye view, positive to draw a right-eye view. The magnitude of OTEP is the distance from either eye to the point midway between the eyes; if R represents the approximate distance from the eye to the objects being drawn, then OTEP may be set using a statement like

```
OTEP = (+ or -) R * TAN(.017453292519943*ANGD/2.)
```

where the constant (.01745...) is just pi/180 and ANGD is the desired difference in the angle between the two views, in degrees; use a value of about 1 or 2 degrees for ANGD.

Note that, if the origin is approximately in the middle of the objects being viewed, then it's probably appropriate to use a value of R computed as follows:

```
R = SQRT((UMID-UORI)**2+(VMID-VORI)**2+(WMID-WORI)**2)
```

TDPARA

SUBROUTINE TDPARA (UA00, VA00, WA00, UV10, VV10, WV10, UV01, VV01, WV01)

```
#include <ncarg/ncargC.h>
void c_tdpara(float ua00, float va00, float wa00, float uv10, float vv10, float wv10
                                                  float uv01, float vv01, float wv01
```

This routine may be thought of as an initialization routine or as just a routine to access certain internal values; it simply transfers the values of its arguments into TDPACK labelled common blocks for later use by other TDPACK routines. These values define a "reference parallelogram". Each point in the plane in which the parallelogram lies can be identified by specifying its "parallelogram coordinates": a pair of reals, one of which may be thought of as an "X" coordinate and the other of which may be thought of as a "Y" coordinate.

Calls to TDPARA may be positioned without regard to calls to TDINIT because the two routines do not affect one another. Redefining the reference parallelogram affects the behavior of each of the routines TDPRPA, TDPRPI, TDGRID, TDLBLA, and TDPLCH, each of which makes use of parallelogram coordinates in some way. Be aware that each of the routines TDGRDS and TDLBLS calls TDPARA to redefine the reference parallelogram and neither of them restores the original definition when it is done.

If a point has "parallelogram coordinates" (XIPA,YIPA), then its actual 3-space coordinates are given by the following equations:

```
U = UA00+XIPA*UV10+YIPA*UV01
V = VA00+XIPA*VV10+YIPA*VV01
W = WA00+XIPA*WV10+YIPA*WV01
```

The point with parallelogram coordinates (0,0) is in what might be thought of as the "lower left" corner of the parallelogram, while the point with parallelogram coordinates (1,1) is in what might be thought of as the "upper right" corner of the parallelogram. **Any** point of the plane in which the reference parallelogram lies can be identified using its parallelogram coordinates, not just the points inside the parallelogram itself.

Note that, although the reference parallelogram doesn’t have to be rectangular, a non-rectangular one is probably not very useful; in fact, a square one defined by unit vectors is probably best, particularly if one is drawing characters in the plane of the reference parallelogram. For example, suppose that you want to write the characters "THE U/V PLANE" in that part of the U/V plane with U values between 0 and 100 and V values between 0 and 200; it is probably best, in this case, to use a reference parallelogram with an origin at (0,0,0), an "X" side with components (1,0,0) and a "Y" side with components (0,1,0). Then, in the call to TDPLCH, one can place the character string at parallelogram coordinates (50,100) and use a character size of 1.5. If one used a reference parallelogram with an origin at (0,0,0), an "X" side with components (100,0,0) and a "Y" side with components (0,200,0), one could place the character string at parallelogram coordinates (.5,.5) and use a character size of .015, but the characters written would be twice as high as they are wide, which is undesirable.

Arguments

UA00, **VA00**, and **WA00** are input reals defining the position of the "origin" of the parallelogram: the point with parallelogram coordinates (0,0).

UV10, **VV10**, and **WV10** are input reals defining the U, V, and W components of the vector from the origin of the parallelogram to the point with parallelogram coordinates (1,0).

UV01, **VV01**, and **WV01** are input reals defining the U, V, and W components of the vector from the origin of the parallelogram to the point with parallelogram coordinates (0,1).

TDCLRS

SUBROUTINE TDCLRS (IWID, IBOW, SHDE, SHDR, IOFC, IOLC, ILMT)

```
#include <ncarg/ncargC.h>
void c_tdclrs(int iwid, int ibow, float shde, float shdr, int iofc, int iolc, int il
```

This routine is called to do the GKS calls defining the following colors on a specified workstation (NSHD is equal to IOLC-IOFC+1, the number of elements in each block of color shades):

```
Color 0 = the background color (black or white, depending on IBOW)
Color 1 = the foreground color (white or black, depending on IBOW)
Color 2 = pure red (RGB values 1,0,0)
Color 3 = pure green (RGB values 0,1,0)
Color 4 = pure blue (RGB values 0,0,1)
Color 5 = pure cyan (RGB values 0,1,1)
Color 6 = pure magenta (RGB values 1,0,1)
Color 7 = pure yellow (RGB values 1,1,0)
Colors IOFC through IOLC = gray shades, from white to black
Colors IOFC+NSHD through IOLC+NSHD = shades of gray
Colors IOFC+2*NSHD through IOLC+2*NSHD = shades of red
Colors IOFC+3*NSHD through IOLC+3*NSHD = shades of green
Colors IOFC+4*NSHD through IOLC+4*NSHD = shades of blue
Colors IOFC+5*NSHD through IOLC+5*NSHD = shades of cyan
Colors IOFC+6*NSHD through IOLC+6*NSHD = shades of magenta
Colors IOFC+7*NSHD through IOLC+7*NSHD = shades of yellow
```

The colors defined by calling TDCLRS may be used for any purpose, but they are particularly useful when calling TDPACK routines to render surfaces.

Arguments

IWID is an input expression of type INTEGER specifying the workstation identifier.

IBOW is an input expression of type INTEGER specifying the basic color scheme (white on black or black on white). If IBOW is 0, the foreground color is white and the background color is black; if IBOW is non-zero, the opposite is true.

SHDE and **SHDR** are input expressions of type REAL, each between 0 and 1, inclusive, specifying how color shades are to be generated. Values of SHDE near 0 call for more intense shades to be used, while values near 1 call for more nearly pastel shades to be used. Values of SHDR near 0 say that a narrower range of shades is to be used, while values near 1 say that a broader range of shades is to be used.

IOFC and **IOLC** are input expressions of type INTEGER specifying the first and last integers in a block of color indices to be used for NSHD shades of gray ranging from pure white to pure black (where NSHD=IOLC-IOFC+1). The next NSHD indices (in numerical order) will be used for the shades of gray selected by SHDE and SHDR; the next NSHD indices after that for selected shades of red, the next NSHD indices after that for selected shades of green, and so on.

ILMT is an input expression of type INTEGER. Using a value between 1 and 7, inclusive, says that only that many blocks of NSHD indices will be defined. For example, if ILMT has the value 4, only the black-to-white scale and the shades of gray, red, and green will be generated; shades of blue, cyan, magenta, and yellow will not be. (This allows one to have more shades of each color at the expense of using fewer colors.) Using a value of ILMT less than 1 or greater than 7 will result in all 8*NSHD sets of color shades being defined.

Parameter-Access Routines

TDGETI

SUBROUTINE TDGETI (PNAM, IVAL)

```
#include <ncarg/ncargC.h>
void c_tdgeti(char *pnam, int *ival)
```

Retrieve the value of an internal parameter. (For names and descriptions of all the internal parameters, look here).

Arguments

PNAM is an input character string, the first three characters of which is the name of an internal parameter of TDPACK whose value is to be retrieved. It is recommended that the other characters of PNAM be used to describe the internal parameter; for example, use 'FOV - FIELD OF VIEW, IN DEGREES' instead of just 'FOV'.

IVAL is an output variable of type INTEGER in which the value of the internal parameter is to be returned. If the internal parameter is intrinsically of type REAL and has the value RVAL, then INT(RVAL) is returned in IVAL.

TDGETR

SUBROUTINE TDGETR (PNAM, RVAL)

```
#include <ncarg/ncargC.h>
void c_tdgetr(char *pnam, float *rval)
```

Retrieve the value of an internal parameter. (For names and descriptions of all the internal parameters, look here).

Arguments

PNAM is an input character string, the first three characters of which is the name of an internal parameter of TDPACK whose value is to be retrieved. It is recommended that the other characters of PNAM be used to describe the internal parameter; for example, use 'FOV - FIELD OF VIEW, IN DEGREES' instead of just 'FOV'.

RVAL is an output variable of type REAL in which the value of the internal parameter is to be returned. If the internal parameter is intrinsically of type INTEGER and has the value IVAL, then REAL(IVAL) is returned.

TDGTRS

SUBROUTINE TDGTRS (IRST, IFC1, IFC2, IFC3, IFC4, ILC1, ILC2, ILTD, USTP, VSTP, WSTP)

```
#include <ncarg/ncargC.h>
void c_tdgtrs(int irst, int *ifc1, int *ifc2, int *ifc3, int *ifc4, int *ilc1,
              int *ilc2, int *iltd, float *ustp, float *vstp, float *wstp)
```

Get the values defining a selected rendering style. (The "rendering styles" affect the appearance of the triangles drawn by a call to the routine TDDTRI.)

Arguments

IRST is an input integer specifying the index of the particular rendering style which is to be retrieved.

IFC1, **IFC2**, **IFC3**, **IFC4**, **ILC1**, **ILC2**, **ILTD**, **USTP**, **VSTP**, and **WSTP** are described here.

TDSETI

SUBROUTINE TDSETI (PNAM, IVAL)

```
#include <ncarg/ncargC.h>
void c_tdseti(char *pnam, int ival)
```

Set the value of an internal parameter. (For names and descriptions of all the internal parameters, look here).

Arguments

PNAM is an input character string, the first three characters of which is the name of an internal parameter of TDPACK whose value is to be changed. It is recommended that the other characters of PNAM be used to describe the internal parameter; for example, use 'FOV - FIELD OF VIEW, IN DEGREES' instead of just 'FOV'.

IVAL is an input expression of type INTEGER whose value is the desired new value of the internal parameter. If the internal parameter is intrinsically of type REAL, it will be given the value

REAL(IVAL).

TDSETR

SUBROUTINE TDSETR (PNAM, RVAL)

```
#include <ncarg/ncargC.h>
void c_tdsetr(char *pnam, float rval)
```

Set the value of an internal parameter. (For names and descriptions of all the internal parameters, look here).

Arguments

PNAM is an input character string, the first three characters of which is the name of an internal parameter of TDPACK whose value is to be changed. It is recommended that the other characters of PNAM be used to describe the internal parameter; for example, use 'FOV - FIELD OF VIEW, IN DEGREES' instead of just 'FOV'.

RVAL is an input expression of type REAL whose value is the desired new value of the internal parameter. If the internal parameter is intrinsically of type INTEGER, it will be given the value INT(RVAL).

TDSTRS

SUBROUTINE TDSTRS (IRST, IFC1, IFC2, IFC3, IFC4, ILC1, ILC2, ILTD, USTP, VSTP, WSTP)

```
#include <ncarg/ncargC.h>
void c_tdstrs(int irst, int ifc1, int ifc2, int ifc3, int ifc4, int ilc1,
              int ilc2, int iltd, float ustp, float vstp, float wstp)
```

Set the values defining a selected rendering style. (The "rendering styles" affect the appearance of the triangles drawn by a call to the routine TDDTRI.)

Arguments

IRST is an input integer specifying the index of the particular rendering style which is to be changed.

IFC1, **IFC2**, **IFC3**, **IFC4**, **ILC1**, **ILC2**, **ILTD**, **USTP**, **VSTP**, and **WSTP** are described here.

Point-Transforming Routines

TDPRPT

SUBROUTINE TDPRPT (UI3D, VI3D, WI3D, XI2D, YI2D)

```
#include <ncarg/ncargC.h>
void c_tdprpt(float ui3d, float vi3d, float wi3d, float *xi2d, float *yi2d)
```

Given the coordinates of a point in 3-space, get the coordinates of its projection in the projection plane.

Arguments

UI3D, **VI3D**, and **WI3D** are input reals specifying the coordinates of a point in 3-space.

XI2D and **YI2D** are output reals specifying the coordinates of the projection in 2-space.

TDPRPA

SUBROUTINE TDPRPA (XIPA, YIPA, XI2D, YI2D)

```
#include <ncarg/ncargC.h>
void c_tdprpa(float xipa, float yipa, float *xi2d, float *yi2d)
```

Given the parallelogram coordinates of a point, get the coordinates of its projection in the projection plane. This routine is essentially the inverse of the routine TDPRPI.

Arguments

XIPA and **YIPA** are input reals specifying the parallelogram coordinates of a point. (The parallelogram is as defined by a prior call to TDPARA.)

XI2D and **YI2D** are output reals specifying the coordinates of the projection of the point in the projection plane.

TDPRPI

SUBROUTINE TDPRPI (XI2D, YI2D, XIPA, YIPA)

```
#include <ncarg/ncargC.h>
void c_tdprpi(float xi2d, float yi2d, float *xipa, float *yipa)
```

Given a point in the projection plane, retrieve the parallelogram coordinates of that point in the reference parallelogram (as defined by the last call to TDPARA) that projects into it. This routine is essentially the inverse of the routine TDPRPA.

Arguments

XI2D and **YI2D** are input reals specifying the coordinates of a point in the projection plane.

XIPA and **YIPA** are output reals specifying the parallelogram coordinates of that point in the reference parallelogram that projects into (XI2D,YI2D).

Line-Drawing Routines

TDLINE

SUBROUTINE TDLINE (UCP1, VCP1, WCP1, UCP2, VCP2, WCP2)

```
#include <ncarg/ncargC.h>
void c_tdline(float ucp1, float vcp1, float wcp1, float ucp2, float vcp2, float wcp2
```

Draw the projection of a line joining two points in 3-space.

Arguments

UCP1, **VCP1**, and **WCP1** are input reals - the 3-space coordinates of the first 3-space point.

UCP2, **VCP2**, and **WCP2** are input reals - the 3-space coordinates of the second 3-space point.

TDLNPA

SUBROUTINE TDLNPA (XCP1, YCP1, XCP2, YCP2)

```
#include <ncarg/ncargC.h>
void c_tdlnpa(float xcp1, float ycp1, float xcp2, float ycp2)
```

Draw the projection of a line joining two points in the reference parallelogram (as defined by the last call to TDPARA).

Arguments

XCP1 and **YCP1** are input reals - the parallelogram coordinates of the first point in the reference parallelogram.

XCP2 and **YCP2** are input reals - the parallelogram coordinates of the second point in the reference parallelogram.

Grid-Drawing Routines

TDGRDS

SUBROUTINE TDGRDS (UMIN, VMIN, WMIN, UMAX, VMAX, WMAX, USTP, VSTP, WSTP, IGRT, IHID)

```
#include <ncarg/ncargC.h>
void c_tdgrds(float umin, float vmin, float wmin, float umax, float vmax, float wmax
              float ustp, float vstp, float wstp, int igrt, int ihid)
```

Draw perimeters, optionally with ticks or grid lines, on the six sides of a box. This routine calls TDPARA and will therefore change the definition of the reference parallelogram.

Arguments

UMIN, **VMIN**, **WMIN**, **UMAX**, **VMAX**, and **WMAX** are real input values, each of which specifies one of the coordinate values defining the box in 3-space. The names of these should make it clear what they are.

USTP, **VSTP**, and **WSTP** are real input values specifying step sizes between ticks or grid lines in the U direction, the V direction, and the W direction, respectively. If one of these values is less than or equal to zero, the ticks or grid lines in the associated direction are omitted.

IGRT is an integer input value of the form 10*IGRN+IGRF, where IGRN is a value specifying what to draw on the near sides of the box and IGRF is a value specifying what to draw on the far sides of the box, where "near" and "far" are as defined by the current line of sight. Each of IGRN and IGRF can have one of the values 0 (draw nothing), 1 (draw just a perimeter), 2 (draw a perimeter with inward-pointing ticks), or 3 (draw a perimeter with a grid). For example, to draw grids on the far side of the box and just perimeters on the near sides of the box, use IGRT = 13.

IHID is an integer input value set to 0 to draw only those sides of the box that cannot be hidden by something inside the box or to 1 to draw only those sides of the box that can be hidden by something inside the box. Standard operating procedure is to call TDGRDS before drawing surfaces inside a box, with IHID set to 1, and then call it again after drawing surfaces inside a box, with IHID set to 0.

TDGRID

SUBROUTINE TDGRID (XBEG, XSTP, NOXS, YBEG, YSTP, NOYS, IGRD)

```
#include <ncarg/ncargC.h>
void c_tdgrid(float xbeg, float xstp, int noxs, float ybeg, float ystp,
              int noys, int igrd)
```

Draw a perimeter, optionally with ticks or grid lines, around the edges of the reference parallelogram defined by the last call to TDPARA. This routine is normally called indirectly (by virtue of a call to TDGRDS), but there is no reason why it should not be called directly.

Arguments

XBEG, **XSTP**, and **NOXS** define where ticks or grid lines are to be drawn along the "X" axis of the parallelogram (at XBEG, XBEG+XSTP, XBEG+2*XSTP, ... XBEG+NOXS*XSTP). XBEG and XSTP are input reals and NOXS is an input integer.

YBEG, **YSTP**, and **NOYS** define where ticks or grid lines are to be drawn along the "Y" axis of the parallelogram (at YBEG, YBEG+YSTP, YBEG+2*YSTP, ... YBEG+NOYS*YSTP). YBEG and YSTP are input reals and NOYS is an input integer.

