PLOT

By: Jan Pedersen (pedersen.PA @ Xerox.com)

Uses: TWODGRAPHICS and PLOTOBJECTS

PLOT is a module designed to assist in the production of analytic graphics. PLOT provides automatic scaling, labeling, incremental modification, generalized selection, and a collection of standard graphics primitives which may be combined to produce interactive plots of great diversity.

PLOT is to some degree object-oriented. The primitive components of a plot are plot objects (e.g. points, lines, etc.). A plot manager maintains a display list of plot objects which are individually responsible for displaying themselves, highlighting themselves, etc. The user constructs a plot incrementally, adding plot objects, while the plot manager handles details such as the appropriate scale for the plot. Each plot object is active, in the sense that it is selectable and may have a menu associated with it. In addition, the plot manager may be directed to modify the appearance of the entire plot through a command menu.

The module is open, in the sense that most default behaviors may be overridden by the user, although it is hoped that the defaults will be sufficient for most applications. A functional interface is provided for programmatic access to all of PLOT's facilities.

The plot manager is abstracted as a datatype of type PLOT, along with a collection of functions which operate on PLOT's. Functions are provided to create PLOT's, manipulate their display lists, and modify default menus. Plot objects are abstracted as instances of datatype PLOTOBJECT. A set of default plot objects are provided, along with a mechanism of defining new plot objects.

Plots exist independently of their representation on the screen. Indeed, it is intended that plots may be displayed on ANY imagestream. However, the most common usage is to display a plot in a window, and a PLOT does have an associated WINDOW which may be opened, closed, etc.

Plots may be hard copied, made into image objects, and dumped to file.

The lispuser's module PLOTEXAMPLES contains a few examples of how PLOT may be used to create high level plotting facilities.

BASIC OPERATION

A plot is abstracted as an instance of datatype PLOT which includes a display list, a property list, and an associated window, among other things. PLOT's may be create via the function CREATEPLOT.

(CREATEPLOT *openflg region title border*) [Function]

Returns a PLOT. If openflg is T then the PLOT's associated window is opened with an empty plot. The other arguments are treated as in CREATEW.

An empty plot is initialized to have a world coordinate system extending from 0.0 to 1.0 on either axis, with no labels or tic marks displayed. As objects are added to the plot, the world coordinate system is grown to accommodate the new objects.

A PLOT has an associated window, which is closed by default. The window is used as the primary display device and may be manipulated with the following functions.

(OPENPLOTWINDOW *plot*) [Function]

Opens the plot's associated window.

Returns the associated window.

(CLOSEPLOTWINDOW *plot*) [Function]

Closes the plot's associated window.

(REDRAWPLOTWINDOW *plot*) [Function]

Redraws, by running down the current display list, the contents of the associated window. Opens the window if it is closed.

(GETPLOTWINDOW *plot*) [Function]

Returns the window associated with plot.

(WHICHPLOT *x y*) [Function]

Returns the PLOT associated with the window (or icon) at position (x . y), or at the current cursor position if x and y are defaulted. x may be a WINDOW, in which case the associated PLOT is returned.

A plot object is abstracted as an instance of datatype PLOTOBJECT. A point plot object is an instance of PLOTOBJECT whose data component describes a point. That is, a point plot object is a subtype of PLOTOBJECT; all plot objects satisfy (type? PLOTOBJECT FOO), but only a point plot object satisfies in addition (PLOTOBJECTSUBTYPE? POINT FOO). A collect of standard plot objects has been implemented, including point, curve, polygon, line, and filled rectangle plot objects. The module is designed so that new objects may defined at any time, but that mechanism is described in a separate document.

PLOTOBJECT's may be added to or deleted from a PLOT. The following functions provide an add facility for the standard objects.

