TransFig: Portable Figures for LaTeX
Version 2.1.

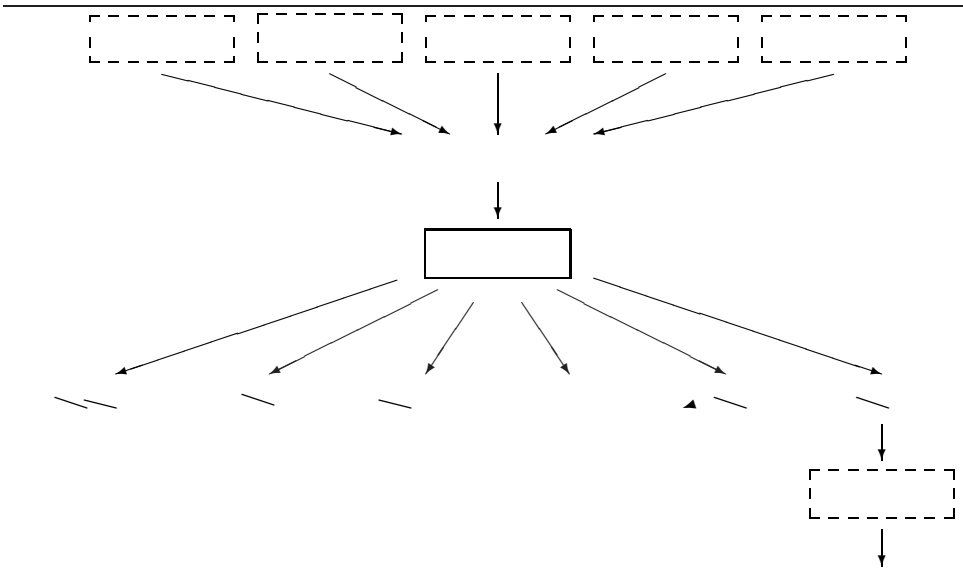
Micah Beck
Department of Computer Science
Ayers Hall, University of Tennessee
Knoxville, TN 3

TransFig is a mechanism for integrating LaTeX documents. Several graphics languages exist which achieve similar ends, but none is widely enough used to be called a standard. TransFig's goal is the portability of LaTeX documents across printers and operating environments. The mechanism in TransFig is Fig code, the graphics code of the Fig interactive graphics editor. TransFig provides an uniform way to translate Fig code into various graphics languages and to integrate that code into a LaTeX document.

1 TransFig

The TransFig package consists of the program, which translates between Fig code and other graphics languages, and the command which can be used to invoke it. The available translators are illustrated in Figure 1 which was included using TransFig. can be used directly to translate from Fig code to the various graphics languages. Each graphics language reuires the user to load a particular set of LaTeX macros and use particular commands to include the gure. TransFig allows these to be hidden.

When the graphics language is specied to the command, it creates a macro le and a make le. The macro le loads any LaTeX macros, and the make le species the appropriate commands to process with LaTeX to create les which load the gures. The user simply the macro le and the names of the les for loading the gures. To switch graphics languages, only to rerun , and then . Make can also be used to keep the translated code up to date when gures change.



```
\documentstyle{article}
\input{transfig}          TransFig macro file
\begin{document}
          ⋮
\begin{figure}
 \begin{center}
 \input{figurei}          i'th TransFig figure
   \end{center}
\end{figure}
          ⋮
```

Figure 2: Layout of a TransFig LaTeX document

1.1 File Name Conventions

Suppose that a document is to include a set of figures which are in TransFig code form. These should be in files with the .fig suffix, for instance figure1.fig, figure2.fig, ...figuren.fig. TransFig will create files figure1.tex, figure2.tex, ...figuren.tex for \input to the LaTeX document, and in some cases will create files with .ltx suffix. Additionally, TransFig creates a file transfig.tex which must be \input at the start of the document.

1.2 TransFig LaTeX Documents

In order to use TransFig, a LaTeX file must follow the format shown in Figure 2. The file transfig.tex must be \input before any TransFig figure is encountered. At the point where figure i is to be inserted, the file figurei.tex is \input.

