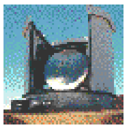  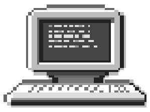 

Up: pgperl page

NAME

pgdisp - X plot display e.g. device \xdisp in PGPLOT

SYNOPSIS

pgdisp [options]

DESCRIPTION

pgdisp is a simplified version of the FIGDISP display server, written by Samuel N. Southard, Jr. for use with the FIGARO image-processing package. The following description is extracted from the FIGDISP manual.

pgdisp is used when setting device to \xdisp in *pgplot*

Use:

Before starting pgdisp, the X11 environment must be set up. The X11 environment consists of setting the DISPLAY environment variable appropriately. For example:

```
% setenv DISPLAY :0
```

or

```
% setenv DISPLAY malama:0.0
```

The first example would cause the display to appear on the local machine. Other values which can be used to accomplish this are "unix:0.0" and "lhost:0.0", where lhost is the name of the local host.

The second example would cause the display to appear on the machine "lo-fan," which could be the either local machine or another machine running an X11 server. Note that the remote machine can be any machine which is running an X11R4 server, not just a Sun. This method is know to work on DECstations.

See the X11 documentation for a more detailed description of DISPLAY. Multiple copies of figdisp/pgdisp may be running at any given time. The only limit is your machine's memory.

Once the display environment variable set, simply typing the line

```
% pgdisp [options] &
```

will start the display server. The options are optional and are discussed below.

If you wish to start up a second copy of pgdisp, type the line

% pgdisp -id # [options] &

represents any integer. (Note that there can not be a copy of figdisp and a copy of pgdisp running on the same screen with the same id.)

The pgdisp server was designed to address the needs of PGPLOT and supplied everything necessary to do so. It uses a default of 16 colors (2 if on a monochrome screen or it can't get 16 colors for some reason), but this can be changed with the lineColors resource. See the section on Color Allocation for a description of how this works. The only other feature provided is a cursor. This is sufficient to support all PGPLOT requirements.

The pgdisp window can be resized arbitrarily. The initial size is determined by the figdisp.lg.geometry resource. Since PGPLOT automatically scales to use the entire window, the pgdisp window should not be resized while a program is accessing it. If this is done, the display server will not crash, but the output will look odd.

The pgdisp window title is updated to show the position of the cursor when the cursor is in the pgdisp window.

Color Allocation

Pgdisp tries to avoid flashing when moving to and from its windows. At the same time, pgdisp tries to use as many colors as the user likes, and it may need to use a private color map to do so. When it is using a private color map, the correct colors will be displayed only when the cursor is in the figdisp window. For more information, see the FIGDISP manual.

The pgdisp window is cleared when pgdisp begins. The first 16 LUT table entries (or as many of them as possible, if the user requested fewer than 16 colors) are set as specified in table 2 (possible values range from 0 to 65535). If the line graphics window is using more than 16 colors, the colors above index 15 are initialized to a greyscale with data value 15 corresponding to black (for the purposes of the greyscale - data value 15 is still assigned as specified in table 2) and the maximum data value corresponding to white. If line graphics are attempted on a greyscale screen, all color components are set to the same value, specifically 0.30*red+ 0.59*green+0.11*blue, where red, green, and blue are the requested values.

Color Index	Red Component	Green Component	Blue Component
0	0	0	0
1	65535	65535	65535

2	65535	0	0
3	0	65535	0
4	0	0	65535
5	0	65535	65535
6	65535	0	65535
7	65535	65535	0
8	65535	32768	0
9	32768	65535	0
10	0	65535	32768
11	0	32768	65535
12	32768	0	65535
13	65535	0	32768
14	21845	21845	21845
15	43690	43690	43690

Table 2. Pgdisp line graphics initial color map entries.

Window Managers

Pgdisp has two aspects which cause its behavior to depend on the window manager. The first of these is how pgdisp decide which color map to use and has already been discussed. The second is the extensive use of the mouse buttons, which interacts with how the window manager determines input focus.