(PLOTPOINT *plot position label symbol menu nodrawflg*) [Function]

Only the plot and position arguments are required. Position is a POSITION in world coordinates. Label is an expression which will be PRIN1 'ed whenever a label is required (typically an atom or a string). Symbol is a BITMAP which will be plotted centered at position. The litatoms CROSS, CIRCLE, STAR are bound to convenient BITMAPS. Symbol defaults to STAR. Menu is either a MENU, a litatom, in which case a MENU of that name must be cached on plot (more about this later), or an item list which may be coerced into a MENU.

If nodrawflg is non-NIL then a point object will be added to the display list of plot, but the associated window will not be updated. If Nodrawflg is NIL, and the plot's associated window is not open, it will be opened.

Returns a POINT PLOTOBJECT.

(PLOTPOINTS *plot positions labels symbol menu nodrawflg*) [Function]

As above except that positions is a list of POSITIONS and labels may also be a list. Reasonable things happen if positions and labels are of unequal length.

Returns a list of POINT PLOTOBJECT's.

(PLOTCURVE *plot positions label style menu nodrawflg*) [Function]

The list of POSITION's defines a piecewise linear curve. Style may be an integer which specifies the line width (in pixels) or a list of (linewidth dashing color), any of which may be NIL; defaults to one. For convenience the atoms DOT, DASH and DOTDASH have been bound to a few dashing patterns.

Returns a CURVE PLOTOBJECT.

(PLOTPOLYGON *plot positions label style menu nodrawflg*) [Function]

As in PLOTCURVE, although a polygon is a closed figure

Returns a POLYGON PLOTOBJECT.

(PLOTTEXT *plot position text label font menu nodrawflg*) [Function]

Text should be a STRING to be printed at position.

Returns a TEXT PLOTOBJECT.

(PLOTFILLEDRECTANGLE *plot left bottom width height label texture borderwidth menu nodrawflg*) [Function]

Texture must be TEXTURE. SHADE1, ..., SHADE8 are bound to some convenient textures. Defaults to SHADE3.

Returns a FILLEDRECTANGLE PLOTOBJECT.

The following two functions add analytic plot objects to the display list of a PLOT. Analytic objects differ from points, curves, etc. by having infinite extents; their appearance on a plot depends on the current world coordinate scale, but adding an analytic object to a plot will not effect the current scale.

(PLOTLINE *plot slope constant label style menu nodrawflg*) [Function]

Slope and constant define an analytic line, y = slope * x + constant. If slope is NIL, it is taken to be infinite; i.e. the line is vertical.

Returns a LINE PLOTOBJECT.

(PLOTGRAPH *plot graphfn nsamples label style menu nodrawflg*) [Function]

Graphfn should be a function of one variable which defines a graph (or the graph of a function) to be drawn on plot. Nsamples is the number of equispaced points along the x-axis of plot at which graphfn is to be sampled when drawn; defaults to 100.

Returns a GRAPH PLOTOBJECT.

Complex objects may be built up from the preceding primitives by defining a compound plot object, which is simply a collection of other plot objects, including other compound objects.

(PLOTCOMPOUND *plot component1 ... componentn typename label menu nodrawflg*) [NoSpread Function]

A compound plot object is specified by listing its components. In addition, a compound plot object may have its own menu and label. The typename field is supplied to allow different compound objects to be differentiated. Drawing a compound object amounts to drawing its components recursively. In general, operations on compound objects are applied recursively.

Components 1 through n are plot objects. Typename is required and serves to tag this compound object, and is accessable via the function COMPOUNDSUBTYPE. Label and menu are as in other plot objects.

Returns a COMPOUND PLOTOBJECT.

All plot objects may be created independently of the previous functions. This is useful if it is desired to create a plot object without entering it on a PLOT's display list. The following functions create and return the standard plot objects.

(CREATEPOINT *position label symbol menu*) [Function]

Returns a POINT PLOTOBJECT.

(CREATECURVE *positions label style menu*) [Function]

Returns a CURVE PLOTOBJECT.

(CREATEPOLYGON *positions label style menu*) [Function]

Returns a POLYGON PLOTOBJECT.