1.3 Using TransFig

The transfig command has the form

```
transfig option... control... filename...
```

Where option is one of the following:

- -L language to translate into the specified language (default pictex).
- -M make file to name the output makefile (default Makefile).
- -T tex file to name the output LaTeX macro file (default transfig.tex).

The language specifier can be eepic, eepicemu, ibmgl, latex, pictex, ps, psfig, pstex, textyl and tpic, indicating translation into LaTeX commands, LaTeX picture environment macros, PostScript, textyl specials, tpic specials. See section 2 for further details about these special languages. The specifier box causes the figures to be replaced by empty boxes of the appropriate size.

A control specifier sets a parameter which governs the translation of a figure to its left in the argument list, until it is overridden. It must be one of the following:

- -m magnification to scale figures by magnification (default 1.0).
- -f font to set the default font family (default cmr).
- -s size to set the default font size (12 default magnification).

Each file name specifies a Fig file, either with or without the .fig suffix. TransFig creates a Makefile to apply fig2dev with the appropriate arguments to the named files, and creates an appropriate .tex file. Thus, to create a Makefile which will translate all figures in a LaTeX directory to LaTeX picture environment, with Computer Modern Bold as the font family, the command would be

```
transfig -L latex -f cmb *.fig
```

After running transfig, simply run make to create the appropriate LaTeX files. Make should be rerun whenever a Fig file is changed to recreate the corresponding LaTeX file. To change between graphics languages, simply run make clean to remove the files created by transfig, and then rerun transfig.

The transfig command can also be used to include figures described in Brian Kernighan's pic graphics language or in PostScript. These graphics formats are distinguished by the file name suffixes .pic and .ps respectively. Translation from PIC is accomplished by invoking the pic2fig program see section 2.3.

1. Text in Figures and Portability

In order to be translatable into different graphics languages, TransFig documents should use only those features which are supported by all of them. In particular, some graphics languages support a richer repertoire of text which is part of the figure than others. LaTeX, for example, allows full use of LaTeX commands in text strings, while PostScript does not.

The standard way to use text in TransFig figures is to include only straightforward text strings with no LaTeX commands. However, if a text object is flagged as special, it is understood to include formatting commands which are specific to the graphics language. A document with special text will not be portable to other output languages.

The standard font specifiers are a small set of generic fonts. However, if a text object is flagged as PS Font, then its font is specified as a PostScript font. When translating such text into PostScript, this font is used. However, translations into other graphics languages will usually require some approximation to the PS font. This approximation may be quite different from the named font.

2 Fig Code and Graphics Languages

TransFig's goal is to provide a framework for tools which maintains the portability of LaTeX documents across printers and operating environments. The central mechanism in TransFig is Fig code, a graphics format which is produced by the Fig interactive graphics editor. Fig code is widely used as an intermediate form for figures, the drivers of which may be attracted to produce compatible output. The reference in Appendix A describes Fig code in more detail.

2.1 Translations From Fig

TransFig currently translates Fig code into the graphics languages LaTeX picture environment, PiCTeX, PostScript, pic, and Textyl specials. The program which accomplishes these translations is fig2dev, which replaces the now-obsolete programs fig2tex, fig2ps, fig2latex, and fig2epic. The transfig command supports the translation of Fig code into tpic specials (see below) using pic, which is not part of the TransFig package. Each language may be appropriate in different environments or for different applications. A short description of each language is given below:

- PiCTeX is a set of TeX macros which implement simple graphics objects directly in TeX. PiCTeX makes no use of pre- or post-processors; the files it generates are completely standard, and can be printed or previewed in any environment where TeX is used. This result is achieved by using TeX arithmetic to do all plotting calculations, and by drawing the figure using the period character as a "brush". As a result PiCTeX is slow and reuires a large amount of TeX memory.
- PostScript is a powerful graphics language which is gaining acceptance as a standard. In an environment where PostScript is translated before being printed, it is usually possible to insert a separately generated PostScript file into a document using the TeX \special command. However, the resulting file can only be previewed using a PostScript previewer, and must be printed on a PostScript printer, such as the Apple LaserWriter. Various options are available for interworking with TeX. The psfig macro automatically scans the file for bounding

box information and generates appropriate TeX spacing and inclusion commands. One limitation of this output is the lack of LaTeX formatting for special objects such as text. The language specifier to the transfig command uses special fig2dev output drivers which separate the figure in text and non-text portions, rendering the former and the latter with LaTeX. The pstex option uses ps to generate the inclusion commands.

