`-F`*fontname*
`-F`*fontname*`:`*size*
`-F:`*size* *fontname* set the postscript font (for use with postscript, aifm, corel and fig). By default, 'Helvetica' is set for PS/Aifm, and 'SwitzerlandLight' for Corel. It can also be 'Times-Roman'. *size* is given in points. *fontname* is ignored for the fig device.

The filename and options can be given in any order.

`orient (`*orientation*`)` [Function File]
Set the default print orientation. Valid values for *orientation* include `"landscape"` and `"portrait"`. If called with no arguments, return the default print orientation.

15.2.7 Interacting with plots

The user can select points on a plot with the `ginput` function or selection the position at which to place text on the plot with the `gtext` function using the mouse.

`[`*x*`, `*y*`, `*buttons*`] = ginput (`*n*`)` [Function File]
Return which mouse buttons were pressed and keys were hit on the current figure. If *n* is defined, then wait for *n* mouse clicks before returning. If *n* is not defined, then `ginput` will loop until the return key is pressed.

b `= waitforbuttonpress ()` [Function File]
Wait for button or mouse press.over a figure window. The value of *b* returns 0 if a mouse button was pressed or 1 is a key was pressed.

See also: [ginput], page 243.

`gtext (`*s*`)` [Function File]
`gtext ({`*s1*`; `*s2*`; ...})` [Function File]
`gtext (...,` *prop*`, `*val*`)` [Function File]
Place text on the current figure using the mouse. The text is defined by the string *s*. If *s* is a cell array, each element of the cell array is written to a separate line. Additional arguments are passed to the underlying text object as properties.

See also: [ginput], page 243, [text], page 237.

15.2.8 Test Plotting Functions

The functions `sombrero` and `peaks` provide a way to check that plotting is working. Typing either `sombrero` or `peaks` at the Octave prompt should display a three-dimensional plot.

`sombrero (`*n*`)` [Function File]
Produce the familiar three-dimensional sombrero plot using *n* grid lines. If *n* is omitted, a value of 41 is assumed.

The function plotted is

```
z = sin (sqrt (x^2 + y^2)) / (sqrt (x^2 + y^2))
```

See also: [surf], page 229, [meshgrid], page 230, [mesh], page 228.

`peaks ()` [Function File]
`peaks (n)` [Function File]
`peaks (x, y)` [Function File]
`z = peaks (...)` [Function File]
`[x, y, z] = peaks (...)` [Function File]

Generate a function with lots of local maxima and minima. The function has the form

$$f(x,y) = 3(1-x)^2 e^{\left(-x^2-(y+1)^2\right)} - 10\left(\frac{x}{5} - x^3 - y^5\right)) - \frac{1}{3}e^{\left(-(x+1)^2-y^2\right)}$$

Called without a return argument, `peaks` plots the surface of the above function using `mesh`. If n is a scalar, the `peaks` returns the values of the above function on a n-by-n mesh over the range `[-3,3]`. The default value for n is 49.

If n is a vector, then it represents the x and y values of the grid on which to calculate the above function. The x and y values can be specified separately.

See also: [surf], page 229, [mesh], page 228, [meshgrid], page 230.

15.3 Graphics Data Structures

15.3.1 Introduction to Graphics Structures

The graphics functions use pointers, which are of class graphics_handle, in order to address the data structures which control graphical displays. A graphics handle may point any one of a number of different object types. The objects are the graphics data structures. The types of objects are: `figure`, `axes`, `line`, `text`, `patch`, and `surface`. (`image` is still available, but is deprecated and will be removed.) Each of these objects has a function by the same name. and, each of these functions returns a graphics handle pointing to an object of corresponding type. In addition, the function `plot` returns a handle pointing to an object of type line, the function `subplot` returns a handle pointing to an object of type axes, and the function `fill` returns a handle pointing to an object of type patch.

Graphics handles may be distinguished from function handles (Section 11.9.1 [Function Handles], page 155) by means of the function `ishandle()`. `ishandle` returns true if its argument is a handle of a graphics object. In addition, the figure object may be tested using `isfigure()`. `isfigure` returns true only if its argument is a handle of a figure. ishghandle() is synonymous with ishandle(). The `whos` function can be used to show the object type of each currently defined graphics handle. (Note: this is not true today, but it is, I hope, considered an error in whos. It may be better to have whos just show graphics_handle as the class, and provide a new function which, given a graphics handle, returns its object type. This could generalize the ishandle() functions and, in fact, replace them.)

The get and set commands are used to obtain and set the properties of graphics objects.

For example, the property "type" of the graphics object pointed to by the graphics handle h may be displayed by:

```
get(h, "type")
```

The allowed properties of a graphics object may be displayed by: `get(h, "");` where h is a handle of a graphics object.