(CREATETEXT *position text label font menu*) [Function]

Returns a TEXT PLOTOBJECT.

(CREATEFILLEDRECTANGLE *left bottom width height label texture style menu*) [Function]

Returns a FILLEDRECTANGLE PLOTOBJECT.

(CREATELINE *slope constant label style menu*) [Function]

Returns a LINE PLOTOBJECT.

(CREATGRAPH *graphfn nsamples label style menu*) [Function]

Returns a GRAPH PLOTOBJECT.

(CREATECOMPOUND *compoundtype components label menu*) [Function]

Components must be a list of PLOTOBJECT's.

Returns a COMPOUND PLOTOBJECT.

Each PLOT has a display list which is nothing more than a list of plot objects. The display list may be manipulated directly via the following functions.

(ADDPLOTOBJECT *plotobject plot nodrawflg*) [Function]

Interns plotobject on the display list of plot, and updates the associated window. The update is suppressed if nodrawflg is non NIL.

One might think of PLOTPOINT as being equivalent to:

```
(ADDPLOTOBJECT  (CREATEPOINT position ....)  plot  nodrawflg)
```

Interns plotobject on the display list of plot, and updates the associated window. The update is suppressed if nodrawflg is non NIL.

Returns plotobject.

(DELETEPLOTOBJECT *plotobject plot nodrawflg nosaveflg*) [Function]

Deletes plotobject from the display list of plot, and updates the associated window accordingly. The update is suppressed if nodrawflg is T. If nosaveflg is T, then the deleted objected will not be saved for possible later undeletion.

Returns plotobject if it was deleted from the display list, else NIL.

A PLOT has collection of properties, some of which are maintained by the plot manager, and others which may be used to cache arbitrary user data. All plot properties are accessed via the function PLOTPROP.

(PLOTPROP *plot prop newvalue*) [NoSpread Function]

If newvalue is absent then the current value of prop is returned. If newvalue is supplied (even if it is NIL) then the value of prop is set and the old value returned. The distinguished prop's PLOTOBJECTS, PLOTSCALE, SELECTEDOBJECT, PLOTWINDOW, PLOTWINDOWVIEWPORT, PLOTPROMPTWINDOW, and PLOTSAVELIST refer system maintained properties plot, and should be treated as read only. Compiles open in some cases.

For example, The display list of plot may be accessed by the expression.

```
(PLOTPROP plot 'PLOTOBJECTS)
```

For convenience in manipulating the property list of a PLOT, the following functions are provided.

(PLOTADDPROP *plot prop itemtoadd firstflg*) [Function]

If the value of prop is a list then itemtoadd is added to the end of the list. If the value of prop is NIL, it is set to (LIST itemtoadd). Firstflg indicates that the new item is to be the first in the list rather than the last. Works only for user defined properties.

Returns the new value.

(PLOTDELPROP *plot prop itemtodelete*) [Function]

If itemtodelete is a member (MEMB) of the prop value, it is deleted. Works only for user defined properties.

Returns NIL if nothing was deleted, else the new value of prop.

(PLOTREMPROP *plot prop*) [Function]

Destructively removes prop from property list of plot. Works only for user defined properties.

Each plot object also has a property list. As with PLOT's, some of the properties are maintained by the system, but the rest may be used to store arbitrary data objects. The property list of a plot object is accessed through the function PLOTOBJECTPROP.

(PLOTOBJECTPROP *object prop newvalue*) [NoSpread Function]

As in PLOTPROP. The distinguished props are OBJECTMENU, OBJECTLABEL, and OBJECTDATA. The property, OBJECTMENU, may be set as well as read; if the newvalue is a list of items, it will be coerced into a menu.

(PLOTOBJECTADDPROP *object prop itemtoadd firstflg*) [Function]

As in PLOTADDPROP. Firstflg indicates that the new item is to be the first in the list rather than the last.

(PLOTOBJECTDELPROP *object prop itemtodelete*) [Function]

As in PLOTDELPROP.

