The XPS-graphics library for X-windows and Postscript

Jos Thijssen
Version 0.1
May 25, 1999

This is a manual for a graphics library for real-time graphics with postscript and X windows. The library is still under development and users are invited to forward any remarks or bugs to `Jos.Thijssen@astro.cf.ac.uk`.

1 General information

People using X windows in a UNIX environment have several beautiful graphics packages and programs at their disposal. For drawing various shapes and forms, the Xfig program is useful, and for making graphs, gnuplot is the appropriate tool. Both programs are easy to use even for inexperienced computer users. For real-time graphics, various libraries are available, like Phigs and GKS. These libraries are extremely versatile and powerful, resulting in a high degree complexity which often discourages novice users.

The XPS-graphics package sacrifices the versality and power of these libraries to user-friendliness. Using it should be no problem for people knowing a bit of Fortran or C programming. The library contains a number of subroutines (functions) that can be called from a Fortran or C program. It provides graphics features like line-drawing, circles, text in various fonts and polygon filling. The last two features are lacking in gnuplot.

The merit of the library lies in its capability to produce real-time graphics, that is, data displayed on the screen by the same program that is producing them. This is useful when one wants to monitor the evolution of a structure and when one is debugging a program. Furthermore, the various fonts and the capability of filling polygons make it a possible alternative to gnuplot in some cases.

2 Description of the library

The XPS library has a restricted set of functions to draw lines, filled polygons and circles on the screen or on the paper. Moreover, it is possible to put text on the screen. All positions are given in *user coordinates*, coordinates defined by the user and which avoid the necessity of converting all positions to the physical screen or postscript coordinates.

In the future, the library will be extended to contain 3D graphics and to make graph drawing more easy.

Any suggestions are welcome.

3 Getting the library and installing it

All necessary files can be found in the directory `/home/orion/spxjmt` in the domain `astro.cf.ac.uk` (you can also send an e-mail to Jos.Thijssen@astro.cf.ac.uk). The directory contains the source files, makefiles, a postscript header file and this documentation (`doc.tex`).

To install the library, put all `.c` and `.h` files in some directory, together with the makefiles `make.lib` and `makefile` and the postscript header file `prelude`. Edit the makefiles to match your local system configuration. There are two makefiles, `make.lib` and `makefile`. `make.lib` is used to build the library file in the directory `XPSDIR`. Just type

```
make -f make.lib
```

and the relevant files will be compiled and stored in a library called libxps.a.

Before compilation and running the resulting executable, define an environment variable `XPSDIR` containing the path to the files `prelude` and `libxps.a`, e.g.

```
setenv XPSDIR /home/jost/XPS
```

To compile an application make sure that the `makefile` of the XPS package is in the same directory as your C or Fortran file and type

```
make myprog
```

for a C-program *myprog*.c or a FORTRAN program *myprog*.f exisiting in your directory. The executable result will be put in a file named *myprog*.

4 Description of functions

The names of the Fortran subroutines or C-functions are equal. A difference is of course that their names are case sensitive for C only.

The functions (subroutines) are now listed in groups according to their action and described. In the following we will not use the word "subroutines", but only "functions", since these are only known in C. Sometimes, the Fortran type CHARACTER** is used, meaning a character string type of arbitrary length, e.g. CHARACTER*64.

- Initplot

  Start the plot.

  `InitPlot(ColorName, width, height, PSFileName, OutPar)`

Parameter	C-Type	FORTRAN-Type	Meaning
ColorName	char *	CHARACTER**	name of the background-color
width	int	INTEGER	width of the plot
height	int	INTEGER	height of the plot
PSFileName	char *	CHARACTER**	name of postscript file
OutPar	int	INTEGER	output specification

  `InitPlot` is the function to be called before any other function of the XPS library. In fact, an error message will be displayed on the screen and the program will exit if other functions are called before `InitPlot`. The parameter *OutPar* must assume values 0, 1 or 2. *OutPar*=0 means that only postscript output will be sent to a file with a name specified by *PSFileName*. *OutPar*=1 means that graphics is produced on an X-window only. *OutPar*=2 means that output is sent to both the X-window and the postscript-file specified by *PSFileName*.