IGRD defines what is to be drawn and has one of the values 1 (draw just a perimeter), 2 (draw a perimeter with inward-pointing ticks), or 3 (draw a perimeter with a grid).

Label-Drawing Routines

TDLBLS

SUBROUTINE TDLBLS (UMIN, VMIN, WMIN, UMAX, VMAX, WMAX, UNLB, VNLB, WNLB, UILB, VILB, WILB, IPCK)

```
#include <ncarg/ncargC.h>
void c_tdlbls(float umin, float vmin, float wmin, float umax, float vmax,
              float wmax, char* unlb, char* vnlb, char* wnlb, char* uilb,
              char* vilb, char* wilb, int ipck)
```

Draw labels on selected sides of a projected box. This routine calls TDPARA and will therefore change the definition of the reference parallelogram; it also changes the value of the internal parameter 'CS2'.

Arguments

UMIN, **VMIN**, **WMIN**, **UMAX**, **VMAX**, and **WMAX** are input expressions of type REAL, each of which specifies one of the coordinate values defining the box in 3-space. The names of these should make it clear what they are.

UNLB, **VNLB**, and **WNLB** are character strings containing numeric labels to be placed on a U axis, a V axis, or a W axis, respectively. The labels need not be in any particular order, but they have to be separated by blanks and each has to be readable using a FORTRAN format of the form "En.0", where "n" is the length of the label.

UILB, **VILB**, and **WILB** are character strings containing informational labels for a U axis, a V axis, or a W axis, respectively.

IPCK is an input integer determining which edges of the box are labelled. If IPCK is zero, all six outer edges are labelled, but if IPCK is non-zero, only three edges are labelled, one set of three if IPCK is negative, a different set if IPCK is positive.

TDLBLA

SUBROUTINE TDLBLA (IAXS, ILBL, NLBL, XAT0, XAT1, YAT0, YAT1, ANGD)

```
#include <ncarg/ncargC.h>
void c_tdlbla(int iaxs, char* ilbl, char* nlbl, float xat0, float xat1,
              float yat0, float yat1, float angd)
```

This routine is called to put labels on a particular edge of a box.

It is assumed that TDPARA has been called to define the reference parallelogram to be a rectangle in 3-space lying in one corner of one face of the box being labelled. The sides of this rectangle are assumed to be vectors of length 1 (that is to say, the rectangle defines a unit square within that face of the box). It is also assumed that the value of the internal parameter 'CS2' has been set in the same way that TDLBLS would reset it, using a code sequence like

```
    CALL TDGETR ('CS1',CSM1)
    CSM2=CSM1*MIN(UMAX-UMIN,VMAX-VMIN,WMAX-WMIN)
    CALL TDSETR ('CS2',CSM2)
```

(where UMIN, UMAX, VMIN, VMAX, WMIN, and WMAX are as defined for a call to TDLBLS).

Arguments

IAXS is an input integer saying which edge of the face is being labelled (1 => left, 2 => right, 3 => bottom, and 4 => top, where the meanings of "left", "right", "bottom", and "top" are defined by the orientation of the reference parallelogram).

ILBL is an input character string to be used as an informational label. If the string is blank, no informational label is written.

NLBL is an input character string containing numeric labels. The labels need not be in any particular order, but they have to be separated by blanks and each has to be readable using a FORTRAN format of the form "En.0", where "n" is the length of the label. If the string is blank, no informational label is written.

XAT0 and **XAT1** are the real values of "X" associated with the left and right edges of the face being labelled, where "left" and "right" are defined in terms of the current reference parallelogram.

YAT0 and **YAT1** are the real values of "Y" associated with the bottom and top edges of the face being labelled, where "bottom" and "top" are defined in terms of the current reference parallelogram.

ANGD is an input real specifying the angle, in degrees, at which the labels are to be written. This angle is defined with reference to the current reference parallelogram.

TDPLCH

SUBROUTINE TDPLCH (XPOS, YPOS, CHRS, SIZE, ANGD, CNTR)

```
#include <ncarg/ncargC.h>
void c_tdplch(float xpos, float ypos, char *chrs, float size, float angd, float cntr
```

Draw a character string at a specified position in the plane of the reference parallelogram.

Arguments

XPOS and **YPOS** are input reals specifying the parallelogram coordinates of a point relative to which the character string is to be positioned.

CHRS is an input character string, suitable for input to the PLOTCHAR routine PLCHHQ.

SIZE is an input real, specifying the width of the characters to be used as a multiple of the "X" dimension of the reference parallelogram. If the reference parallelogram is not a square, distortion of the characters is to be expected: for example, if the reference parallelogram is twice as wide as it is high, then the same will be true of characters drawn by TDPLCH.

ANGD is an input real specifying the angle, in degrees, at which the character string is to be drawn relative to the reference parallelogram (0 => in the direction of positive X, 90 => in the direction of positive Y, 180 => in the direction of negative X, and so on).

CNTR is an input real specifying how the character string is to be centered relative to the point (XPOS,YPOS) within the reference parallelogram. If CNTR = -1, the leading edge of the character string will be placed at (XPOS,YPOS); if CNTR = 0, the center of the character string will be placed at (XPOS,YPOS); and if CNTR = +1, the trailing edge of the character string will be placed at (XPOS,YPOS). Other values may be used, with the logical results; for example, if CNTR = -.5, the point halfway between the leading edge and the center of the character string will be placed at (XPOS,YPOS).

Surface-Drawing Routines

TDDTRI

SUBROUTINE TDDTRI (RTRI, MTRI, NTRI, ITWK)

```
#include <ncarg/ncargC.h>
void c_tddtri(float *rtri, int mtri, int *ntri, int *itwk)
```

Draw the triangles defined by a triangle list (arguments RTRI, MTRI, and NTRI). The triangles are drawn in the order specified by the contents of the permutation array ITWK.

The exact manner in which each triangle is drawn depends on the rendering style selected for it. Generally, it is first filled with a user-selected color; if a wire-frame image is desired and no lines are to be considered hidden, this step may be skipped. The fill color can be a specific one chosen by the user (sometimes, it is useful to pick the background color) or one selected from a range of user-specified colors according to the angle between a line from the light source to the center of the triangle and the normal to the plane of the triangle (which results in a shading of the surface). After the triangle is filled, line segments representing its boundary and/or its intersection with planes of constant U, V, and/or W may be drawn in a user-specified contrasting color.

Arguments

RTRI is a real input array, dimensioned 10 by MTRI, in which a list of triangles has been stored, probably by means of calls to TDSTRI, TDITRI, and/or TDMTRI, and sorted, probably by means of a call to TDOTRI.

If you put triangles in the triangle list directly, keep in mind the following: For each value of *i* from 1 to NTRI, RTRI(1,*i*), RTRI(2,*i*), and RTRI(3,*i*) are the U, V, and W coordinates of the first vertex of the triangle, RTRI(4,*i*), RTRI(5,*i*), and RTRI(6,*i*) are the U, V, and W coordinates of the second vertex of the triangle, RTRI(7,*i*), RTRI(8,*i*), and RTRI(9,*i*) are the U, V, and W coordinates of the third vertex of the triangle, and RTRI(10,*i*)=REAL(*irst*), where *irst* is the index of the desired rendering style array for the triangle. The order of the vertices is meaningful: they must be given in counter-clockwise order as viewed from the "top". It is better if all of the triangles in a triangle list are about the same size.

MTRI is an input integer specifying the second dimension of RTRI and thus the maximum number of triangles the triangle list will hold.

NTRI is an input integer specifying the number of triangles currently in the triangle list. (It is the user's responsibility to zero this initially; its value is increased by each call to a triangle-generating routine like TDSTRI, TDITRI, or TDMTRI and may be increased by a call to the triangle-ordering routine TDOTRI.)

ITWK is an integer input array, dimensioned at least NTRI, containing a permutation of the integers from 1 to NTRI (probably generated by calling the routine TDOTRI).

TDSTRI

SUBROUTINE TDSTRI (U, NU, V, NV, W, LW1D, RTRI, MTRI, NTRI, IRST)

```
#include <ncarg/ncargC.h>
void c_tdstri(float *u, int nu, float *v, int nv, float *w, int lw1d,
```

```
               float *rtri, int mtri, int *ntri, int irst)
```

Add triangles defining a simple surface to the triangles in a triangle list.

Arguments

U is a real input array, dimensioned NU, in which are stored values of an independent variable *u*. It must be the case that U(1) < U(2) < ... U(NU-1) < U(NU).

NU is an input integer specifying the dimension of the array U.

V is a real input array, dimensioned NV, in which are stored values of an independent variable *v*. It must be the case that V(1) < V(2) < ... V(NV-1) < V(NV).

NV is an input integer specifying the dimension of the array V.

W is a real input array, dimensioned NU by NV and having FORTRAN first dimension LW1D, in which are stored values of a dependent variable *w(u,v)*. The points (((U(I),V(J),W(I,J)),I=1,NU),J=1,NV) define a surface that one wishes to draw.

LW1D is an input integer specifying the FORTRAN first dimension of the array W. It must be the case that LW1D is greater than or equal to NU.

RTRI is a real input/output array, dimensioned 10 by MTRI, in which a list of triangles is stored.

MTRI is an input integer specifying the second dimension of RTRI and thus the maximum number of triangles the triangle list will hold.

NTRI is an input/output integer specifying the number of triangles currently in the triangle list. It is the user's responsibility to zero this initially and its value is increased by each call to a triangle-generating routine like TDSTRI. If NTRI becomes equal to MTRI, TDSTRI does not take an error exit; instead, it just stops generating triangles. Therefore, it's a good idea, after calling TDSTRI, to check the value of NTRI against the dimension MTRI; if they're equal, it probably means that the triangle list filled up and that the rendered surface will be incomplete.

IRST is an input integer specifying the index of the rendering style to to be used for the triangles added to the triangle list by this call. Look here for descriptions of the internal parameters defining the rendering styles, here for a description of the routine used to get the values of these internal parameters, and here for a description of the routine used to set their values.

TDITRI

SUBROUTINE TDITRI (U, NU, V, NV, W, NW, F, LF1D, LF2D, FISO, RTRI, MTRI, NTRI, IRST)

```
#include <ncarg/ncargC.h>
```

```
void c_tditri(float *u, int nu, float *v, int nv, float *w, int nw,
              float *f, int lf1d, int lf2d, float fiso, float *rtri,
              int mtri, int *ntri, int irst)
```

Add triangles defining an isosurface to the triangles in a triangle list.

Arguments

U is a real input array, dimensioned NU, in which are stored values of an independent variable *u*. It must be the case that U(1) < U(2) < ... U(NU-1) < U(NU).

NU is an input integer specifying the dimension of the array U.

V is a real input array, dimensioned NV, in which are stored values of an independent variable *v*. It must be the case that V(1) < V(2) < ... V(NV-1) < V(NV).

NV is an input integer specifying the dimension of the array V.

W is a real input array, dimensioned NW, in which are stored values of an independent variable *w*. It must be the case that W(1) < W(2) < ... W(NW-1) < W(NW).

NW is an input integer specifying the dimension of the array W.

F is a real input array having first and second FORTRAN dimensions LF1D and LF2D, respectively, in which are stored the values of a dependent variable *f(u,v,w)*. F(I,J,K) is the value of the function *f* at the position (U(I),V(J),W(K)); the equation "*f(u,v,w)=FISO*" defines a surface that one wishes to draw.

LF1D and **LF2D** are the first and second FORTRAN dimensions of the array F. It must be the case that LF1D is greater than or equal to NU and that LF2D is greater than or equal to NV.

FISO is that value of the function *f* defining the isosurface to be drawn.

RTRI is a real input/output array, dimensioned 10 by MTRI, in which a list of triangles is stored.

MTRI is an input integer specifying the second dimension of RTRI and thus the maximum number of triangles the triangle list will hold.

NTRI is an input/output integer specifying the number of triangles currently in the triangle list. It is the user's responsibility to zero this initially and its value is increased by each call to a triangle-generating routine like TDITRI. If NTRI becomes equal to MTRI, TDITRI does not take an error exit; instead, it just stops generating triangles. Therefore, it's a good idea, after calling TDITRI, to check the value of NTRI against the dimension MTRI; if they're equal, it probably means that the triangle list filled up and that the rendered surface will be incomplete.

IRST is an input integer specifying the index of the rendering style to to be used for the triangles added to the triangle list by this call. Look here for descriptions of the internal parameters defining the rendering styles, here for a description of the routine used to get the values of these internal parameters, and here for a description of the routine used to set their values.

TDMTRI

SUBROUTINE TDMTRI (IMRK, UMRK, VMRK, WMRK, SMRK, RTRI, MTRI, NTRI, IRST, UMIN, VMIN, WMIN, UMAX, VMAX, WMAX)

```
#include <ncarg/ncargC.h>
void c_tdmtri(int imrk, float umrk, float vmrk, float wmrk, float smrk,
              float *rtri, int mtri, int *ntri, int irst, float umin,
              float vmin, float wmin, float umax, float vmax, float wmax)
```

Add triangles defining a 3D marker to the triangles in a triangle list.

Arguments

IMRK is an input expression of type INTEGER having an absolute value between 1 and 5, inclusive, specifying the type of marker to be generated. If the value of IMRK is less than zero, the triangles will not be clipped against the sides of the data box, otherwise, they will.

UMRK, **VMRK**, and **WMRK** are input expressions of type REAL specifying the 3-space coordinates of the center point of the marker.

SMRK is an input expression of type REAL specifying the radius of the marker in 3-space.

RTRI is a real input/output array, dimensioned 10 by MTRI, in which a list of triangles is stored.

MTRI is an input integer specifying the second dimension of RTRI and thus the maximum number of triangles the triangle list will hold.

NTRI is an input/output integer specifying the number of triangles currently in the triangle list. It is the user's responsibility to zero this initially and its value is increased by each call to a triangle-generating routine like TDMTRI. If NTRI becomes equal to MTRI, TDMTRI does not take an error exit; instead, it just stops generating triangles. Therefore, it's a good idea, after calling TDMTRI, to check the value of NTRI against the dimension MTRI; if they're equal, it probably means that the triangle list filled up and that the rendered marker will be incomplete.

IRST is an input integer specifying the index of the rendering style to to be used for the triangles added to the triangle list by this call. Look here for descriptions of the internal parameters defining the rendering styles, here for a description of the routine used to get the values of these internal parameters, and here for a description of the routine used to set their values.

UMIN, **VMIN**, **WMIN**, **UMAX**, **VMAX**, and **WMAX** are input expressions of type REAL, each of which specifies one of the coordinate values defining the data box in 3-space. The names of these should make it clear what they are.

TDOTRI

SUBROUTINE TDOTRI (RTRI, MTRI, NTRI, RTWK, ITWK, IORD)

```
#include <ncarg/ncargC.h>
void c_tdotri(float *rtri, int mtri, int *ntri, float *rtwk, int *itwk, int iord)
```

This routine, given a list of NTRI triangles in the array RTRI and a real scratch array RTWK of length at least MTRI x 2 , determines the order in which the triangles are to be rendered and returns a permutation of the integers from 1 to NTRI in the array ITWK, defining that permutation.

The caller may select any of three ways in which the triangles are to be ordered, the first two of which are essentially identical: When the argument IORD is given the value 0, the distances of the midpoints of the triangles from the viewpoint are computed and the triangles are sorted by decreasing order of these distances. When IORD is given the value -1, the result is the same, except that the distances of the farthest points of the triangles from the viewpoint are computed and the triangles are put in decreasing order of those distances. Both of these possibilities are appropriate for situations in which the triangles represent smooth surfaces that do not intersect each other or themselves; the occasional small errors in the resulting rendering order should be acceptable.

If any of the triangles in the list intersect each other or if the surfaces being depicted are too rough, then the third option should be used: When IORD is given the value +1, TDOTRI executes an algorithm taken from the reference "Computer Graphics Principles and Practice", by Foley and Van Dam. It starts by ordering the triangles as if IORD had the value -1 (using distances of the far points of the triangles from the viewpoint), but then it checks for situations in which this ordering is in error and fixes the errors. Executing this algorithm can be time-consuming, so it should not be done unless it is really necessary; one possible way to proceed might be to use IORD = -1 while checking out a code and then use IORD = +1 only when doing final plots.

Sometimes, when IORD = +1, triangles must be broken into smaller triangles, thereby increasing the total number of triangles in RTRI. If, as a result of this, NTRI becomes equal to MTRI, no error exit is taken; instead, TDOTRI just returns control to the caller. Therefore, it’s a good idea, after calling TDOTRI, to check the value of NTRI against the dimension MTRI; if they’re equal, it probably means that the triangle list filled up and that using the permutation returned in ITWK will result in an incorrect rendering of the triangles.

Arguments

RTRI is a real input/output array, dimensioned 10 by MTRI, in which a list of triangles has been stored, probably by means of calls to TDSTRI and/or TDITRI. As described above, the number of triangles in the list may increase as a result of calling TDOTRI.

MTRI is an input integer specifying the second dimension of RTRI and thus the maximum number of triangles the triangle list will hold.

NTRI is an input/output integer specifying the number of triangles currently in the triangle list. It is the user’s responsibility to zero this initially; its value is increased by each call to a triangle-generating routine like TDSTRI or TDITRI and may be increased by a call to TDOTRI.

RTWK is a real scratch array, dimensioned at least MTRIx2.

ITWK is an integer output array, dimensioned at least MTRI, returned containing a permutation of the integers from 1 to NTRI, specifying the order in which the triangles ought to be rendered.