LaTeX picture environment is a restricted graphics facility implemented within LaTeX. It is a standard part of every LaTeX version, is processed quickly, and does not require a large internal TeX memory. However, not every graphics object which can be described with Fig code can be drawn using the LaTeX picture environment. Restrictions include a limited set of slopes at which lines can be drawn, and no ability to drawn splines.

EPIC is an enhanced version of the LaTeX picture environment which removes many restrictions. It uses no facilities outside of those needed for the LaTeX picture environment.

EEPIC is a further enhancement of epic which uses tpic specials to implement general graphics objects. It is also subject to the same software reuirements as tpic.

TeXtyl specials are a set of special commands which produce graphics instructions in the DVI file produced by LaTeX. The resulting DVI file must be postprocessed using Textyl to produce a PostScript program, which transforms it into a stack which uses its own line drawing fonts.

tpic specials are a set of special commands which produce graphics instructions in the DVI file produced by LaTeX. However, the graphics in the resulting DVI file can only be previewed or printed using software which understands these commands.

IBM-GL IBM-GL International Business Machines Graphics Language and HP-GL Hewlett-Packard Graphics Language are compatible languages which drive a variety of IBM and HP pen plotters.

When LaTeX processes the transfig.tex, it will print the message TransFig: figures in language indicating which graphics language is in use.

2.2 The Fig Graphics Editor

The interpretation of Fig code was originally developed by the Fig graphics editor and the program f2p, which translates Fig code into PostScript. The most recent version is V2.1 it is implemented by Version 2.1 of the Fig graphics editor, which runs under SunView, and by Version 2.1 of XFig.

Previous versions of Fig code which have been widely used are V2.0. The V2.1 format is in effect a unification of these two incompatible formats, and supersedes both of them. TransFig supports Fig code formats V1.3, V1.4, V2.0 as well as the latest format.

2.3 Other Fig Compatible Programs

The following programs use Fig code as a graphics description, and thus are compatible with TransFig:

- The numerical plotting program gnuplot can optionally produce output in Fig code format.
- The PIC-to-Fig translator pic2fig translates the language of Brian Kernighan's graphics preprocessor for Troff, Tbl and Eqn. This allows users to create figures without employing a graphics editor.

The Fig code produced by these programs can be viewed and edited with the Fig graphics editor.

3 Related Software

Software availability is subject to change, and the list may not be complete or up to date.

EPIC is an enhancement of TEX's picture environment which removes many restrictions. It uses only the facilities which TEX's picture environment provides. EPIC was developed by Sunil Podar at the State University of New York at Stonybrook, and is available via anonymous FTP from SUN.SOE.CLARKSON.EDU.

EEPIC is a further enhancement of EPIC which uses tpic specials (see below) to implement general graphics objects. It is a superset of the EPIC requirements (tpic, although there is an emulation package which will implement most of EEPIC using the same facilities as EPIC). EEPIC was developed by Conrad Kwok at the University of California, and is available via anonymous FTP from SUN.SOE.CLARKSON.EDU.

Fig is an interactive graphics editor in the style of MacDraw, which runs under the SunTools SunView windowing system. It produces its own output code which can be translated into a variety of graphics languages, including Postscript, and LaTeX's picture environment.

Fig was developed by Supoj Sutanthavibul at the University of Texas at Austin, supoj@SALLY.UTEXAS.EDU, and is available via anonymous FTP from SALLY.

Fig 2.1 is a version of Fig which implements various enhancements to the user interface, and uses Fig code V2.1. Fig 2.1 was developed by various contributors. It is available via anonymous FTP from FTP.CS.CORNELL.EDU.

Fig2dev translates from Fig code to LaTeX picture, PiCTeX, Postscript, LaTeX picture environment commands, EPIC macros, eTeX style, and the pic graphics language. It is part of the TransFig package, and supports Fig code V2.1.

Fig2tex, Fig2ps, Fig2latex, Fig2epic, Fig2pic are Fig code translation programs which were distributed as part of earlier versions of the TransFig package. They have been replaced by g2dev (see above).