DEFAULT MOUSE BUTTON ACTIONS

The user may interact with a plot through its associated window. A plot provides two default menu's, the RIGHT menu, which pops up if the right mouse button is depressed within a plot's window, and typically contains items relevant to the plot as a whole, and the MIDDLE menu, which pops up if the middle mouse button is depressed, and typically contains items relevant to the currently selected plot object. The left mouse button is used exclusively for selection. The right menu may optionally be fixed to the right hand side of the plot window for easy reference. In summary:

Left Button

While depressed will select the closest plot object.

Middle Button

Pops up a menu of default actions on the selected object

Right Button

Pops up a menu of default actions on the plot as a whole

DEFAULT MIDDLE MENU ITEMS

Label

Label the selected object. Either a default location for the label is selected (for point plot objects), or the user is queried for a location.

Unlabel

If the object is label, remove the label.

Relabel

Change the object's label

Delete

Remove the object from the plot. May be undeleted later.

DEFAULT RIGHT MENU ITEMS

Layout

Create a SKETCH of the contents of the PLOT. Requires SKETCH and SKETCHSTREAM to be loaded.

Redraw

Redraw the plot

Rescale

Compute a new scale for both the X and the Y axis based on the objects currently displayed. May also rescale the X or Y axis separately.

Extend

Extend the axes slightly on either side so plot objects occuring on the borders may become visible. May be applied separately to either axis.

Labels

Change the marginal labels. May either Choose a margin explicitly, or respond to query.

Tics

Enable or disable marginal tics.

Undelete

Restore the last plot object deleted. Subsidiary items allow selected objects to be restored.

Deselect

Deselects the current selected object.

The default menus may be altered or superceded altogether. Each plot object may either use the default middle menu, another cached menu, or provide its own individual menu.

Menus are described by item lists of the form (label function helpstring [(subitems ....)]). Function may be a litatom in which case the function is called with one argument, plot, for right menu items,or two arguments, plotobject and plot, for all other menus. If function is a list the CAR of the list is a APPLIED to `(CONS PLOTOBJECT (CONS PLOT (CDR list)))`, etc.

The following functions facilitate modifying existing menus, and creating new menus.

(PLOTMENU *plot menuname newmenu*) [NoSpread Function]

Plot and menuname are required. If newmenu is not present, then the current value of menu menuname is returned. Menuname may be RIGHT or MIDDLE, in which case the default menus are referred to, or any LITATOM, in which case the cached menu by that name is referred to. Menus other than RIGHT or MIDDLE will typically be specialized menus for particular plot objects. If present, newmenu must be a MENU.

(PLOTMENUITEMS *plot menuname menuitems*) [NoSpread Function]

Plot and menuname are required. If menuitems is not present, then the current item list for the MENU menuname is returned. If menuitems is present, then menu menuname is replaced with a new menu with items list menuitems. All the properties (if any) of the old menu are copied over. Menuname may be one of RIGHT or MIDDLE, in which case the operations refer to the default right or middle mouse button menus or any other LITATOM, in which case the operations refer to a menu cached on plot by that name. Menus other than RIGHT or MIDDLE will typically be specialized menus for particular plot objects.

(PLOTADDMENUITEMS *plot menuname itemstoadd*) [Function]

Itemstoadd must be a list of menu items. Adds each item in itemstoadd to the end of the item list for menu menuname and replaces menu menuname with a new MENU having the appropriate item list.

Returns the the new item list for menuname.

(PLOTDELMENUITEM *plot menuname itemstodelete*) [Function]

Itemstodelete must be a list of items. For each element of itemstodelete, if it is a LITATOM, then deletes the item whose CAR is EQ to it. If it is a LISTP, then deletes the item EQUAL to it. Replaces menu menuname with a new MENU having the appropriate item list.

Returns NIL if no items were deleted, else the new item list.