IORD is an integer input expression that says how the triangles are to be ordered. The value 0 implies ordering by decreasing distance of the triangle midpoints from the eye, -1 implies ordering by decreasing distance of the triangle farpoints from the eye, and +1 implies ordering by decreasing distance of the triangle farpoints from the eye, with adjustments made by running an algorithm from the reference "Computer Graphics Principles and Practice", by Foley and Van Dam.

TDSORT

SUBROUTINE TDSORT (RWRK, NRWK, IORD, IWRK)

```
#include <ncarg/ncargC.h>
void c_tdsort(float *rwrk, int nwrk, int iord, int *iwrk)
```

This is an internal routine used for sorting. It is not intended that a user should call this routine directly, but doing so should do no harm. Given an array of NRWK reals in an array RWRK and an "ordering flag" IORD, TDSORT returns a permutation vector IWRK such that, for every I and J such that 1.LE.I.LE.J.LE.NWRK, if IORD is zero, then RWRK(IWRK(I)).LE.RWRK(IWRK(J)), else RWRK(IWRK(I)).GE.RWRK(IWRK(J)).

Simplified-Interface Routines

TDEZ2D

SUBROUTINE TDEZ2D(NX,NY,X,Y,Z,RMULT,THETA,PHI,IST)

```
#include <ncarg/ncargC.h>
void c_tdez2d (int nx, int ny, float *x, float *y, float *z,
               float rmult, float theta, float phi, int ist)
```

This subroutine uses other entries in TDPACK to draw a surface plot. TDEZ2D is meant to be a simplified interface for the purpose of quickly drawing a plot. TDEZ2D sacrifices much of the flexibility and functionality of TDPACK in its attempt to be simple.

Arguments

NX, **NY**, **X**, **Y**, and **Z** define the input data for the surface. X is dimensioned for NX, Y is dimensioned for NY, and Z(I,J) is the data value at (X(I),Y(J)) for I=1,NX and J=1,NY.

RMULT, **THETA**, and **PHI** are real input values specifying an eye position (the point from which the surface will be viewed), and are defined as follows:

- RMULT is a multiplier of the diagonal length (DL) of the smallest box containing the surface to be drawn.
- THETA is an angle (in degrees) in the XY plane measured positive counter-clockwise from the X axis
- PHI is an angle (in degrees) measured from the positive Z axis toward the XY plane.

Thus, the coordinate (RMULT*DL,THETA,PHI) is the spherical coordinate for the eye position. If RMULT = THETA = PHI = 0., a default eye position (2.5,-55.,70.) is chosen.

TDEZ2D calculates the point looked at to be the midpoint of the surface.

IST is a style index defining the colors used to shade the surface. The legal values for IST are as follows:

IST	Description
1	produce a wire-frame surface
2	use gray shades underneath; gray shades on top
3	use gray shades underneath; red shades on top
4	use gray shades underneath; green shades on top
5	use gray shades underneath; blue shades on top
6	use gray shades underneath; cyan shades on top
7	use gray shades underneath; gray shades on top
8	use gray shades underneath; magenta shades on top

If IST is positive, then black is used for the background color and white for the foreground color; if IST is the negative of any of the above values, then white is used for the background color and black for the foreground color. If IST falls outside of the legal range, it is defaulted to 6.

When TDEZ2D is called, a color table is defined for all open workstations that implements IST as described above. *This color table will supersede any color table that has been previously defined.* The color table that is defined is:

Color index	Colors
0	black if IST is positive; white if IST is negative
1	white if IST is positive; black if IST is negative
2	red
3	green
4	blue
5	cyan
6	magenta
7	yellow
8-37	grayscale from white to black
38-67	shades of gray
68-97	shades of red
98-127	shades of green
128-157	shades of blue
158-187	shades of cyan
188-217	shades of magenta
218-247	shades of yellow

TDEZ2D does *not* call FRAME.

If the image is too small, decrease the size of RMULT; if the image is too large, increase the size of RMULT.

Example:

```
CALL TDEZ2D(NX,NY,X,Y,Z,0.,0.,0.,6)
```

will draw a surface in shades of cyan with a black background and with a default eye position selected.

TDEZ3D

SUBROUTINE TDEZ3D(NX,NY,NZ,X,Y,Z,U,VALUE,RMULT,THETA,PHI,IST)

```
#include <ncarg/ncargC.h>
```

```
void c_tdez3d (int nx, int ny, int nz, float *x, float *y, float *z,
               float *u, float value, float rmult, float theta, float phi,
               int ist)
```

This subroutine uses other entries in TDPACK to draw an isosurface plot. TDEZ3D is meant to be a simplified interface for the purpose of quickly drawing a plot. TDEZ3D sacrifices much of the flexibility and functionality of TDPACK in its attempt to be simple.

Arguments

NX, **NY**, **NZ**, **X**, **Y**, **Z**, and **U** define the input data for the isosurface where **VALUE** is the isovalue. X is dimensioned for NX, Y is dimensioned for NY, Z is dimensioned for NZ, and U(I,J,K) is the data value at (X(I),Y(J),Z(K)) for I=1,NX, J=1,NY, and K=1,NZ.

RMULT, **THETA**, and **PHI** are real input values specifying an eye position (the point from which the surface will be viewed), and are defined as follows:

- RMULT is a multiplier of the diagonal length (DL) of the grid box specified ty X, Y, and Z.
- THETA is an angle (in degrees) in the XY plane measured positive counter-clockwise from the X axis
- PHI is an angle (in degrees) measured from the positive Z axis toward the XY plane.

Thus, the coordinate (RMULT*DL,THETA,PHI) is the spherical coordinate for the eye position. If RMULT = THETA = PHI = 0., a default eye position (2.5,-55.,70.) is chosen.

TDEZ3D calculates the point looked at to be the midpoint of the grid box specified by X, Y, and Z.

IST is a style index defining the colors used to shade the surface. The legal values for IST are:

IST	Description
1	produce a wire-frame surface
2	use gray shades underneath; gray shades on top
3	use gray shades underneath; red shades on top
4	use gray shades underneath; green shades on top
5	use gray shades underneath; blue shades on top
6	use gray shades underneath; cyan shades on top
7	use gray shades underneath; gray shades on top
8	use gray shades underneath; magenta shades on top

If IST is positive, then black is used for the background color and white for the foreground color; if IST is the negative of any of the above values, then white is used for the background color and black for the foreground color. If IST falls outside of the legal range, it is defaulted to 6.

When TDEZ3D is called, a color table is defined for all open workstations that implements IST as described above. *This color table will supersede any color table that has been previously defined.* The color table that is defined is:

Color index	Colors
0	black if IST is positive; white if IST is negative
1	white if IST is positive; black if IST is negative
2	red
3	green
4	blue
5	cyan
6	magenta
7	yellow
8-37	grayscale from white to black
38-67	shades of gray
68-97	shades of red
98-127	shades of green
128-157	shades of blue
158-187	shades of cyan
188-217	shades of magenta
218-247	shades of yellow

TDEZ3D does *not* call FRAME.

If the image is too small, decrease the size of RMULT; if the image is too large, increase the size of RMULT.

Example:

```
CALL TDEZ3D(NX,NY,NZ,X,Y,Z,U,1.,0.,0.,0.,6)
```

will draw an isosurface (using an isovalue of 1.) in shades of cyan with a black background and with a default eye position selected.

TDPACK Internal Parameters

Each "internal parameter" of TDPACK is a variable, in a TDPACK common block, whose value has a well-defined effect on the behavior of one or more TDPACK routines. All of the internal parameters have default values. Parameter-access routines allow one to retrieve the value of an internal parameter or to change its value. The value of a parameter may be changed at any time prior to calling a routine whose behavior is affected by it.

The package TDPACK has the following internal parameters:

Parameters Accessed Using TDGETI, TDGETR, TDSETI, and TDSETR

Most of TDPACK’s internal parameters are identified by a three-character mnemonic name, have a single real or integer value, and are accessed by calling one of the routines TDGETI or TDGETR (to get a parameter value) or TDSETI or TDSETR (to set a parameter value); these internal parameters are described below, in alphabetical order.

’CS1’ (Character Size 1)

This is a multiplier for the sizes of all characters drawn by the routine TDLBLS.

The default value of ’CS1’ is 1.

'CS2' (Character Size 2)

This is a multiplier for the sizes of all characters drawn by the routine TDLBLA. Exactly how it is set depends on how the current reference parallelogram has been defined. If the current reference parallelogram has sides of length 1, then it is appropriate to set

```
'CS2'='CS1'*MIN(UMAX-UMIN,VMAX-VMIN,WMAX-WMIN)
```

where UMIN, UMAX, VMIN, VMAX, WMIN, and WMAX are defined like the arguments of TDLBLS. Note that, if you call TDLBLS, it resets 'CS2' in this fashion, and it does **not** restore the original value before return.

The default value of 'CS2' is 1.

'FOV' (Field Of View)

When 'SET' has a non-zero value, specifying that TDINIT is to call the SPPS routine SET, then 'FOV' specifies the desired field of view, in degrees, to be seen in the projection plane; its value is used to compute appropriate values for the arguments in the call to SET.

The default value of 'FOV' is 20.

'HND' (Handedness)

A zero value specifies that the UVW coordinate system is to be right-handed (the default), while a non-zero value says that it is to be left-handed. Changing the value of 'HND' mostly affects the behavior of TDINIT - it changes the values of the projection coefficients precomputed and stored in TDPACK common blocks - but it also affects the behavior of the routines TDDTRI and TDLBLS; for this reason, you should not change the value of 'HND' between a call to TDINIT and an ensuing call to one of the other two routines, as this would lead to inconsistent behavior. (Most likely, you will decide what the handedness of your coordinate system is, change the value of 'HND' accordingly, and leave that value unchanged thereafter.)

The default value of 'HND' is 0, specifying a right-handed coordinate system.

'LSU', 'LSV', and 'LSW' (Light Source Position)

If one or more of these three values is non-zero, then the 3-space point ('LSU','LSV','LSW') is the position of the light source assumed by TDDTRI in rendering the triangles in the triangle list.

The default values are all 0., which, by convention, specifies a light source at the position (UMID,VMID,WMID), where UMID, VMID, and WMID are as specified in the last call to TDINIT.

'SET' (Do-SET-Call Flag)

If this flag is equal to 1 (the default), calling TDINIT will result in a call to the SPPS routine SET to define the mapping from the "user coordinate system" (the X/Y projection plane) to the "fractional coordinate system" (the same as the GKS NDC space); that call will be done in such a way as to give a field of view of 'FOV' degrees and to use the portion of the plotter frame defined by the internal parameters 'VPL', 'VPR', 'VPB', and 'VPT'. If 'SET' is given the value 0, no such call to SET will be done and the user will be expected to have called it. In the future, other non-zero values of 'SET' may be implemented which will result in TDINIT's doing a call to SET according to some other criterion.

The default value of 'SET' is 1, specifying that a SET call should be done by TDINIT.

'SHD' (Shading Type Flag)

The value of 'SHD' affects the behavior of TDDTRI when it is picking the fill color to be used for a triangle and the rendering style for the triangle implies that the color index is to be chosen by computing the angle between two vectors - the normal to the triangle and the vector from its center to the light source - and then mapping that angle into a specified range of color indices (thus shading the surface).

If the value of 'SHD' is zero, angles between 0 and 90 are mapped linearly into the specified range of color indices, from lowest to highest, while angles between 90 and 180 are mapped linearly into that same range, from highest to lowest. The effect of this is essentially to use two light sources, one at the position ('LSU','LSV','LSW') and the other at an exactly opposite position relative to the center of the triangle.

If the value of 'SHD' is non-zero, angles between 0 and 180 are mapped linearly into the specified range of color indices, from lowest to highest. The effect of this is essentially to use a single light source, at the position ('LSU','LSV','LSW').

Note that, in both cases, it is the user's responsibility to define the colors associated with the color indices in the specified range to achieve the desired effect. (Normally, the colors associated with the beginning of the range of color indices should be the lightest and the colors associated with the end of that range should be the darkest.) For help in defining colors, see the code of the examples.

The default value of 'SHD' is zero.

'STE' (Stereo Type Flag)

When 'STE' is zero (the default), then stereo pairs are drawn using a single image plane which is perpendicular to the line joining a point midway between the eyes to the point looked at. If 'STE' is set non-zero, then the view from each eye is drawn using an image plane which is perpendicular to the line joining that eye to the point looked at. (In practice, this probably makes little difference.)

Changing this parameter changes the values of the projection coefficients precomputed by TDINIT and stored in TDPACK common blocks.

The default value of 'STE' is 0, specifying that stereo pairs should be drawn using a single image plane perpendicular to the line from the midpoint of the eyes to the point looked at.

'VPB', 'VPL', 'VPR', and 'VPT' (Viewport Specifiers)

The values of these parameters specify the positions of the bottom, left, right, and top edges, respectively, of a desired viewport within the plotter frame. Each is given a value between 0 and 1, inclusive, specifying the position as a fraction of the distance across the plotter frame from left to right or from bottom to top.

These values are used when the SPPS routine SET is called by TDINIT (which happens when the internal parameter 'SET' has a non-zero value).

The default values of 'VPB', 'VPL', 'VPR', and 'VPT' are .05, .05, .95, and .95, respectively, specifying the use of most of the plotter frame.

Parameters Accessed in Other Ways

Rendering-Style Arrays

The internal parameter arrays that define rendering styles are accessed using the routines TDGTRS (to get the complete definition of a specific rendering style) and TDSTRS (to redefine a specific rendering style). Sixty-four different rendering styles may be defined, indexed 1 through 64; each is defined by an array of ten quantities, seven of which are of type INTEGER (named IFC1, IFC2, IFC3, IFC4, ILC1, ILC2, and ILTD) and three of which are of type REAL (named USTP, VSTP, and WSTP). These quantities may be described as follows:

IFC1 and **IFC2** are color indices specifying a range of colors to be used for the "bottom" side of a surface (where function values are less than the value on the surface). If IFC1 is negative, filling of triangles seen from the "bottom" is turned off. If IFC1 is zero or greater, but IFC2 is less than or equal to it, the color with index IFC1 is used. If IFC1 is zero or greater and IFC2 is greater than IFC1, then a range of color indices is specified; colors near the beginning of that range are used for triangles that are nearly perpendicular to the line of sight, while colors near the end of that range are used for triangles more nearly parallel to the line of sight. (Normally, one should make triangles perpendicular to the line of sight lighter than those parallel to the line of sight.)

IFC3 and **IFC4** are color indices specifying a range of colors to be used for the "top" side of a surface (where function values are greater than the value on the surface). If IFC3 is negative, filling of triangles seen from the "top" is turned off. If IFC3 is zero or greater, but IFC4 is less than or equal to it, the color with index IFC3 is used. If IFC3 is zero or greater and IFC4 is greater than IFC3, then a range of color indices is specified; colors near the beginning of that range are used for triangles that are nearly perpendicular to the line of sight, while colors near the end of that range are used for triangles more nearly parallel to the line of sight. (Normally, one should make triangles perpendicular to the line of sight lighter than those parallel to the line of sight.)

ILC1 is the color index specifying a color to be used for lines drawn on the "bottom" side of a surface. If ILC1 is negative, the drawing of these lines is turned off.

ILC2 is the color index specifying a color to be used for lines drawn on the "top" side of a surface. If ILC2 is negative, the drawing of these lines is turned off.

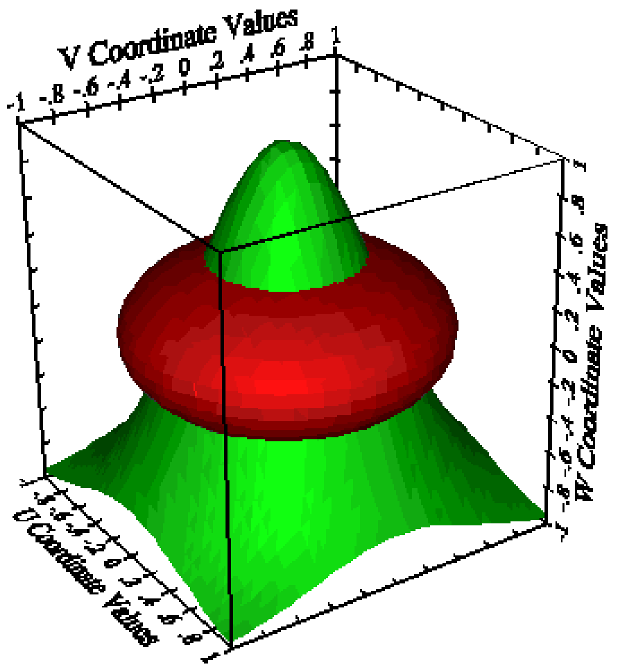
ILTD is a flag, which, if set non-zero, turns on the drawing of the edges of the individual triangles into which surfaces have been decomposed.

USTP, **VSTP**, and **WSTP** are the distances between slices in the U, V, and W directions, respectively. If a given value is zero, the associated slice lines are not drawn.

Note: It is the responsibility of the user to call the GKS routine GSCR to define all of the colors to be used.

