Cl-plplot User Manual

February, 2008

1 Introduction

Cl-plplot provides a CFFI based interface to the PLplot graphics library. The PLplot graphics library supports the drawing of many different types of 2D and 3D graphs using a large variety of output devices including X11, postscript and png. PLplot can also be used to make scaled drawing, diagrams, etc...

At present, cl-plplot consists of two packages, one is low-level interface to the PLplot library and the other is high-level plotting interface that attempts to make using PLplot with Lisp easier. It has been tested on OS-X and debian linux with SBCL. Since it interfaces to the PLplot library using CFFI, it should work with any combination of lisp implementation and operating system that is supported by CFFI.

2 Installation

Cl-plplot depends on a working installation of the PLplot plotting library. This is available as a debian package, but it is also not too hard to compile from source on most systems. You will need PLplot version 5.8.0 or later.

Once you have installed the PLplot library cl-plplot will hopefully not have any trouble finding the library. If you do encounter difficulties with cl-plplot finding the PLplot library, then the you will probably need to edit the function load-libraries in /cl-plplot/src/system. All that should be necessary is to provide the appropriate path to the library file 'libplplotd'.

3 The Low Level Interface to PLplot

3.1 Introduction

The low level interface `:cl-plplot-system` provides both 'direct' and 'wrapped' access to the functions in the PLplot library. What this means is that each of the functions in the PLplot library is available in two ways. The 'direct' form expects that all arguments are in a form appropriate for handing off to the C library function. In cases where the C function returns an array or a string it will be returned as C pointer. On the other hand, the 'wrapped' form will automatically convert Lisp variables to the appropriate C variables, fill in any array size arguments and then call the 'direct' form. In the case where the C function returns an array or a string the 'wrapped' form will automatically convert this into the appropriate Lisp type. Furthermore, the 'wrapped' form takes care of allocating and deallocating the necessary memory to make the C function call. The 'wrapped' form is recommended for general use, but the 'direct' form is also available in situations where speed may be important or you are already using C pointers and do not want to pay the overhead of converting them back and forth to Lisp variables.

3.2 Example (the PLplot arrows function)

```
(defcfun ("plarrows" c-plarrows)
         :void
         (u *plflt)
         (v *plflt)
         (x *plflt)
         (y *plflt)
         (n plint)
         (scale plflt)
         (dx plflt)
         (dy plflt))

(defun plarrows (u v x y scale dx dy)
  (let ((c-u (make-ptr u :double #'(lambda (x)
                                     (coerce x 'double-float))))
        (c-v (make-ptr v :double #'(lambda (x)
                                     (coerce x 'double-float))))
        (c-x (make-ptr x :double #'(lambda (x)
                                     (coerce x 'double-float))))
        (n (length y))
        (c-y (make-ptr y :double #'(lambda (x)
                                     (coerce x 'double-float)))))
    (unwind-protect
        (c-plarrows c-u
                    c-v
                    c-x
                    c-y
                    (funcall #'(lambda (x) (round x)) n)
```

```
                           (funcall #'(lambda (x)
                                        (coerce x 'double-float)) scale)
                           (funcall #'(lambda (x)
                                        (coerce x 'double-float)) dx)
                           (funcall #'(lambda (x)
                                        (coerce x 'double-float)) dy))
             (progn
              (foreign-free c-u)
              (foreign-free c-v)
              (foreign-free c-x)
              (foreign-free c-y)))))
```

Notes

- The name of the PLplot function as defined in the PLplot manual is used for the 'wrapped' form and the name of the 'direct' has `c-` appended onto the front. This convention is followed for most PLplot functions.
- The function `make-ptr` handles the creation of a C array from a Lisp vector.
- The argument `n` required by PLplots arrows function is automatically determined from the length of the vector y and does not need to be passed in.
- The call to the PLplot library function is wrapped with `unwind-protect` so that if the C function fails the memory occupied by `c-u`, `c-v`, `c-x` and `c-y` is still freed.

3.3 Exceptions

- There are a few exceptions to the above naming conventions for the 'direct' and 'wrapped' forms. Typically these occur for functions that have very complicated arguments, for example functions that required callbacks. When in doubt the best approach is probably to peruse `src/system/api.lisp` where all of the PLplot functions are defined.
- Not all of the PLplot library functions are available in cl-plpot. Some were deemed to be too esoteric, or better handled by an equivalent Lisp function. However, if you find a function that you feel should be supported please let me know.

3.4 Supported PLplot Functions

The current best reference is the manual that comes with PLplot. TODO: a list of the supported PLplot functions and their arguments.

4 The High Level Plotting Package

4.1 Introduction

The high level interface `:cl-plplot` tries to make PLplot a little more Lispy using a CLOS based approach. You create window objects, then associate axis objects, text objects and plot objects to the window objects. When everything is set you call the `render` method on the window object to display the plot using the display device of your choice.

