SHOW-PLANE AND FRIENDS
USER'S MANUAL

Robert Heller

COINS Technical Report 87-15

January 1987

Abstract

This user manual describes how to use the standard image operators *Show-Plane*, *Show-Plane-Edge*, *Show-Plane-Vector*, *Show-Lines*, *Show-Histogram*, *Show-Bar-Graph*, *Show-Points*, *and Show-Line-Graph*. These image operators are intended to provide a uniform method of displaying image and non-image data on a variety of graphics devices.

Contents

1 Introduction

The *Show-Plane* standard image operator displays a *plane* on a specified device as an intensity display. *Show-Plane* returns the *plane* that was passed to it as an argument. The *Show-Plane-Edge* standard image operator displays one or two planes on a specified device as an edge display. If *Show-Plane-Edge* is given only one plane it performs a one-by-two convolution to generate the horizontal and vertical edges, otherwise the two planes contain the horizontal and vertical edges. The *Show-Plane-Vector* standard image operator displays a pair of planes on a specified device as a vector field display. The *Show-Lines* standard image operator displays a *line array* on a specified device. *Show-Lines* returns the *line array* that was passed to it as an argument. The *Show-Histogram* and *Show-Bar-Graph* standard image operators display bar graphs, *Show-Histogram* takes a sequence of data values and *Show-Bar-Graph* gets passed a function that returns each data value to plot. The *Show-Points* and *Show-Line-Graph* standard image operators draw line graphs. *Show-Points* takes a 2D array of x,y values and *Show-Line-Graph* takes a function that returns two values (X and Y) to be plotted.

All of the functions described in this manual use a device specification. The device specification describes a "virtual" display device to use for the display. Included in this "virtual" device are such things as the size, shape, and location of the "virtual" display on the phyisical device (know as the *view port*), intensity and color mappings, commentary, the phyisical device to use, along with various device dependent parameters.

2 General Discussion

The eight functions described in this document fall into six catagories, depending on the type of display generated.

2.1 Image Displays

Show-Plane is used to display an image plane as an intensity or color display. Image displays are simply a mapping from image data values to varing

degrees of "intensity" on a display device. Some display devices support color, in which case two or more planes can be displayed in different color "intensities". Some devices cannot display multiple grey levels directly, so the grey level effect is generated with dithering. *Show-Plane* will call *stats-plane* if needed to compute MINVAL and MAXVAL, if these values are not present on the *planes'* association list. These values are needed as default values for the minimum and maximum intensity values used to scale the image values to display intensities.

2.2 Edge Displays

Show-Plane-Edge is used to display one or two planes as an edge display. The display consists of horizontal and vertical edges draw at the right and bottom edges of the image pixels. Edge displays are usually a rendering of some sort of edge information. *Show-Plane-Edge* can take either one or two input planes. If passed only one plane, *Show-Plane-Edge* does a 1x2 convolution to generate the horizontal and vertical edges. *Show-Plane-Edge* will call *stats-plane* if needed to compute MINVAL and MAXVAL, if these values are not present on the *planes'* association list. These values are needed as default values for the minimum and maximum intensity values used to scale the image values to display intensities.

2.3 Vector Field Displays

Show-Plane-Vector is used to generate a vector field display. The display consists of vectors with optional arrow heads and tails, which are use to indicate the directionality of the vectors. The input planes contain either displacements in rectanglar coordinates $(\Delta r, \Delta c)$ or polar coordinates (ρ, θ). The Vector displays are usually a rendering of some sort of motion or gradiant information.

2.4 Line Displays

Show-Lines is used to display line displays. The data involved is straight line data extracted by a line-based segmentation image operator. The data

array passed to *Show-Lines* is in image-coordinate space. *Show-Lines* assumes that the end points for each line are ordered, with the "start" point in the first two columns of each row and the "end" point in the second two columns of each row. This ordering can be displayed using the arrow-head-descriptor argument to cause *Show-Lines* to display arrow heads at the "end" points of the lines. *Show-Lines* uses either the plane and/or the limits arguments to determine the coordinate space and will then clip the line data to fit.

2.5 Bar Graphs

Show-Histogram and *Show-Bar-Graph* generate bar graph displays. These two functions generate a "bar graph" type display. *Show-Histogram* takes a sequence of histogram buckets as an argument and *Show-Bar-Graph* takes a function as an argument. The "inner loop" is basically the same for both of these functions.

The bar graph functions generate their displays using the these conventions:

- The left axis (the "Y" axis) is 1/10 of the width of the view-port to the right of the left edge of the view-port. The bottom axis (the "X" axis) is 1/10 the height of the view-port up from the bottom of the view-port.

- Axis tick labels are displayed using a text size of (1/60, 1/60) (height, width) of the view-port. The "X" axis tick labels are centered under the tick marks 1/15 of the view-port height below the "X" axis. The "Y" axis tick labels are drawn from the left edge of the view-port with the bottom of the text even with the tick point.

- "X" axis ticks are 1/40 if the view-port height long and "Y" axis are 1/30 of the view-port width long.

- The bars are drawn in an upright position with the minimim value point at the "X" axis. The bars themselves are scaled to use only 80% of the vertical size of the view-port for a maximum sized bar.

Bar width and spacing are computed as follows:

$$\begin{aligned} \texttt{Bar-To-Bar} &= \frac{\texttt{Device-Width} \times (9/10)}{\texttt{Number-Of-Bars}} \\ \texttt{Bar-Offset} &= \texttt{Bar-To-Bar} \times \texttt{Bar-Spacing} \\ \texttt{Bar-Width} &= \texttt{Bar-To-Bar} \times (1 - (\texttt{Bar-Spacing} \times 2)) \end{aligned}$$

`Device-Width` is the width of the device view-port. `Bar-To-Bar` is the distance between bar centers. `Bar-Offset` is the space between the left axis and the leftmost edge of the first bar. `Bar-Width` is the actual width of the bars.

- There are always 11 "Y" axis ticks. There is a "X" axis tick for each bar displayed. The "X" axis ticks are centered under the bars.

2.6 Line Graphs

Show-Points and *Show-Line-Graph* generate line graph displays. These two functions generate a "line graph" type display. *Show-Points* takes a 2D array as an argument and *Show-Line-Graph* takes a function as an argument. The "inner loop" is basically the same for both of these functions.

The line graph functions generate their displays using the these conventions:

- The left axis (the "Y" axis) is 1/10 of the width of the view-port to the right of the left edge of the view-port. The bottom axis (the "X" axis) is 1/10 the height of the view-port up from the bottom of the view-port.

- Axis tick labels are displayed using a text size of (1/60, 1/60) (height, width) of the view-port. The "X" axis tick labels are centered under the tick marks 1/15 of the view-port height below the "X" axis. The "Y" axis tick labels are drawn from the left edge of the view-port with the bottom of the text even with the tick point.

- "X" axis ticks are 1/40 if the view-port height long and "Y" axis are 1/30 of the view-port width long.

- There are as many axis ticks on each axis as there are data points.

3 Brief Functional Descriptions

llvs:show-plane

SYNOPSIS

llvs:show-plane plane &key device mask-plane mask-value limits

DESCRIPTION

Show-plane generates an intensity display of a plane on one of the available graphics devices. `Plane` is the plane to be displayed. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details"). The `:mask-plane`, `:mask-value`, and `:limits` keyword parameters are standard and do all of the standard things: `:mask-plane` and `:mask-value` provide for a mask plane and mask value list and `:limits` provide for specifying a display "window".

RETURNS

The first argument `plane`.

llvs:show-plane-edge

SYNOPSIS

llvs:show-plane-edge horz-plane &optional vert-plane &key device mask-plane mask-value limits

DESCRIPTION

Show-plane-edge generates an edge display of one or two planes on one of the available graphics devices. If `vert-plane` is `NIL`, a one-by-two operation is performed on `horz-plane` to generate the horizontal and vertical edges, otherwise, `horz-plane` and `vert-plane` are

the horizontal and vertical edge planes, respectively. The :device keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details"). The :mask-plane, :mask-value, and :limits keyword parameters are standard and do all of the standard things: :mask-plane and :mask-value provide for a mask plane and mask value list and :limits provide for specifying a display "window".

RETURNS

The two values: the two input planes (second value is NIL if no vertical edge plane).