TDPACK Examples

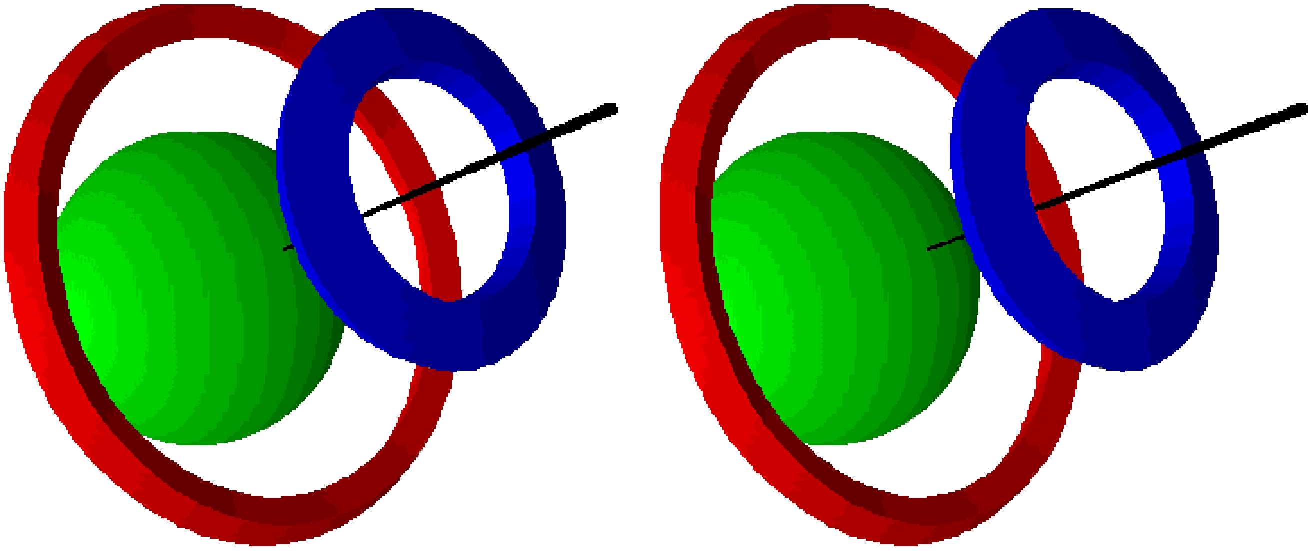
Example 1



Example 1 is meant to illustrate the use of the basic TDPACK routines. It includes all of the code necessary to draw the stereo picture above, showing two intersecting surfaces, rendered in different styles, enclosed in a box, with tick marks and labels on selected edges of the box.

On a system on which NCAR Graphics has been installed, you can obtain the code for this example, compile it, and execute it by typing the command "ncargex tdex01". This will create an NCAR Graphics metafile called "tdex01.ncgm".

Example 2



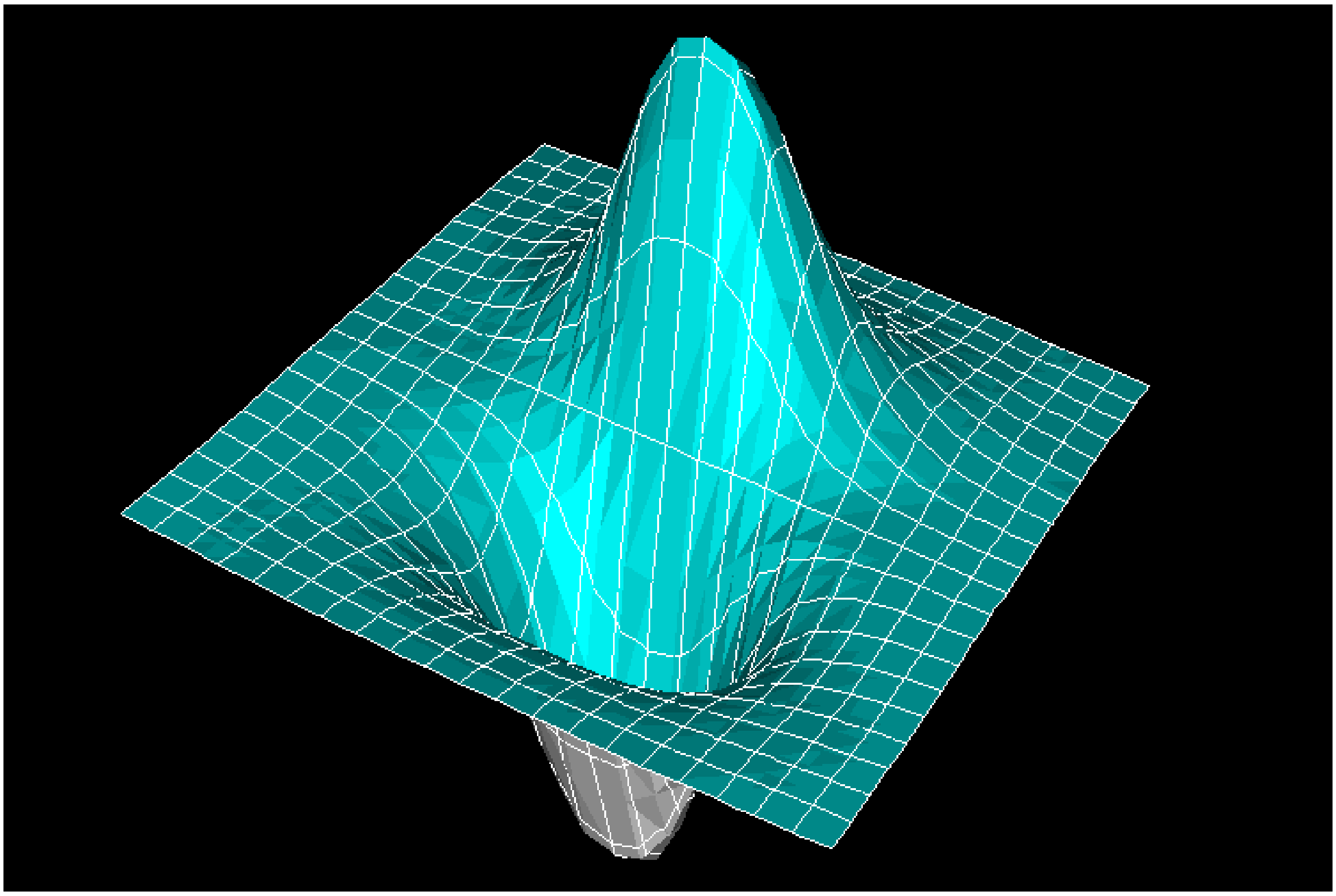
Example 2 is a figure that was used to illustrate a scientific article. The green sphere represents the earth, the red and blue rings elements from a model grid, and the narrow black cone a ray emanating from the center of the earth at a specific latitude and longitude. Note that the values of the internal parameters 'LSU', LSV', and 'LSW' have been changed to move the light source from the position of the eye to a point somewhere off to the left and that the value of the internal parameter 'SHD' has been changed to change the nature of the shading from double-light-source mode to single-light-source mode.

The reason that none of the objects in the figure casts a shadow on any of the others is that the shading algorithm is a simple one that does not use ray-tracing.

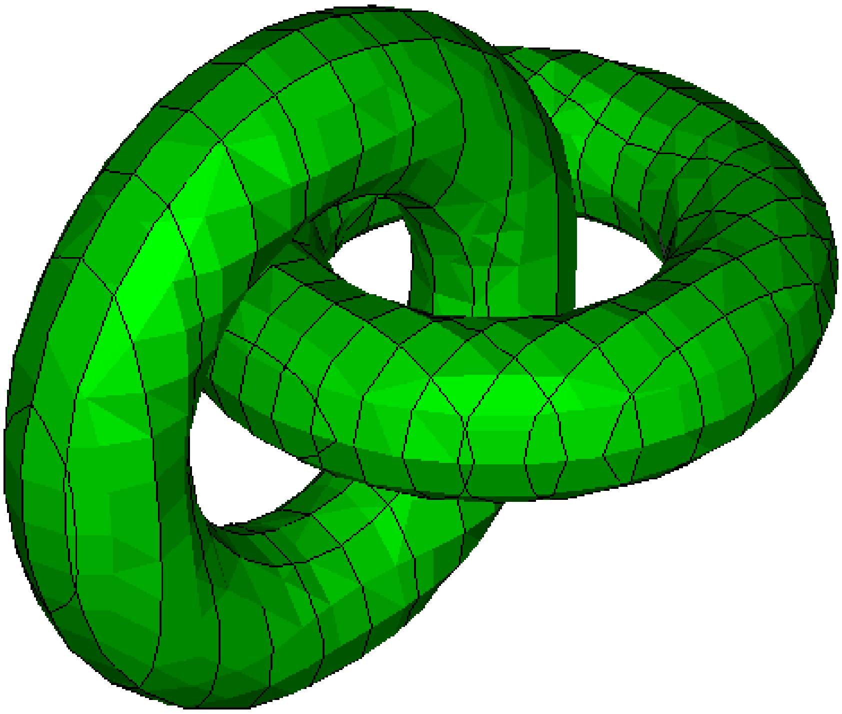
On a system on which NCAR Graphics has been installed, you can obtain the code for this example, compile it, and execute it by typing the command "ncargex tdex02". This will create an NCAR Graphics metafile called "tdex02.ncgm".

Example 3

Example 3 is meant to illustrate the use of the simplified-interface routines. It produces the following two frames, of which the first illustrates the use of TDEZ2D and the second illustrates the use of TDEZ3D:



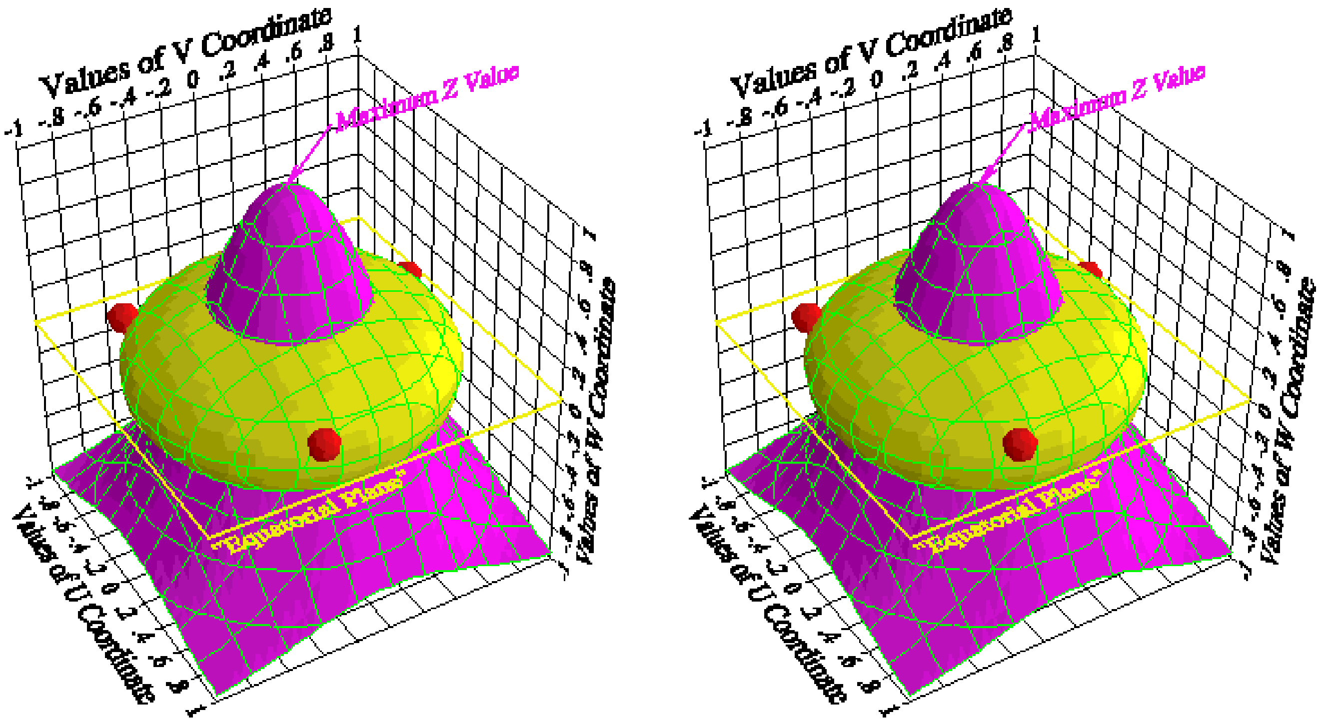
(tdex03, frame 1)



(tdex03, frame 2)

On a system on which NCAR Graphics has been installed, you can obtain the code for this example, compile it, and execute it by typing the command "ncargex tdex03". This will create an NCAR Graphics metafile called "tdex03.ncgm".

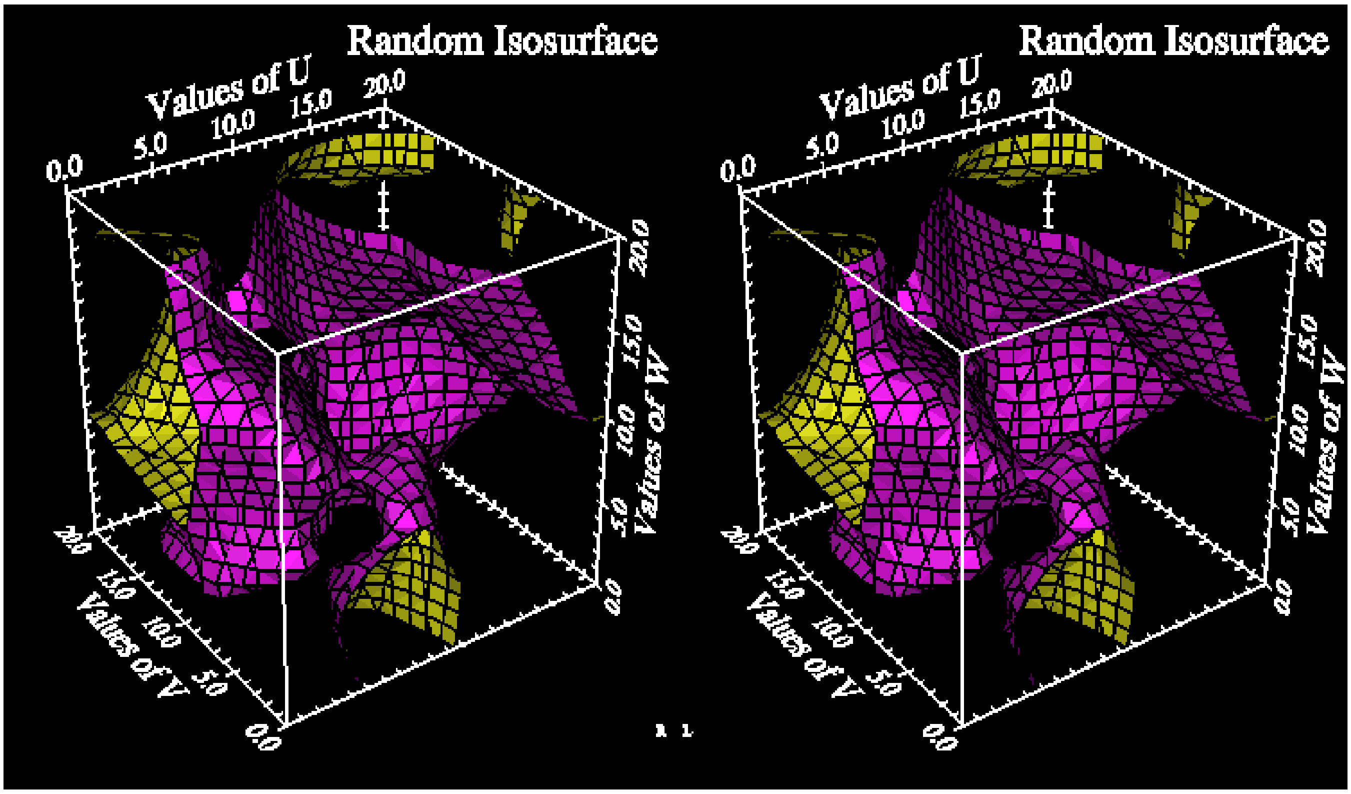
C Example 1



C example 1 is meant to illustrate the use of the C interface to the basic TDPACK routines. It includes all of the code necessary to draw the stereo picture above, showing two intersecting surfaces, rendered in different styles, enclosed in a box, with tick marks and labels on selected edges of the box. There are also three marks (in red) and some labels.

On a system on which NCAR Graphics has been installed, you can obtain the code for this example, compile it, and execute it by typing the command "ncargex c_tdex01". This will create an NCAR Graphics metafile called "c_tdex01.ncgm".

TDPACK Demo



The TDPACK demo is a program that allows one to investigate the capabilities of TDPACK in an interactive manner. On a system on which NCAR Graphics has been installed, simply execute the command

```
tdpackdemo
```

This will create an X window containing an initial display of a simple surface. Commands may then be typed in the original window, directing the demo program to change the X-window display in various ways. A little experimentation should suffice to learn what is possible; a good place to start is with an "H" command, which will give you a list of all the possible commands.

The code of the TDPACK demo program may be viewed here.