This approach was adopted in part to make graphing easier in a multi-threaded environment. The PLplot library supports drawing multiple graphs simultaneously using the concept of plot streams. However using these in a multi-threaded environment would be a challenge as just about every call to a PLplot library function would have to prefaced with a function call to set the correct stream. As all the calls to the PLplot library are gauranteed to happen during `render` one only needs to threadlock around this method to avoid confusing PLplot.

4.2 The Object Layout

```
window object
 |--axis object (1 for the x axis, 1 for the y-axis)
 |   |--axis-label object (0 or more per axis object)
 |       |--text-item object (1 per axis-label object)
 |--axis-label object (0 or more)
 |--plot object (0 or more)
 |--text-label object (0 or more)
 |   |--text-item object (1 per text-label object)
 |--color-table object (1 per window)
 |--extended-color-table object (1 per window)
```

Note that this is **NOT** an inheritance diagram. Each of the above objects is a distinct entity.

4.3 Object Definitions

The Window Object

This is the main object without which it is impossible to actually create a plot. It always contains references to two axis objects, one for the x-axis and one for the y-axis. It can also contain references to additional axis-label objects that might be used to provide a title for the plot as well as any other axis labeling that is needed, references to plot objects that will be drawn in the window and references to text labels to draw in the window. When you call `render` on a window object it determines the coordinate system and size of the plotting rectangle in which the plots will appear, then calls the `render` methods of the axis-label, text-label and plot objects to create the final plot.

The 3D-Window Object

This is the main window object for drawing 3D plots. At present these include surface mesh plots (3D-mesh) and solid surface plots (surface-plot). This object inherits from the Window object and adds a z-axis as well as altitude and azimuthal viewing angles.

The Axis Object

This object is used to set minimum and maximum values to plot in the window, as well as specifying how to label the axis, major tick interval, minor tick interval, ticks, grid, ...

The Axis-label Object

This object is used to draw text relative to one of the sides of the plotting rectangle. It specifies where to draw the text relative to one of the edges of the rectangle as well as in what orientation.

The 3D-axis-label Object

This object inherits from the axis-label object. It uses a different convention for specifying which axis to label.

The Text-label Object

This is like the axis-label object except that it is typically drawn inside the plotting rectangle and its location is determined by the plot coordinate system.

The 3D-text-label Object

This object inherits from the text-label objects. It is used to draw '3D' text and as such requires a number of additional parameters to specify how to draw the text.

The Text-item Object

This object is used to actually draw the text. It specifies font, size, color, superscript, ...

The Plot Object

This object is used to convert data into a plot. In its most generic form it contains two functions, one that returns its preference for the x and y (and z if relevant) ranges of the window coordinate system and another that draws the object in the current window using `:cl-plplot-system` function calls. Specialized forms currently include x-y-plot, bar-graph and contour-plot.

The 3D Plot Object

This object inherits from the plot object. Specialized forms currently include the 3D-mesh and surface-plot.

The Color-table Object

This object handles the use of PLplot's color map 0. Typically it will consist of 16 colors, each of with contains red, green and blue values as well as symbol (like `:blue`) that you can use to refer to the color. This color table is used by PLplot to draw essentially all of the 'basic' items, i.e. lines, text, axises. You can have more then 16 colors but not all of the graphics devices will handle that many colors. Some in fact may only handle black and white.

The Extended-color-table Object

This object handles the use of PLplot's color map 1. Typically this color table will contain anywhere between 128 to 256 different colors, though, again, not all graphics devices will handle so many colors. It used for shading plots, such as contour plots, where it is often desirable to have continuous range of colors rather than a few blocky colors.

4.4 Examples

Note: These are also available in `src/examples/window-examples.lisp`.

X versus Y plots

```
(defun x-y-plot ()
  (let* ((x (my-make-vector 40 #'(lambda(x) (* 0.1 x))))
 (y (my-make-vector 40 #'(lambda(x) (* (* 0.1 x) (* 0.1 x)))))
 (p (new-x-y-plot x y))
 (w (basic-window)))
    (add-plot-to-window w p)
    (render w "xwin")))
```

The vectors x and y are created using the function `y = x^2`. A x-y-plot object is then created to plot the vector x versus the vector y. Finally a window object is created in which the x-y-plot object p can be plotted. The x-y-plot object p is added to the window w by the function `add-plot-to-window`. Then the window is drawn by calling the `render` method and specifying what PLplot graphics device to use for the graphical output ('xwin' is the X11 graphics device).