  The size of the plot (the *plotting area*) is determined by *width* and *height*. These are the sizes (in pixels) of an X-window. Choosing both parameters equal to 600 yields a square postscript plot on an A4 sheet of paper filling more or less the width of that paper.

Further reduction and squeezing can be done "by hand" in the postscript file, by changing the scale-parameters.

If output is sent to the X-window only, a postscript name still has to be provided but this is not used.

- Framing

  Specify user (world-) coordinates

  `Framing(Xll, Yll, Xur, Yur)`

Parameter	C-Type	FORTRAN-Type	Meaning
Xll	double	REAL*8	lower-left X-user-coordinate
Yll	double	REAL*8	lower-left Y-user-coordinate
Xur	double	REAL*8	upper-right X-user-coordinate
Yur	double	REAL*8	upper-right Y-user-coordinate

  `Framing` is the function to be called before starting the actual drawing. An error message will be displayed on the screen when an attempt is made to draw e.g. a line before `Framing` was called.

  `Framing` specifies user (or world) coordinates for further drawing. These coordinates specify an cartesian frame on the plot window in the users' own coordinates. *Xll*... *Yur* specify the lower left and upper right corner of the plotting area in user coordinates.

  All plotting routines use user-coordinates for arguments specifying positions.

- EndPlot

  Stop the program

  `Endplot()`

  If a stop-button is installed (see below), the program will wait until this button has been pressed. If no stop-button is installed, the program will terminate immediately.

- PutStartButton

  Put a start-button on the screen

  `PutStartButton()`

  At any time, a start-button can be put on the X-window screen. The execution is then suspended until the start button is clicked with the mouse. This may be helpful if drawing is going too fast and should be interrupted for a moment to inspect the drawing completed so far.

  This function has no effect on the postscript output.

- PutStopButton

  Put a stop button on the screen

  `PutStopButton()`

  At any time, a stop-button can be put on the X-window screen. The execution is then terminated as soon as the stop button is clicked with the mouse. Useful at the end of the program to enjoy the picture before exiting.

This function has no effect on the postscript output.

- Draw

  Draw a line form starting point to final point as specified by arguments

  `Draw(X1, Y1, X2, Y2)`

Parameter	C-Type	FORTRAN-Type	Meaning
X1	double	REAL*8	X-user-coordinate of starting point
Y1	double	REAL*8	Y-user-coordinate of starting point
X2	double	REAL*8	X-user-coordinate of final point
Y2	double	REAL*8	Y-user-coordinate of final point

  Draws a line from *(X1, Y1)* to *(X2, Y2)*. Clipping is performed on the border of the plotting area, i.e. the lines are cut off at their intersection with the border of the plotting area.

- DrawTo

  Draw a line from the last pen position to point specified by argument

  `DrawTo(X, Y)`

Parameter	C-Type	FORTRAN-Type	Meaning
X	double	REAL*8	X-user-coordinate of next point
Y	double	REAL*8	Y-user-coordinate of next point

  For functions involving line drawing and writing, the point to which the fictitious pen has moved after the last such action, is kept in memory. `DrawTo` draws a line form this last point to the point specified by argument.

  In postscript, a line drawn to the current position results in no drawing.

  Clipping is performed on the border of the plotting area.

- SetPoint

  Plot a point at *(X, Y)*

  `SetPoint(X, Y)`

Parameter	C-Type	FORTRAN-Type	Meaning
X	double	REAL*8	X-user-coordinate of point
Y	double	REAL*8	Y-user-coordinate of point

  Plots a point at the location specified by user coordinates *(X, Y)* if this point lies inside the plotting area. The last pen position is moved to this point, so a subsequent call to `DrawTo` will draw a line form the point *(X, Y)* to the point specified in the `DrawTo` call.