(PLOT.FIXRIGHTMENU *plot fixedflg*) [NoSpread Function]

Fixedflg is optional. If not present that the current state of the right menu of plot is returned; T implies the right menu is fixed. If Fixedflg is supplied the right menu state is correspondingly changed.

The middle button menu for a particular plot object is a property of that plot object, and may be accessed via the function PLOTOBJECTPROP. For example, the expression,

```
(PLOTOBJECTPROP object 'OBJECTMENU)
```

will return the current middle button menu for object. If the OBJECTMENU property is NIL, then the system default MIDDLE menu is used, if it is a LITATOM, than a specialized cached menu by that name is used, finally, if it is a MENU, then that menu is used.

Two default fonts are provided, a large font for labels and a small font for tic marks. Both may be reset and that aspect of a plot will change accordingly with the next redraw.

LARGEPLOTFONT [Variable]

Default value: (Gacha 12 BRR)

SMALLPLOTFONT [Variable]

Default value: (Gacha 8 MRR)

Detailed Operation

Most visible aspects of a PLOT may be changed programmatically. The following functions allow the user to specify labels, etc., as well as override the default algorithms for drawing tics, etc.

(PLOTLABEL *plot margin newlabel nodrawflg*) [NoSpread Function]

Plot and position are required. Margin must be one of TOP, BOTTOM, LEFT, OR RIGHT. If newlabel is absent, then the current margin label is returned (may be NIL). If newlabel is present then the margin label is set to newlabel. The display is automatically updated unless nodrawflg is non NIL.

(PLOTTICS *plot margin newvalue nodrawflg*) [NoSpread Function]

Plot and margin are required. Margin must be one of TOP, BOTTOM, LEFT, OR RIGHT. If newvalue is absent, returns the tic status of that margin. NIL implies no tics or labels, T implies both. If newvalue is present, then sets margin's tic status. The display is automatically updated unless nodrawflg is non NIL.

The appearance of the tic marks will also depend on the tic generation method employed. The default is simply to make down tics at "pretty" intervals from the max to the min of each axis in world coordinates. However, non-numeric tic marks, and other behaviors are user specifiable by the function PLOTTICMETHOD.

(PLOTTICMETHOD *plot margin newmethod nodrawflg*) [NoSpread Function]

Plot and margin are required. Margin must be one of TOP, BOTTOM, LEFT, OR RIGHT.If newmethod is absent, returns the current tic method for margin margin. Newmethod may be one of NIL, implying the default tic method, a list of CONS pairs (value . label), in which case label (if non-NIL) will be printed at value, or a list of numbers, which is equivalent to ((value . value) ...) or a function which will be called with args, margin plotscale plot, and should return a list as above. Plotscale is a datatype which descibes the current scale of the plot.

(DEFAULTTICMETHOD *margin plotscale plot*) [Function]

The result depends on the ticinfo field of plotscale, which should be an instance of the PLOTSCALE datatype. The ticinfo field will be an instance of datatype TICINFO. If its ticinc field is a number (the usual case) then it returns a list of numbers, starting at ticmin and ending at ticmax in increments of ticinc, otherwise returns ticinc (should be a list).

When a plot object is added to a plot, the scale of the plot is adjusted so that the object is visible. This is accomplished by comparing the extent (in world coordinates) of the object with the current scale of the plot. If the scale needs to be enlarged, a new interval is chosen for each axis which is guaranteed to include the object and also be some multiple of a "round" increment -- in other words, a pretty tic interval. The default behavior of this scaling algorithm may be altered in several ways.

The pretty tic interval is determined by the TICFN for each axis. The default uses the function SCALE to find a suitable interval. This may be altered by supplying a TICFN other than the default.

Given a pretty tic interval, the default is to simply use the end points of that interval as the endpoints of the scale for each axis. This may be altered by supplying a SCALEFN other than the default.

In other words the actually displayed interval (for each axis) in world coordinates (what I will call the plot interval) is separated from the pretty tic interval (for each axis). The pretty tic interval is computed first, then the plot interval is computed in the presence of that information. This separation is useful if the user wishes to plot objects in a coordinate system different from the one used to display tic marks.