F2p, F2ps are the original Fig code translation programs. These programs are out of date and have been subsumed by g2dev (see above).

GnuPlot is a numerical plotting program which can optionally produce Fig code format. GnuPlot was developed by a group of people, including Thomas Williams and Colin Kelly of Pixar Corp (INFO-GNUPLOT@SUN.COM), and David F. Kotz of Duke University (DFK@DUKE.CS.DUKE.EDU). GnuPlot is available via anonymous FTP from DUKE.

LaTeX is a standard macro package used for describing TeX documents. Part of this package is the LaTeX picture environment, a restricted graphics facility. The capabilities of this facility are described in sections of the LaTeX Document Preparation System by Leslie Lamport.

Pic2 g is a version of Brian Kernighan's graphics preprocessor for Troff. Pic2 g, which is a modified form of pic (see below), has been altered to produce Fig code.

PiCTeX is a set of macros for describing graphics in TeX. PiCTeX is implemented entirely within TeX, and requires no pre- or post processing programs or special fonts. The main problem in using PiCTeX is its slow operation all calculations are done using TeX's integer arithmetic and large memory reuirements. Many PiCTeX users have turned to C implementations of TeX in order to obtain memory sizes larger than are possible using the standard WebPascal version.

PiCTeX was developed by Michael Wichura at the University of Chicago (wichura@GALTON.UCHICAGO.EDU), and is available via anonymous FTP from A.CS.UIUC.EDU. It is also included as contributed software with the UniTeX distribution.

Plot2 g translates figures from the Unix plot file format to Fig code. Plot2 g was developed by Richard Murphy of Rice University (RICE.EDU), and is available via anonymous FTP from QED.RICE.EDU.

TeXtyl is a DVI file postprocessor which translates \special commands into its own set of drawing fonts. The result of this is a standard DVI file which can be printed using any DVI driver, as long as its drawing fonts are available to TeXtyl. It is available via anonymous FTP from VENUS.YCC.YALE.EDU.

tpic is a version of Brian Kernighan's graphics preprocessor for Troff. tpic has been altered to produce TeX \special commands which are understood by some print drivers and previewers. For information

about distribution topics, contact Tim Morgan of the University of California at Irvine, tmorgan@ICS.UCI.EDU.

TransFig was developed by Micah Beck with major contributions by Frank Schmuck, now of IBM, and Conrad Kwok of UC Davis. It is available via anonymous FTP from FTP.CS.CORNELL.EDU.

X g is a version of the Fig graphics editor which can be compiled for Suntools or X Windows Version 11 windowing systems. X g is part of the contributed software distributed with the X Window System and can be obtained by anonymous FTP from EXPO.LCS.MIT.EDU.

Xpic is a graphical editor similar to Fig which runs under X Version 11. Xpic was developed by Mark Moraes at the University of Toronto, moraes@csri.toronto.edu and is available via anonymous FTP from ai.toronto.edu.

A Fig Code V2.1 Reference Guide

PLEASE NOTE: This guide has not been updated to the current protocol. Please refer to the xfig document FORMAT3.0 for a complete description.

A Fig code version V2.1 file has the following structure:

```
#FIG 2.1
global parameters
object description
object description
    ⋮
```

A.1 Comment Lines

The very first line is a comment line containing the Fig version. Programs which interpret Fig code verify compatibility by checking the first line for this comment. All other lines which contain a # in the first column are treated as comments and are ignored.

A.2 Global Parameters

The first non-comment line consists of two global parameters:

```
fig_resolution coordinate_system
```

Fields in a line of a Fig file are separated by blanks or tabs. Blank lines are ignored. Fig files are free format. The fields of lines in Fig files are described in this guide by tables like the one below. The fields must appear in the order given in the table.

Type	Field	Units values
int	fig_resolution	pixels/inch Fig value: 0
int	coordinate_system	1: origin at lower left corner 2: origin at upper left corner Fig value: 2

The Type column specifies the type of the field, and is either float, integer or string. The notation following the type indicates units or ranges. Integer values of -1 are

interpreted as default values in this field. The rightmost column of this table either defines the units in which the field is expressed, or lists the values which the field can take. The notation DEFAULT in this column indicates that no value other than the default values are allowed. It is intended that future versions of Fig will define other values for these fields, but that these will remain legal, thus providing backward compatibility.