---

llvs:show-plane-vector

SYNOPSIS

llvs:show-plane-vector radius-or-row-plane angle-or-col-plane
&key device coord-type
coding-method vector-length
length-clipping tail-location tail-size
arrow-head-descriptor mask-plane
mask-value limits

DESCRIPTION

Show-plane-vector generates a vector field display of a pair of planes on one of the available graphics devices. Radius-or-row-plane and angle-or-col-plane are the planes to be displayed. If there is only an angle-component plane, there is a global constant Unit-Vector-Plane which is a level 0 plane of size (1,1) with a value of 1 and a background value of 1 available for use as a radius plane. This plane can be used to display unit vector fields. The :device keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details"). The :mask-plane, :mask-value, and :limits keyword parameters are standard and

do all of the standard things: :mask-plane and :mask-value provide for a mask plane and mask value list and :limits provide for specifying a display "window". This function has a number of special keyword parameters. These parameters are:

:coord-type This is the coordinate system to use. There are two coordinate systems posible. Polar coordinates (rho, theta) and rectanglar coordinates (row,column). In polar coordinate mode, the first input plane contains the radius (rho) values and the second plane contains the angle (theta) values. In rectanglar mode, the first plane contains the row offsets and the second plane contains the column offsets. Posible values for this parameter are:

:polar use polar coordinate system

:rectangular use rectangular coordinate system

Default - :rectangular.

:coding-method This is the coding method to use. There are two coding methods. Length coding (vector length = magnitude) and intensity coding (brightness = magnitude). Posible values for this parameter are:

:length use length coding

:intensity use intensity coding

Default - :length.

:vector-length This is the maximum vector length. This is the maximum length the vectors should be. If length coding, this is the length that corresponds to the value for :maximum-intensity. If intensity coding, this is how long the vectors are. Should be a number, in image coordinates. Default: min(delta-row, delta-col).

:length-clipping Cliping flag. Can be :TRUE or :FALSE. If this parameter is :TRUE, then no vectors longer than :vector-length will be drawn. Only used with length coding. Default :FALSE.

`:tail-location` The location of the tail as a pair (row . col) of rational numbers (fraction of image pixel). The tail is a small rectangle drawn at the vectors' tails. Default: (1/2 . 1/2) (center of the image pixel).

`:tail-size` The size of the tail as a rational number (fraction of image pixel height). If the size is smaller then 2x2 display pixels, the tail is not drawn. Default: 1/10.

`:arrow-head-descriptor` The arrow head descriptor. This parameter describes the arrow heads to be drawn to the head of the vector. This is either a pair of numbers or `:no-arrow-heads`. If it is a pair the car is the arrow length as a number (if neg, then relative to the length of the vector, if pos then in absolute image coords) and the cdr is the arrow spread as the tangent of the half-angle. If it is `:no-arrow-heads`, then no arrow heads are drawn. Default: `:no-arrow-heads`.

RETURNS

Two values: the input planes.

---

llvs:show-lines

SYNOPSIS

llvs:show-lines lines &key pl device limits arrow-head-discriptor

DESCRIPTION

This function displays the lines specified by the `lines` argument, which must be an n by m maxrix (m >= 4). The first two elements of each row are the row and column position of the beginning of the line. The third and fourth elements are the row and column position of the end of the line. Either the `limits` or the `pl` argument must be supplied so that the position of the lines on the page may be determined. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details").

The `arrow-head-descriptor` argument provides for putting arrow headed on the lines as with `Show-Plane-Vector`.

RETURNS

The first argument `lines`.

---

llvs:show-histogram

SYNOPSIS

llvs:show-histogram histogram &key device valleys peaks start end bar-spacing x-axis-p y-axis-p x-axis-labels y-axis-labels

DESCRIPTION

This function displayes the histogram (as a bar graph) specified by the `histogram` argument, which must be a sequence. The sequence can be any Common LISP sequence type object - *Show-histogram* uses the Common LISP functions *reduce*, *elt*, and *length* functions on the histogram argument. Each element (from `start` to `end`, inclusive (defaults are 0 and *length*(`histogram`) 1) is a histogram bucket. `Bar-spacing` specifies 1/2 of the space between the bars as a fraction of the total bar width. `X-axis-p` and `y-axis-p` control whether or not axes are drawn. `X-axis-labels` and `y-axis-labels` are alternative labels for the axis ticks and are either sequences or `NIL`. If a label sequence is `NIL`, the ticks are labeled with the tick value. The `valleys` and `peaks` arguments are sequences of bucket numbers which mark valleys or peaks. The valleys are marked with a vertival line and the peaks are marked with the peak's value above the top of the bar. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details").

RETURNS

The first argument `histogram`.

---

llvs:show-bar-graph

SYNOPSIS

llvs:show-bar-graph bar-graph-function &key max-value min-value device function-arguments start end bar-spacing x-axis-p y-axis-p x-axis-labels y-axis-labels

DESCRIPTION

This function draws a bar graph. The `bar-graph-function` argument must be a function of at least one argument. It is called repeatedly to fetch the value for each bar. It is pased an index between `start` and end, inclusive, and `function-arguments` as its argument list. Is should return a value between `min-value` and `max-value`, inclusive. `Bar-spacing` specifies 1/2 of the space between the bars as a fraction of the total bar width. `X-axis-p` and `y-axis-p` control whether or not axes are drawn. `X-axis-labels` and `y-axis-labels` are alternative labels for the axis ticks and are either sequences or `NIL`. If a label sequence is `NIL`, the ticks are labeled with the tick value. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details").

RETURNS

T.

---

llvs:show-points

SYNOPSIS

llvs:show-points points &key device start end max-x-value min-x-value max-y-value min-y-value point-label x-axis-p y-axis-p x-axis-labels y-axis-labels

DESCRIPTION

This function displays a line graph. The `points` argument must be

an n by m matrix (m > 2). The first element of each row is the X coord and the second element is the Y coord of the points to graph. The ranges of the X and Y values is specified by `max-x-value`, `min-x-value`, `max-y-value`, and `min-y-value`. The `start` and `end` arguments specify the part of the `points` array to process. `Point-label` is either `NIL` or a single character to be draw at each data point. `X-axis-p` and `y-axis-p` control whether or not axes are drawn. `X-axis-labels` and `y-axis-labels` are alternative labels for the axis ticks and are either sequences or `NIL`. If a label sequence is `NIL`, the ticks are labeled with the tick value. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details").

RETURNS

The first argument, `points`.

llvs:show-line-graph

SYNOPSIS

llvs:show-line-graph point-function &key device start end max-x-value min-x-value max-y-value min-y-value point-label x-axis-p y-axis-p x-axis-labels y-axis-labels function-arguments

DESCRIPTION

This function draws a line graph. The `point-function` argument must be a function which returns two values, X and Y, which are the seccessive points to graph. The function is passed an index (from `start` to `end`, inclusive) and `function-arguments` as its argument list. The ranges of the X and Y values is specified by `max-x-value`, `min-x-value`, `max-y-value`, and `min-y-value`. The `start` and `end` arguments specify the part of the `points` array to process. `Point-label` is either `NIL` or a single character to be draw at each data point. `X-axis-p` and `y-axis-p` control whether or not

axes are drawn. `X-axis-labels` and `y-axis-labels` are alternative labels for the axis ticks and are either sequences or `NIL`. If a label sequence is `NIL`, the ticks are labeled with the tick value. The `:device` keyword parameter specifies a device name (as a keyword) or a device specification (as an association list) (see below under "Device Dependent Device Associations Details").

RETURNS

T.

---

4 Device Specification Association List

The device specification association list is used to specify one or more parameters for controlling the display. These parameters control such things as the device to use, display positioning, color(s) to use, etc.

It is intended that the user will develop a set of these lists for their own use. *Show-Plane*, *Show-Plane-Edge*, *Show-Plane-Vector*, etc. always deposit the association list last used in `*show-plane-descriptor*` in the `llvs` package whether this list was the input argument or generated through interaction with the user. The user can capture this list and save it for future use.

4.1 Device Common Associations

These associations are common to all display devices.

4.1.1 The View Port

The view port is that portion of the display device that is used for the display. It is possible to generate multiple displays of the same plane or different planes in different parts of the display. Device independent coordinates are used to avoid device dependencies. Also, the origin is the same as for image planes.

4.1.2 Absolute And Negative Displays

The `:absolute` and `:negative` parameters are used to control how the display is to be generated. The `:absolute` flag is useful for displaying only magnitude information. Planes are sometimes generated with signed results, but only the magnitude (and not the sign) is important or interesting. Also, sometimes the data is numerically inverted or a photographic-like negative is desired. The `:negative` flag handles this. These parameters are only used by *Show-Plane* and *Show-Plane-Edge*.