Bar Graphs

```
(defun bar-graph ()
  (let* ((y (my-make-vector 10 #'(lambda(x) (* (* 0.2 x) (* 0.2 x)))))
 (b (new-bar-graph nil y :fill-colors (vector :grey)))
 (w (basic-window)))
    (add-plot-to-window w b)
    (render w "xwin")))
```

A vector y is created using the function `y = (0.2 * x)^2`. This vector is used to make a bar-graph object in which each of the bars will be filled with the color grey. As in the X versus Y plot example, a window w is created, the bar-graph object b is added to this window and then the window is rendered using the X11 graphics device.

Contour Plots

```
(defun contour-plot ()
  (let ((c (new-contour-plot
             (my-make-matrix 50 50 #'(lambda (x y)
                                        (my-contour-plot-fn x y)))
             :x-min 0.0 :x-max 1.0 :y-min 0.0 :y-max 1.0
```

```
                 :fill-type :smooth))
     (w (basic-window)))
         (add-plot-to-window w c)
         (render w "xwin")))
```

A contour plot object is created from a 2D matrix of data. Additionally the coordinate system of the matrix is specified with `:x-min`, `:x-max`, `:y-min` and `:y-max` and the contour plot fill type is specified with `:fill-type`. The function `my-make-matrix` is defined in `src/examples/window-examples.lisp`.

3D mesh plots

```
     (defun 3d-plot-1 ()
       (let ((c (new-3d-mesh nil nil
                 (my-make-matrix 50 50 #'(lambda (x y)
                                          (my-contour-plot-fn x y)))
                 :line-color :blue))
     (w (basic-3d-window :altitude 30 :azimuth 60)))
         (add-plot-to-window w c)
         (render w "xwin")))
```

A 3D-mesh object is created from a 2D matrix of data. A 3D-window object is created in which to draw the 3D-plot object, with the viewing angle specified by `:altitude` and `:azimuth`. The plot is added to the window and is then rendered using the X11 graphics device.

3D surface plots

```
     (defun surface-plot-1 ()
       (let ((c (new-surface-plot nil nil
                 (my-make-matrix 50 50 #'(lambda (x y)
                                          (my-contour-plot-fn x y)))
         :line-color :blue))
     (w (basic-3d-window :altitude 30 :azimuth 60)))
         (add-plot-to-window w c)
         (render w "xwin")))
```

This example is essentially the same as the mesh plot example, except that we now create a surface-plot object.

4.5 Cl-plplot Functions in Alphabetical order

add-axis-label-to-axis

Argument List

(a-axis a-axis-label)

Documentation

adds a-axis-label to a-axis.

add-color-to-color-table

Argument List

(a-color-table new-color)

Documentation

adds a new color #(r g b :smybol) to the end of a color table.

add-plot-to-window

Argument List

(a-window a-plot)

Documentation

add-plot-to-window, adds a-plot to a-window.

add-text-label-to-window

Argument List

(a-window a-text-label)

Documentation

add-text-label-to-window, adds a-text-label to a-window.

backspace

Argument List

none.

Documentation

This (and greek-char, hershey-char, italic-font, normal-font, number-symbol, overline, roman-font, script-font, subscript, superscript, underline and unicode-char) exist to insert PLplot escape sequences into a string. Cl-plplot uses the default PLplot escape character "#".

basic-window

Argument List

(&key (x-label x-axis) (y-label y-axis) (title cl-plplot) x-axis-min x-axis-max y-axis-min y-axis-max (background-color *background-color*) (foreground-color *foreground-color*))

Documentation

creates a basic window object with ready-to-go axises.

basic-3d-window

Argument List

(&key (x-label x-axis) (y-label y-axis) (z-label z-axis) (title cl-plplot) x-axis-min x-axis-max y-axis-min y-axis-max z-axis-min z-axis-max (altitude 60) (azimuth 30) (background-color *background-color*) (foreground-color *foreground-color*))

Documentation

creates a basic 3D window object with ready-to-go axises.

bring-to-front

Argument List

(a-window a-plot)

Documentation

organizes the plots so that a-plot is drawn on top.

default-color-table

Argument List

none.

Documentation

returns the default color table.

edit-3D-mesh

Argument List

(a-3d-mesh &key line-width line-style line-color grid-type contour-options curtain)

Documentation

Edits the visual properties of a 3D-mesh plot object.

- Set the line width with :line-width (integer, 0 means no line).
- Set the line style with :line-style (integer between 1 and 8).
- Set the line color with :line-color symbol.

- Set the grid type with :grid-type to one of (:gridx, :gridy or :gridxy).
- Set the contour options with :contour-options to one of (:magnitude-contour, :base-contour or :both).
- Set the whether or not display a curtain with :curtain

edit-3D-window

Argument List

(a-3d-window &key x-axis y-axis z-axis title foreground-color background-color window-line-width window-font-size viewport-x-min viewport-x-max viewport-y-min viewport-y-max plots text-labels color-table altitude azimuth)

Documentation

edit-3D-window, edits the visual properties of a 3D-window.