The current state of each axis of a PLOT is cached in the plot property plotscale, whose value is an instance of datatype PLOTSCALE. A PLOTSCALE has three fields for each axis, one which contains an instance of AXISINTERVAL, describing the actual plot interval for that axis, another which contains an instance of TICINFO, which describes the pretty tic interval for that axis, and a third which is a simply a place to cache a user supplied TICFN and SCALEFN.

(PLOTTICFN *plot axis ticfn nodrawflg*) [NoSpread Function]

Ticfn is optional. If not present the current ticfn for the indicated axis is returned. If supplied, the state of that axis is correspondingly updated. A ticfn is called with args min, max, and plot and should return an instance of TICINFO. If the state of plot is changed, the appropriate axis is rescaled. A value of NIL implies the default ticfn.

(DEFAULTTICFN *min max* -- -- --) [Function]

The default ticfn for each axis. Uses the function SCALE to find a suitable pretty tic interval.

(PLOTSCALEFN *plot axis scalefn nodrawflg*) [NoSpread Function]

Scalefn is optional. If not present the current scalefn for that axis of plot is returned. If supplied, the state of that axis is updated. A scalefn is called with four arguments, the min and max extent (in world coordinates) on that axis of the plotobjects currently displayed, the TICINFO for that axis, and the plot;

the scalefn should return an AXISINTERVAL which will determine the scale for that axis of plot. A value of NIL implies the default scalefn.

(DEFAULTSCALEFN *min max ticinfo*) [Function]

The default scalefn for each axis.

Returns an AXISINTERVAL with endpoints identical to the endpoints of ticinfo.

(ADJUSTSCALE? *extent plot*) [Function]

Determines whether extent will fit into the current viewing area of plot. If so, returns NIL. If not, returns T and updates the plotscale of plot.

(EXTENTOFPLOT *plot*) [Function]

Computes the current extent of plot by mapping EXTENTOBJECT down the display list. Returns an EXTENT.

To be precise, the scaling algorithm operates as follows; a min and max extent of the data is computed (via EXTENTOFPLOT or entered manually in the case manual rescaling), then CHOOSETICS is called, which returns an instance of TICINFO. CHOOSETICS either uses a default TICFN, or one supplied by the user, The default TICFN, calls SCALE repeatedly to find an "optimal" tic interval in world coordinates. Once the TICINFO instance has been computed, CHOOSESCALE is called with the original min, max and the TICINFO, and returns an instance of AXISINTERVAL, which will determine the actually displayed plot interval. Again, CHOOSESCALE either uses a default SCALEFN, or one supplied by the user. The default SCALEFN simply uses the end points of the passed in pretty tic interval as the end points of the AXISINTERVAL which it returns. Finally, the PLOT is redrawn with the new scale -- notice that the plot interval may either be larger or smaller than the pretty tic interval; the margin drawing routines are robust enough to deal with all cases.

For example, suppose the world coordinates are in centigrade and it is desired to produce a pretty tic interval in units of Fahrenheit (this is an easy case since the transformation between scales is linear -- more about that later). The user would then supply a TICFN which would transform the incoming min and max to Fahrenheit , apply the default TICFN on the transformed min and max, obtain a TICINFO in Fahrenheit, transform the fields of that record back to Centigrade, and return that record. Note, it is always assumed that the fields of a returned TICINFO are in the units of the world coordinate system. The rest of the machinery would then go through as before.

A tricker example is one in which it is desired to produce unequispaced tic marks. Suppose the data were plotted on a log scale (that is, log was applied BEFORE plotting the data).The default algorithm would produce a pretty tic interval in the log scale. It might be desired instead to produce one pretty in the original scale. The user would then supply a TICFN which would exponentiate the incoming min and max, apply the default TICFN on the transformed min and max, obtain a TICINFO in the original scale, then return a TICINFO in the logscale. Note; since equispaced tic marks in the orginal scale are not equispaced in the log scale, the TICINC field of the returned TICINFO would be a list of the unequispaced tic marks values, rather than a number.