4.1.3 Maximum And Minimum Intensity

The `:maximum-intensity` and `:minimum-intensity` parameters are used to stretch or compress the display range. These two parameters function the same way that the old VISION system used MINVAL and MAXVAL in functions like *showpl*. These two parameters specify the two end points of a straight (linear) line where the `:minimum-intensity` is at the low end and `:maximum-intensity` is at the high end. This line describes a linear function whose output is screen intensity and whose input is the pixel values. These parameters are only used by *Show-Plane*, *Show-Plane-Vector*, and *Show-Plane-Edge*. *Show-Plane-Vector* only uses `:maximum-intensity`.

4.1.4 Comments

The comment related parameters (`:comments`, `:comment-location` and `:comment-size`) are used to generate comments, captions or titles for displays.

4.1.5 Erasing And Page Feeding

The `:erase-screen` and `:new-page` parameters are functionally equivalent. Both names are allowed to allow for more readable (and understandable) association lists.

4.1.6 Pixel Size And Shape

The `:aspect-ratio` and `:pixel-size-ratio` parameters control the size and shape of the pixels. These parameters are only used by the functions

which deal in image coordinates.

4.1.7 Common Device Association List Details

Specifically, the following device-common associations are recognized:

`:view-port` This is view port to use. The value should itself be an association list with the keys `:top`, `:bottom`, `:left` and `:right`. The values for these keys are rational numbers from 0 to 1. Zero is the top/left of the device's screen or page and one is the bottom/right of the device's screen or page. The defaults are :top 0, :bottom 1 :left 0 :right 1 - that is the whole screen or page.

`:absolute` If this parameter is specified with a non-NIL value, use the absolute values. The default is NIL - that is do not use absolute values of the pixels in the plane.

`:negative` If this parameter is specified with a non-NIL value, generate a "negative" intensity display. The default is NIL - that is generate a "positive" display[1].

`:device` This specifies the device to use. (See below under "Device Dependent Device Associations Details".) There is no default device - a device *MUST* be specified, unless a call was made to *llvs-use-device*, in which case the device named in the most recent call to *llvs-use-device* is used.

`:maximum-intensity` This specifies the pixel value that maps to the maximum brightness of the graphic device. This value need not be in the plane's range (between MINVAL and MAXVAL). The default is the plane's MAXVAL or the maximum of the absolute values of MAXVAL and MINVAL if `:absolute` is non-NIL.

`:minimum-intensity` This specifies the pixel value that maps to the minimum brightness of the graphic device. This value need not be in the plane's range (between MINVAL and MAXVAL). The default is the plane's MINVAL or 0 if `:absolute` is non-NIL.

[1] A "positive" display is defined as maximum "brightness" equals maximum intensity.

`:erase-screen` or `:new-page` If either of these parameters is specified with a non-NIL value, the screen is erased or a new page is generated. These two parameters are interchangeable. Both are accepted for semantic reasons. If both are supplied, then a logical or is performed on the two values. The default for both is NIL.

`:comments` This is a list of zero or more comment strings. Each string is displayed on a single line. The first comment is placed at the comment location and each successive comment is placed below the previous comment. The default is NIL.

`:comment-location` This is a list of two rational numbers (x y) specifying the location of the first comment. This is the location of a point above and to the left of the first character and is in terms of the view-port. The default is (0 31/32)[2].

`:comment-size` This is a list of two rational numbers (h w) specifying the size of the comment characters. It is in terms of the view-port. The default is (1/32 1/32)[3].

`:aspect-ratio` This is the aspect ratio of the pixels. It is a pure rational number. The default for this parameter and `:pixel-size-ratio` are computed to make the plane fill the view-port.

`:pixel-size-ratio` This is the display-to-image pixel size ratio. It is a pair of pure rational numbers and is the ratio of the size (h w) of the pixels over the size (h w) of the view port. The default is computed from the size of image plane and is adjusted to fit the view-port.

Each supported device will recognize additional associations. Also, unused associations are ignored without warning. This means that association lists can be developed to work on more than one device.

[2]This is 16 device pixels up from the bottom at the extreme left on a 512x512 display (such as the COMTAL or the Grinnells with the default view port).

[3]This is 16x16 pixels per character on a 512x512 display (such as the COMTAL or the Grinnells with the default view port).

4.2 Device Dependent Associations

These associations are specific to particular devices. Some are common to some devices (i.e. :color is common to color devices, but is not used with monochrome devices).

4.2.1 Color And Configuration

The `:color` and `:configuration` association is used only with the devices which are capable of color and/or multiple configurations. For the newer Grinnells the `:color` association selects a bank of display memory.

4.2.2 Extra Erasing Associations

For the grinnell devices, there are two extra erasing options. Mostly these are used to clear away extraneous (previous) display information.

4.2.3 High Speed Transfers

The newer Grinnells, the COMTAL and the IP8500 have the ability to take raw byte-pixel data and display this more or less directly. The transfer rate is fast, but no scaling is possible. Also, only specifically sized images are transferable this way. The `:fast` flag controls this option. This flag is only used by *Show-Plane*.

4.2.4 Printer Related Associations

For the printer type devices (the LGP's and the Printronix) there is a collection of associations relating to generating a print job. These include queue name, forms type and other print queue related options.

Since these devices cannot be written to directly, a file name is needed to use as the name of the file used to hold plot commands. There is an option to determine whether a new file is generated or an existing file is appended to. Also, there are options to control such things as if and where to insert "page feeds".

4.2.5 COMTAL Handling

The COMTAL is not really a graphics device in the same sense as the other devices supported. Instead it is an image processing subsystem. The two COMTAL-specific associations, `:display-plane-type` and `:plane-mapping`, determine which of the 7 possible COMTAL planes to write to. There are presently 3 image planes (512x512x8 bits) and 4 graphic planes (512x512x1 bits). To actually cause a display, you need to send the COMTAL commands (separate functions are be provided) or enter commands at the COMTAL's terminal.

4.2.6 The Tektronix 4662 Plotter

This plotter is not very sophisticated. It is a single-pen plotter. It needs to have paper feed to it by hand. The `:pause-message` association can be used to display a message on the user's terminal and read back a newline, effectively pausing the system while the user changes the paper and/or pen (the message can inform the user as to what is needed).

4.2.7 Device Dependent Device Associations Details

`:user` or `:terminal` This is the user's terminal. If the terminal can do pixel graphics then pixel graphics are done. Otherwise, printable characters are used in a way similar to Brian Burns's *tonepl*. Device-dependent keys:

`:terminal-type` This is the terminal type to use. Allowed values are:

`:sun` Monochrome SUN Workstation terminal.	NYI
`:sun-color` Color SUN Workstation terminal.	NYI
`:regis` [4] RIGIS-compatable terminal (GIGI's and VT125's).	
`:vt240` [5] VT240 terminal.	
`:lispm` Monochrome LISP Machine terminal[6].	NYI

[4] *Show-Plane-Edge* and *Show-Plane-Vector* are not yet implemented for regis terminals.

[5] *Show-Plane-Edge* and *Show-Plane-Vector* are not yet implemented for the VT240 terminal.

[6] Only allowed on the LISP Machine version.

:lispm-color Color LISP Machine terminal[7]. NYI

:tektronix4107 Tektronix 4107 color terminal.

:dumb Dumb terminal. This is the default.

:color [8] Specifies what color to display the image in. Only used for color terminal types.

:grinnell1 This is the first grinnell. Also known as the old grinnell. It has the following device dependent keys: NYI

:color [9] This is the color to use. Possible values are :red, :green, :blue, :red-overlay, :green-overlay or :blue-overlay. The default is :green. If an overlay color is specified, only 1 bit per pixel can be displayed.

:configuration This the the configuration to use. One of the following are allowed:

:config444 This is the 444 grinnell configuration. This configuration sets up the grinnell to use 4 bits for each red, green, and blue plus one bit of overlay for each color. It is the default configuration unless the :color key is set to :green.

:config888 This is the same as :config444. An informatative message is generated if this is used and it is treated as if :config444 had been specified.

:config565 This is the 565 grinnell configuration. This configuration sets up the grinnell to use 5 bits for each red and blue and 6 bits for green. The overlay bits are not available. It is an error to specify this configuration and an overlay color at the same time.

:config080 This is the 080 grinnell configuration. This configuration sets up the grinnell to use 7 bits for green and

[7] Only allowed on the LISP Machine version.

[8] For *Show-Plane-Vector* there are two keyword values: :vector-color and :tail-color.

[9] For *Show-Plane-Vector* there are two keyword values: :vector-color and :tail-color.

one bit for each overlay color. It is an error to specify this configuration with colors :red or :blue. This is the default if the color is :green.

:erase-ov If this parameter is specified with a non-NIL value, then the overlay planes are cleared. It is ignored if the configuration is 565. The default is NIL.

:erase-non-ov If this parameter is specified with a non-NIL value, then the non-overlay planes are cleared. The default is NIL.