- Set x-axis to a new object of type axis with :x-axis.
- Set y-axis to a new object of type axis with :y-axis.
- Set z-axis to a new object of type axis with :z-axis.
- Set title to a new object of type axis-label with :title.
- Set the foreground color with :foreground-color.
- Set the background color with :background-color.
- Set the pen width for drawing the border with :window-line-width.
- Set the font size for the tick labels with :window-font-size.
- Set the location of the left border with :viewport-x-min.
- Set the location of the right border with :viewport-x-max.
- Set the location of the bottom border with :viewport-y-min.
- Set the location of the top border with :viewport-y-max.
- Set :plots to a list of plot objects to change the plots associated with a window.
- Set :text-labels to a list of text-label objects to change the text-labels associated with a window.
- Set :color-table to a new color table object to change the colors of a plot.
- Set the observer altitude in degrees with :altitude.
- Set the observer azimuth in degrees with :azimuth.

edit-axis

Argument List

(a-axis &key axis-min axis-max major-tick-interval minor-tick-number properties)

Documentation

edit-axis, edits an axis.

- set the minimum value with :axis-min.
- set the maximum value with :axis-max.
- set the spacing between major ticks with :major-tick-interval.
- set the spacing between minor ticks with :minor-tick-interval.
- set the properties with :properties. this should be a list containing zero of more of the following symbols:

  :draw - draw axis on both sides of the window.

  :draw-bottom/left - draw axis on the bottom/left side of the window.

  :draw-top/right - draw axis on the top/right side of the window.

  :fixed-point - use fixed point labels.

  :major-tick-grid - draw a grid on the graph at the major ticks.

  :minor-tick-grid - draw a grid on the graph at the minor ticks.

  :invert-ticks - draw ticks inward rather than outwards.

  :log-axis - draw the axis on a log scale.

  :major-tick-labels-above/right - draw the tick labels above/right of the ticks.

  :major-tick-labels-below/left - draw the tick labels below/left of the ticks.

  :minor-ticks - draw minor ticks.

  :major-ticks - draw major ticks.

edit-axis-label

Argument List

(a-axis-label &key axis-text-item side displacement location orientation)

Documentation

edit-axis-label, edits a axis-label object.

- set axis-text-item to a new object of class text-item with :axis-text-item.
- set which axis to draw the label on with :side.
- set the displacement from the edge of the the graph with :displacement.
- set the location with along the side of the graph with :location.
- set the orientation with (:parallel or :perpendicular) with :orientation.

edit-3D-axis-label

Argument List

(a-axis-label &key axis-text-item side displacement location orientation primary/secondary)

Documentation

edit-3D-axis-label, edits a 3D-axis-label object.

- set axis-text-item to a new object of class text-item with :axis-text-item.
- set which axis to draw the label on with :side (:x, :y or :z).
- set the displacement from the edge of the the graph with :displacement.
- set the location with along the side of the graph with :location.
- set the orientation with (:parallel or :perpendicular) with :orientation.
- Set which axis to label (:primary or :secondary) with :primary/secondary

edit-bar-graph

Argument List

(a-bar-graph &key side-by-side line-colors fill-colors line-width filled)

Documentation

edit-bar-graph, edits the visual properties of a bar-graph.

- set whether the bars are plotted side-by-side or on top of each other with :side-by-side.
- set the line colors of the bars with :line-colors.
- set the fill colors of the bars with :fill-colors.
- set the line width of the bars with :line-width.
- set whether or not the bars are filled with :filled.

edit-contour-plot

Argument List

(a-contour-plot &key line-color line-width fill-type fill-colors)

Documentation

edit-contour-plot, edits the visual properties of a contour plot.

- set the line color with :line-color (this should be a color symbol in the current color table).
- set the line width with :line-width (integer, 0 means no line).
- set the fill-type with :fill-type (:none :block :smooth).
- set the fill-colors with :fill-colors (should be a vector of color symbols)

edit-surface-plot

Argument List

(a-surface-plot &key line-width line-style line-color light-source surface-options)

Documentation

edit-surface-plot, edits the visual properties of a solid surface plot.

- Set the line width with :line-width (integer, 0 means no line).
- Set the line style with :line-style (integer between 1 and 8).
- Set the line color with :line-color symbol.
- Move the light-source to a new position with :light-source #(x y z).
- Change the surface-options with :surface-options to a list including zero or more of:

    :faceted
    :base-contours
    :surface-contours
    :curtain
    :magnitude-coloring

edit-text-item

Argument List

(a-text-item &key the-text text-color text-justification font-size)

Documentation

edit-text-item, edits a text-item object.

- set the text with :text.
- set the color of the text with :text-color symbol.
- set the justification with :text-justification (0.0 = left justified, 1.0 = right justified).
- set the font-size with :font-size (relative to the default size).

edit-text-label

Argument List

(a-text-label &key label-text-item text-x text-y text-dx text-dy)

Documentation

edit-text-label, edits a text-label object.