The basic unit of position in Fig files is the pixel. A Fig figure described at this resolution, the figure can be drawn on a higher resolution device. Pixels are square, and this resolution represents position resolution in both the x and y dimensions.

Some values are expressed as symbols and their numerical values are also listed. These symbols are defined in the header file object.h.

A.3 Object Descriptions

The rest of the file contains objects described in terms of six types, appearing on separate lines:

1. Ellipse.
2. Polyline, including Polygons and Boxes.
3. Spline, including Closed/Open Control/Interpolated Splines.
4. Text.
5. Circular Arc.
6. Compound object which is composed of one or more objects.

The following group of common fields appear in several of these objects, and so the are described here, and later are simply referred to as the common fields.

Type	Field	Units values	
int	line_style	SOLID_LINE	0
		DASH_LINE	1
		DOTTED_LINE	2
int	line_thickness	pixels	
int	color	DEFAULT	
		BLACK_COLOR	0
		BLUE_COLOR	1
		GREEN_COLOR	2
		CYAN_COLOR	3
		RED_COLOR	
		MAGENTA_COLOR	
		YELLOW_COLOR	
		WHITE_COLOR	
int	depth	no units	
int	pen	DEFAULT	
int	fill_style	DEFAULT	
		WHITE_FILL	1
		BLACK_FILL	21
oat	style_val	pixels	

For the dashed line style, the style val specifies the length of a dash. For dotted lines it indicates the gap between consecutive dots.

Depth determines which filled objects will obscure other objects, with the objects of greater depth being obscured. If two objects have the same depth that overlap, the object which occurs first in the file is obscured.

The values between WHITE_FILL and BLACK_FILL define a gray scale for many graphics languages cannot fully implement area fill.

The color field will be extended in the future to allow encoding of 24 bit byte RGB specifiers: one for line color and one for fill color.

Arrow lines are used to describe optional arrowheads for Arc, Polyline, and Spline objects. If an object has a forward arrow, then the line describing it follows the object description. If an object has a backward arrow

line describing it follows the object description, if there is one.

An arrow line consists of the following fields

Type	Field	Units values
int	arrow_type	DEFAULT
int	arrow_style	DEFAULT
float	arrow_thickness	DEFAULT
float	arrow_width	pixels
float	arrow_height	pixels

The pen field can only take the value DEFAULT. It is intended for extensions to Fig code will define other values for this field. Its intended use is to define the shape of the pen used in drawing objects. It will be the stipple pattern for line filling. The default pen will be black filling.

A.3.1 Ellipse Objects

Type	Field	Units values	
int	object_code	O_ELLIPSE	1
int	sub_type	T_ELLIPSE_BY_RAD	1
		T_ELLIPSE_BY_DIA	2
		T_CIRCLE_BY_RAD	3
		T_CIRCLE_BY_DIA	
	common fields		
int	direction	1	
float	angle	radians	
int	center_x,center_y	pixels	
int	radius_x,radius_y	pixels	
int	start_x,start_y	pixels	
int	end_x,end_y	pixels	

The Ellipse object describes an ellipse or a circle. The point center_x,center_y with radii radius_x and radius_y, and whose x-axis is rotated angle by from the horizontal. If the object describes a circle, then radius_x and radius_y must be equal.

The fields start_x, start_y, end_x and end_y are used only by Fig, and are not used in drawing the object. If the ellipse is specified by its radius, then

(startx, starty) is (centerx, centery), and (endx, endy) is a corner of a box which bounds the ellipse. If the ellipse is specified by d (startx, starty) and (endx, endy) are the two corners of the box which bound the ellipse.

A.3.2 Polyline Objects

Type	Field	Units values
int	object_code	O_POLYLINE 2
int	sub_type	T_POLYLINE 1 T_BOX 2 T_POLYGON 3 T_ARC_BOX T_EPS_BOX
	common fields	
int	radius	no units
int	forward_arrow, backward_arrow	0: no arrow 1: arrow

The Polyline object description has an additional points line following any arrow lines. The line consists of a sequence of coordinate pairs followed by the pair which marks the end of the line.