The plot scale of each axis may be manipulated directly through the following functions.

(PLOTAXISINTERVAL *plot axis newinterval nodrawflg*) [Function]

Plot and axis are required. Axis must be one of X, or Y. If newinterval is NIL , returns the current AXISINTERVAL for that axis. If newinterval is non-NIL it must be an AXISINTERVAL.

(PLOTTICINFO *plot axis newticinfo nodrawflg*) [Function]

Plot and axis are required. Axis must be one of X, or Y. If newticinfo is NIL , returns the current TICINFO for that axis. If newticinfo is non-NIL it must be a TICINFO.

On occasion it is useful to clean out an existing plot instead of creating a new one.

(PLOT.RESET *plot xscale yscale flushmargins flushprops nodrawflg*) [Function]

Returns plot to a pristine state. If xscale and yscale are provided, the scale of the plot is set accordingly.

Finer control over the behavior of plot objects is possible through the following functions.

(TRANSLATEPLOTOBJECT *plotobject dx dy plot nodrawflg*) [Function]

Moves plotobject dx, dy in world coordinates and updates the associated window accordingly. The update is suppressed if nodrawflg is non NIL.

(DRAWPLOTOBJECT *plotobject plot*) [Function]

Draw plotobject in the window asssociated with plot. As with all the display functions, the window should be opened beforehand. DRAWOBJECT does NOT check that the window is open.

APPLY's the plotobject's DRAWFN.

(ERASEPLOTOBJECT *plotobject plot*) [Function]

APPLY's the plotobject's ERASEFN

(HIGHLIGHTPLOTOBJECT *plotobject plot*) [Function]

Invoked when a plotobject is selected

(LOWLIGHTPLOTOBJECT *plotobject plot*) [Function]

Invoked when a plotobject is deselected

(EXTENTOFPLOTOBJECT *plotobject plot*) [Function]

Computes the extent of plotobject in world coordinates.

Returns an EXTENT, which has fields MAXX, MINX, etc.

(DISTANCETOPLOTOBJECT *plotobject streamposition plot*) [Function]

Returns the "distance" to plotobject from streamposition in stream coordinates. Value returned may be a FIXP or a FLOATP, but is always a distance in stream coordinates.

(CLOSESTPLOTOBJECT *plot streamposition*) [Function]

Returns the "closest" plotobject on plot's display list to streamposition.

(DESELECTPLOTOBJECT *plot*) [Function]

Deselects the current selected object of plot

Plot objects also have "afterfns". That is, functions which are optionally invoked after some standard operation. These are stored as plot object properties with distinguished names, and invoked with at least two args, the plotobject and the plot.

WHENADDEDFN [Property]

The WHENADDEDFN is called with three arguments, plotobject, plot, and nodrawflg

WHENDELETEDFN [Property]

The WHENDELETEDFN is called with four arguments, plotobject, plot, nodrawflg, and nosaveflg.

WHENDRAWNFN [Property]

The WHENDRAWNFN is called with three arguments, plotobject, viewport and plot.

WHENERASEDFN [Property]

WHENHIGHLIGHTEDFN [Property]

WHENLOWLIGHTEDFN [Property]

WHENTRANSLATEDFN [Property]

A PLOT has two associated windows, the mainwindow in which the graphics, labels, tics, etc. are displayed and an attached promptwindow. The mainwindow is cached as plot property and may be accessed via the function PLOTPROP. A function is provided for easy access to the prompt window.

(PLOTPROMPT *text plot*) [Function]

Text is output in the one character high prompt window of plot.

PLOT's may be drawn in ANY imagestream (but only interacted with in the PLOT's associated window). The following function is the fundamental draw primitive.