TDPACK Demo Program - Code

```
      PROGRAM TDDEMO
C
C Declare parameters specifying the maximum sizes of the data-defining
C arrays and the size of the triangle-list array.
C
        PARAMETER (IMAX=101,JMAX=101,KMAX=101,MTRI=100000)
C
C Declare local dimensioned variables to hold data defining a simple
```

```
C surface or an isosurface.
C
        DIMENSION U(IMAX),V(JMAX),W(KMAX),S(IMAX,JMAX),F(IMAX,JMAX,KMAX)
C
C Declare the local dimensioned variable in which the triangle list is
C to be constructed and a couple of temporary variables used in sorting
C the list.
C
        DIMENSION RTRI(10,MTRI),RTWK(MTRI,2),ITWK(MTRI)
C
C Declare some character variables into which to encode numeric and
C informational labels for the axes.
C
        CHARACTER*128 UNLB,VNLB,WNLB,UILB,VILB,WILB,PILB
C
C Declare some character variables in which formats for encoding the
C numeric labels are kept.
C
        CHARACTER*16 UFMT,VFMT,WFMT
C
C Define a character variable into which to read a file name.
C
        CHARACTER*128 FLNM
C
C Define a character variable into which to read commands.
C
        CHARACTER*1 COMD
C
C Declare a variable to hold the names of the various mark types.
C
        CHARACTER*15 IMKT(5)
C
C Define the conversion constant from degrees to radians.
C
        DATA DTOR / .017453292519943 /
C
C Set the default handedness of the coordinate system.  This must match
C the default value of the TDPACK internal parameter 'HND'.  Zero means
C a right-handed system, one a left-handed system.
C
        DATA IHND / 0 /
C
C Set the default value of the parameter that says whether or not the
C X workstation will be updated as changes are made.  (When making a
C movie, it's a good idea to set this to zero.)
C
        DATA IUXW / 1 /
C
C Set the default value of the surface selector.
C
        DATA ISAS / -1 /
C
C Set the default values of the dimensions of the data arrays.
C
        DATA IDIM,JDIM,KDIM / 21,21,21 /
C
C Set the default value of the flag that says whether the basic color
C scheme will be white on black (IBOW=0) or black on white (IBOW=1).
C
        DATA IBOW / 0 /
C
```

```
C Set the default values of parameters determining the eye position, the
C position of the point looked at, and the field of view.
C
        DATA ANG1,ANG2,REYE,RAPT,FOVP / -35.,25.,2.9,0.,20. /
C
C Set the default value of the TDPACK character multiplier.
C
        DATA RCS1 / 1.25 /
C
C Set the default value of the parameter that says to compute new center
C positions for the exponentials used to generate dummy surfaces and
C isosurfaces.  Basically, with ICEN = 0, the same surface will be
C generated by consecutive calls to GENDAT or to TDGNDT, even if the
C dimensions of the data array are changed, but, with ICEN = 1, a
C different surface will be generated each time.
C
        DATA ICEN / 0 /
C
C Set the default cut-off values for generated isosurfaces.
C
        DATA FIS1,FIS2 / 95.,95. /
C
C Set the default value of the multiplier for the random number term
C in the code used to compute the surface/isosurface data.
C
        DATA RNDM / 0. /
C
C Set the default value of the parameter that says what kinds of grids
C are to be drawn.
C
        DATA IGDF / 13 /
C
C Set the default value of parameters that specify the grid spacing.
C
        DATA UGIN,VGIN,WGIN / 1.,1.,1. /
C
C Set the default value of parameters that specify the label spacing.
C
        DATA ULIN,VLIN,WLIN / 5.,5.,5. /
C
C Set the default values of parameters specifying how the numeric labels
C are to be encoded.
C
        DATA UFMT / '(21(1X,F5.1))' /
        DATA VFMT / '(21(1X,F5.1))' /
        DATA WFMT / '(21(1X,F5.1))' /
C
C Set the default values of the U, V, and W axis labels and the plot
C label.
C
        DATA UILB / 'U Coordinate Values' /
        DATA VILB / 'V Coordinate Values' /
        DATA WILB / 'W Coordinate Values' /
C
        DATA PILB / ' ' /
C
C Set the default value of the parameter that controls fading of axis
C labels for rotation angles near multiples of 90 degrees.
C
        DATA IFDE / 0 /
C
```

```
C Set the default values of parameters used to position the plot label.
C
        DATA PILX,PILY / 0. , .54 /
C
C Set the default value of the flag that says whether or not to draw a
C 3D grid of marks.
C
        DATA IMRK / 0 /
C
C Set the default value of the mark size parameter.
C
        DATA SMRK / .025 /
C
C Set the default value of parameters that specify the mark spacing.
C
        DATA UMKS,VMKS,WMKS / 5.,5.,5. /
C
C Set the default value of the parameter that determines how the
C triangles are ordered.  This will be used as the final argument in
C calls to TDOTRI.
C
        DATA IORD / 0 /
C
C Define default values of the shading parameters.
C
        DATA SHDE,SHDR,ISHD,ANG3,ANG4
     +       /.15,  1.,   0,  0.,  0./
C
C Define default values of the rendering-style parameters.
C
        DATA IFCB,IFCT,ILCB,ILCT,IDTR,USIN,VSIN,WSIN
     +       /  4,   2, 116, 116,   0,  1.,  1.,  1./
C
C Set a couple of counters that keep track of how many NCGM and
C PostScript frames have been saved.
C
        DATA NFNG,NFPS / 0,0 /
C
C Set the default value of the stereo-view flag.  The user can set it
C negative to get stereo views on a single frame or positive to get
C stereo views on separate frames.  In either case, its absolute value
C is the angle, in degrees, between the lines of sight of the two views.
C
        DATA STVF / 0. /
C
C When STVF is set negative. WOSW is the width of each stereo window,
C as a fraction of the width of the plotter frame.  If this value is
C set greater than .5, the windows will overlap slightly.
C
        DATA WOSW / .5 /
C
C Define the names of the various mark types.
C
        DATA IMKT / 'a tetrahedron.','an octahedron.','a cube.',
     +                          'an icosahedron.', 'a "sphere".'/
C
C Open GKS.
C
        CALL GOPKS (6,0)
C
C Open and activate an X workstation.
```

```
C
        CALL GOPWK (1,0,8)
        CALL GACWK (1)
C
C Open and activate an NCGM workstation.
C
        CALL GOPWK (2,0,1)
        CALL GACWK (2)
C
C Open and activate a PostScript workstation.
C
        CALL GOPWK (3,0,20)
        CALL GACWK (3)
C
C Turn clipping off.
C
        CALL GSCLIP (0)
C
C Select a "fill area interior style" of "solid".
C
        CALL GSFAIS (1)
C
C Double the line width.
C
        CALL GSLWSC (2.)
C
C Define colors on all the workstations.
C
        DO 102 IWID=1,3
          CALL TDCLRS (IWID,IBOW,SHDE,SHDR,101,116,0)
  102   CONTINUE
C
C Save the values of SHDE and SHDR.
C
        SSHE=SHDE
        SSHR=SHDR
C
C Deactivate the NCGM and PostScript workstations.
C
        CALL GDAWK (2)
        CALL GDAWK (3)
C
C Select font number 25, turn on the outlining of filled fonts, set the
C line width to 1, and turn off the setting of the outline color.
C
        CALL PCSETI ('FN - FONT NUMBER',25)
        CALL PCSETI ('OF - OUTLINE FLAG',1)
        CALL PCSETR ('OL - OUTLINE LINE WIDTH',1.)
        CALL PCSETR ('OC - OUTLINE LINE COLOR',-1.)
C
C Compute values defining the surface to be depicted.
C
  103   PRINT * , 'Generating U data.'
C
        DO 104 I=1,IDIM
          U(I)=REAL(I-1)
  104   CONTINUE
C
        UMIN=U(   1)
        UMAX=U(IDIM)
        UMID=.5*(UMIN+UMAX)
```

```
C
        PRINT * , 'Generating V data.'
C
        DO 105 J=1,JDIM
          V(J)=REAL(J-1)
  105   CONTINUE
C
        VMIN=V(   1)
        VMAX=V(JDIM)
        VMID=.5*(VMIN+VMAX)
C
        PRINT * , 'Generating W data.'
C
        DO 106 K=1,KDIM
          W(K)=REAL(K-1)
  106   CONTINUE
C
        WMIN=W(   1)
        WMAX=W(KDIM)
        WMID=.5*(WMIN+WMAX)
C
C Generate data representing a simple surface, a sphere, a torus, two
C interlocking tori, three joined tori, or any of the above with some
C added random noise, depending on the value of ISAS.
C
        IF (ISAS.LT.0) THEN
          PRINT * , 'Generating a simple random surface.'
          CALL GENDAT (S,IMAX,IDIM,JDIM,23,23,WMIN,WMAX,ICEN)
          IF (RNDM.NE.0.) CALL ADDRAN (F,IMAX,IDIM,JDIM,WMIN,WMAX,RNDM)
        ELSE IF (ISAS.EQ.0) THEN
          PRINT * , 'Generating a random isosurface.'
          CALL TDGNDT (F,IMAX,JMAX,IDIM,JDIM,KDIM,23,23,0.,190.,ICEN)
          IF (RNDM.NE.0.) CALL TDADRN (F,IMAX,JMAX,IDIM,JDIM,KDIM,
     +                                                0.,190.,RNDM)
        ELSE IF (ISAS.GE.1.AND.ISAS.LE.5) THEN
          PRINT * , 'Generating a canned isosurface.'
          FMIN=+1.E36
          FMAX=-1.E36
          DO 109 I=1,IDIM
            UTMP=1.-2.*(REAL(IDIM-I)/REAL(IDIM-1))
            DO 108 J=1,JDIM
              VTMP=1.-2.*(REAL(JDIM-J)/REAL(JDIM-1))
              DO 107 K=1,KDIM
                WTMP=1.-2.*(REAL(KDIM-K)/REAL(KDIM-1))
                IF      (ISAS.EQ.1) THEN
                  F(I,J,K)=100.*SQRT(UTMP**2+VTMP**2+WTMP**2)
                ELSE IF (ISAS.EQ.2) THEN
                  F(I,J,K)=380.*
     +                     SQRT((SQRT(UTMP**2+VTMP**2)-.7)**2+WTMP**2)
                ELSE IF (ISAS.EQ.3) THEN
                  F(I,J,K)=380.*
     +                     MIN(SQRT((SQRT((1.35*UTMP+.35)**2+
     +                          (1.35*VTMP)**2)-.7)**2+(1.35*WTMP)**2),
     +                         SQRT((SQRT((1.35*UTMP-.35)**2+
     +                          (1.35*WTMP)**2)-.7)**2+(1.35*VTMP)**2))
                ELSE IF (ISAS.EQ.4) THEN
                  F(I,J,K)=760.*MIN(
     +                     SQRT((SQRT(UTMP**2+VTMP**2)-.7)**2+WTMP**2),
     +                     SQRT((SQRT(WTMP**2+UTMP**2)-.7)**2+VTMP**2),
     +                     SQRT((SQRT(VTMP**2+WTMP**2)-.7)**2+UTMP**2))
                ELSE IF (ISAS.EQ.5) THEN
```

```
                F(I,J,K)=4.*FIS1*(UTMP**2+VTMP**2-.75*WTMP**2)
              END IF
              FMIN=MIN(FMIN,F(I,J,K))
              FMAX=MAX(FMAX,F(I,J,K))
  107       CONTINUE
  108     CONTINUE
  109   CONTINUE
        IF (RNDM.NE.0.) CALL TDADRN (F,IMAX,JMAX,IDIM,JDIM,KDIM,
     +                                        FMIN,FMAX,RNDM)
      END IF
C
C Get triangles representing the surface, to be rendered using rendering
C style 1.
C
  110 PRINT * , 'Generating triangles representing the surface.'
C
      NTRI=0
C
      IF (ISAS.LT.0.OR.(ISAS.GE.99.AND.ISAS.LE.105)) THEN
        CALL TDSTRI (U,IDIM,V,JDIM,S,IMAX,RTRI,MTRI,NTRI,1)
        IF (NTRI.LT.MTRI) THEN
          PRINT * , 'Percentage of triangle space used by TDSTRI:',
     +                           INT(100.*REAL(NTRI)/REAL(MTRI))
        ELSE
          PRINT * , 'Triangle space overflow on call to TDSTRI.'
        END IF
      ELSE
        IF (FIS1.EQ.FIS2) THEN
          CALL TDITRI (U,IDIM,V,JDIM,W,KDIM,F,IMAX,JMAX,FIS1,
     +                                   RTRI,MTRI,NTRI,1)
        ELSE
          CALL TDITRI (U,IDIM,V,JDIM,W,KDIM,F,IMAX,JMAX,FIS1,
     +                                   RTRI,MTRI,NTRI,2)
          CALL TDITRI (U,IDIM,V,JDIM,W,KDIM,F,IMAX,JMAX,FIS2,
     +                                   RTRI,MTRI,NTRI,3)
        END IF
        IF (NTRI.LT.MTRI) THEN
          PRINT * , 'Percentage of triangle space used by TDITRI:',
     +                           INT(100.*REAL(NTRI)/REAL(MTRI))
        ELSE
          PRINT * , 'Triangle space overflow on call to TDITRI.'
        END IF
      END IF
C
C If it's been requested, put a 3D grid of marks inside the viewing box.
C
      IF (IMRK.NE.0) THEN
C
        IF (UMKS.GT.0..AND.UMAX-UMIN.LE.20.*UMKS) THEN
          UEPS=1.E-3*(UMAX-UMIN)
          IST1=INT((UMIN-UEPS)/UMKS+.5+SIGN(.5,UMIN-UEPS))
          IST2=INT((UMAX+UEPS)/UMKS-.5+SIGN(.5,UMAX+UEPS))
        ELSE
          IST1=1
          IST2=0
        END IF
C
        IF (VMKS.GT.0..AND.VMAX-VMIN.LE.20.*VMKS) THEN
          VEPS=1.E-3*(VMAX-VMIN)
          JST1=INT((VMIN-VEPS)/VMKS+.5+SIGN(.5,VMIN-VEPS))
          JST2=INT((VMAX+VEPS)/VMKS-.5+SIGN(.5,VMAX+VEPS))
```

```
          ELSE
            JST1=1
            JST2=0
          END IF
C
          IF (WMKS.GT.0..AND.WMAX-WMIN.LE.20.*WMKS) THEN
            WEPS=1.E-3*(WMAX-WMIN)
            KST1=INT((WMIN-WEPS)/WMKS+.5+SIGN(.5,WMIN-WEPS))
            KST2=INT((WMAX+WEPS)/WMKS-.5+SIGN(.5,WMAX+WEPS))
          ELSE
            KST1=1
            KST2=0
          END IF
C
          IF (IST1.LE.IST2.AND.JST1.LE.JST2.AND.KST1.LE.KST2) THEN
            SIZE=SMRK*MIN(UMAX-UMIN,VMAX-VMIN,WMAX-WMIN)
            DO 903 I=IST1,IST2
              UVAL=UMKS*REAL(I)
              DO 902 J=JST1,JST2
                VVAL=VMKS*REAL(J)
                DO 901 K=KST1,KST2
                  WVAL=WMKS*REAL(K)
                  CALL TDMTRI (IMRK,UVAL,VVAL,WVAL,SIZE,
     +                         RTRI,MTRI,NTRI,4,
     +                         UMIN,VMIN,WMIN,UMAX,VMAX,WMAX)
  901           CONTINUE
  902         CONTINUE
  903       CONTINUE
          ELSE
            PRINT * , 'Omitting marks - either one of the specified'
            PRINT * , 'mark intervals is non-positive or the marks'
            PRINT * , 'would be too crowded.'
          END IF
C
        END IF
C
C Tell the user what the minima and maxima are.
C
  111   PRINT * , 'U minimum and maximum: ',UMIN,UMAX
        PRINT * , 'V minimum and maximum: ',VMIN,VMAX
        PRINT * , 'W minimum and maximum: ',WMIN,WMAX
C
C Define numeric and informational labels for the plot.
C
        PRINT * , 'Encoding labels for the plot.'
C
        IF (ULIN.GT.0.) THEN
          IF (UMAX-UMIN.LE.20.*ULIN) THEN
            UEPS=1.E-3*(UMAX-UMIN)
            IST1=INT((UMIN-UEPS)/ULIN+.5+SIGN(.5,UMIN-UEPS))
            IST2=INT((UMAX+UEPS)/ULIN-.5+SIGN(.5,UMAX+UEPS))
            WRITE (UNLB,UFMT) (ULIN*REAL(I),I=IST1,IST2)
          ELSE
            PRINT *, 'Omitting too-dense numeric labels in U direction.'
            UNLB=' '
          END IF
        END IF
C
        IF (VLIN.GT.0.) THEN
          IF (VMAX-VMIN.LE.20.*VLIN) THEN
            VEPS=1.E-3*(VMAX-VMIN)
```

```
            JST1=INT((VMIN-VEPS)/VLIN+.5+SIGN(.5,VMIN-VEPS))
            JST2=INT((VMAX+VEPS)/VLIN-.5+SIGN(.5,VMAX+VEPS))
            WRITE (VNLB,VFMT) (VLIN*REAL(J),J=JST1,JST2)
          ELSE
            PRINT *, 'Omitting too-dense numeric labels in V direction.'
            VNLB=' '
          END IF
        END IF
C
        IF (WLIN.GT.0.) THEN
          IF (WMAX-WMIN.LE.20.*WLIN) THEN
            WEPS=1.E-3*(WMAX-WMIN)