Thus, for example,

```
h=figure();
get(h,"type")
ans = figure

get(h, "");
error: get: ambiguous figure property name ; possible matches:

__backend__            currentobject          paperposition          toolbar
__enhanced__           deletefcn              paperpositionmode      type
__modified__           dockcontrols           papersize              uicontextmenu
__myhandle__           doublebuffer           papertype              units
__plot_stream__        filename               paperunits             userdata
alphamap               handlevisibility       parent                 visible
beingdeleted           hittest                pointer                windowbuttondownfcn
busyaction             integerhandle          pointershapecdata      windowbuttonmotionfcn
buttondownfcn          interruptible          pointershapehotspot    windowbuttonupfcn
children               inverthardcopy         position               windowbuttonwheelfcn
clipping               keypressfcn            renderer               windowstyle
closerequestfcn        keyreleasefcn          renderermode           wvisual
color                  menubar                resize                 wvisualmode
colormap               mincolormap            resizefcn              xdisplay
createfcn              name                   selected               xvisual
current_point          nextplot               selectionhighlight     xvisualmode
currentaxes            numbertitle            selectiontype
currentcharacter       paperorientation       tag

aa=axes();
get(aa,"");
error: get: ambiguous axes property name ; possible matches:

__modified__            drawmode               tickdir                xtickmode
__myhandle__            fontangle              tickdirmode            yaxislocation
activepositionproperty  fontname               ticklength             ycolor
alim                    fontsize               tightinset             ydir
alimmode                fontunits              title                  ygrid
ambientlightcolor       fontweight             type                   ylabel
beingdeleted            gridlinestyle          uicontextmenu          ylim
box                     handlevisibility       units                  ylimmode
busyaction              hittest                userdata               yminorgrid
buttondownfcn           interpreter            view                   yminortick
cameraposition          interruptible          visible                yscale
camerapositionmode      key                    x_normrendertransform  ytick
cameratarget            keybox                 x_projectiontransform  yticklabel
cameratargetmode        keypos                 x_rendertransform      yticklabelmode
cameraupvector          keyreverse             x_viewporttransform    ytickmode
cameraupvectormode      layer                  x_viewtransform        zcolor
cameraviewangle         linestyleorder         xaxislocation          zdir
cameraviewanglemode     linewidth              xcolor                 zgrid
children                minorgridlinestyle     xdir                   zlabel
clim                    nextplot               xgrid                  zlim
climmode                outerposition          xlabel                 zlimmode
clipping                parent                 xlim                   zminorgrid
color                   plotboxaspectratio     xlimmode               zminortick
colororder              plotboxaspectratiomode xminorgrid             zscale
createfcn               position               xminortick             ztick
currentpoint            projection             xscale                 zticklabel
dataaspectratio         selected               xtick                  zticklabelmode
dataaspectratiomode     selectionhighlight     xticklabel             ztickmode
deletefcn               tag                    xticklabelmode
```

The root figure has index 0. Its properties may be displayed by:

```
get(0,"");
```

15.3.2 Graphics Objects

Plots in Octave are constructed using the following *graphics objects*. Each graphics object has a set of properties that define its appearance and may also contain links to other graphics objects. Graphics objects are only referenced by *handle*.

root figure — The parent of all figure objects. The index for the root figure is defined to be 0.

figure — A figure window.

axes — A set of axes. This object is a child of a figure object and may be a parent of line, text, image, patch, or surface objects.

line — A line in two or three dimensions.

text — Text annotations.

image — A bitmap image.

patch — A filled polygon, currently limited to two dimensions.

surface — A three-dimensional surface.

To determine whether a variable is a graphics object index or a figure index, use the functions `ishandle` and `isfigure`.

`ishandle (h)` [Built-in Function]
Return true if *h* is a graphics handle and false otherwise.

`ishghandle (h)` [Function File]
Return true if *h* is a graphics handle and false otherwise.

`isfigure (h)` [Function File]
Return true if *h* is a graphics handle that contains a figure object and false otherwise.

The function `gcf` returns an index to the current figure object, or creates one if none exists. Similarly, `gca` returns the current axes object, or creates one (and its parent figure object) if none exists.

`gcf ()` [Function File]
Return the current figure handle. If a figure does not exist, create one and return its handle. The handle may then be used to examine or set properties of the figure. For example,

```
fplot (@sin, [-10, 10]);
fig = gcf ();
set (fig, "visible", "off");
```

plots a sine wave, finds the handle of the current figure, and then makes that figure invisible. Setting the visible property of the figure to `"on"` will cause it to be displayed again.

See also: [get], page 247, [set], page 247.