:grinnell2 This is the second grinnell. Also known as the new grinnell. It has the following device dependent keys:

:color [10] This is the color to use. Possible values are :red, :green, :blue, :red-overlay, :green-overlay, :blue-overlay or :white-overlay. The default is :green. If an overlay color is specified, only 1 bit per pixel can be displayed.

:configuration This the the configuration to use. One of the following are allowed:

:config888 This is the 888 grinnell configuration. This configuration sets up the grinnell to use 8 bits for each red, green, and blue plus one bit of overlay for each color. It is the default configuration.

:config444 This is the same as :config888. An informatative message is generated if this is used and it is treated as if :config888 had been specified.

:config800 This is the 800 grinnell configuration. This configuration sets up the grinnell to use 8 bits for red and one bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :green or :blue.

:config080 This is the 080 grinnell configuration. This configuration sets up the grinnell to use 8 bits for green and one

[10] For *Show-Plane-Vector* there are two keyword values: :vector-color and :tail-color.

bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :red or :blue.

:config008 This is the 008 grinnell configuration. This configuration sets up the grinnell to use 8 bits for blue and one bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :red or :green.

:erase-ov If this parameter is specified with a non-NIL value, then the overlay planes are cleared. The default is NIL.

:erase-non-ov If this parameter is specified with a non-NIL value, then the non-overlay planes are cleared. The default is NIL.

:fast This specifies whether or not to use fast mode. This flag is only checked for unsigned-byte images which fit the view port exactly (ie one-to-one image pixel to display pixel[11]) and where there is no mask-plane and delta row and delta column both are equal 1. If non-NIL, a high-speed transfer is done bypassing all scaling adjustments. The default is T.

:grinnell3 This is the third grinnell. It is the one we got from IBM[12]. It has the following device dependent keys: NYI

:color [13] This is the color to use. Possible values are :red, :green, :blue, :red-overlay, :green-overlay, :blue-overlay or :white-overlay. The default is :green. If an overlay color is specified, only 1 bit per pixel can be displayed.

:configuration This the the configuration to use. One of the following are allowed[14]:

[11] Display pixels can be either single or double height or single or double width.

[12] It is not presently hooked up.

[13] For *Show-Plane-Vector* there are two keyword values: :vector-color and :tail-color.

[14] These configurations are guess work. I won't know for sure until I look at the documentation for this beast. This is probably close, though.

:config888 This is the 888 grinnell configuration. This configuration sets up the grinnell to use 8 bits for each red, green, and blue plus one bit of overlay for each color. It is the default configuration.

:config444 This is the same as :config888. An informatative message is generated if this is used and it is treated as if :config888 had been specified.

:config800 This is the 800 grinnell configuration. This configuration sets up the grinnell to use 8 bits for red and one bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :green or :blue.

:config080 This is the 080 grinnell configuration. This configuration sets up the grinnell to use 8 bits for green and one bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :red or :blue.

:config008 This is the 008 grinnell configuration. This configuration sets up the grinnell to use 8 bits for blue and one bit for each overlay color. The display is in black and white. It is an error to specify this configuration with colors :red or :green.

:erase-ov If this parameter is specified with a non-NIL value, then the overlay planes are cleared. The default is NIL.

:erase-non-ov If this parameter is specified with a non-NIL value, then the non-overlay planes are cleared. The default is NIL.

:fast [15] This specifies whether or not to use fast mode. This flag is only checked for unsigned-byte images which fit the view port exactly (i.e. one-to-one image pixel to display pixel[16]) and where there is no mask-plane and delta row and delta column both are equal to 1. If non-NIL, a high-speed transfer is done, bypassing all scaling adjustments. The default is T.

[15] Assumes that the beast is like the grinnell2.

[16] Display pixels can be either single or double height or single or double width.

:comtal [17] This is the COMTAL. It has the following device dependent keys:

:display-plane-type This specifies whether a COMTAL image plane (8 bits per pixel) or a COMTAL graphic plane (1 bit per pixel) is to be used. Legal values are :image or :graphic. The default is :image.

:plane-mapping This specifies which COMTAL plane to use. If :display-plane-type is :image this parameter can have values of 1, 2 or 3 and if :display-plane-type is :graphic this parameter can values of 1, 2, 3 or 4. The default is 1.

:fast This specifies whether or not to use fast mode. This flag is only checked for unsigned-byte images which fit the view port exactly (i.e. one-to-one image pixel to display pixel) and where there is no mask-plane and delta row and delta column both are equal 1 and the display plane type is of type :image. If non-NIL, a high-speed transfer is done, bypassing all scaling adjustments. The default is T.

:symbolics-lgp This is the Symbolics LGP. It has the following device dependent keys:

:print-now If this is specified as non-NIL, the output file is queued to the print queue for printing. The default is NIL (ie don't print it now).

:file-name The name of the plot file. The default is "PLOTLGP.DAT".

:append-to-file This flag specifies whether a new plot file should be opened or not. If T, an existing file is appended to. The default is NIL.

:page-feed-before This flag specifies whether or not a "page feed" is generated before the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

[17] *Show-Plane-Edge* and *Show-Plane-Vector* are not yet implemented for the COMTAL.

:page-feed-after This flag specifies whether or not a "page feed" is generated after the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

:queue The name of the queue to place the print job on. Only checked if :print-now is non-NIL. The default is "SYS$LASER"[18].

:form The name of the printer form type to use. Only checked if :print-now is non-NIL. The default is "LASER"[19].

:job-name The job name to use. Only checked if :print-now is non-NIL. The default is the file name.

:copies The number of copies to print. Only checked if :print-now is non-NIL. The default is 1.

:notify If this is specified as a non-NIL value, the user will be notified when the print job completes. Only checked if :print-now is non-NIL. The default is T.

:flag-page If this is specified as a non-NIL value, a flag page will be generated. Only checked if :print-now is non-NIL. The default is NIL.

:trailer-page If this is specified as a non-NIL value, a trailer page will be generated. Only checked if :print-now is non-NIL. The default is T.

:delete If this is specified as a non-NIL value, the file is deleted after printed. Only checked if :print-now is non-NIL. The default is T.

:ln03 This is the LN03 laser printer. It has the following device dependent keys:

:print-now If this is specified as non-NIL, the output file is queued to the print queue for printing. The default is NIL (ie don't print it now).

[18]This is provided to handle sites with more than one Symbolics LGP print queue.
[19]This is provided to handle sites with more than one laser printer form type.

:file-name The name of the plot file. The default is "PLOTLN03.DAT".

:append-to-file This flag specifies whether a new plot file should be opened or not. If T, an existing file is appended to. The default is NIL.

:page-feed-before This flag specifies whether or not a "page feed" is generated before the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

:page-feed-after This flag specifies whether or not a "page feed" is generated after the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

:queue The name of the queue to place the print job on. Only checked if :print-now is non-NIL. The default is "VIS$LASER"[20].

:form The name of the printer form type to use. Only checked if :print-now is non-NIL. The default is "LN03"[21].

:job-name The job name to use. Only checked if :print-now is non-NIL. The default is the file name.

:copies The number of copies to print. Only checked if :print-now is non-NIL. The default is 1.

:notify If this is specified as a non-NIL value, the user will be notified when the print job completes. Only checked if :print-now is non-NIL. The default is T.

:flag-page If this is specified as a non-NIL value, a flag page will be generated. Only checked if :print-now is non-NIL. The default is T.

:trailer-page If this is specified as a non-NIL value, a trailer page will be generated. Only checked if :print-now is non-NIL. The default is NIL.

[20]This is provided to handle sites with more than one LN03 LGP print queue.
[21]This is provided to handle sites with more than one laser printer form type.

:delete If this is specified as a non-NIL value, the file is deleted after printed. Only checked if :print-now is non-NIL. The default is T.

:line-width This is how thick to draw the lines in device pixels. The default is 3, which gives results simular to the LGP, except the lines are nice and black.

:line-delta This is the step value used when drawing lines that are not exactly vertical or horizonatal. These lines are actually made up of many little vertical or horizontal lines. This parameter controls the spacing between these little lines. A value of 1 gives a spacing of 0 pixels between these little lines. The default is 1.

:printronix [22] This is the Printronix printer. It has the following device dependent keys:

:print-now If this is specified as non-NIL, the output file is queued to the print queue for printing. The default is NIL (i.e. don't print it now).

:file-name The name of the plot file. The default is "PLOTPL.DAT".

:append-to-file This flag specifies whether a new plot file should be opened or not. If T, an existing file is appended to. The default is NIL.

:page-feed-before This flag specifies whether or not a "page feed" is generated before the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

:page-feed-after This flag specifies whether or not a "page feed" is generated after the plot has been generated. This is used when more than one plot is to be generated in a given file. The default is T.