            KST1=INT((WMIN-WEPS)/WLIN+.5+SIGN(.5,WMIN-WEPS))
            KST2=INT((WMAX+WEPS)/WLIN-.5+SIGN(.5,WMAX+WEPS))
            WRITE (WNLB,WFMT) (WLIN*REAL(K),K=KST1,KST2)
          ELSE
            PRINT *, 'Omitting too-dense numeric labels in W direction.'
            WNLB=' '
          END IF
        END IF
C
C Set the overall character multiplier for TDPACK.
C
        CALL TDSETR ('CS1',RCS1)
C
C Define rendering styles 1, 2, and 3.
C
        PRINT * , 'Defining rendering styles.'
C
        IF (IFCB.LT.0.OR.IFCB.GT.7.OR.SHDE.EQ.0.) THEN
          IFC1=IFCB
          IFC2=IFCB
        ELSE
          IFC1=101+IFCB*16
          IFC2=116+IFCB*16
        END IF
C
        IF (IFCT.LT.0.OR.IFCT.GT.7.OR.SHDE.EQ.0.) THEN
          IFC3=IFCT
          IFC4=IFCT
        ELSE
          IFC3=101+IFCT*16
          IFC4=116+IFCT*16
        END IF
C
        IF (USIN.NE.0..AND.UMAX-UMIN.LE.50.*USIN) THEN
          USIM=1.
        ELSE
          IF (USIN.NE.0.)
     +    PRINT * , 'Omitting too-dense slice lines in U direction.'
          USIM=0.
        END IF
C
        IF (VSIN.NE.0..AND.VMAX-VMIN.LE.50.*VSIN) THEN
          VSIM=1.
        ELSE
          IF (VSIN.NE.0.)
     +    PRINT * , 'Omitting too-dense slice lines in V direction.'
          VSIM=0.
        END IF
C
```

```
        WSIM=1.
C
        IF (WSIN.NE.0..AND.WMAX-WMIN.LE.50.*WSIN) THEN
          WSIM=1.
        ELSE
          IF (WSIN.NE.0.)
     +    PRINT * , 'Omitting too-dense slice lines in W direction.'
          WSIM=0.
        END IF
C
        CALL TDSTRS (1,IFC1,IFC2,IFC3,IFC4,ILCB,ILCT,IDTR,
     +                    USIM*USIN,VSIM*VSIN,WSIM*WSIN)
        CALL TDSTRS (2, 109, 116,IFC3,IFC4,  -1,ILCT,IDTR,
     +                    USIM*USIN,VSIM*VSIN,WSIM*WSIN)
        CALL TDSTRS (3,IFC1,IFC2, 109, 116,ILCB,  -1,IDTR,
     +                    USIM*USIN,VSIM*VSIN,WSIM*WSIN)
        CALL TDSTRS (4, 109, 116, 213, 228,  -1,  -1,   0,
     +                           0.,       0.,       0.)
C
C Check for too-dense grid lines.
C
        PRINT * , 'Checking grid parameters.'
C
        IF (UGIN.GT.0.) THEN
          IF (UMAX-UMIN.LE.50.*UGIN) THEN
            UGIM=1.
          ELSE
            PRINT * , 'Omitting too-dense grid lines in U direction.'
            UGIM=0.
          END IF
        END IF
C
        IF (VGIN.GT.0.) THEN
          IF (VMAX-VMIN.LE.50.*VGIN) THEN
            VGIM=1.
          ELSE
            PRINT * , 'Omitting too-dense grid lines in V direction.'
            VGIM=0.
          END IF
        END IF
C
        IF (WGIN.GT.0.) THEN
          IF (WMAX-WMIN.LE.50.*WGIN) THEN
            WGIM=1.
          ELSE
            PRINT * , 'Omitting too-dense grid lines in W direction.'
            WGIM=0.
          END IF
        END IF
C
C Determine the positions of the eye and the point looked at.
C
        PRINT * , 'Determining eye position.'
C
        DOFB=SQRT((UMAX-UMIN)**2+(VMAX-VMIN)**2+(WMAX-WMIN)**2)
        DEYE=REYE*DOFB
        DAPT=RAPT*DOFB
C
        UEYE=UMID+DEYE*COS(DTOR*ANG1)*COS(DTOR*ANG2)
        VEYE=VMID+DEYE*SIN(DTOR*ANG1)*COS(DTOR*ANG2)
        WEYE=WMID+DEYE*SIN(DTOR*ANG2)
```

```
C
        UAPT=UMID-DAPT*COS(DTOR*ANG1)*COS(DTOR*ANG2)
        VAPT=VMID-DAPT*SIN(DTOR*ANG1)*COS(DTOR*ANG2)
        WAPT=WMID-DAPT*SIN(DTOR*ANG2)
C
C Compute some derivative quantities that are needed to put a label on
C the plot.
C
        UDNL=UMID-UEYE
        VDNL=VMID-VEYE
        WDNL=WMID-WEYE
C
        IF (IHND.EQ.0) THEN
          UDNX=VMID-VEYE
          VDNX=UEYE-UMID
          WDNX=0.
          UDNY=VDNX*WDNL-VDNL*WDNX
          VDNY=WDNX*UDNL-WDNL*UDNX
          WDNY=UDNX*VDNL-UDNL*VDNX
        ELSE
          UDNX=VEYE-VMID
          VDNX=UMID-UEYE
          WDNX=0.
          UDNY=VDNL*WDNX-VDNX*WDNL
          VDNY=WDNL*UDNX-WDNX*UDNL
          WDNY=UDNL*VDNX-UDNX*VDNL
        END IF
C
        DNOM=SQRT(UDNX*UDNX+VDNX*VDNX+WDNX*WDNX)
        UDCX=UDNX/DNOM
        VDCX=VDNX/DNOM
        WDCX=WDNX/DNOM
C
        DNOM=SQRT(UDNY*UDNY+VDNY*VDNY+WDNY*WDNY)
        UDCY=UDNY/DNOM
        VDCY=VDNY/DNOM
        WDCY=WDNY/DNOM
C
        CALL TDGETR ('CS1',CSM1)
        CSM2=CSM1*MIN(UMAX-UMIN,VMAX-VMIN,WMAX-WMIN)
C
C Initialize TDPACK.
C
        PRINT * , 'Initializing TDPACK.'
C
        CALL TDINIT (UEYE,VEYE,WEYE,UAPT,VAPT,WAPT,UAPT,VAPT,WAPT+1.,0)
        CALL GETSET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C
C If the X workstation is not to be updated, jump to get user input.
C
        IF (IUXW.EQ.0) GO TO 112
C
C Clear the X workstation.
C
        CALL GCLRWK (1,1)
C
C Draw the plot label, if any.  The plot label is positioned in a
C rectangle that is perpendicular to the line of sight and passes
C through the midpoint of the data box; some algebra is required to
C come up with the direction cosines of the sides of that rectangle.
C We can then call TDPARA to define it as the current reference
```

```
C parallelogram and TDPLCH to draw a label relative to the rectangle.
C
        IF (PILB.NE.' ') THEN
          PRINT * , 'Drawing plot label.'
          CALL TDPARA (UMID-.5*DOFB*(UDCX+UDCY),
     +                 VMID-.5*DOFB*(VDCX+VDCY),
     +                 WMID-.5*DOFB*(WDCX+WDCY),
     +                 UDCX,VDCX,WDCX,UDCY,VDCY,WDCY)
          CALL TDPLCH ((.5+PILX)*DOFB,(.5+PILY)*DOFB,
     +                 PILB(1:LNBPCS(PILB)),.04*CSM2,0.,0.)
        END IF
C
C Put some simple labels on the axes of the plot.  If the last argument
C in the call to TDLBLS is a zero, all six outer edges of the projection
C of the box are labelled; if the argument is non-zero, then only three
C edges are labelled (which three depends on the sign of the argument),
C which gives one set of U axis labels, one set of V axis labels, and
C one set of W axis labels.  If the label fading flag is set and the
C viewing angle is close to a multiple of 90 degrees, the polyline and
C fill area color indices are set so as to do the fade (useful mostly
C when a movie is being made).
C
        IF (IGDF.NE.0) THEN
C
          PRINT * , 'Drawing axis labels.'
C
          IF (IFDE.NE.0) THEN
            ANGM=MOD(ANG1,90.)
            IF (ANGM.LT. 0.) ANGM=90.+ANGM
            IF (ANGM.GT.45.) ANGM=90.-ANGM
            MANG=101+INT(ANGM)
          ELSE
            MANG=116
          END IF
C
          IF (MANG.GT.101) THEN
            IF (MANG.GE.102.AND.MANG.LE.115) THEN
              CALL SFLUSH
              IF (IBOW.EQ.0) THEN
                CALL GSPLCI (217-MANG)
                CALL GSFACI (217-MANG)
              ELSE
                CALL GSPLCI (MANG)
                CALL GSFACI (MANG)
              END IF
            END IF
            CALL TDLBLS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                   UNLB,VNLB,WNLB,UILB,VILB,WILB,1)
            IF (MANG.GE.102.AND.MANG.LE.115) THEN
              CALL SFLUSH
              CALL GSPLCI (1)
              CALL GSFACI (1)
            END IF
          END IF
C
C Draw the sides of the box that could be hidden.
C
          PRINT * , 'Drawing far sides of box.'
C
          CALL TDGRDS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                 UGIM*UGIN,VGIM*VGIN,WGIM*WGIN,IGDF,1)
```

```
C
        END IF
C
C Draw all the triangles.
C
        PRINT * , 'Rendering surface.'
C
        CALL TDOTRI (RTRI,MTRI,NTRI,RTWK,ITWK,IORD)
        CALL TDDTRI (RTRI,MTRI,NTRI,ITWK)
C
C Draw the sides of the box that cannot be hidden.
C
        IF (IGDF.NE.0) THEN
C
          PRINT * , 'Drawing near sides of box.'
C
          CALL TDGRDS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                 UGIM*UGIN,VGIM*VGIN,WGIM*WGIN,IGDF,0)
C
        END IF
C
C Flush buffers and update the X workstation.
C
        CALL SFLUSH
        CALL GUWK (1,0)
C
C Let the user enter a command.
C
  112   PRINT * , ' '
        PRINT * , 'Enter a command (H for help, Q for quit):'
C
        READ  '(A1)', COMD
C
        IF      (COMD.EQ.'D'.OR.COMD.EQ.'d') THEN
C
          IF (ISAS.GE. 99) ISAS=ISAS-100
          ISAS=MAX(-1,MIN(5,ISAS))
          PRINT * , ' '
          PRINT * , 'Change the type of data displayed (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            PRINT * , ' '
            PRINT * , 'The current surface type selector is ',ISAS
            PRINT * , ' '
            PRINT * , 'Enter -1 for a random surface, 0 for a random'
            PRINT * , 'isosurface, 1-5 for various canned isosurfaces,'
            PRINT * , '6 or greater to read a data file.'
            CALL TDRDIN (ITMP,ISAS)
            IF (ITMP.LE.5) THEN
              ISAS=MAX(-1,ITMP)
            ELSE
              PRINT * , ' '
              PRINT * , 'Enter the name of a data file:'
              READ '(A128)' , FLNM
              OPEN (13,FILE=FLNM(1:LNBPCS(FLNM)),STATUS='OLD',
     +                                  FORM='FORMATTED',ERR=113)
              PRINT * , 'File opened successfully.'
              READ (13,'(2I4)',ERR=113) IDIM,JDIM
              IF (IDIM.LT.1.OR.IDIM.GT.IMAX.OR.
     +            JDIM.LT.1.OR.JDIM.GT.JMAX) THEN
                PRINT * , 'Dimensions read are out of range!'
```

```
                GO TO 112
              END IF
              READ (13,'(6E12.0)',ERR=113,END=114)
     +                                      UMIN,UMAX,VMIN,VMAX,WMIN,WMAX
              UMID=.5*(UMIN+UMAX)
              VMID=.5*(VMIN+VMAX)
              WMID=.5*(WMIN+WMAX)
              READ (13,'(6E12.0)',ERR=113,END=114) (U(I),I=1,IDIM)
              READ (13,'(6E12.0)',ERR=113,END=114) (V(I),I=1,JDIM)
              READ (13,'(6E12.0)',ERR=113,END=114)
     +                                ((S(I,J),I=1,IDIM),J=1,JDIM)
              CLOSE (13)
              ISAS=ISAS+100
              GO TO 110
  113         PRINT * , ' '
              PRINT * , 'Error trying to read data file!'
              GO TO 112
  114         PRINT * , ' '
              PRINT * , 'Premature EOF on data file!'
              GO TO 112
            END IF
          END IF
          PRINT * , ' '
          PRINT * , 'Change the data dimensions (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            PRINT * , ' '
            PRINT * , 'Current dimensions are as follows:'
            PRINT * , '  1st data dimension: ',IDIM
            PRINT * , '  2nd data dimension: ',JDIM
            PRINT * , '  3rd data dimension: ',KDIM
            PRINT * , ' '
            PRINT * , 'Enter the new 1st data dimension:'
            CALL TDRDIN (IDIM,IDIM)
            IDIM=MAX(5,MIN(IMAX,IDIM))
            PRINT * , 'Enter the new 2nd data dimension:'
            CALL TDRDIN (JDIM,JDIM)
            JDIM=MAX(5,MIN(JMAX,JDIM))
            PRINT * , 'Enter the new 3rd data dimension:'
            CALL TDRDIN (KDIM,KDIM)
            JDIM=MAX(5,MIN(KMAX,KDIM))
          END IF
          IF (ISAS.LE.0) THEN
            PRINT * , ' '
            PRINT * , 'Use old centers for exponentials (Y or N)?'
            READ  '(A1)', COMD
            IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
              ICEN=0
            ELSE
              ICEN=1
            END IF
          END IF
          GO TO 103
C
        ELSE IF (COMD.EQ.'G'.OR.COMD.EQ.'g') THEN
C
          PRINT * , ' '
          PRINT * , 'Current value of grid type selector: ',IGDF
          PRINT * , ' '
          PRINT * , 'Enter new value of grid type selector (a 2-digit'
          PRINT * , 'integer).  The first digit applies to the near'
```

```
PRINT * , 'sides of the box and the second to the far sides'
PRINT * , 'of the box.  Each digit is a 0 (nothing), a 1'
PRINT * , '(perimeter only), a 2 (perimeter plus ticks),'
PRINT * , 'or a 3 (perimeter plus grid lines):'
CALL TDRDIN (IGDF,IGDF)
IGDF=10*MAX(0,MIN(3,IGDF/10))+MAX(0,MIN(3,MOD(IGDF,10)))
IF (IGDF.NE.0) THEN
  PRINT * , ' '
  PRINT * , 'Change spacing of grid lines (Y or N)?'
  READ  '(A1)', COMD
  IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
    PRINT * , ' '
    PRINT * , 'Current spacing of grid lines:'
    PRINT * , '  Grid-line spacing in U: ',UGIN
    PRINT * , '  Grid-line spacing in V: ',VGIN
    PRINT * , '  Grid-line spacing in W: ',WGIN
    PRINT * , ' '
    PRINT * , 'Enter new value for grid-line spacing in U:'
    CALL TDRDRN (UGIN,UGIN)
    UGIN=MAX(0.,UGIN)
    PRINT * , 'Enter new value for grid-line spacing in V:'
    CALL TDRDRN (VGIN,VGIN)
    VGIN=MAX(0.,VGIN)
    PRINT * , 'Enter new value for grid-line spacing in W:'
    CALL TDRDRN (WGIN,WGIN)
    WGIN=MAX(0.,WGIN)
  END IF
  PRINT * , ' '
  PRINT * , 'Change spacing of numeric labels (Y or N)?'
  READ  '(A1)', COMD
  IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
  PRINT * , ' '
    PRINT * , 'Current spacing of numeric labels:'
    PRINT * , '  Numeric-label spacing in U: ',ULIN
    PRINT * , '  Numeric-label spacing in V: ',VLIN
    PRINT * , '  Numeric-label spacing in W: ',WLIN
    PRINT * , ' '
    PRINT * , 'Enter new value for label spacing in U:'
    CALL TDRDRN (ULIN,ULIN)
    ULIN=MAX(0.,ULIN)
    PRINT * , 'Enter new value for label spacing in V:'
    CALL TDRDRN (VLIN,VLIN)
    VLIN=MAX(0.,VLIN)
    PRINT * , 'Enter new value for label spacing in W:'
    CALL TDRDRN (WLIN,WLIN)
    WLIN=MAX(0.,WLIN)
  END IF
  PRINT * , ' '
  PRINT * , 'Change formats for numeric labels (Y or N)?'
  READ  '(A1)', COMD
  IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
  PRINT * , ' '
    PRINT * , 'Current formats for numeric labels:'
    PRINT * , '  Format for U labels: ',UFMT(1:LNBPCS(UFMT))
    PRINT * , '  Format for V labels: ',VFMT(1:LNBPCS(VFMT))
    PRINT * , '  Format for W labels: ',WFMT(1:LNBPCS(WFMT))
    PRINT * , ' '
    PRINT * , 'Formats must not be more than 16 characters'
    PRINT * , 'long and must not specify a line more than'
    PRINT * , '128 characters long.  Enter them carefully.'
    PRINT * , ' '
```

```
            PRINT * , 'Enter new U format:'
            READ '(A16)' , UFMT
            PRINT * , 'Enter new V format:'
            READ '(A16)' , VFMT
            PRINT * , 'Enter new W format:'
            READ '(A16)' , WFMT
          END IF
          PRINT * , ' '
          PRINT * , 'Change informational labels (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            PRINT * , ' '
            PRINT * , 'Current U axis label:',UILB(1:LNBPCS(UILB))
