TransFig: Portable Figures for LaTeX
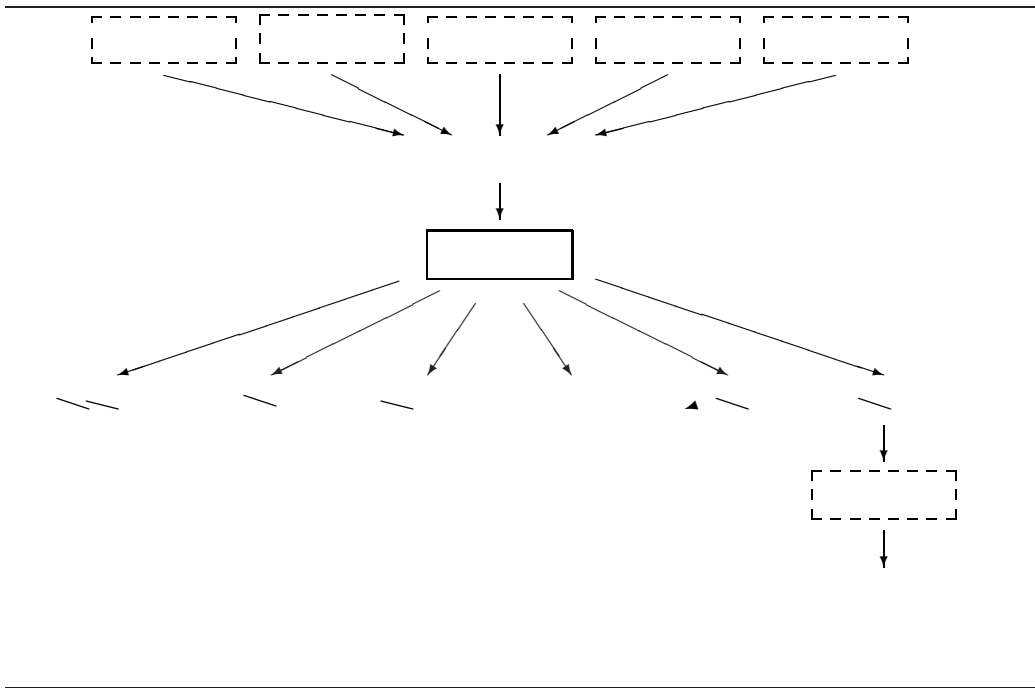# Version 2.1.

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

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

1 TransFig

The TransFig package consists of the fig2dev program, which translates between Fig code and other graphics languages, and the transfig command which can be used to invoke it. The available translations are illustrated in Figure 1 which was included using TransFig. Fig2dev can be used directly to translate from Fig code to the various graphics languages. However, each graphics language requires the user to load a particular set of TeX particular commands to include the figure. TransFig allows these to be hidden.

When the graphics language is specified to the transfig command, it creates a macro file and a make file. The macro file loads a set of TeX macros, and the make file specifies the appropriate commands to create which load the figures. The user simply input the macro file and the names of the files for loading the figures. To switch graphics languages, it is necessary only to rerun transfig, and then make. Make can also be used to keep the translated code up to date when figures change.

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

Figure 2: Layout of a TransFig document

1.1 File Name Conventions

Suppose that a document is to include a set of gures which are stored in Fig code form. These should be in les with the name gure x, for instance figure1.fig, figure2.fig,...figuren.fig. TransFig will create le figure1.tex, figure2.tex,...figuren.tex for \input to the LaTeX document, and in some cases will create les with other suxi es. Additionally, TransFig creates a le transfig.tex which must be \input at the start of the document.

1.2 Trans Fig LaTeX Documents

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

1.3 Using TransFig

The transfig command has the form

        transfig option... control... lename...

Where option is one of the following:

- -L language to translate into the language specified; default pictex.
- -M make le to name the output make le; default Makefile.
- -T tex le to name the output TeX macro le; default transfig.tex.

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

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

- -m magnification to scale gures by magnification; default 1.0.
- -f font to set the default font family; default
- -s size to set the default font size; default 11 magnification

Each le name specifies a Fig le, either with or without the .fig sufx. TransFig creates a Makefile to apply fig2dev with the appropriate arguments to the named les, and creates an appropriate transfig.tex le. Thus, to create a Makefile which will translate all gures in a 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 les. Make should be rerun whenever a Fig le is changed to recreate the corresponding LaTeX le. To change between graphics languages, simply run make clean to remove the les 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 names .pic and .xps respectively. Translation from PIC is accomplished by invoking pic2fig program see section 2.3.