- DrawCircle

  Draws a circle

  `DrawCircle(X, Y, Radius)`

Parameter	C-Type	FORTRAN-Type	Meaning
X	double	REAL*8	X-user-coordinate of center
Y	double	REAL*8	Y-user-coordinate of center
Radius	double	REAL*8	Radius in user-coordinates

A circle around *(X,Y)* is drawn. The radius is in user coordinates used along the X-axis. The circle will always be drawn with two equal axes, even if the scale of the X and Y axes are not identical.

- DrawRectangle

  Draws a rectangle

  `DrawRectangle(X1, Y1, X2, Y2)`

Parameter	C-Type	FORTRAN-Type	Meaning
X1	double	REAL*8	X-user-coordinate of lower-left point
Y1	double	REAL*8	Y-user-coordinate of lower-left point
X2	double	REAL*8	X-user-coordinate of upper-right point
Y2	double	REAL*8	Y-user-coordinate of upper-right point

  A rectangle is drawn. Opposite points are specified by *(X1, Y1)* and *(X2,Y2)* resp.

- FillRectangle

  Draws a rectangle filled black

  `FillRectangle(X1, Y1, X2, Y2)`

Parameter	C-Type	FORTRAN-Type	Meaning
X1	double	REAL*8	X-user-coordinate of lower-left point
Y1	double	REAL*8	Y-user-coordinate of lower-left point
X2	double	REAL*8	X-user-coordinate of upper-right point
Y2	double	REAL*8	Y-user-coordinate of upper-right point

  A rectangle is drawn and filled black. Opposite points are specified by *(X1, Y1)* and *(X2,Y2)* resp.

- FillPolygon

  Draws a polygon filled black

  `FillPolygon(Points, NPoints)`

Parameter	C-Type	FORTRAN-Type	Meaning
Points	double[]	REAL*8 (npoints,2)	Array containing points
NPoints	int	REAL*8	number of points

  Draws a polygon through *Points* and fills it black. Points should in Fortran be a `REAL*8` array of dimension precisely *NPoints* $\times$ *2*. This array specifies the points defining the polygon to be filled. *NPoints* is the number of points of the polygon.

- SetLineStyle

  Set the current line style

  `SetLineStyle(Style)`

Parameter	C-Type	FORTRAN-Type	Meaning
Style	int	INTEGER	code for line style (see below)

A call to this function sets the new line style. All drawing after this call will be done in the new line style, until a new call to this function is performed. There exist four different predefined line styles. The default style is 0, corresponding to solid line style. 1 means dashed line style, 2 dotted, and 3 dashed-dotted line style. The size of the dashes and the distances between the dots are determined by the current dash length (see `SetDashLength`).

- SetDashLength

  Set the current dash length

  `SetDashLength(Length)`

Parameter	C-Type	FORTRAN-Type	Meaning
Length	int	INTEGER	new dash length

This function is used to set the dash length for drawing until another call to this function is made. In dashed line style, the dashes are twice the dash length and they are separated by white spaces of one dash length. The dots in dotted line style are separated by one dash length. In dash-dotted style, the dashes are two dash lengths long, and dashes and dots are separated by a dash length. `InitPlot` sets the dash length equal to 5 for X-windows. The dash length for postscript is automatically rescaled to an equivalent length.

- SetLineWidth

  Set the current line width

  `SetLineWidth(Width)`

Parameter	C-Type	FORTRAN-Type	Meaning
Width	int	INTEGER	new line width

This function is used to set the line width to a new value, *width*. The width is given in pixels, default is 1.

- SetNamedColor

  Set X-window color by specifying color name

  `SetNamedColor (ColorName)`

Parameter	C-Type	FORTRAN-Type	Meaning
ColorName	char *	CHARACTER **	color name

This function has no effect on the postscript output. For X-output, the color of subsequent drawing can be set by specifying the name of the desired color. Possible color names can be found in directory `/usr/local/X11R5/lib/X11` or a similar one, depending on the implementation of X. This method of specifying colors by their is very convenient, but slow.