Most window managers have two methods of determining which window gets input focus: click-to-type, in which the user must click on the window he wants to get input, and follow-pointer, in which the window under the pointer gets input. If the user has specified click-to-type, before he can pan the image, manipulate the LUTs, or any other mouse controlled function, he must click on the figdisp window. If this click is done inside the figdisp window, instead of on the border, figdisp will also get this click and will processes it, which will result in the image being panned (if the left mouse button is used).

There are at least two ways around this. The first is to use follow-pointer focus. The way to do this varies from window manager to window manager. For the OpenLook Window Manager, the line

```
OpenWindows.SetInput:     followmouse
```

should be placed into the user's ~/.Xdefaults file or olwm should be started with the -f or -follow flag. For the Motif window manager, the line

```
Mwm*keyboardFocusPolicy:       pointer
```


accomplishes the same thing.

The second way to prevent unwanted panning in figdisp is by telling the window manager not to pass the select event (clicking on the window) to the program (figdisp). This is accomplished in the Motif window manager by adding the line

```
Mwm*passSelectButton:     false
```

to the user's ~/.Xdefaults file.

Note that for changes to the .Xdefaults file to take affect, the resource database must be updated and the window manager must be re-started. The easiest way to do this is to exit X windows and start it up again.

Customizing Figdisp

Figdisp allows the user to customize everything about how it interacts with the user. Table 3 contains the list of command line options and the corresponding X resources that figdisp recognizes. Every resource is prefaced by the name of the program (i.e. a listing of .display means the resource is figdisp.display if the program was started up as figdisp or and pgdisp.display if the program was started up as pgdisp). A detailed discussion of the effects of each of these resources has been given in preceeding sections.

Command Line Option	X resource	Default	Notes
-display disp	.display	none	The display on which the display server should run.

```
-id #               .id       0    The  id  number  of  this
copy of                                figdisp or pgdisp.  An
                              arbitrary number of copies  of
                              figdisp/pgdisp  may  be run at
                              the same time, as long as each
                              one's id number is unique.

-geometry      WxH[+x+y]           512x512   This       flag
corresponds to                              .bm.geometry in
figdisp  and                               .lg.geometry  in
pgdisp

-bmGeometry WxH[+x+y]    .bm.geometry   512x512   The bitmap
window geometry.

-lgGeometry  WxH[+x+y]    .lg.geometry   512x512   The  line
graphics window                               geometry.
```



```
-patchGeometry WxH[+x+y] .patchgeometry      The default  is
large  enough  for                               11 rows and
columns of the                              chosen font

-cmapGeometry WxH[+x+y]  .cmapgeometry  256x32    The  color
map window geometry.

-locationGeometry WxH[+x+y] .locationgeometry     The  loca-
tion              window              (shows             the
                        128x128   portion of the image vis-
able as                                compared to the entire
image)                                geometry.

-colors  #      .bm.maxcolors  225  The  initial  number  of
colors  -maxColors  #                      figdisp will try
to use in  the                               default  color
map for bitmap                              graphics.

-minColors #        .bm.mincolors  200  The  minimum  number
of  colors                            figdisp will use in
the  default                               color  map.   If
this  many  colors                               can not be
allocated, a private                              color map
will be used.

-privateColors   #    .bm.private.maxcolors    The   initial
number          of          colors          -maxPrivateColors
#          65536     figdisp  will  try   to   use   in   a
                              private  color  map.   This is
```

decreased to the maximum
number of colors supported on
the chosen display.

-minPrivateColors # .bm.private.mincolors The minimum
number of colors 170 figdisp will
try to use in a private color
map. If this many colors can
not be allocated figdisp will
give up.

-lineColors # .lineColors 16 The number of colors
to use for line graphics.

-visual .visual Any The visual to use.
Accepted values include the
X11 visuals PseudoColor and
GrayScale, as well as Default
(only the default visual is
allowed) and Any (any visual
is allowed) for either pgdisp
or figdisp. Pgdisp also
allows the X11 visual classes

StaticGray, StaticColor,
DirectColor, and TrueColor.