            PRINT * , 'Current V axis label:',VILB(1:LNBPCS(VILB))
            PRINT * , 'Current W axis label:',WILB(1:LNBPCS(WILB))
            PRINT * , 'Current plot label:  ',PILB(1:LNBPCS(PILB))
            PRINT * , ' '
            PRINT * , 'Enter new U axis label:'
            READ '(A128)' , UILB
            PRINT * , 'Enter new V axis label:'
            READ '(A128)' , VILB
            PRINT * , 'Enter new W axis label:'
            READ '(A128)' , WILB
            PRINT * , 'Enter new plot label:'
            READ '(A128)' , PILB
          END IF
          PRINT * , ' '
          PRINT * , 'Change informational-label positions (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            PRINT * , ' '
            PRINT * , 'Current plot label position (horizontal):',PILX
            PRINT * , 'Current plot label position (vertical):  ',PILY
            PRINT * , ' '
            PRINT * , 'Enter new plot label position (horizontal):'
            CALL TDRDRN (PILX,PILX)
            PILX=MAX(-1.,MIN(1.,PILX))
            PRINT * , 'Enter new plot label position (vertical):'
            CALL TDRDRN (PILY,PILY)
            PILY=MAX(-1.,MIN(1.,PILY))
          END IF
          PRINT * , ' '
          IF (IFDE.NE.0) THEN
            PRINT * , 'Axis labels are currently faded for azimuth ang
     +les near 90 degrees.'
          ELSE
            PRINT * , 'Axis labels are not currently faded for azimuth
     + angles near 90 degrees.'
          END IF
          PRINT * , ' '
          PRINT * , 'Fade labels for azimuth angles near 90 degrees (Y
     + or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            IFDE=1
          ELSE
            IFDE=0
          END IF
        END IF
        GO TO 111
C
```

```
        ELSE IF (COMD.EQ.'H'.OR.COMD.EQ.'h') THEN
C
          PRINT * , ' '
          PRINT * , 'Possible commands are as follows:'
          PRINT * , '  D  =>  Data change'
          PRINT * , '  G  =>  Grid change'
          PRINT * , '  H  =>  Help'
          PRINT * , '  M  =>  Miscellaneous changes'
          PRINT * , '  Q  =>  Quit'
          PRINT * , '  R  =>  Refresh/rendering change'
          PRINT * , '  S  =>  Save plot (NCGM and/or PostScript)'
          PRINT * , '  V  =>  View change'
          GO TO 112
C
        ELSE IF (COMD.EQ.'M'.OR.COMD.EQ.'m') THEN
C
          PRINT * , ' '
          IF (IBOW.EQ.0) THEN
            PRINT * , 'The color scheme flag is now 0 (white on black).'
          ELSE
            PRINT * , 'The color scheme flag is now 1 (black on white).'
          END IF
          PRINT * , ' '
          PRINT * , 'Enter new color scheme flag (0 => WOB, 1 => BOW):'
          CALL TDRDIN (ITMP,IBOW)
          ITMP=MAX(0,MIN(1,ITMP))
          IF (ITMP.NE.IBOW) THEN
            IBOW=ITMP
            CALL GCLRWK (1,1)
            DO 115 IWID=1,3
              IF (IWID.NE.1) CALL GACWK (IWID)
              IF (IBOW.EQ.0) THEN
                CALL GSCR (IWID,0,0.,0.,0.) ! black
                CALL GSCR (IWID,1,1.,1.,1.) ! white
              ELSE
                CALL GSCR (IWID,0,1.,1.,1.) ! white
                CALL GSCR (IWID,1,0.,0.,0.) ! black
              END IF
              IF (IWID.NE.1) CALL GDAWK (IWID)
  115       CONTINUE
            SSHE=SHDE
            SSHR=SHDR
          END IF
C
          PRINT * , ' '
          IF (IUXW.EQ.0) THEN
            PRINT * , 'The X-workstation-update flag is now 0 (off).'
          ELSE
            PRINT * , 'The X-workstation-update flag is now 1 (on).'
          END IF
          PRINT * , ' '
          PRINT * , 'Enter new X-workstation-update flag (0 => off, 1 =>
     + on):'
          CALL TDRDIN (IUXW,IUXW)
          IUXW=MAX(0,MIN(1,IUXW))
C
          PRINT * , ' '
          IF (IHND.EQ.0) THEN
            PRINT * , 'The handedness flag is now 0 (right-handed).'
          ELSE
            PRINT * , 'The handedness flag is now 1 (left-handed).'
```

```
        END IF
        PRINT * , ' '
        PRINT * , 'Enter new handedness flag (0 => right, 1 => left):'
        CALL TDRDIN (IHND,IHND)
        IHND=MAX(0,MIN(1,IHND))
        CALL TDSETI ('HND',IHND)
C
        PRINT * , ' '
        PRINT * , 'Current value of stereo-view flag: ',STVF
        PRINT * , ' '
        PRINT * , 'Enter a new value of the stereo-view flag, which'
        PRINT * , 'is set non-zero to make the S command save stereo'
        PRINT * , 'views.  Use a negative value to get each stereo'
        PRINT * , 'pair on a single frame or a positive value to use'
        PRINT * , 'separate frames.  The absolute value is the angle,'
        PRINT * , 'in degrees, between the lines of sight of the two'
        PRINT * , 'views.  (A value of one or two degrees seems to'
        PRINT * , 'work best):'
        CALL TDRDRN (STVF,STVF)
        STVF=MAX(-10.,MIN(10.,STVF))
C
        PRINT * , ' '
        PRINT * , 'Current character size multiplier: ',RCS1
        PRINT * , ' '
        PRINT * , 'Enter new value for character size multiplier:'
        CALL TDRDRN (RCS1,RCS1)
        RCS1=MAX(.1,MIN(10.,RCS1))
C
        PRINT * , ' '
        PRINT * , 'Multiplier for random term in dummy data: ',RNDM
        PRINT * , ' '
        PRINT * , 'Enter new multiplier for random term:'
        CALL TDRDRN (RNDM,RNDM)
        RNDM=MAX(0.,RNDM)
C
        PRINT * , ' '
        PRINT * , 'Current triangle-ordering flag: ',IORD
        PRINT * , ' '
        PRINT * , 'Enter new triangle-ordering flag (0 => sort on  '
        PRINT * , 'distances to triangle centers, -1 => sort on    '
        PRINT * , 'distances to triangle farpoints, +1 => sort on  '
        PRINT * , 'distances to triangle farpoints and then run an '
        PRINT * , 'algorithm to fix ordering errors):'
        CALL TDRDIN (IORD,IORD)
        IORD=MAX(-1,MIN(+1,IORD))
C
        PRINT * , ' '
        PRINT * , 'Current flag for the grid of 3D marks: ',IMRK
        IF (IMRK.EQ.0) THEN
          PRINT * , 'No grid of 3D marks is to be generated.'
        ELSE
          PRINT * , 'Grid of 3D marks is to be generated, each ',
     +                    IMKT(IABS(IMRK))(1:LNBPCS(IMKT(IABS(IMRK))))
          IF (IMRK.LT.0) THEN
            PRINT * , 'Marks are not to be clipped by the data box.'
          ELSE
            PRINT * , 'Marks are to be clipped by the data box.'
          END IF
        END IF
        IMKO=IMRK
        SMKO=SMRK
```

```
        UMKO=UMKS
        VMKO=VMKS
        WMKO=WMKS
        PRINT * , ' '
        PRINT * , 'Enter new value of flag for the grid of 3D marks'
        PRINT * , '(0 => no marks, 1 => tetrahedron, 2 => octahedron,'
        PRINT * , '3 = > cube, 4 => icosahedron, 5 => sphere; use a'
        PRINT * , 'negated value to omit clipping by data box.'
        CALL TDRDIN (IMRK,IMRK)
        IMRK=MAX(-5,MIN(5,IMRK))
        IF (IMRK.NE.0) THEN
          PRINT * , ' '
          PRINT * , 'Current value of mark size parameter:   ',SMRK
          PRINT * , ' '
          PRINT * , 'Enter new mark size (as a fraction of the'
          PRINT * , 'smallest side of the data box):'
          CALL TDRDRN (SMRK,SMRK)
          SMRK=MAX(.001,MIN(1.,SMRK))
          PRINT * , ' '
          PRINT * , 'Current mark spacing parameters:',UMKS,VMKS,WMKS
          PRINT * , ' '
          PRINT * , 'Enter new mark spacing in U:'
          CALL TDRDRN (UMKS,UMKS)
          UMKS=MAX(0.,UMKS)
          PRINT * , 'Enter new mark spacing in V:'
          CALL TDRDRN (VMKS,VMKS)
          VMKS=MAX(0.,VMKS)
          PRINT * , 'Enter new mark spacing in W:'
          CALL TDRDRN (WMKS,WMKS)
          WMKS=MAX(0.,WMKS)
        END IF
C
        PRINT * , ' '
        PRINT * , 'Current isosurface cut-off value 1: ',FIS1
        PRINT * , 'Current isosurface cut-off value 2: ',FIS2
        PRINT * , ' '
        PRINT * , 'Isosurface cut-off values must be between 0'
        PRINT * , 'and 200 (the initial values were 95 and 95).'
        PRINT * , 'If the two values are equal, one isosurface'
        PRINT * , 'is specified; if the two values are different,'
        PRINT * , 'two isosurfaces are specified.'
        PRINT * , ' '
        PRINT * , 'Enter new isosurface cut-off value 1:'
        OFS1=FIS1
        CALL TDRDRN (FIS1,FIS1)
        FIS1=MAX(0.,MIN(200.,FIS1))
        PRINT * , 'Enter new isosurface cut-off value 2:'
        OFS2=FIS2
        CALL TDRDRN (FIS2,FIS2)
        FIS2=MAX(0.,MIN(200.,FIS2))
        IF (FIS1.GT.FIS2) THEN
          FIS0=FIS1
          FIS1=FIS2
          FIS2=FIS0
        END IF
C
        IF (IMRK.NE.IMKO.OR.SMRK.NE.SMKO.OR.UMKS.NE.UMKO.OR.
     +        UMKS.NE.UMKO.OR.VMKS.NE.VMKO.OR.WMKS.NE.WMKO.OR.
     +        FIS1.NE.OFS1.OR.FIS2.NE.OFS2) THEN
          GO TO 110
        ELSE
```

```
            GO TO 111
          END IF
C
        ELSE IF (COMD.EQ.'Q'.OR.COMD.EQ.'q') THEN
C
          CALL GDAWK (1)
          CALL GCLWK (1)
          CALL GCLWK (2)
          IF (NFPS.NE.0) THEN
            CALL GACWK (3)
            CALL GCLRWK (3,1)
            CALL GDAWK (3)
          END IF
          CALL GCLWK (3)
          CALL GCLKS
          STOP
C
        ELSE IF (COMD.EQ.'R'.OR.COMD.EQ.'r') THEN
C
          PRINT * , ' '
          PRINT * , 'Redrawing the picture.'
          PRINT * , ' '
          PRINT * , 'Change the shading parameters (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
            PRINT * , ' '
            PRINT * , 'Current shading parameter values are as follows:'
            PRINT * , '  Shading parameter 1:   ',SHDE
            PRINT * , '  Shading parameter 2:   ',SHDR
            PRINT * , '  TDPACK parameter SHD:  ',ISHD
            PRINT * , '  Light source azimuth:  ',ANG3
            PRINT * , '  Light source elevation:',ANG4
            PRINT * , ' '
            PRINT * , 'Note that if both angles are zero, the light'
            PRINT * , 'source is at the position of the viewer.'
            PRINT * , ' '
            PRINT * , 'Enter new shading parameter 1 (0 => shading off,'
            PRINT * , 'near 0 => brighter colors, near 1 => pastels).'
            CALL TDRDRN (SHDE,SHDE)
            SHDE=MAX(0.,MIN(1.,SHDE))
            PRINT * , 'Enter new shading parameter 2 (near 0 => small'
            PRINT * , 'range of shades, near 1 => full range).'
            CALL TDRDRN (SHDR,SHDR)
            SHDR=MAX(0.,MIN(1.,SHDR))
            PRINT * , 'Enter new value of TDPACK internal parameter SHD'
            PRINT * , '(0 => shading over a 90-degree range of triangle'
            PRINT * , 'orientations, 1 => over a 180-degree range):'
            CALL TDRDIN (ISHD,ISHD)
            ISHD=MAX(0,MIN(1,ISHD))
            CALL TDSETI ('SHD',ISHD)
            PRINT * , 'Enter light source angle 1 (azimuth, in degrees):
     +'
            CALL TDRDRN (ANG3,ANG3)
            ANG3=SIGN(MOD(ABS(ANG3),360.),ANG3)
            PRINT * , 'Enter light source angle 2 (elevation, in degrees
     +):'
            CALL TDRDRN (ANG4,ANG4)
            ANG4=MAX(-89.999,MIN(+89.999,ANG4))
            IF (ANG3.NE.0..OR.ANG4.NE.0.) THEN
              UCLS=UMID+100.*DOFB*COS(DTOR*ANG3)*COS(DTOR*ANG4)
              VCLS=VMID+100.*DOFB*SIN(DTOR*ANG3)*COS(DTOR*ANG4)
```

```
            WCLS=WMID+100.*DOFB*SIN(DTOR*ANG4)
          ELSE
            UCLS=0.
            VCLS=0.
            WCLS=0.
          END IF
          CALL TDSETR ('LSU',UCLS)
          CALL TDSETR ('LSV',VCLS)
          CALL TDSETR ('LSW',WCLS)
          IF (SHDE.NE.0..AND.(SHDE.NE.SSHE.OR.SHDR.NE.SSHR)) THEN
            CALL GCLRWK (1,1)
            DO 117 IWID=1,3
              IF (IWID.NE.1) CALL GACWK (IWID)
              CALL TDCLRS (IWID,IBOW,SHDE,SHDR,101,116,0)
              IF (IWID.NE.1) CALL GDAWK (IWID)
  117       CONTINUE
            SSHE=SHDE
            SSHR=SHDR
          END IF
        END IF
        PRINT * , ' '
        PRINT * , 'Change the rendering parameters (Y or N)?'
        READ  '(A1)', COMD
        IF (COMD.EQ.'Y'.OR.COMD.EQ.'y') THEN
          PRINT * , ' '
          PRINT * , 'Usable color indices are as follows:'
          PRINT * , '  -1 => if line color, line not drawn; if fill co
     +lor, no fill done'
          PRINT * , '   0 => background or 101-116 => shades from all-
     +white to all-black'
          PRINT * , '   1 => foreground or 117-132 => shades of gray (
     +light to dark)'
          PRINT * , '   2 => red        or 133-148 => shades of red (l
     +ight to dark)'
          PRINT * ,  '   3 => green      or 149-164 => shades of green
     + (light to dark)'
          PRINT * , '   4 => blue       or 165-180 => shades of blue (
     +light to dark)'
          PRINT * , '   5 => cyan       or 181-196 => shades of cyan (
     +light to dark)'
          PRINT * , '   6 => magenta    or 197-212 => shades of magent
     +a (light to dark)'
          PRINT * , '   7 => yellow     or 213-228 => shades of yellow
     + (light to dark)'
          PRINT * , '  Colors with indices 117 through 228 are affecte
     +d by the values of'
          PRINT * , '  shading flags.  Using a fill color index 0 thro
     +ugh 7 when shading'
          PRINT * , '  is turned on implies using shades of that color
     + (for example, use'
          PRINT * , '  of the value 4 implies the use of color indices
     + 165 through 180).'
          PRINT * , ' '
          PRINT * , 'Current rendering values are as follows:'
          PRINT * , '  Bottom/top fill colors:      ',IFCB,IFCT
          PRINT * , '  Bottom/top line colors:      ',ILCB,ILCT
          PRINT * , '  Triangle-drawing flag:       ',IDTR
          PRINT * , '  U/V/W slice separations:     ',USIN,VSIN,WSIN
          PRINT * , ' '
          PRINT * , 'Enter new bottom fill color:'
          CALL TDRDIN (IFCB,IFCB)
```

```
            IFCB=MAX(-1,IFCB)
            PRINT * , 'Enter new top fill color:'
            CALL TDRDIN (IFCT,IFCT)
            IFCT=MAX(-1,IFCT)
            PRINT * , 'Enter new bottom line color:'
            CALL TDRDIN (ILCB,ILCB)
            ILCB=MAX(-1,ILCB)
            PRINT * , 'Enter new top line color:'
            CALL TDRDIN (ILCT,ILCT)
            ILCT=MAX(-1,ILCT)
            PRINT * , 'Enter new triangle-drawing flag (0 or 1):'
            CALL TDRDIN (IDTR,IDTR)
            IDTR=MAX(0,MIN(1,IDTR))