1. Text in Figures and Portability

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

The standard way to use text in TransFig figures is as straight text with no LaTeX commands. However, if a text object is flagged as special, it is understood to include formatting commands which are interpreted by the graphics language. A document with special text will not be portable to all 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 specifier is a PostScript font. When translating such text into PostScript, this font name is used. However, translations into other graphics languages will make some approximate of the PS font This approximation may be quite different from the name

2 Fig Code and Graphics Languages

TransFig's goal is to provide a framework for tools which maintain the portability of TeX 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. It is widely used as an intermediate form for figures, hence other graphics tools 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 three graphics languages, the LaTeX picture environment, PiCTeX, PostScript, and LaTeX tpic 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 tpic, 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 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 requires a large amount of TeX memory.

   PostScript is a powerful graphics language which is gaining acceptance as a standard. In an environment where dvi code 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 previewer, and must be printed on a printer, such as the Apple LaserWriter.

   Various options are available for integration with TeX. The psfig macro automatically scans the file for bounding

box information and generates appropriate TeX spacing and i
clusion commands. One limitation of its output is the lack of
L&TeX formatting for special objects text. The language speci-
er to the transfig command uses special fig2dev output
drivers which separate the gure in text and non-text portio
rendering the former and the latter in TeX. The epstex
option uses gs to generate the inclusion commands.

L&TeX picture environment is a restricted graphics facility impl
within TeX. It is a standard part of every version of L
is processed uickly, and does not reuire a large internal T
memory. However, not every graphics object which can be de-
scribed with Fig code can be drawn using the L&TeX picture en-
vironment. Restrictions include a limited set of slopes at w
lines can be drawn, and no ability to drawn splines.

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

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

TeXtyl specials are a set of special commands which produce
graphics instructions in the overlay produced by TeX. The re-
sulting I le must be postprocessed using the TeXtyl pro-
gram, which transforms it into a standard which uses
its own line drawing fonts.

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

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

When L&TeX 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 defined by the graphics editor and the program f2p, which translates Fig code into graphics language. The most recent version is V2.1 it is implemented by Version Fig graphics editor, which runs under SunView, and by XFig Version 2.

Previous versions of Fig code which have been widely used are V2.0. The V2.1 format is in effect a unification of these feature compatible formats, and superseds both of them. TransFig supporting Fig code formats V1.3, V1., V these formats. as well as the older formats 3.a

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, into Fig code. This allows users to create figures without employing a graphics editor.

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

3   Related Software

Software availability is subject to change, and this list may not be completely up to date.

EPIC is an enhancement of the LaTeX picture environment which removes many restrictions. It uses only the facilities which LaTeX implement the environment. 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 subject to the same software reuirements as tpic, although there is an emulation package which will implement most of EPIC using the same facilities. EEPIC was developed by Conrad Kwok at the University of California at Davis, 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 code in order which can be translated into a variety of graphics languages including Postscript, and LaTeX PiCT

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 PiCTs, Postscript, the LaTeX picture environment commands, PIC macros, Xtyl, and the 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, in
   Thomas Williams and Colin Kelly of Pixar corp INFO-GNUPLOT@SUN.COM,
   and David F. Kotz of Duke Univ dfk@DUKE.CS.DUKE.EDU. Gnu-
   plot is available via anon FTP from DUKE.

LaTeX is a standard macro package used for describing TeX documents in
   of this package is the X picture environment, a restricted graphics facil-
   ity. The capabilities of this facility are codes of TeX, in sect
   Document Preparation System by Leslie Lamport.

Pic2 g is a version of Brian Kernighan graphics preprocessor for Troff.
   Pic2 g, which is a modi ed from c see below, has been altered to
   produce Fig code.

PiCTeX is a set of macros for describing graphic documents. PiCT
   is implemented entirely within X standard and requires no pre- or post-
   processing programs or special fonts. The main problem in us
   is its slow operation all calculations are X's done using arith-
   metic and large memory reuirements. Many BeCT have turned to
   C implementations of X in T order to obtain memory sizes larger than are
   possible using the standard WebPascal version.

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

Plot2 g translates gures from the Unix plot le format to Fig code. P
   developed by Richard Murphy of Rice Univ rnm@RICE.EDU, and
   is available via anon FTP from QED.RICE.EDU.

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

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

about distribution, or contact Tim Morgan of the University of California at Irvine, morgan@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.

Xfig is a version of the Fig graphics editor which can be compiled for Suntools or X Windows Version 11 windowing systems. Xfig 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 Window 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 3.col. Please refer to the xfig document FORMAT3.0 for a complete des

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 format. 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. Lines terminate object descriptions. The fields of lines in Fig files are described in this guide by tables like the one below. The fields must appear given in the table.

Type	Field	Units values
int	fig_resolution	pixelsinch 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 char, integer or string. The notation following the type indicates that 0 at-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. While figures are described at this resolution, the figure can be drawn at a lower or higher resolution. Pixels are square, and so 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 object.h file.