-help .showhelp True Whether or not an initial
help -nohelp screen should be
displayed.

Table 3. Pgdisp command line options and resources

If you are already familiar with X resources, Table 2, plus the meaning of all the resources, which can be found earlier in this document, is all you need to know. If not, continue reading.

Figdisp looks for its resource values in the following places (precedence is from top to bottom):

The command line

The XENVIRONMENT or ~/.hostname files

The server resource database (the xrdb command) or the ~/.Xdefaults file

The Figdisp or Pgdisp file in XAPPLRESDIR or /usr/lib/X11/app-defaults

The compile time defaults.

You should note that because of this precedence, the defaults specified by your system manager in the Figdisp or Pgdisp file will take priority over the compile-time defaults, so the listed default values may not agree with those on your system.

Customizing Pgdisp with Command-Line Options

The easiest way to experiment with figdisp's options is to use the command line. Doing this is as simple as looking up the command-line option which corresponds to the resource you want to specify, and then using it on the command line. For example, changing the key used to exit figdisp to "q" is as simple as

```
% figdisp -quit q &
```

Some figdisp command-line options do not require an argument. For example, let's say that in addition to changing the quit key, you wanted to disable the initial message figdisp produces referring you to this manual and to the help key. You would do this by the command


```
% figdisp -quit q -nohelp &
```

Now let's say that in addition to all of the above, you wanted to specify that the bitmap graphics window started out 256 pixels wide, 512 pixels tall, 100 pixels from the left edge of the screen, and 200 pixels from the top edge of the screen, the line graphics window to start out 512 pixels wide and 256 pixels high (but didn't care where on the screen it appeared), the font used for labels to be 5x8 (a very small font), PostScript output to go to the file ~/fdout.ps, the display server to try to allocate a maximum of 128 colors and a minimum of 100 colors in the default color map, and the display server to exit if it has to allocate a private color map. This would take the command

```
% figdisp -quit q -nohelp -bmGeometry 256x512+100+200
-lgGeometry 512x256 -font 5x8 -psFile ~/fdout.ps -colors 128
-minColors 100 -maxPrivateColors 1 -minPrivateColors 2 &
```

Since the minimum number of colors allowable in the private color map is less than the maximum, figdisp will never use a private color map. This technique can also be used to require figdisp to use a private color map.

Note that the options can come in any order and the flags can be abbreviated to the smallest unique string, so the last command could also be

```
% figdisp -minP 2 -maxP 1 -minC 100 -colo 128 -ps ~/fdout.ps
-fon 5x8 -lg 512x256 -bm 256x512+100+200 -noh -q q &
```

As was mentioned in Starting Figdisp, if you are using Figaro figdisp has been aliased so that you do not need to specify the "&" to start figdisp in the background.

Customizing Figdisp with your ~/.Xdefaults file

As you no doubt realized, typing in a long command line as in the final example can get to be very awkward. This is where the server resource database (which is usually read from the ~/.Xdefaults file using the command xrdb, usually located in ~/.xinitrc) comes in handy. The .Xdefaults file has the same format as all the other resource files listed (XENVIRONMENT, /usr/lib/X11/app-defaults/Figdisp, etc), so this section applies to them as well.

Assume that you had decided that the final example in the previous section was how you wanted figdisp to start up. You would then edit your ~/.Xdefaults file to include the lines:


```
figdisp.quit:              q  figdisp.showhelp:         False
figdisp.bm.geometry:                         256x512+100+200
figdisp.lg.geometry:                                 512x256
figdisp.font:            5x8
figdisp.psFile:              ~/fdout.ps
figdisp.bm.maxcolors:        128
figdisp.bm.mincolors:        100
figdisp.bm.private.maxcolors: 1
figdisp.bm.private.mincolors: 2
```

If your server has a server resource database, you would need to update it by either re-starting X windows or using the xrdb command. Once this was done, figdisp would automatically come up as if you had typed in all those

```
options.  You could still override your choices by  using  a
command line option.
```

Up: pgperl page

JAC unix support (helpme@jach.hawaii.edu)