- set text-item to a new object of class text-item with :label-text-item.
- set the x location of the text with :text-x.
- set the y location of the text with :text-y.
- set dx for drawing text at an angle with :text-dx.
- set dy for drawing text at an angle with :text-dy.

edit-3D-text-label

Argument List

(a-text-label &key label-text-item text-x text-y text-z text-dx text-dy text-dz text-sx text-sy text-sz)

Documentation

edit-3D-text-label, edits a 3D-text-label object.

- set text-item to a new object of class text-item with :label-text-item.
- set the x location of the text with :text-x.
- set the y location of the text with :text-y.
- set the z location of the text with :text-z.
- set dx for drawing text at an angle with :text-dx.
- set dy for drawing text at an angle with :text-dy.
- set dz for drawing text at an angle with :text-dz.
- set sx for shearing text with :text-sx.
- set sy for shearing text with :text-sy.
- set sz for shearing text with :text-sz.

edit-window

Argument List

(a-window &key x-axis y-axis title foreground-color background-color window-line-width window-font-size viewport-x-min viewport-x-max viewport-y-min viewport-y-max plots text-labels color-table)

Documentation

edit-window, edits a window object.

- set x-axis to a new object of type axis with :x-axis.
- set y-axis to a new object of type axis with :y-axis.
- set title to a new object of type axis-label with :title.
- set the foreground color with :foreground-color.
- set the background color with :background-color.
- set the pen width for drawing the border with :window-line-width.
- set the font size for the tick labels with :window-font-size.
- set the location of the left border with :viewport-x-min.
- set the location of the right border with :viewport-x-max.
- set the location of the bottom border with :viewport-y-min.
- set the location of the top border with :viewport-y-max.
- set :plots to a list of plot objects to change the plots associated with a window.

- set :text-labels to a list of text-label objects to change the text-labels associated with a window.
- set :color-table to a new color table object to change the colors of a plot.

edit-window-axis

Argument List

(a-window which-axis &key axis-min axis-max major-tick-interval minor-tick-number properties)

Documentation

allows the user to edit the axis of a window. which-axis should be one of the symbols :x or :y. see edit-axis for a more detailed explanation of the meaning of the different key words.

edit-x-y-plot

Argument List

(a-x-y-plot &key line-width line-style symbol-size symbol-type color)

Documentation

edit-x-y-plot, edits the visual properties of a x-y-plot.

- set the line width with :line-width (integer, 0 means no line).
- set the line style with :line-style (integer between 1 and 8).
- set the symbol size with :symbol-size (1.0 is the defaul size, 0.0 means no symbols).
- set the symbol type with :symbol-type (integer or nil to use the default types).
- set the color with :color symbol.

get-cursor

Argument List

(a-window device &key (size-x 600) (size-y 500))

Documentation

get the location (in window coordinates) of the next mouse click. in order to do this the window must first be rendered so that the user has something to click on.

greek-char

Argument List

none.

hershey-char

Argument List

none.

italic-font

Argument List

none.

new-3D-mesh

Argument List

(data-x data-y data-z &key contour-levels (copy t) (line-width 1) (line-style 1) (line-color *foreground-color*) (grid-type :grid-xy) contour-options curtain)

Documentation

new-3D-mesh, creates a new 3D mesh (surface) plot object.

- data-x specifies the x values of the points in data-z. If data-x is nil then data-z will be plotted against its row index in x.
- data-y specifies the y avlues of the points in data-z. If data-y is nil then data-z will be plotted against its column index in y.
- data-z is a 2D array of z values for the plot.
- contour-levels specifies the levels at which to draw contours, if desired. If this is not specified, default values are chosen.
- If copy is true then copies of data-x, data-y and data-z will be made, otherwise reference will be kept to the original vectors.
- line-width should be an integer line width, or zero if no line is desired
- line-style specifies what style line to draw (if a line is drawn), this should be a number between 1 and 8.
- line-color is the color to use for the lines in the plot.
- grid-type should be one of :gridx, :gridy or :gridxy. This specifies whether to draw lines only in x, only in y, or in both dimensions between the data points.
- contour-options should be one of nil, :magnitude-contour, :base-contour or :both. nil - no contours.

  :magnitude-contour - draw contour lines on the plot.

:base-contour - draw contour lines on the x-y plane below the plot.

:both - draw both magnitude and base contours.

- curtain should be t or nil. This specifies whether to draw a 'curtain' around the edges of the plot."

new-3D-window

Argument List

(&key x-axis y-axis z-axis title (window-line-width 1.0) (window-font-size *font-size*) (foreground-color *foreground-color*) (background-color *background-color*) (viewport-x-min 0.1) (viewport-x-max 0.9) (viewport-y-min 0.1) (viewport-y-max 0.9) plots text-labels color-table (altitude 60) (azimuth 30))

Documentation

new-3D-window, creates and returns a new 3D-window object.