(DRAWPLOT *plot stream streamviewport streamregion*) [Function]

Stream is any imagestream. Streamviewport is a viewport on that stream that defines the the world to stream transformation. Streamregion is a region in stream coordinates that will contain the entire image (for a window it will be the CLIPPINGREGION). Streamviewport is usually the result of ADJUSTVIEWPORT.

For more information about viewport, consult the documentation for the **TWODGRAPHICS** module.

(ADJUSTVIEWPORT *viewport streamregion plot*) [Function]

Viewport is a VIEWPORT whose parentstream is the imagestream of interest. Streamregion is a region in stream coordinates that will contain the entire image.

Adjusts the Streamsubregion and Worldregion of viewport to reflect the current scale and margin setting of plot.

(MINSTREAMREGIONSIZE *stream plot*) [Function]

Returns a CONS pair (minwidth . minheight) of the plot in stream coordinates.

A plot has "afterfns" for two major operations, opening and closing the plotwindow. These are stored as plot properties with distinguished names. The values of these properties may be a single function or a list of functions which are called in sequence with the plot as an argument.

WHENOPENEDFN [Property]

WHENCLOSEDFN [Property]

PLOT's may be copied, made into image objects, dumped onto files, sent in the mail, etc.

(COPYPLOT *plot*) [Function]

Returns a copy of plot. The user defined properties require special handling. If there exists a plot prop COPYFN, which may be function or list of functions, the function (or functions) will be invoked with the arguments newplot plot and propname for each user defined property on plot. If the function returns a non-NIL value, it will be used as the value of propname on newplot. In the case of a list of functions, the first non-NIL value (traveling from the head to the tail of the list of functions) will be used as the new prop value. Otherwise the prop will be HCOPYALL'ed.

(COPYPLOTOBJECT *plotobject plot*) [Function]

Returns a copy of plotobject. The protocol for copying objectprops is similar to plot props. The plotobject may have a COPYFN prop which may be a function or list of functions. The function (or functions) will be invoked with the arguments newplotobject plotobject plot propname. The first non-NIL value will be used as the prop value else the property will be HCOPYALL'ed.

(PRINTPLOT *plot stream*) [Function]

Writes out an HREADable symbolic representation of plot on stream. Again, user defined properties require special handling. If there exists a plot prop PUTFN, which may be function or list of functions, the function (or functions) will be invoked with the arguments plot propname and stream for each user defined property on plot. If the function returns a non-NIL value, it is assumed an HREADable representation of the prop value has been written out on stream. In the case of a list of functions, the functions will invoked one at time, starting from the head of the list, until a non-NIL result is obtained. If there is no PUTFN, or the function (or none of the functions) returns a non-NIL value, the prop is HPRINT'ed.

Lists of the form ((FUNCTION function) arg) are recognized by the inverse of PRINTPLOT, READPLOT, to imply that function should be called with plot and arg as arguments at HREAD time, and the value returned to be the prop value.

(PRINTPLOTOBJECT *plotobject plot stream*) [Function]

Writes out an HREADable symbolic representation of plotobject on stream. As in PRINTPLOT user defined object properties require special handling. The protocol is the same as in PRINTPLOT.

The following data types have HPRINT macros and need no special handling: FONTDESCRIPTOR, MENU, PLOT, and PLOTOBJECT.

A file package command has been defined to simplyfy dumping PLOT's on files.

(PLOTS . *plots*) [FilePkgCom]

The syntax is identical to VARS.

A plot image object is fully supported.

(CREATEPLOTIMAGEOBJ *plot*) [Function]

Returns an image object which contains a copy of plot. These image objects can also be created by copy-selecting from a plot window into a host window (e.g. TEdit or Sketch) that supports image objects. Such a selection will ask whether the plot should be inserted as a bitmap or a plot, the latter case constructing a plot image object. Buttoning on the image object provides the option of reshaping the plot or creating a separate plot window in which the plot can be modified. Closing the plot window will ask whether the new plot should be reinserted in the host.