:queue The name of the queue to place the print job on. Only checked if :print-now is non-NIL. The default is "SYS$PRINT"[23].

[22] *Show-Plane-Edge* and *Show-Plane-Vector* are not yet implemented for the printronix printer.

[23] This is provided to handle sites with more than one Printronix print queue.

:form The name of the printer form type to use. Only checked if :print-now is non-NIL. The default is "DEFAULT"[24].

:job-name The job name to use. Only checked if :print-now is non-NIL. The default is the file name.

:copies The number of copies to print. Only checked if :print-now is non-NIL. The default is 1.

:notify If this is specified as a non-NIL value, the user will be notified when the print job completes. Only checked if :print-now is non-NIL. The default is T.

:flag-page If this is specified as a non-NIL value, a flag page will be generated. Only checked if :print-now is non-NIL. The default is T.

:trailer-page If this is specified as a non-NIL value, a trailer page will be generated. Only checked if :print-now is non-NIL. The default is NIL.

:delete If this is specified as a non-NIL value, the file is deleted after printed. Only checked if :print-now is non-NIL. The default is T.

:plotter This is the Tektronix 4662 plotter. It has the following device dependent keys:

:pause-message This is a string to print out to *query-io*. *Show-Plane* will then wait for a RETURN (newline) from *query-io* before continuing. If this is NIL, *Show-Plane* won't print anything and won't wait. This is useful because the plotter does not have automatic paper feeding nor multiple pens. The default is NIL.

:ip8500 This is the Gould DeAnza IP8500 Image processor. It has the following device dependent keys:

[24]This is provided to handle sites with more than one Printronix printer form type.

`:fast` This specifies whether or not to use fast mode. This flag is only checked for unsigned-byte images which fit the view port exactly (i.e. one-to-one image pixel to display pixel) and where there is no mask-plane and delta row and delta column both are equal 1. If non-NIL, a high-speed transfer is done, bypassing all scaling adjustments. The default is T.

`:memory-channel-number` [25] This is the memory channel to use. Should be in the range 0-3.

`:extended-memory` This flag specifies whether or not to use the extended memory. If non-NIL, then extended memory is used, otherwise normal memory is used. The extended memory is 1024 by 1024 pixels. Normal memory is 512 by 512. Only 512 by 512 pixels can be displayed. The default is NIL (normal memory).

[25] For *Show-Plane-Vector* there are two keyword values: `:vector-memory-channel-number` and `:tail-memory-channel-number`.

5 Examples

These examples show how each of the functions described in this manual might be called.

Example 5.1: Display an image.

This is an example of how to display an image. The global variable `im1i8` is bound to the plane to be displayed and the display device is Grinnell 2's green image bank, in the fourth quadrent.

```
Lisp> (show-plane im1i8 :device '(,@grinnell2-green
                                  ,@view-port-q4))
#<PLANE ...>
```

Example 5.2: Display a region segmentation.

This is an example of how to display a region segmentation. The global `l6is` is bound to a region segmentation plane and the display device is the Tektronix 4662 plotter.

```
Lisp> (llvs-use-device :plotter)
:PLOTTER

Lisp> (show-plane l6is nil :device (append
                                    absolute
                                    max-intensity-1
                                    '((:minimum-intensity
                                         . 0)
                                     ) ))
#<PLANE ...> ;
NIL
```

Example 5.3: Display a gradient field.

This is an example of a gradient field displayed with *Show-Plane-Vector*. The input planes are mag and theta and the display device is the Tektronix 4107 terminal.

```
Lisp> (stats-plane mag)  ; get maxval of mag
#<PLANE ...>

Lisp> (llvs-use-device :user)
:TERMINAL

Lisp> (show-plane-vector mag theta :device
                                   '((:terminal-type
                                        . :tektronix4107)
                                     (:vector-color . :red)
                                     (:tail-color . :green)
                                     (:maximmum-intensity
                                       . ,(cdr
                                            (assoc
                                              ':maxval
                                         (plane-associations
                                                mag)))))
                                   :vector-length 2.0
                                   :coord-type :polar
                                   :arrow-head-descriptor
                                     ' '(-0.3 . 0.7)
                                   :tail-size 1/8
                                   )
#<PLANE ...> ;
#<PLANE ...>
```

Example 5.4: Display line results.

This is an example of how to display the line results (such as from Brian's Lines). The global variable *lines* is bound to an array of line data. The plane the lines are from is 16i and the display device is Grinnell

2's white overlay plane, in 080 configuration. The overlay planes are erased before the lines are displayed. Arrow heads are also drawn.

```
Lisp> (show-lines *lines* :pl 16i :device
                                  (append
                                    erase-ov
                                    grinnell2-white-ov
                                    grinnell-config080)
                                  :arrow-head-descriptor
                                      '(-0.3 . 0.7))
#<ARRAY #x123345>
```

Example 5.5: Display a simple histogram

This is an example of displaying a histogram. The global valiable *histo* contains the histogram. The display device is the Tektronix 4107 terminal, in yellow. Any existing graphics display is cleared.

```
Lisp>(show-histogram *histo*
                     :device
                      '((:device . :terminal)
                        (:erase-screen . t)
                        (:terminal-type . :tektronix4107)
                        (:color . :yellow))
#<ARRAY #x44457>
```

Example 5.6: Display a bar graph of a function's value.

This is an example of a bar graph using a function to generate the bar height values. The function used fetches the data from a defstruct and does some arithmetic adjustments.

```
;;; Generate a bar graph of percentages of used vaxen
;;;
;;; first the data structure:
;;
(defstruct vaxen-usage
  (name        "" :type string)     ; the name of the VAX
  (peak-users  0  :type integer)    ; peak number of users
  (max-users   32 :type integer)    ; max number of users
                                    ; (at a time)
  (cpu-type '|750| :type symbol)    ; cpu type
   )

;;; the data itself
;;;

(setf *vax-use-data* (make-array 4)
      (aref *vax-use-data* 0)
        (make-vaxen-usage :name "VAX7"
                          :peak-users 5
                          :max-users 6)
      (aref *vax-use-data* 1)
        (make-vaxen-usage :name "VAX8"
                          :peak-users 10
                          :max-users 10)
      (aref *vax-use-data* 2)
        (make-vaxen-usage :name "VAX9"
                          :peak-users 4
                          :max-users 8)
      (aref *vax-use-data* 3)
        (make-vaxen-usage :name "VAX10"
                          :peak-users 2
                          :max-users 4)
     )

;;; define the function to fetch the data:
```

```
(defun get-percent-max-usage (index data-array)
   (let ((peak-usage (vaxen-usage-peak-users
                        (aref data-array index)))
         (max-users  (vaxen-usage-max-users
                        (aref data-array index))))
        (/ peak-usage max-users) ))

;;; select the device: the ln03
;;

(llvs-use-device :ln03)


;;; now draw the graph
;;;

(show-bar-graph #'get-percent-max-usage
                :start 0 :end 3
                :min-value 0 :max-value 1
                :function-arguments
                     (list *vax-use-data*)
                :x-axis-labels
                     (map 'vector
                          #'vaxen-usage-name
                          *vax-use-data*)
                :y-axis-labels
                  '("0%" "10%" "20%"
                    "30%" "40%" "50%"
                    "60%" "70%" "80%"
                    "90%" "100%")
                :device '((:print-now . t))
                )
```

Example 5.7: Draw a line graph from a points array

This example displays the data in the global variable *points* as a line graph on Tektronix 4662 plotter. The data values go from $-\pi$ to π in the X direction and from -1 to 1 in the Y direction. There are 20 data points.

```
Lisp>(show-points *points* :device
                           '((:device . :plotter)
(:pause-message . "Please insert the RED pen")
(:comments "Data values from (TEST-OPERATOR L6I) call.")
                             (:comment-location 0 33/32)
                             (:view-port (:bottom . 5/6))
                          :start 0 :end 19
                          :min-x-value (- PI)
                          :max-x-value PI
                          :min-y-value -1
                          :max-y-value 1)
#<ARRAY #x90827>
```

Example 5.8: Draw a sine curve

This example uses the Common LISP function *sin* to generate a sine curve plot on the IP8500:

```
Lisp>(llvs-use-device :ip8500)
:IP8500

Lisp>(show-line-graph #'(lambda (i)
                          (let ((x (* (/ i 20) (* PI 2))))
                            (values x (sin x)) ))
                      :start 0 :end 20
                      :min-x-value 0 :max-x-value (* PI 2)
                      :min-y-value -1 :max-y-value 1
                      )
T
```