- x-axis is a object of type axis.
- y-axis is a object of type axis.
- z-axis is a object of type axis.
- title is a object of type axis-label.
- foreground-color is a color symbol in the current color table.
- background-color is a color symbol in the curretn color table.
- window-line-width is a floating point number specifying the pen width to use when drawing the border & the tick marks.
- window-font-size is the font size to use for the tick mark labels.
- viewport-x-min (0.0 - 1.0) is the location of the left side of the border in the device window.
- viewport-x-max (0.0 - 1.0) is the location of the right side of the border.
- viewport-y-min (0.0 - 1.0) is the location of the bottom side of the border.
- viewport-y-max (0.0 - 1.0) is the location of the top side of the border.
- plots is a list of plot objects.
- text-labels is a list of text-label objects.
- color-table specifies what color table to use.
- altitude specifies the angle by which to rotate the plot around the x axis.
- azimuth specified the angle by which to rotate the plot around the z axis.

new-axis

Argument List

(&key axis-min axis-max (major-tick-interval 0) (minor-tick-number 0) (properties *axis-properties*) axis-labels)

Documentation

new-axis, creates and returns an axis object.

- axis-min is the minimum value for the axis.
- axis-max is the maximum value for the axis.
- major-tick-interval is the spacing between major ticks (0 means use plplot default).
- minor-tick-number is the number of minor ticks to put between the major ticks.
- properties is a list of symbols as explained in edit-axis.
- axis-labels is a list of axis-label objects.

new-axis-label

Argument List

(axis-text-item side displacement &key (location 0.5) (orientation parallel))

Documentation

new-axis-label, creates and returns a new axis label.

- axis-text-item is a object of type text-item.
- side is one of :top, :bottom, :left or :right.
- displacement specifies the distance from the edge of the graph in units of the default font-size.
- location is the position of the label along the side of the graph (0.0 - 1.0).
- orientation is one :parallel or :perpendicular.

new-3D-axis-label

Argument List

(axis-text-item side displacement &key (location 0.5) (orientation parallel) (primary/secondary :primary))

Documentation

new-3D-axis-label, creates and returns a new 3D axis label.

- axis-text-item is a object of type text-item.
- side is one of :x, :y or :z.
- displacement specifies the distance from the edge of the graph in units of the default font-size.
- location is the position of the label along the side of the graph (0.0 - 1.0).
- orientation is one :parallel or :perpendicular.
- primary/secondary is one of :primary or :secondary. This specifies which of two possible choices for each axis should be labeled.

new-bar-graph

Argument List

(x data &key bar-widths side-by-side line-colors fill-colors (line-width 1.0) (filled t) (copy t))

Documentation

creates a new bar-graph plot object.

- x is an array of size (n) specifying the centers of the bars with x[i] < x[i+1]. if x is nil then data will be plotted against its index.
- data should be an array of size (n x m), where m > 0.
- bar-widths should be an array of size (n). it will specify the full width of each bar. defaults are chosen if this is not specified.
- side-by-side is t or nil. it specifies whether to draw the bars on top of each other or next to each other.
- line-colors should be an array of symbols of size (m) specifying colors in the current color table.
- fill-colors should be an array of symbols of size (m) specifying what color to use when filling in the bars.
- line-width is a number specifying how wide of a line to draw around the bar.
- filled specifies whether or not the bars are filled.
- if copy is true, then copies are made of x, data and widths, otherwise references are kept to the original vectors.

new-color-table

Argument List

(&optional rgb-colors)

Documentation

creates a new color table instance from a vector of rgb triples, which also includes a symbol to refer to the color by. for example: #((0 0 0 :black) (128 128 128 :grey) (255 255 255 :white)).

new-contour-plot

Argument List

(data &key contour-levels (line-color *foreground-color*) (line-width 1) (fill-type none) fill-colors x-min x-max x-mapping y-min y-max y-mapping (copy t))

Documentation

creates a new contour plot.

- data is a 2d array of z values.
- contour-levels is a 1d vector of floats specifying the levels at which the contours should appear. if this is nil, then default contours are created based on the minimum and maximum values in data.
- line-color is a symbol specifying which color to use in the current color table for the contour lines.
- line-width is an integer specifying what size line to use for the contours (or zero for no line).
- fill-type is one of :none (contours only), :block (a single color between each contour) or :smooth (color varies continously between contours).
- fill-colors is a (optional) vector of colors from the current color table to use when fill-type is :block.
- x-min & y-min specify the location of data(0, 0) in plot coordinates.
- x-max & y-max specify the location of data(max, max) in plot coordinates.
- x-mapping & y-mapping are either 1d vectors or 2d matrices specifying how to map points in data into plot coordinates. If they are 1d vector, then data(i,j) is mapped to [x-mapping(i), y-mapping(j)]. If they are 2d vectors then data(i,j) is mapped to [x-mapping(i,j), y-mapping(i,j)]. They must be used together and both must be the same type & also match the dimensions of data. They will override x-min, y-min, x-max and y-max if they are specified.

new-custom-plot-object

Argument List

(min-max-function render-function)

Documentation

allows the user to create their own custom plot object.