            PRINT * , 'Enter new U slice separation (0 => none):'
            CALL TDRDRN (USIN,USIN)
            USIN=MAX(0.,USIN)
            PRINT * , 'Enter new V slice separation (0 => none):'
            CALL TDRDRN (VSIN,VSIN)
            VSIN=MAX(0.,VSIN)
            PRINT * , 'Enter new W slice separation (0 => none):'
            CALL TDRDRN (WSIN,WSIN)
            WSIN=MAX(0.,WSIN)
          END IF
          GO TO 111
C
        ELSE IF (COMD.EQ.'S'.OR.COMD.EQ.'s') THEN
C
          PRINT * , ' '
          IF (STVF.EQ.0.) THEN
            PRINT * , 'Saving just the current image.  (An M command'
            PRINT * , 'can be used to turn on saving of stereo pairs.)'
          ELSE
            PRINT * , 'Saving a stereo pair of the current image.'
          END IF
          PRINT * , ' '
          PRINT * , 'Do you want to save to an NCAR CGM file (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.NE.'Y'.AND.COMD.NE.'y') THEN
            ISNG=0
          ELSE
            ISNG=1
            CALL GACWK (2)
            NFNG=NFNG+1
            IF (NFNG.GT.1) CALL GCLRWK (2,1)
          END IF
          PRINT * , ' '
          PRINT * , 'Do you want to save to a PostScript file (Y or N)?'
          READ  '(A1)', COMD
          IF (COMD.NE.'Y'.AND.COMD.NE.'y') THEN
            ISPS=0
          ELSE
            ISPS=1
            CALL GACWK (3)
            NFPS=NFPS+1
            IF (NFPS.GT.1) CALL GCLRWK (3,1)
          END IF
          IF (ISNG.NE.0.OR.ISPS.NE.0) THEN
            CALL GDAWK(1)
            OFFS=-(DEYE+DAPT)*TAN(DTOR*ABS(STVF)/2.)
  118       PRINT * , ' '
            IF      (OFFS.LT.0.) THEN
```

```
          PRINT * , 'Initializing TDPACK for left-eye view.'
        ELSE IF (OFFS.GT.0.) THEN
          PRINT * , 'Initializing TDPACK for right-eye view.'
        ELSE
          PRINT * , 'Initializing TDPACK for single view.'
        END IF
        CALL TDINIT (UEYE,VEYE,WEYE,UAPT,VAPT,WAPT,
     +                             UAPT,VAPT,WAPT+1.,OFFS)
        CALL GETSET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
        IF (OFFS.NE.0..AND.STVF.LT.0.) THEN
          IF (OFFS.LT.0.) THEN
            CALL SET  (1.-WOSW,1.,.5-.5*WOSW,.5+.5*WOSW,
     +                        XWDL,XWDR,YWDB,YWDT,LNLG)
          ELSE
            CALL SET  (  0., WOSW,.5-.5*WOSW,.5+.5*WOSW,
     +                        XWDL,XWDR,YWDB,YWDT,LNLG)
          END IF
        END IF
        IF (PILB.NE.' ') THEN
          PRINT * , 'Drawing plot label.'
          CALL TDPARA (UMID-.5*DOFB*(UDCX+UDCY),
     +                 VMID-.5*DOFB*(VDCX+VDCY),
     +                 WMID-.5*DOFB*(WDCX+WDCY),
     +                 UDCX,VDCX,WDCX,UDCY,VDCY,WDCY)
          CALL TDPLCH ((.5+PILX)*DOFB,(.5+PILY)*DOFB,
     +                 PILB(1:LNBPCS(PILB)),.04*CSM2,0.,0.)
        END IF
        IF (IGDF.NE.0) THEN
          PRINT * , 'Drawing axis labels.'
          IF (IFDE.NE.0) THEN
            ANGM=MOD(ANG1,90.)
            IF (ANGM.LT. 0.) ANGM=90.+ANGM
            IF (ANGM.GT.45.) ANGM=90.-ANGM
            MANG=101+INT(ANGM)
          ELSE
            MANG=116
          END IF
          IF (MANG.GT.101) THEN
            IF (MANG.GE.102.AND.MANG.LE.115) THEN
              CALL SFLUSH
              IF (IBOW.EQ.0) THEN
                CALL GSPLCI (217-MANG)
                CALL GSFACI (217-MANG)
              ELSE
                CALL GSPLCI (MANG)
                CALL GSFACI (MANG)
              END IF
            END IF
            CALL TDLBLS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                   UNLB,VNLB,WNLB,UILB,VILB,WILB,1)
            IF (MANG.GE.102.AND.MANG.LE.115) THEN
              CALL SFLUSH
              CALL GSPLCI (1)
              CALL GSFACI (1)
            END IF
          END IF
          PRINT * , 'Drawing far sides of box.'
          CALL TDGRDS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                 UGIM*UGIN,VGIM*VGIN,WGIM*WGIN,IGDF,1)
        END IF
        PRINT * , 'Rendering surface.'
```

```
          CALL TDOTRI (RTRI,MTRI,NTRI,RTWK,ITWK,IORD)
          CALL TDDTRI (RTRI,MTRI,NTRI,ITWK)
          IF (IGDF.NE.0) THEN
            PRINT * , 'Drawing near sides of box.'
            CALL TDGRDS (UMIN,VMIN,WMIN,UMAX,VMAX,WMAX,
     +                   UGIM*UGIN,VGIM*VGIN,WGIM*WGIN,IGDF,0)
          END IF
          IF      (OFFS.LT.0.) THEN
            CALL PLCHHQ (.98*XWDL+.02*XWDR,.98*YWDB+.02*YWDT,'L',.006,
     +                                                          0.,0.)
          ELSE IF (OFFS.GT.0.) THEN
            CALL PLCHHQ (.02*XWDL+.98*XWDR,.98*YWDB+.02*YWDT,'R',.006,
     +                                                          0.,0.)
          END IF
          CALL SFLUSH
          IF (ISNG.NE.0) CALL GUWK (2,0)
          IF (ISPS.NE.0) CALL GUWK (3,0)
          IF (OFFS.LT.0.) THEN
            OFFS=-OFFS
            IF (STVF.GT.0.) THEN
              IF (ISNG.NE.0) CALL GCLRWK (2,1)
              IF (ISPS.NE.0) CALL GCLRWK (3,1)
            END IF
            GO TO 118
          END IF
          IF (ISNG.NE.0) CALL GDAWK (2)
          IF (ISPS.NE.0) CALL GDAWK (3)
          CALL GACWK(1)
        END IF
        GO TO 112
C
      ELSE IF (COMD.EQ.'V'.OR.COMD.EQ.'v') THEN
C
        PRINT * , ' '
        PRINT * , 'Changing view angle and distance.  Current'
        PRINT * , 'values are as follows:'
        PRINT * , '  Viewing angle 1 (azimuth):                  ',ANG1
        PRINT * , '  Viewing angle 2 (elevation):                ',ANG2
        PRINT * , '  Distance to eye (diagonal multiple):        ',REYE
        PRINT * , '  Distance to aim point (diagonal multiple): ',RAPT
        PRINT * , '  Field of view (in degrees):                 ',FOVP
        PRINT * , ' '
        PRINT * , 'Enter viewing angle 1 (azimuth, in degrees):'
        CALL TDRDRN (ANG1,ANG1)
        ANG1=SIGN(MOD(ABS(ANG1),360.),ANG1)
        PRINT * , 'Enter viewing angle 2 (elevation, in degrees):'
        CALL TDRDRN (ANG2,ANG2)
        ANG2=MAX(-89.999,MIN(+89.999,ANG2))
        PRINT * , 'Enter distance to eye (diagonal multiple):'
        CALL TDRDRN (REYE,REYE)
        REYE=MAX(.51,MIN(51.,REYE))
        PRINT * , 'Enter distance to aim point (diagonal multiple):'
        CALL TDRDRN (RAPT,RAPT)
        RAPT=MAX(-.9*REYE,MIN(100.*REYE,RAPT))
        PRINT * , 'Enter desired field of view (in degrees):'
        CALL TDRDRN (FOVP,FOVP)
        FOVP=MAX(1.,MIN(179.,FOVP))
        CALL TDSETR ('FOV',FOVP)
        GO TO 111
C
      ELSE
```

```
C
          PRINT * , ' '
          PRINT * , 'Illegal command.  Try again.'
C
          GO TO 112
C
        END IF
C
      END


      SUBROUTINE GENDAT (DATA,IMAX,IDIM,JDIM,MLOW,MHGH,DLOW,DHGH,ICEN)
C
C This is a routine to generate test data for two-dimensional graphics
C routines.  Given an array "DATA", dimensioned "IMAX x 1", it fills
C the sub-array ((DATA(I,J),I=1,IDIM),J=1,JDIM) with a two-dimensional
C field of data having approximately "MLOW" lows and "MHGH" highs, a
C minimum value of exactly "DLOW" and a maximum value of exactly "DHGH".
C If ICEN is non-zero, new centers are computed for the exponentials,
C yielding a whole new field.
C
C "MLOW" and "MHGH" are each forced to be greater than or equal to 1
C and less than or equal to 25.
C
C The function used is a sum of exponentials.
C
        DIMENSION DATA(IMAX,*),CCNT(3,50)
C
        SAVE NLOW,NHGH,NCNT,CCNT
C
        DATA NCNT / 0 /
C
        IF (ICEN.NE.0.OR.NCNT.EQ.0) THEN
C
          NLOW=MAX0(1,MIN0(25,MLOW))
          NHGH=MAX0(1,MIN0(25,MHGH))
          NCNT=NLOW+NHGH
C
          DO 101 ICNT=1,NCNT
            CCNT(1,ICNT)=FRAN()
            CCNT(2,ICNT)=FRAN()
            IF (ICNT.LE.NLOW) THEN
              CCNT(3,ICNT)=-1.
            ELSE
              CCNT(3,ICNT)=+1.
            END IF
  101     CONTINUE
C
        END IF
C
        DMIN=+1.E36
        DMAX=-1.E36
C
        DO 104 I=1,IDIM
          XPOS=REAL(I-1)/REAL(IDIM-1)
          DO 103 J=1,JDIM
            YPOS=REAL(J-1)/REAL(JDIM-1)
            DATA(I,J)=0.
            DO 102 K=1,NCNT
              TEMP=-50.*((XPOS-CCNT(1,K))**2+(YPOS-CCNT(2,K))**2)
              IF (TEMP.GE.-20.) DATA(I,J)=DATA(I,J)+CCNT(3,K)*EXP(TEMP)
```

```
  102       CONTINUE
            DMIN=MIN(DMIN,DATA(I,J))
            DMAX=MAX(DMAX,DATA(I,J))
  103     CONTINUE
  104   CONTINUE
C
        DO 106 I=1,IDIM
          DO 105 J=1,JDIM
            DATA(I,J)=(DATA(I,J)-DMIN)/(DMAX-DMIN)*(DHGH-DLOW)+DLOW
  105     CONTINUE
  106   CONTINUE
C
        RETURN
C
      END


      SUBROUTINE ADDRAN (DATA,IMAX,IDIM,JDIM,DLOW,DHGH,RNDM)
C
        DIMENSION DATA(IMAX,*)
C
C This subroutine is called to add a random quantity to each element of
C the IDIM x JDIM array DATA, whose first FORTRAN dimension is IMAX.
C
C The magnitude of the random quantity is stated as a multiple (RNDM)
C of the range of the data.
C
        DMUL=RNDM*(DHGH-DLOW)
C
        DMIN=+1.E36
        DMAX=-1.E36
C
C Add random quantities to the data.
C
        DO 102 I=1,IDIM
          DO 101 J=1,JDIM
            DATA(I,J)=DATA(I,J)+DMUL*FRAN()
            DMIN=MIN(DMIN,DATA(I,J))
            DMAX=MAX(DMAX,DATA(I,J))
  101     CONTINUE
  102   CONTINUE
C
C Readjust the data to have the high and low values as before.
C
        DO 104 I=1,IDIM
          DO 103 J=1,JDIM
            DATA(I,J)=(DATA(I,J)-DMIN)/(DMAX-DMIN)*(DHGH-DLOW)+DLOW
  103     CONTINUE
  104   CONTINUE
C
C Done.
C
        RETURN
C
      END


      SUBROUTINE TDGNDT (DATA,IMAX,JMAX,IDIM,JDIM,KDIM,MLOW,MHGH,
     +                                        DLOW,DHGH,ICEN)
C
C This is a routine to generate test data for three-dimensional graphics
```

```
C routines.  Given an array "DATA", dimensioned "IMAX x JMAX x ...", it
C fills the sub-array (((DATA(I,J,K),I=1,MX),J=1,MY),K=1,MZ) with a
C three-dimensional field of data having approximately "MLOW" lows and
C "MHGH" highs, a minimum value of exactly "DLOW" and a maximum value
C of exactly "DHGH".  If ICEN is non-zero, new centers are computed for
C the exponentials, yielding a whole new field.
C
C "MLOW" and "MHGH" are each forced to be greater than or equal to 1
C and less than or equal to 25.
C
C The function used is a sum of exponentials.
C
        DIMENSION DATA(IMAX,JMAX,*),CCNT(4,50)
C
        SAVE NLOW,NHGH,NCNT,CCNT
C
        DATA NCNT / 0 /
C
        IF (ICEN.NE.0.OR.NCNT.EQ.0) THEN
C
          NLOW=MAX0(1,MIN0(25,MLOW))
          NHGH=MAX0(1,MIN0(25,MHGH))
          NCNT=NLOW+NHGH
C
          DO 101 ICNT=1,NCNT
            CCNT(1,ICNT)=FRAN()
            CCNT(2,ICNT)=FRAN()
            CCNT(3,ICNT)=FRAN()
            IF (ICNT.LE.NLOW) THEN
              CCNT(4,ICNT)=-1.
            ELSE
              CCNT(4,ICNT)=+1.
            END IF
  101     CONTINUE
C
        END IF
C
        DMIN=+1.E36
        DMAX=-1.E36
C
        DO 105 I=1,IDIM
          XPOS=REAL(I-1)/REAL(IDIM-1)
          DO 104 J=1,JDIM
            YPOS=REAL(J-1)/REAL(JDIM-1)
            DO 103 K=1,KDIM
              ZPOS=REAL(K-1)/REAL(KDIM-1)
              DATA(I,J,K)=0.
              DO 102 ICNT=1,NCNT
                TEMP=-50.*((XPOS-CCNT(1,ICNT))**2+
     +                     (YPOS-CCNT(2,ICNT))**2+
     +                     (ZPOS-CCNT(3,ICNT))**2)
                IF (TEMP.GE.-20.) DATA(I,J,K)=DATA(I,J,K)+
     +                                     CCNT(4,ICNT)*EXP(TEMP)
  102         CONTINUE
              DMIN=MIN(DMIN,DATA(I,J,K))
              DMAX=MAX(DMAX,DATA(I,J,K))
  103       CONTINUE
  104     CONTINUE
  105   CONTINUE
C
        DO 108 I=1,IDIM
```

```
          DO 107 J=1,JDIM
            DO 106 K=1,KDIM
              DATA(I,J,K)=DLOW+((DATA(I,J,K)-DMIN)/
     +                           (DMAX       -DMIN))*(DHGH-DLOW)
  106       CONTINUE
  107     CONTINUE
  108   CONTINUE
C
        RETURN
C
      END


      SUBROUTINE TDADRN (DATA,IMAX,JMAX,IDIM,JDIM,KDIM,DLOW,DHGH,RNDM)
C
        DIMENSION DATA(IMAX,JMAX,*)
C
C This subroutine is called to add a random quantity to each element of
C the IDIM x JDIM x KDIM array DATA, whose first two FORTRAN dimensions
C are IMAX and JMAX.
C
C The magnitude of the random quantity is stated as a multiple (RNDM)
C of the range of the data.
C
        DMUL=RNDM*(DHGH-DLOW)
C
        DMIN=+1.E36
        DMAX=-1.E36
C
C Add random quantities to the data.
C
        DO 103 I=1,IDIM
          DO 102 J=1,JDIM
            DO 101 K=1,KDIM
              DATA(I,J,K)=DATA(I,J,K)+DMUL*FRAN()
              DMIN=MIN(DMIN,DATA(I,J,K))
              DMAX=MAX(DMAX,DATA(I,J,K))
  101       CONTINUE
  102     CONTINUE
  103   CONTINUE
C
C Readjust the data to have the high and low values as before.
C
        DO 106 I=1,IDIM
          DO 105 J=1,JDIM
            DO 104 K=1,KDIM
              DATA(I,J,K)=(DATA(I,J,K)-DMIN)/(DMAX-DMIN)*
     +                                        (DHGH-DLOW)+DLOW
  104       CONTINUE
  105     CONTINUE
  106   CONTINUE
C
C Done.
C
        RETURN
C
      END


      FUNCTION FRAN()
C
```

```
C Pseudo-random-number generator.
C
        DOUBLE PRECISION X
        SAVE X
        DATA X / 2.718281828459045 /
        X=MOD(9821.D0*X+.211327D0,1.D0)
        FRAN=REAL(X)
        RETURN
      END


      SUBROUTINE TDRDIN (IVAL,IDEF)
C
C This routine reads a line of input to get an integer value IVAL.  If
C the line is entirely blank, the default value IDEF is returned.  If
C the line contains an illegal character, another line is obtained.
C
        CHARACTER*128 LINE
C
C Assume the default value will be returned until we find otherwise.
C
        IVAL=IDEF
C
C Get a line from standard input.
C
        READ '(A128)' , LINE
C
        IF (LINE.NE.' ') THEN
          READ (LINE,*) IVAL
        END IF
C
C Done.
C
        RETURN
C
      END


      SUBROUTINE TDRDRN (RVAL,RDEF)
C
C This routine reads a line of input to get a real number RVAL.  If
C the line is entirely blank, the default value RDEF is returned.  If
C the line contains an illegal character, another line is obtained.
C
        CHARACTER*128 LINE
C
C Assume the default value will be returned until we find otherwise.
C
        RVAL=RDEF
C
C Get a line from standard input.
C
        READ '(A128)' , LINE
C
        IF (LINE.NE.' ') THEN
          READ (LINE,*) RVAL
        END IF
C
C Done.
C
        RETURN
```

```
C
      END
```