$x_1\ y_1\ x_2\ y_2 \ldots x_n\ y_n$

Type	Field	Units values
int	$x_i\ y_i$	pixels

The Polyline object describes a piecewise linear curve, starting at $x_1\ y_1$ and passing through each point $x_i\ y_i$ for i = 2 ... n. If sub_type is T_BOX or T_POLYGON then $x_1\ y_1$ and $x_n\ y_n$ must be identical. If sub_type is T_BOX, then the line segments must all be a vertically oriented rectangle. If sub_type is T_ARC_BOX, then the corners of the box are drawn with circular arcs, the size of which are determined by the radius field. Many output modes draw T_ARC_BOX object as simple boxes.

The T_EPS_BOX object is a simple box filled with a figure described by an imported Encapsulated PostScript (EPS) file. Following any arrow lines which are ignored is an EPS specification line, consisting of a flag indicating the vertical orientation of the figure, and the name of the EPS file. Following that is a points line describing the two opposite corners of the figure.

Type	Field	Units values
int	flipped	0: normal orientation 1: flipped
string	filename	

A.3.3 Spline Objects

Type	Field	Units values	
int	object_code	O_SPLINE	3
int	sub_type	T_OPEN_NORMAL	0
		T_CLOSED_NORMAL	1
		T_OPEN_INTERPOLATED	2
		T_CLOSED_INTERPOLATED	3
common fields			
int	forward_arrow, backward_arrow	0: no arrow 1: arrow	

The Spline object description has the following any arrow line which has the same format as described above for the Polyline object. If the sub_type of the spline is T_OPEN_INTERPOLATED or T_CLOSED_INTERPOLATED, then an additional control points follows the points line. The line consists of a seuence of x, y coordinate pairs for each point in the points line. Note that whereas the points line contains integers, the control line consists of floating point number pairs

$$lx_1\ ly_1\ rx_1\ ry_1\ lx_2\ ly_2\ rx_2\ ry_2\ \ldots\ lx_n\ ly_n\ rx_n\ ry_n$$

Type	Field	Units values
float	$lx_i\ ly_i\ rx_i\ ry_i$	pixels

The interpretation of Spline objects is more complex than descriptions, and is discussed in section A..

A.3. Text Objects

Type	Field	Units values
int	object_type	O_TEXT
int	sub_type	T_LEFT_JUSTIFIED 0 T_CENTER_JUSTIFIED 1 T_RIGHT_JUSTIFIED 2
int	font	DEFAULT ROMAN 1 BOLD 2 ITALICS 3 SANSSERIF TYPEWRITER
oat	font_size	points
int	pen	DEFAULT
int	color	DEFAULT
int	depth	no units
oat	angle	radians
int	text_flags	no units
oat	height, length	pixels
int	x, y	pixels
string	string	

The positioning of the string is specified by the sub_type value. The values T_LEFT_JUSTIFIED, T_CENTER_JUSTIFIED, and T_RIGHT_JUSTIFIED specify that (x, y) is the left end, center and right end of the baseline. The height and length fields specify the box that the text fits into. These are accurate only for the fonts used by Fig.

The string field is an ASCII string terminated by the '\0' character. This terminating character is not a part of the string. None may contain the new-line character '\n'. Some output modes will interpret ISO encoded European accents not found in the ASCII character set.

The text_flags field is a bit vector which specifies various settable properties of the text object. Each flag corresponds to a bit position in the field. The default value of each flag is FALSE.

Flag	Bit mask in binary	Description
RIGID_TEXT	0001	Font size doesn't scale inside compound
SPECIAL_TEXT	0010	Text includes formatting commands
PSFONT_TEXT	0100	Font field specifies PS font
HIDDEN_TEXT	1000	Fig editor should not display text

The RIGID_TEXT flag is used to preserve the absolute size of text objects when inside compound and that compound is scaled by the SPECIAL_TEXT flag is used to inhibit the escaping of formatting commands when translating text to LaTeX or Troff, in order to allow the user to inject such commands directly into the figure. The PSFONT_TEXT flag changes the interpretation of the font field. Rather than selecting from the limited choices shown in the table above, the field is interpreted as selecting from the full set of PostScript fonts. A text object with the PSFONT_TEXT flag set may not be fully translatable into output forms other than PostScript. The HIDDEN_TEXT flag is meaningful only to graphics editors, and specifies that the text should not be displayed on the screen. This is most useful for special text objects which contain very long formatting command strings.