A Standard Constants Available

There are a collection of useful constants for use in association lists passed in to the :DEVICE argument to SHOW-PLANE. These constants can be combined using either the APPEND function or using a backquote form:

```
> ;; display a plane on the comtal in the comtal's image 1 in
> ;; the upper left quadrant.
> (show-plane some-plane :device (append comtal-image-1
>  view-port-q1))
> ;; same thing, but using backquote
> (show-plane some-plane :device `(,@comtal-image-1
>  ,@view-port-q1))
```

The constants are as follows:

COMTAL-IMAGE-1 - COMTAL image plane 1
COMTAL-IMAGE-2 - COMTAL image plane 2
COMTAL-IMAGE-3 - COMTAL image plane 3
COMTAL-GRAPHIC-1 - COMTAL graphic plane 1
COMTAL-GRAPHIC-2 - COMTAL graphic plane 2
COMTAL-GRAPHIC-3 - COMTAL graphic plane 3
COMTAL-GRAPHIC-4 - COMTAL graphic plane 4

GRINNELL1-RED - grinnell1 red plane
GRINNELL1-GREEN - grinnell1 green plane
GRINNELL1-BLUE - grinnell1 blue plane
GRINNELL1-RED-OV - grinnell1 red overlay plane
GRINNELL1-GREEN-OV - grinnell1 green overlay plane
GRINNELL1-BLUE-OV - grinnell1 blue overlay plane

GRINNELL2-RED - grinnell2 red plane
GRINNELL2-GREEN - grinnell2 green plane
GRINNELL2-BLUE - grinnell2 blue plane
GRINNELL2-RED-OV - grinnell2 red overlay plane
GRINNELL2-GREEN-OV - grinnell2 green overlay plane
GRINNELL2-BLUE-OV - grinnell2 blue overlay plane
GRINNELL2-WHITE-OV - grinnell2 white overlay plane

GRINNELL3-RED - grinnell3 red plane
GRINNELL3-GREEN - grinnell3 green plane
GRINNELL3-BLUE - grinnell3 blue plane
GRINNELL3-RED-OV - grinnell3 red overlay plane
GRINNELL3-GREEN-OV - grinnell3 green overlay plane
GRINNELL3-BLUE-OV - grinnell3 blue overlay plane
GRINNELL3-WHITE-OV - grinnell3 white overlay plane

GRINNELL-CONFIG444 - grinnell1 444 configuration
GRINNELL-CONFIG888 - grinnell[2,3] 888 configuration
GRINNELL-CONFIG565 - grinnell1 565 configuration
GRINNELL-CONFIG080 - grinnell[1,2,3] 080 configuration
GRINNELL-CONFIG800 - grinnell[2,3] 800 configuration
GRINNELL-CONFIG008 - grinnell[2,3] 008 configuration

VIEW-PORT-Q1 - upper left quadrant view port
VIEW-PORT-Q2 - upper right quadrant view port
VIEW-PORT-Q3 - lower left quadrant view port
VIEW-PORT-Q4 - lower right quadrant view port

MAX-INTENSITY-1 - maximum intensity of 1.0
MAX-INTENSITY-2 - maximum intensity of 2.0
MAX-INTENSITY-4 - maximum intensity of 4.0
MAX-INTENSITY-8 - maximum intensity of 8.0
MAX-INTENSITY-16 - maximum intensity of 16.0
MAX-INTENSITY-32 - maximum intensity of 32.0
MAX-INTENSITY-64 - maximum intensity of 64.0
MAX-INTENSITY-128 - maximum intensity of 128.0
MAX-INTENSITY-256 - maximum intensity of 256.0

SLOW - force scaling

NEGATIVE - produce negative display
ABSOLUTE - use absolute values
ERASE - erase screen
ERASE-OV - erase overlay planes (Grinnells only)
ERASE-NON-OV - erase non-overlay planes (Grinnells only)

B View Port Generating Functions

This appendix describes several functions useful for generating view ports for *Show-Plane* and friends.

B.1 Functions

llvs:view-port-of-plane

SYNOPSIS

llvs:view-port-of-plane plane1 plane2

DESCRIPTION

This function computes a view-port association list that represents that portion of `plane2` (the *base plane*) `plane1` (the *sub-plane*) corresponds to. This is done by adjusting for relative sizes and locations. Note: if `plane1` is larger than `plane2`, an over-size view port is generated (this may be ok - if it is then used to compute a sub-view-port of a another view-port, the final result may be in range).

RETURNS

An association list with keys `:top`, `:bottom`, `:left` and `:right`.

---

llvs:view-port-of-view-port

SYNOPSIS

llvs:view-port-of-view-port inner-view-port outer-view-port

DESCRIPTION

This function computes the view port of a view port. That is, it computes a "sub-view-port" of a view port. `Outer-view-port` is considered the "main" view port and `inner-view-port` is the desired "piece".

RETURNS

The new view-port describing the piece (`inner-view-port`) in terms of `outer-view-port`.

---

llvs:compute-new-view-port-for-show-plane

SYNOPSIS

llvs:compute-new-view-port-for-show-plane new-plane
old-plane

DESCRIPTION

This macro generates a new view port to display `new-plane` on top of `old-plane`. It gets the view port used to display `old-plane` from `llvs:*show-plane-descriptor*`. It uses both *view-port-of-plane* to compute a sub-view-port and *view-port-of-view-port* to compute a final view-port from the view-port generated by *view-port-of-plane* and the view-port extracted from `llvs:*show-plane-descriptor*`.

RETURNS

The new view port.

Example B.1: Display a region segmentation atop the intensity plane

Display a region segmentation atop the intensity plane. The segmentation is over a portion of the intensity plane.

```
> ;; *intensity* is the intensity plane and *seg* is the
> ;; segmentation.
> (show-plane *intensity*
>             :device '(,@grinnell2-red ; use "red" bank
>                       ,@grinnell-config800 ; disp in BW
>                       ,@erase ; erase screen
>                       ))
#S(PLANE ...)
```

```
> (show-plane-edge *seg* nil   ; use white overlay
>                  :device '(,@grinnell2-white-ov
>                           ,@grinnell-config800
>  ;; compute new view-port
> ,@(compute-new-view-port-for-show-plane *seg* *intensity*)
>                           ))
#S(PLANE ...)
NIL
```

Example B.2: Generate the first quadrant of the first quadrant

This code segment computes a new view-port association which describes the upper left quadrant of the upper left quadrant. This new view-port describes the upper left sixteenth of the display area.

```
> (defconstant view-port-q11
>              (acons :view-port
>                     (view-port-of-view-port
>                       (cdr (assoc :view-port view-port-q1))
>                       (cdr (assoc :view-port view-port-q1))
>                        )
>                     nil))
VIEW-PORT-Q11
>view-port-q11
((:VIEW-PORT (:TOP . 0) (:BOTTOM . 1/4) (:LEFT . 0)
             (:RIGHT . 1/4)))
```

C Show-Plane Helper Functions (Advanced Topics)

This appendix describes the internal (not exported) LISP functions used by Show-Plane and friends. These functions are intended for system programmers who are writing additional device drivers or addition Show-mumble type functions.

C.1 Introduction

In the process of writing Show-Plane, Show-Plane-Edge, and Show-Plane-Vector, I wrote a number of helper functions to handle much of the "grunge work" that Show-Plane (and friends) needed to do: things like filling out the device specification with defaults and checking for bogus values in the device specification. In all of these function the `device` argument is meant to be the (posibly modified) device specification defined for the *show-plane*, et. al. `:device` keyword argument.

C.2 Processing Defaults

These functions handle the processing of the device specification. This includes filling in default values and checking for illegal values. The source code for these functions is in file SHOW PLANE.LSP.

There are three constants and five functions. The main function (llvs::process-common-defaults) calls the other four functions, using the three constants where needed. For standard sorts of things, you only need to call llvs::process-common-defaults. If you need to do non-standard things or some sort of special processing you'll have to call the four lower level functions, possibly using the three constants. You should be sure to process the view-port before processing the pixel sizes.

---

llvs::default-view-port

DESCRIPTION

This constant is the normal default view-port. It is a view-port association list representing the whole "screen".

---

llvs::default-comment-location

DESCRIPTION

This constant is the default comment location. It is the two element list `(0 31/32)`.[26]

---

llvs::default-comment-size

DESCRIPTION

This constant is the default comment size. It is the two element list `(1/32 1/32)`.[27]

---

llvs::process-common-defaults

SYNOPSIS

llvs::process-common-defaults device plane limits

DESCRIPTION

This function processes all of the "common" defaults for Show-Plane. This includes: the view-port, the comment size and location, the pixel sizes, and the intensities (minimum and maximum). `Device` is the device keyword argument supplied by the user. Plane is the plane to be processed (used for fetching `:minval` and `:maxval` if needed). Note: if either `:minval` or `:maxval` is not on the plane's association list and either `:minimum-intensity` or `:maximum-intensity` are not on the device specification, this function will call *stats-plane*. Limits is the (fully expended) processing limits. It is used to compute the pixel sizes.