- min-max-function must be either nil or a function of no arguments that returns the vector #(xmin xmax ymin ymax) that specifies the ideal window size for this object.
- render-function must be a function of one argument that specifies how to render the plot object. generally the rendering will be done with a bunch of calls to functions in the cl-plplot-system module. the current plot number will be passed to this function.

new-extended-color-table

Argument List

(&key control-points (color-table-size 128))

Documentation

creates a new extended color table instance from a vector of control points. for example: #((0.0 0.0 0.0 0.0) (1.0 255 255 255)) will create a gray scale color table. see plscmap1l in the plplot documentation for a more thorough explanation.

new-surface-plot

Argument List

(data-x data-y data-z &key contour-levels (copy t) (line-width 1) (line-style 1) (line-color *foreground-color*) light-source surface-options)

Documentation

Creates a new 3D (solid surface) plot.

- data-x specifies the x values of the points in data-z. If data-x is nil then data-z will be plotted against its row index in x.
- data-y specifies the y avlues of the points in data-z. If data-y is nil then data-z will be plotted against its column index in y.
- data-z is a 2D array of z values for the plot.
- contour-levels specifies the levels at which to draw contours, if desired. If this is not specified, default values are chosen.
- If copy is true then copies of data-x, data-y and data-z will be made, otherwise reference will be kept to the original vectors.
- line-width should be an integer line width, or zero if no line is desired
- line-style specifies what style line to draw (if a line is drawn), this should be a number between 1 and 8.
- line-color is the color to use for the lines in the plot.
- light-source is a 3 element vector #(x y z) specifying the location of the light source that will illuminate the plot surface.
- surface-options is list containing zero or more of the following symbols:

  :faceted - a network of lines is drawing connecting the points that make up the surface.

  :base-contours - a contour plot is also drawn in the base xy plane.

  :surface-contours - contour levels are drawn on the surface of the plot.

  :curtain - a curtain between the borders of the surface and the base xy plane.

  :magnitude-coloring - the surface is colored according to the z value of the plot. If this is not set then surface is colored according the intensity of the reflected light from light source."

new-text-item

Argument List

(the-text &key (text-color *foreground-color*) (text-justification 0.5) (font-size *font-size*))

Documentation

new-text-item, creates and returns a new text item.

- text is a string specifying the text.
- text-color is a symbol specifying the text color.
- text-justification specifies how to center the string relative to its
- reference point. 0.0 - 1.0, where 0.5 means the string is centered.
- font-size sets the fontsize relative to the default font size.

new-text-label

Argument List

(label-text-item text-x text-y &key (text-dx 0.0) (text-dy 0.0))

Documentation

new-text-label, creates and returns a new text-label object.

- text-item should be an object created by new-text-item.
- text-x specifies the x location of the text reference point (in window coordinates).
- text-y specifies the y location of the text reference point.
- text-dx and text-dy specify the location of a second reference point (in window coordinates) the text is drawn along a line connecting (text-x,text-y) and (text-x + text-dx, text-y + text-dy).

new-3D-text-label

Argument List

(label-text-item text-x text-y text-z &key (text-dx 0.0) (text-dy 0.0) (text-dz 0.0) (text-sz 0.0) (text-sz 0.0) (text-sz 0.0))

Documentation

new-3D-text-label, creates and returns a new 3D-text-label object.

- text-item should be an object created by new-text-item.
- text-x specifies the x location of the text reference point (in window coordinates).
- text-y specifies the y location of the text reference point.
- text-z specifies the z location of the text reference point.
- text-dx, text-dy and text-dz specify the location of a second reference point (in window coordinates) the text is drawn along a line connecting (text-x,text-y and text-z) and (text-x + text-dx, text-y + text-dy, text-z + text-dz).

- test-sx, text-sy and text-sz specify the location of a third reference point (in window coordinates) the text is sheared to be parallel to a line connecting (text-x,text-y and text-z) and (text-x + text-sx, text-y + text-sy, text-z + text-sz).

new-window

Argument List

(&key x-axis y-axis title (window-line-width 1.0) (window-font-size *font-size*) (foreground-color *foreground-color*) (background-color *background-color*) (viewport-x-min 0.1) (viewport-x-max 0.9) (viewport-y-min 0.1) (viewport-y-max 0.9) plots text-labels color-table)

Documentation

new-window, creates and returns a new window object.