PS Font	Value
Times-Roman	1
Times-Italic	2
Times-Bold	3
Times-BoldItalic	4
AvantGarde	5
AvantGarde-BookOblique	6
AvantGarde-Demi	7
AvantGarde-DemiOblique	8
Bookman-Light	9
Bookman-LightItalic	10
Bookman-Demi	11
Bookman-DemiItalic	12
Courier	13
Courier-Oblique	14
Courier-Bold	15
Courier-BoldItalic	16
Helvetica	17
Helvetica-Oblique	18
Helvetica-Bold	19
Helvetica-BoldOblique	20
Helvetica-Narrow	21
Helvetica-Narrow-Oblique	22
Helvetica-Narrow-Bold	23
Helvetica-Narrow-BoldOblique	24
NewCenturySchlbk-Roman	25
NewCenturySchlbk-Italic	26
NewCenturySchlbk-Bold	27
NewCenturySchlbk-BoldItalic	28
Palatino-Roman	29
Palatino-Italic	30
Palatino-Bold	31
Palatino-BoldItalic	32
Symbol	33
ZapfChancery-MediumItalic	34
ZapfDingbats	35

A.3. Arc Objects

Type	Field	Units values
int	object_code	O_ARC
int	sub_type	T_3_POINT_ARC 1
	common fields	
int	direction	0: clockwise 1: counter
int	forward_arrow, backward_arrow	0: no arrow 1: arrow
float	center_x,center_y	pixels
int	x_1 y_1 x_2 y_2 x_3 y_3	pixels

The Arc object describes a circular arc centered at the point center$_x$ center$_y$, starting at x_1 y_1, passing through x_2 y_2, and ending at x_3 y_3. It is drawn either clockwise of counter-clockwise as specified by direction. Note that this description is quite overdetermined, as the center and direction of the arc can be deduced from the coordinates of the three points which are specified.

A.3. Compound Objects

Type	Field	Units values
int	object_type	O_COMPOUND
int	upperright_corner_x	pixels
int	upperright_corner_y	
int	lowerleft_corner_x	
int	lowerleft_corner_y	

The Compound object description describes a compound object bounded by the rectangle determined by the points

upperright_corner_x upperright_corner_y
lowerleft_corner_x lowerleft_corner_y

It consists of all the objects following it until an object of type END_COMPOUND is encountered. Compound objects may be nested.

A. Splines

Specifying the interpretation of a Spline object is more problematic than other graphics objects. A graphics object can be viewed as having two parts: an abstract description of the object and a set of appearance parameters which specify how the object is to be represented. For example, a circular arc has a well understood abstract definition, independent of the width of the line used to draw it. Unfortunately, the abstract specification of splines is not. The following descriptions come at second hand the author of this is not versed in spline algorithms, and so may have garbled them. Hopefully, they will give the knowledgeable reader some idea of the intended meaning of Spline objects.

Fig splines come in two major varieties: B-splines and Interpolated. Each of these is available in open or closed subtypes. If the sub_type field has the value T_OPEN_NORMAL or T_CLOSED_NORMAL then it describes a B-spline. In these cases, the points line which follows contains the control points for the spline. The spline does not actually pass through these points, but they determine where it will pass, which is generally near the control points. B-splines are quite smooth.

If the sub_type field has the value T_OPEN_INTERPOLATED or T_CLOSED_INTERPOLATED then it describes an interpolated spline. In these cases, the points line which follows contains the points through which the spline will pass. In addition, each point follows the points line, which specifies two control points, lx_i ly_i and rx_i ry_i for each interpolation point. The ith section of the interpolated spline is drawn using the Bezier with the four points x_i y_i, rx_i ry_i, lx_{i+1} ly_{i+1}, and x_{i+1} y_{i+1}. Interpolated splines are not as smooth as B-splines.

For either type of closed splines, the first and last points on x_1 y_1 and x_n y_n are identical. For closed interpolated splines, the last pair of control points on the control point line, lx_n ly_n and rx_n ry_n are the same as lx_1 ly_1 and rx_1 ry_1 respectively.