A.3 Object Descriptions

The rest of the file contains objects descriptions of six types, depending on

1. Ellipse.

2. Polyline, including Polygons and Boxes.

3. Spline, including Closed/Open Control/Interpolated Sp

. Text.

. Circular Arc.

. Compound object which is composed of one or more objects.

The following group of common fields appear in several object descriptions, and so the are described here, and later are simply referred to by the indicator 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 speci es the length of a dash. For dotted lines it indicates the gap between consecutive do

Depth determines which lled objects will obscure other lled objects, with the objects of greater depth being obscured. If two objects that th same depth overlap, the object which occurs rst in the le is obscur Edg

The values between WHITE_FILL and BLACK_FILL de ne a gray scale many graphics languages cannot fully implement area ll.

The color eld will be extended in the future to allow encoding of two byte RGB speci ers: one for line color and one for ll color.

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

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

An arrow line consists of the following elds

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

The pen eld can only take the value DEFAULT. It is intended that future extensions to Fig code will de ne other values for this eld. Its intended use is to de ne the shape of the pen used in drawing objects. It includes the stipple pattern for line lling. The default pen is a black circular pen.

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	

<div align="center">common elds</div>

int	direction	1
oat	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 whose center is at center_x,center_y with radii radius_x and radius_y, and whose x-axis is rotated by angle from the horizontal. If the object describes a circle, then radius_x and radius_y must be eual.

The elds 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 speci ed by 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, then 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	

<div align="center">common elds</div>

Type	Field	Units values
int	radius	no units
int	forward_arrow,	0: no arrow
	backward_arrow	1: arrow

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

x_1 y_1 x_2 y_2 $:$ $:$ x_n y_n

Type	Field	Units values
int	x_i y_i	pixels

The Polyline object describes a piecewise ling ra the point x_1 y_1 and passing through each x_i point 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 sub_type is T_BOX, then the line segments must all be a vertically oriented rec 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. Many output modes draw T_ARC_BOX object as simple boxes.

The T_EPS_BOX object is a simple box lled with a gure described by an imported Encapsulated PostScript EPS le. Following any which are ignored is an EPS speci cation, consisting of a ag indicating the vertical orientation of the gure, and the name of the import EPS lowing that is a points line describing the two opposite corners of t

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 a points line 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 line follows the points line. The line consists of a seuence of control coordinate pairs for each point in the points line. Note that where as the points line contains integers, the control line consists of ﬂoating point number

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

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

The interpretation of Spline objects is more complex than other object 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 speci ed by the sub_type. 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, respectively. The height and length elds specify the box that the text ts into. These are speci cat accurate only for the fonts used by Fig.

The string eld is an ASCII string terminated by the character '\0'. The terminating character is not a part of the string. No string 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 eld is a bit vector which speci es various settable properties of the text object. Each ag corresponds to a bit in the plot, the default value of each 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. The SPECIAL_TEXT flag is used to inhibit the escaping of formatting commands when translating text to TeX 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 set shown in the table above, the field is interpreted as selecting from the table of PostScript fonts. A text object with the PSFONT_TEXT flag set may not be fully translatable into output forms other than PostScript. Finally, the HIDDEN_TEXT field 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 may include 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 elds		
int	direction	0: clockwise
		1: counter
int	forward_arrow,	0: no arrow
	backward_arrow	1: arrow
oat	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 speci ed by direction. Note that this description is uite overdetermined, as the center and direction of the arc can be deduced from the three points which are speci ed.

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 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 whose type is O_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 which make up the object and a set of appearance parameters which specify how that object is to be represented. For example, a circular arc has a well understood abstract de nition, independent of the we wish to draw it. Unfortunately, the abstract speci cation of splines. The following descriptions come at second hand the author is not versed in spline algorithms, and so may have garbled them. Hopefully, the knowledgeable reader some idea of the intended meaning of Sp

Fig splines come in two major varieties: B-splines and Inter. Each of these is available in open or closed subtype. eld has the value T OPEN NORMAL or T CLOSED NORMAL then it describes a B-spline. In these cases, the points line which follow control points the for the spline. The spline does not actually pass through, but they determine where it will pass, which is generally the control points. B-splines are uite smooth.

If the sub type eld has the value T OPEN INTERPOLATED or T CLOSED INTERPOLATED then it describes an interpolated spline. In these cases, the points line which follows interpolation points through which the spline will pass. In addition, a line follows the points line, which speci es 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 Bez with the four points x_i y_i , rx_i rx_i , lx_{i1} ly_{i1} , and x_{i1} y_{i1} . Interpolated splines are not as smooth as B-splines.

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