- SetNamedBackground

  Set X-window color by specifying color name

  `SetNamedBackground (ColorName)`

Parameter	C-Type	FORTRAN-Type	Meaning
ColorName	char *	CHARACTER **	color name

  This function has no effect on the postscript output. For X-output, the color of the background drawing can be set by specifying the name of the desired background color. Possible color names can be found in directory `/usr/local/X11R5/lib/X11` or a similar one, depending on the implementation of X.

- MoveTo

  Move current position to point specified by argument

  `MoveTo(X, Y)`

Parameter	C-Type	FORTRAN-Type	Meaning
X	double	REAL*8	X-user-coordinate of point
Y	double	REAL*8	Y-user-coordinate of point

  Moves the current position to *(X, Y)*. Use of this function is to write text or numbers starting at a position specified by *(X, Y)*.

- SetFont

  Sets font style and size

  `SetFont(FontName, Weight, Orientation, Size)`

Parameter	C-Type	FORTRAN-Type	Meaning
FontName	char *	CHARACTER **	font name
Weight	char *	CHARACTER **	font weight
Orientation	char *	CHARACTER **	orientation
Size	int	INTEGER	font size

  SetFont sets the font style and size for writing text and numbers on the screen. The character-type arguments are case-insensitive.

  The *FontName* must be a string starting with "T" or "t" (for "Times"), "H" or "h" ("Helvetica), "C" or "c" ("Courier"), "N" or "n" ("NewCenturySchlbk") or "S" or "s" ("Symbol"). These are some of the most common adobe-fonts. It is believed that this choice will satisfy most of the needs.

  Weight is either a string starting with "m" (from "medium") or "b" ("bold"). Furthermore, *Orientation* a string starting with either "r" ("roman"), "i" ("italic") or "o" ("oblique"). Oblique may only be used with fonts Courier and Helvetica, italic only with Times and NewCenturySchlbk. Symbol can only have roman and medium for *Orientation* and *Weight* resp.

  Size is typically 12, 14, 18, 20 etc.

  This function may cause an error when a font cannot be found by the X-server. Furthermore, not all postscript printers have all these fonts at their disposal. Sometimes, a font is not found when using ghostscript to preview the plot, but it comes out of the printer correctly...

- WriteText

  Plots *Text* starting at the current pen position.

  `WriteText(Text)`

Parameter	C-Type	FORTRAN-Type	Meaning
Text	char *	CHARACTER**	text to be written

  Writes *Text* starting from the current position. Most often, a call to this function is preceded by a call to `MoveTo` or to another write-function.

  The left end of the plotted text starts at the *X*–coordinate of the current position. The *Y*–coordinate of the current position specifies the *mean* height of the written text, that is, not the bottom line.

  The current position becomes the right end of the text plotted.

- WriteInt

  Plot the *Number* using *CharNum* characters

  `WriteInt(Number, CharNum)`

Parameter	C-Type	FORTRAN-Type	Meaning
Number	int	INTEGER	number to be written
CharNum	int	INTEGER	number of characters used to write *Number*

  Writes *Number* using *CharNum* digits starting from the current position. Most often, a call to this function is preceded by a call to `MoveTo` or to another write-function.

  The left end of the plotted number starts at the current *X*–coordinate. The current *Y*–coordinate specifies the *mean* height of the written text, that is, not the bottom line.

  The current position becomes the right end of the text plotted.

- WriteFloat

  Write a real number

  `WriteFloat(Number, TotChar, DecChar)`

Parameter	C-Type	FORTRAN-Type	Meaning
Number	double	REAL*8	number to be written
CharNum	int	INTEGER	total number of characters used to write *Number*
DecChar	int	INTEGER	number of decimals used to write *Number*

  Writes *Number* using *TotChar* characters and *DecChar* decimals starting from the current position. Most often, a call to this function is preceded by a call to `MoveTo` or to another write-function.

  The left end of the plotted number starts at the current *X*–coordinate. The current *Y*–coordinate specifies the *mean* height of the written text, that is, not the bottom line.

  The current position becomes the right end of the text plotted.