- x-axis is a object of type axis.
- y-axis is a object of type axis.
- title is a object of type axis-label.
- foreground-color is a color symbol in the current color table.
- background-color is a color symbol in the current color table.
- window-line-width is a floating point number specifying the pen width to use when drawing the border & the tick marks.
- window-font-size is the font size to use for the tick mark labels.
- viewport-x-min (0.0 - 1.0) is the location of the left side of the border in the device window.
- viewport-x-max (0.0 - 1.0) is the location of the right side of the border.
- viewport-y-min (0.0 - 1.0) is the location of the bottom side of the border.
- viewport-y-max (0.0 - 1.0) is the location of the top side of the border.
- plots is a list of plot objects.
- text-labels is a list of text-label objects.
- color-table specifies what color table to use.

new-x-y-plot

Argument List

(x y &key (copy t) (line-width 1) (line-style 1) (symbol-size 0.0) symbol-type (color *foreground-color*) x-error y-error)

Documentation

creates a new x-y plot.

- if x is nil then y will be plotted against its index.
- if copy is true then copies of x,y,x-error and y-error will be made, otherwise references will be kept to the original vectors.
- line-width should be an integer line width, or zero if no line is desired.
- line-style specifies what style line to draw (if a line is drawn), this should be a number between 1 and 8.
- symbol-size specified how big to draw the symbols (1.0 is standard size). if it is zero the symbols are not drawn.
- symbol-type should be set to a number (that specifies a symbol) if you want specific types of symbols, otherwise default symbol types are used.
- color is the color to use when plotting the lines and symbols, it should be a symbol that specifies a color in the current color table. if it is not specified then the current foreground color will be used.
- x-error should be a vector of the same length as x that contains the size of the error bars in x.
- y-error is for error bars in y.

normal-font

Argument List

none.

number-symbol

Argument List

none.

overline

Argument List

none.

remove-axis-label-from-axis

Argument List

(a-axis &optional a-axis-label)

Documentation

remove-axis-label-from-axis, destructively removes a-axis-label from a-axis. if a-axis-label is not specified then the last axis-label is removed.

remove-color-from-color-table

Argument List

(a-color-table &optional color-to-remove)

Documentation

removes color-to-remove, if specified, or the last color if not.

remove-plot-from-window

Argument List

(a-window &optional a-plot)

Documentation

remove-plot-from-window, destructively removes a-plot from a-window. if a-plot is not specified then the last plot is removed.

remove-text-label-from-window

Argument List

(a-window &optional a-text-label)

Documentation

remove-text-label-from-window, destructively removes a-text-label from a-window. if a-text-label is not specified then the last text-label is removed.

render

Argument List

(a-window device &key filename (size-x 600) (size-y 500))

Documentation

renders a window and it associated plots and labels using device.

- device: a string naming a plplot graphical device such as 'xwin'.
- filename: where to save the graph for file based devices.
- size-x: the size of the window in x (pixels).
- size-y: the size of the window in y (pixels).

if you are using cl-plplot in a multi-threaded environment you should thread lock prior to calling render, as the plplot library only handles rendering one plot at a time.

roman-font

Argument List

none.

script-font

Argument List

none.

send-to-back

Argument List

(a-window a-plot)

Documentation

organizes the plots so that a-plot is drawn on the bottom.

set-color-table

Argument List

(a-window a-extended-color-table)

Documentation

sets the color table associated with a-window to a-color-table. returns the old color table. (set-color-table (cl-plplot::window cl-plplot::color-table)) method documentation: sets the color table associated with a-window to a-color-table. returns the old color table.

set-foreground-color

Argument List

(color)

Documentation

switches the pen to the desired foreground color, or the default foreground color if the desired color cannot be found.

subscript

Argument List

none.

superscript

Argument List

none.

underline

Argument List

none.

unicode-char

Argument List

none.

update-color

Argument List

(a-color-table color-symbol new-color)

Documentation

changes the color specified by color-symbol to new-color, which should be a rgb triple in the form #(r g b).

x-y-z-data-to-grid

Argument List

(data x-grid y-grid &key (algorithm grid-csa) optional-data)

Documentation

calls the plplot function plgriddata to turn irregulary spaced data as (x,y,z) points into the 2d array data[i,j] = z. please see the plplot manual for further documenation.

- data is either a 3 x n matrix of (x,y,z) points, or a list containing (x-vector, y-vector, z-vector).
- x-grid specifies the locations of the grid points in the x direction.
- y-grid specifies the locations of the grid points in the y direction.

- algorithm is one of :grid-csa, :grid-dtli, :grid-nni, :grid-nnidw, :grid-nnli or :grid-nnaidw and specifies what algorithm to use to grid the data.
- optional-data is a floating point number used in some of the algorithms for determining how to best grid the data.

5 Index