[26] This is 16 device pixels up from the bottom at the extreme left on a 512x512 display (such as the COMTAL or the Grinnells with the default view port).

[27] This is 16x16 pixels per character on a 512x512 display (such as the COMTAL or the Grinnells with the default view port).

RETURNS

A new device specification, with defaults appended onto the input argument device. Note: this device specification might be EQ to the input device specification if no processing was done.

llvs::process-default-view-port

SYNOPSIS

llvs::process-default-view-port device def-view-port

DESCRIPTION

This function processes the view-port specification in `device`. If there is no view-port or only a partial view-port, values are taken from def-view-port (the default view-port, usually `llvs::default-view-port`). If the view-port is illegal (larger than the screen or of zero or negative size), an error is generated (via *vis-error*).

RETURNS

A device specification with a legal, fully specified view-port. This object might be EQ to the device argument if the view-port that was specified was complete. If the view-port had to be added to, a new LISP list is created.

---

llvs::process-default-comments

SYNOPSIS

llvs::process-default-comments device def-comment-size def-comment-location

DESCRIPTION

This function processes the comment size and location specifications. `Device` is the device specification. `Def-comment-size` (usually `llvs::default-comment-size`) and `def-comment-location` (usually `llvs::default-comment-location`) are the default size and location. The

comment location is not checked for being inside the view-port - this is to allow placing comments outside of the view-port (desirable on the printer type devices) - this may cause bogus results with some devices (i.e. the grinnell). If the `:comments` keyword is not present in the device specification, no processing is done.

RETURNS

A device specification. It might be EQ to the device argument if the input device specification was complete or had no comments.

---

llvs::process-default-pixels

SYNOPSIS

llvs::process-default-pixels device limits

DESCRIPTION

This function computes and fills in the values for the pixel size and aspect ratio. It uses `limits` to determine the "size" of the virtual plane and the view-port specification (in `device`) to determine the "size" of the virtual device. (*Process-default-view-port* must have already been called to setup the view-port).

RETURNS

A device specification. It might be EQ if no processing was done.

---

llvs::process-default-intensities

SYNOPSIS

llvs::process-default-intensities device maxval minval

DESCRIPTION

This function process the minimum and maximum intensity values. `Maxval` and `minval` are the default values to use (usually fetched from the plane's association list (after running *stats-plane* if needed)).

RETURNS

A device specification. It might be EQ to the input specification if no processing was needed.

C.3 General Utilitity Functions

These functions perform various general-purpose operations needed by show-plane and related functions. The source code for these functions is in file SHOW PLANE.LSP.

---

llvs::compute-pixel-height

SYNOPSIS

llvs::compute-pixel-height device-vp pixel-size-ratio

DESCRIPTION

This function computes the pixel height in device coordinates. `Device-vp` is the view-port, in *device-dependent* coordinates, and `pixel-size-ratio` is the value of the `:pixel-size-ratio` association of the device specification. The result of this function is generally passed to the low level C routines to provide size information for use in graphics size computation (as the DY argument for a rectangle drawing function for example).

RETURNS

The pixel height as an integer.

llvs::compute-pixel-width

SYNOPSIS

llvs::compute-pixel-width device-vp pixel-size-ratio

DESCRIPTION

This function computes the pixel width in device coordinates. `Device-vp` is the view-port, in *device-dependent* coordinates, and `pixel-size-ratio` is the value of the `:pixel-size-ratio` association of the device specification. The result of this function is generally passed to the low level C routines to provide size information for use in graphics size computation (as the DX argument for a rectangle drawing function for example).

RETURNS

The pixel width as an integer.

llvs::compute-device-height

SYNOPSIS

llvs::compute-device-height device-vp

DESCRIPTION

This function computes the height of the device in device coordinates. `Device-vp` is the view-port, in *device-dependent* coordinates. This value can be passed to C code for use in graphics size computations. This function is called by *llvs::compute-pixel-height*.

RETURNS

The device height as an integer.

llvs::compute-device-width

SYNOPSIS

llvs::compute-device-width device-vp

DESCRIPTION

This function computes the width of the device in device coordinates. `Device-vp` is the view-port, in *device-dependent* coordinates. This value can be passed to C code for use in graphics size computations. This function is called by *llvs::compute-pixel-width*.

RETURNS

The device width as an integer.

C.4 Device Dependent Functions

These functions are usefull for writing compatable Show-Mumble type functions. The source code for these functions is in separate files separated by device type or class.

C.4.1 Grinnells

These functions are in the file SHPL GRINNELL.LSP.

---

llvs::process-grinnell-defaults

SYNOPSIS

llvs::process-grinnell-defaults grinnell-type device

DESCRIPTION

This function does the device-specific default processing for the grinnells. `Grinnell-type` is one of `:grinnell1`, `:grinnell2`, or `:grinnell3`. Note: only `:grinnell2` is fully implemented. The associations processed are `:color`, `:configuration`, and `:fast`. The defaults are: `:green` for `:color`, `:config080` (if `:color` equals `:green`) or `:config888`[28]/`:config444`[29] (if `:color` is not `:green`) for `:configuration`, and T for `:fast`.

RETURNS

A device specifivation. It might be EQ if no processing was done.

[28] If `:grinnell2` or `:grinnell3`.
[29] If `:grinnell1`.

llvs::show-plane-grinnell2-comments

SYNOPSIS

llvs::show-plane-grinnell2-comments comments location size color device-view-port

DESCRIPTION

This function displays comments on Grinnell2. `Comments` is a list of strings and/or symbols. `Location`, `size`, and `color` are the values of the `:comment-location`, `:comment-size`, and `:color` associations of the device specification. `Device-view-port` is the view-port in *device-dependent* coordinates. Note: does a *grinnell2:gr2_flush*.

RETURNS

T

---

llvs::convert-to-grinnell-view-port

SYNOPSIS

llvs::convert-to-grinnell-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in Grinnell coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The Grinnell coordinates are integers in the range 0 to 512 inclusive, with the origin at the lower left.

RETURNS

A device dependent view-port association list.

C.4.2 Comtal

These functions are in the file SHPL COMTAL.LSP.

llvs::process-comtal-defaults

SYNOPSIS

llvs::process-comtal-defaults device

DESCRIPTION

This function processes the defaults for the Comtal. The device associations processed are `:display-plane-type`, `:plane-mapping`, and `:fast`. The defaults are `:image` for `:display-plane-type`, 1 for `:plane-mapping`, and T for `:fast`.

RETURNS

A device specification. It might be EQ to the input device specification if no processing was done.

llvs::convert-to-comtal-view-port

SYNOPSIS

llvs::convert-to-comtal-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in Comtal coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The Comtal coordinates are integers in the range 0 to 512 inclusive, with the origin at the upper left.

RETURNS

A device dependent view-port association list.

C.4.3 IP8500

These functions are in the file SHPL IP8500.LSP.

llvs::process-ip8500-defaults

SYNOPSIS

llvs::process-ip8500-defaults device

DESCRIPTION

This function processes defaults for the IP8500. Associations processed are `:memory-channel-number` and `:fast`. The defaults are 0 for `:memory-channel-number` and T for `:fast`.

RETURNS

A device specification. It might be EQ to the passed in device specification if not processing was done.

---

llvs::show-plane-ip8500-comments

SYNOPSIS

llvs::show-plane-ip8500-comments comments location size memchan device-view-port

DESCRIPTION

This function displays the comments on the IP8500. `Comments` is a list of strings and/or symbols. `Location`, `size`, and `memchan` are the values of the `:comment-location`, `:comment-size`, and `:memory-channel-number` associations of the device specification. `Device-view-port` is the view-port in *device-dependent* coordinates.

RETURNS

NIL

llvs::convert-to-ip8500-view-port

SYNOPSIS

llvs::convert-to-ip8500-view-port view-port
extended-memory-?

DESCRIPTION

This function converts the device-independent view port to a view port in IP8500 coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The IP8500 coordinates are integers in the range 0 to either 512 or 1024 inclusive, with the origin at the lower left. The `extended-memory-?` flag specifies whether or not to use extended memory. Normal memory is 512x512 and extended memory is 1024x1024. `Extended-memory-?` is usually the value of the `:extended-memory` association of the device specification.

RETURNS

A device dependent view-port association list.

C.4.4 Plotter

These functions are in the file SHPL_PLOTTER.LSP.

---

llvs::show-plane-plotter-comments

SYNOPSIS

llvs::show-plane-plotter-comments comments location size
device-view-port

DESCRIPTION

This function plots the comments on the plotter. `Comments` is a

list of strings and/or symbols. `Location` and `size` are the values of the `:comment-location` and `:comment-size` associations of the device specification. `Device-view-port` is the view-port in *device-dependent* coordinates.

RETURNS

NIL

---

llvs::compute-plotter-view-port

SYNOPSIS

llvs::compute-plotter-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in plotter coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The plotter coordinates are integers in the range 0 to 3072 inclusive, with the origin at the lower left.

RETURNS

A device dependent view-port association list.

C.4.5 Printers, General

These functions are in the file SHPL PRINTERS.LSP.

---

llvs::process-printer-defaults

SYNOPSIS

llvs::process-printer-defaults device default-file default-queue default-form

DESCRIPTION

This function processes general printer defaults. Associations processed are: `:file-name`, `:notify`, `:delete`, `:queue`, `:form`, `:job-name`, and `:copies`. The defaults for `:file-name`, `:queue`, and `:form` are specified by the `default-file`, `default-queue`, and `default-form` arguments. The rest of the defaults are: name part of the `file-name` for `:job-name`, T for `:notify` and `:delete`, and 1 for `:copies`[30].

RETURNS

A device specification. It might be EQ to the device specification passed in, if no processing was done.

C.4.6 Symbolics Laser Printer

These functions are in the file SHPL_LASER.LSP.

---

llvs::show-plane-symb-lgp-comments

SYNOPSIS

llvs::show-plane-symb-lgp-comments comments location size device-view-port

DESCRIPTION

This function plots the comments on the Symbolics LGP. `Comments` is a list of strings and/or symbols. `Location` and `size` are the values of the `:comment-location` and `:comment-size` associations of the device specification. `Device-view-port` is the view-port in *device-dependent* coordinates.

[30]Some of these parameters are VMS dependent, specifically, `:notify` and `:form`.

RETURNS

NIL

---

llvs::compute-symbolics-lgp-view-port

SYNOPSIS

llvs::compute-symbolics-lgp-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in Symbolics LGP coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The Symbolics LGP coordinates are integers in the range 250 to 1850 in the X axis (left to right) and 600 to 2200 in the Y axis (bottom to top) inclusive, with the origin at the lower left.

RETURNS

A device dependent view-port association list.

C.4.7 LN03

These functions are in the file SHPL_LN03.LSP.

---

llvs::process-ln03-defaults

SYNOPSIS

llvs::process-ln03-defaults device

DESCRIPTION

This function processes the two LN03 specific device associations: `:line-width` and `:line-delta`.

RETURNS

A new device association with `:line-width` and `:line-delta` included. It might be EQ to the device specification passed in, if no processing was done.

---

llvs::show-plane-ln03-comments

SYNOPSIS

llvs::show-plane-ln03-comments comments location size device-view-port

DESCRIPTION

This function plots the comments on the LN03 LGP. `Comments` is a list of strings and/or symbols. `Location` and `size` are the values of the `:comment-location` and `:comment-size` associations of the device specification. `Device-view-port` is the view-port in *device-dependent* coordinates.

RETURNS

NIL

---

llvs::compute-ln03-view-port

SYNOPSIS

llvs::compute-ln03-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in LN03 LGP coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The LN03 LGP coordinates are integers in the range 0 to 2400 in the X axis (left to right) and 360 to 2760 in the Y axis (top to bottom) inclusive, with the origin at the upper left.

RETURNS

A device dependent view-port association list.

C.4.8 Printronix

These functions are in the file SHPL.PINTRONIX.LSP.

llvs::show-plane-printronix-comments

SYNOPSIS

llvs::show-plane-printronix-comments comments

DESCRIPTION

This function prints the comments on the Printronix printer. `Comments` is a list of strings and/or symbols. The comments are allways printed underneath the plot.

RETURNS

NIL

---

llvs::compute-printronix-view-port

SYNOPSIS

llvs::compute-printronix-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in Printronix printer coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The Printronix printer coordinates are integers in

the range 0 to 768 in the X axis (left to right) and 0 to 896 in the Y axis (top to bottom) inclusive, with the origin at the upper left. Note: the view-port is only used for size - the plot is allways plotted from the upper left corner of the page.

RETURNS

A device dependent view-port association list.

C.4.9 Terminals, General

These functions are in the file SHPL TERMINALS.LSP.

llvs::process-terminal-defaults

SYNOPSIS

llvs::process-terminal-defaults device

DESCRIPTION

This function processes general terminal defaults. Association processed are: `:terminal-type` and `:color`. The default value for `:terminal-type` is `:dumb`. The `:color` association is only processed if the terminal type specified is a color terminal (`:regis`, `:vt240`, or `:tektronix4107`). The default is `:white` for the `:regis` and `:vt240` types and `:black-and-white` for the `:tektronix4107`.

RETURNS

A device specification. It might be EQ to the device specification passed in, if no processing was done.

C.4.10 Dumb Terminal

These functions are in the file SHPL.DUMBTERM.LSP.

---

llvs::compute-dumb-terminal-view-port

SYNOPSIS

llvs::compute-dumb-terminal-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in dumb terminal coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The dumb terminal coordinates are integers in the range 0 to the terminal's width setting (set with the DCL command $ SET TERM/WIDTH www) in the X axis (left to right) and 0 to the terminal's page length (set with the DCL command $ SET TERM/PAGE ppp) in the Y axis (top to bottom) inclusive, with the origin at the upper left. Note: the view-port is only used for size - the plot is allways plotted from the upper left corner of the screen or page.

RETURNS

A device dependent view-port association list.

C.4.11 Regis, VT120, and VT240 Terminals

These functions are in the file SHPL REGISTERM.LSP.

llvs::compute-regis-terminal-view-port

SYNOPSIS

llvs::compute-regis-terminal-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in REGIS terminal coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The REGIS terminal coordinates are integers in the range 0 to 480 inclusive, with the origin at the upper left.

RETURNS

A device dependent view-port association list.

---

llvs::convert-to-regis-color-code

SYNOPSIS

llvs::convert-to-regis-color-code color

DESCRIPTION

This function returns the REGIS color code number that corresponds to the color name keyword passed in. The input keywords are `:blue`, `:red`, `:magenta`, `:green`, `:cyan`, `:yellow`, and `:white`. The result is in the range of 1 to 7 inclusive, with `:blue` mapping to 1, `:red` mapping to 2, etc.

RETURNS

An integer in the range of 1 to 7 inclusive.

---

llvs::show-plane-regis-erase-screen

SYNOPSIS

llvs::show-plane-regis-erase-screen

DESCRIPTION

This function clears the screen (graphics) on the REGIS terminal.

RETURNS

NIL

llvs::show-plane-regis-terminal-comments

SYNOPSIS

llvs::show-plane-regis-terminal-comments comments color location size device-view-port

DESCRIPTION

This function displays the comments on the REGIS terminal. `Comments`, `location`, and `size` are the values of the `:comments`, `:location`, and `:size` associations of the device specification. `Color` is a REGIS color-code, as an integer from 1 to 7 (i.e. the result of applying *llvs::convert-to-regis-color-code* to the value of the `:color` association of the device specification). `Device-view-port` is the view-port in *device-dependent* coordinates.

RETURNS

NIL

C.4.12 Tektronix 4107 and Related Terminals

These functions are in the file SHPL T4107TERM.LSP.

---

llvs::compute-tek4107-view-port

SYNOPSIS

llvs::compute-tek4107-view-port view-port

DESCRIPTION

This function converts the device-independent view port to a view port in Tektronix 4107 terminal coordinates. The `view-port` argument is the value of the `:view-port` association of the device specification and must be fully filled in. The device independent coordinates are rational numbers in the range of 0 to 1 inclusive, with the origin at the upper left. The Tektronix 4107 terminal coordinates are integers in the range 0 to 3072 inclusive, with the origin at the lower left.

RETURNS

A device dependent view-port association list.

---

llvs::show-plane-tek4107-comments

SYNOPSIS

llvs::show-plane-tek4107-comments comments location size device-view-port color

DESCRIPTION

This function displays comments on the Tektronix 4107 terminal. `Comments`, `location`, and `size` are the values of the `:comments`, `:location`, and `:size` associations of the device specification. `Device-view-port` is the view-port in device coordinates and `color` is one of the Tektronix 4107 color numbers legal for *tek4017:tek graphtext colour*.

RETURNS

NIL

C.5 VMS Directory Structure

The files described above are in a directory tree that can be accessed with these logical names: