The PDF Panel

Nelson H. F. Beebe
Center for Scientific Computing
University of Utah
Department of Mathematics, 322 INSCC
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
Email: beebe@math.utah.edu, beebe@acm.org, beebe@computer.org, beebe@ieee.org (Internet)
URL: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 585 1640, +1 801 581 4148

Introduction

The TUG'2001 Portable Document Format (PDF) Panel convened on Tuesday, August 14, 2001, with members Nelson H. F. Beebe, Hans Hagen (chair), Martin Schröder, Don Story, and Hàn Thế Thành, with many comments and questions from the audience.

The list of topics that was projected on the screen makes up the sectional headings in what follows.

Any errors or omissions in this article are solely the fault of this reporter. The text is based on cryptic notes that I typed into a computer file immediately after the panel session, but has been expanded into English, with literature and Web references.

PDF design goals

Adobe Systems defined the Portable Document Format (PDF) in a specification published in 1993 [7], and updated in 2000 [3] and 2001 [4]. Adobe maintains a developer's Web site for PDF at http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html where technical notes and electronic versions of the specifications may be found.

The PDF language shares much of the syntax of PostScript, but unlike PostScript, is not a true programming language. Several technological goals influenced the development of PDF:

- The data format must be identical on all operating system platforms.
- The data format must be public.
- A compact page description is desirable, to reduce data storage and transfer costs. While later versions of PostScript defined additional compression algorithms, their use has primarily been for compressing bitmaps, and programmers have to work quite hard to produce compact PostScript. [Tom Rokicki's dvips and my dvialw have both taken this about as far as is feasible: both produce much more compact output than the majority of PostScript-producing packages.]
- Page independence is needed to support high-speed parallel printing, to speed up screen display, and to permit network transmission of document fragments.
- Random access to individual page descriptions is needed to speed processing.
- Single-pass file generation is required.
- It should be possible to incrementally update page description files, while retaining the complete original contents. Examples include addition of reader comments in overlay notes, replacement of low-resolution figures by high-resolution ones, and repair of minor typographical errors.
- A simplified, nonextensible, page-description language permits simpler and faster implementations of page rendering software.
- Font embedding promotes document portability.
- Font subsetting may reduce file size, and, for some vendors, solves the font copyright issue. For more information about this, see the TEX *Font Panel* article in these proceedings.
- The format should be extensible, so that new object types (e.g., audio and video) can be added in the future, without invalidating software that recognizes only older formats.
- It should be possible to encrypt file contents, and prevent (or at least, strongly discourage) certain actions, such as data extraction and printing.

These goals have all been met in Adobe's implementations of PDF processing software.

Interestingly, TEX DVI files, defined fifteen years earlier, fulfill all of these requirements, except for font embedding, encryption, and incremental updating.

PDF advantages

Publishers and print shops like PDF, because such files are less troublesome to deal with than PostScript files often are. Numerous magazines and newspapers are now printed locally from master PDF files shipped electronically, saving the significant expense and delay of long-distance transportation of printed matter.

Some printer vendors exploit page independence to achieve very high performance: IBM has a PDF printer with 24 CPUs simultaneously rendering PDF page images to print at more than 400 pages/minute.

At least one PDF file viewer is freely available for each of the major platforms, including a hand-held Personal Digital Assistant (PDA), so the vast majority of computer users can view PDF files without cost. Besides Adobe's free Acrobat Reader, and their commercial Capture, Catalog, Distiller, InDesign, Photoshop, and Illustrator tools, there are Ghostscript (`http://sourceforge.net/projects/ghostscript/`), Ghostview, gv, and xpdf (`http://www.foolabs.com/xpdf/`) for viewing and printing, pdf2ps for printing, pdftotext for extracting raw text, and Ghostscript's ps2pdf and Frank Siegert's PStill (`http://www.wizards.de/~frank/pstill.html`) for converting PostScript to PDF.

The availability of multiple independent implementations is critical for demonstrating the sufficiency of the published PDF specification. It also promotes market competition, and gives users alternatives when the inevitable nasty software bug arises.

Apple's MacOS X operating system uses PDF as the native screen description format. There were early attempts to use PostScript for that purpose by Sun, with the *Network extensible Window System*, NeWS [8], in the late 1980s, and by NeXT, with Display PostScript [2, 9], in the early 1990s. Regrettably, processing power at the time was insufficient to make those efforts successful.

Adobe developed a special simplified generic PDF-producing printer driver, PDFWriter, for Microsoft Windows and Apple MacOS. This has made it possible for software vendors on those platforms to add PDF output capability with relatively little effort. Regrettably, output quality is sometimes inferior to what Distiller can produce, leading to user confusion and dissatisfaction. Adobe Acrobat 5 now installs a PostScript driver instead of PDFWriter.

PDF supports the notion of 'thumbnails': small bitmap images of pages that can be quite helpful in navigating through those documents where pages have recognizably different appearance. It also has bookmarks and hypertext links.

PDF viewers also offer magnification, which can be quite helpful in overcoming low screen resolution, or compensating for vision impairment.

Newer PDF viewers provide for page rotation, which is essential for reading documents with tables in landscape orientation.

Adobe offers a free PDF file creation service on the Web at `https://createpdf.adobe.com` that can be used to convert files from a variety of current desktop publishing and bitmap graphics file formats to PDF.

PDF has been extended to handle *forms*: documents with boxes to be filled out and transmitted electronically. The U.S. Internal Revenue Service provides income tax forms this way.

PDF and TEX

Hàn Thế Thành's important Ph.D. thesis research that led to pdfTEX has shown how TEX users can directly enjoy the benefits of PDF. The close coupling between typesetter and device driver makes some things possible that would perhaps be impractical in the conventional TEX → DVI → PostScript → PDF production path.

Elsewhere in these proceedings, Don Story shows how JavaScript can be used with TEX and PDF to create interactive documents, and Hans Hagen's fine work with ConTEXt and PDF is almost magical.

The hyperref package, written by Sebastian Rahtz, Heiko Oberdiek, and others, modifies LATEX sectional and cross-referencing commands to emit TEX `\special` commands to record hypertext links that some DVIware, and pdfTEX, can deal with. PDF supports such links, so PDF file viewing is automatically enhanced with navigational links. The package is available in the CTAN archives at `ftp://ctan.tug.org/tex-archive/macros/latex/contrib/supported/hyperref/`.

PDF and document archiving

In my view, the open specification and wide acceptance of PDF is very likely to ensure that it can be used for 'long-term' document storage, something that cannot be said for *any* of the proprietary desktop publishing formats.

Nevertheless, because PDF is a page description language, rather than a document markup language, it is still best to preserve document input forms, provided those are open and, possibly de-facto, standard.

PDF disadvantages: availability

Despite the praise of the previous sections, PDF is imperfect.

PDF implementations do not always agree with the specification, and Adobe's software often precedes the specification by months, or even a few years, as happened with PostScript Level 3. Third-party software developers then face the Herculean task of trying to reverse engineer the specification from experiments with Adobe's software. The development of both Ghostscript and pdfTEX has been significantly delayed by such problems.

Adobe's initial support of PDF for Apple MacOS, IBM PC DOS and OS/2, Microsoft Windows, and several flavors of UNIX (Compaq/DEC OSF/1, GNU/Linux on Intel x86, Hewlett-Packard HP-UX, IBM AIX, and Sun SunOS and Solaris) was encouraging. After all, a file format can hardly have the term 'Portable' in its name if it is not usable almost everywhere.

Sadly, Adobe's original commitment to broad support of PDF has been sharply curtailed. While the free Acrobat Reader component is offered for a number of platforms and human languages (see `http://www.adobe.com/products/acrobat/alternate.html`), the Acrobat product family with Distiller and Exchange has been completely dropped on all but MacOS and Windows. This is extremely troublesome, when Adobe markets PDF as a ubiquitous solution for page description.

The Acrobat releases for UNIX systems have an un-UNIX like command line, and lack support for path searching to find needed files. They also ship without any manual pages, a deficiency that I remedied locally. I donated my work back to Adobe for free and unfettered future distribution. Since the UNIX product line was dropped, that did not happen, so I am willing to make that documentation available on request to licensees of the product. For copyright reasons, I cannot place it in a public archive.

Were it not for Aladdin Ghostscript, users on other platforms would be mostly unable to produce PDF files at all. While the *Aladdin Free Public License* is quite generous, it does restrict commercial re-use, which, among other things, means that Aladdin Ghostscript cannot be included on TUG's annual TEX Live CD-ROM. Instead, TEX Live has to use the approximately one-to-two-years-older GNU release of Ghostscript.

It is never a good idea to rely on *any* software product that has a sole implementation, or runs only on a single platform. Software is complex, and even the yet-to-be-written perfect software package can be crippled by errors in the compiler, or run-time libraries, or operating system, or even hardware. Scientific experiments are never considered reliable until they have been independently reproduced. Software use is, after all, just another kind of experiment, and experience should have taught us to be highly skeptical of the outcome of any change to input data, or to program code.

Thanks to the fine work of Karel Skoupý and the NTS team [16], even TEX now has an independent implementation, although METAFONT still does not.

PDF disadvantages: complexity

PDF is compact because of data compression, and use of a binary, rather than ASCII, representation. Although the latter is possible, and was originally touted as an advantage of PDF [7], in practice, binary encoding is now almost universally used.

Compression and binary encoding both introduce a serious problem: data transformations that were formerly simple in uncompressed plain text now become immensely more complicated. A great many of the problems posted to the PDF user and developer mailing lists would have relatively simple solutions with plain text files.

What is needed is a standard tool for dumping PDF into a text format that can be edited, then converted back to the binary form, much as Geoffrey Tobin's extremely useful dv2dt and dt2dv tools (`ftp://ctan.tug.org/tex-archive/dviware/dtl`) do for DVI files, and Lee Hetherington's and Eddie Kohler's t1disasm and t1asm utilities (`ftp://ctan.tug.org/tex-archive/fonts/utilities/t1utils`) do for Type 1 outline font files. To my knowledge, no such freely-distributable tool exists for PDF files.

PDF's numbered, rather than named, object structure means that modifications generally require complete parsing of PDF, because objects must be renumbered if any are added or removed. Any future PDF disassembler/assembler tool must take this into account: it should be possible to hide this design blemish entirely.

The PDF file structure makes it impossible to simply concatenate multiple PDF documents to obtain a single document, something that is generally problem free with PostScript files.

Until I wrote this article, I knew of no generally-available free software that can combine PDF files, although there are commercial products for desktop systems that do so.

Now, with the TEX Live distribution,[1] it is as simple as this:

```
texexec --pdfarrange *.pdf --result=all
```

The resulting `all.pdf` file will contain all of the PDF files listed on the command line.

The binary format is also a serious problem for indexing of document collections, such as by Web search engines, or search tools like glimpse (see `http://webglimpse.net/`) or mg [19]. All of these need a PDF disassembler. Adobe's Acrobat Catalog product for indexing PDF file collections is platform specific, and GUI based, making it useless for many applications.

Except for dvipdfm (`ftp://ctan.tug.org/tex-archive/dviware/dvipdfm/`), TEX DVI drivers are incapable of dealing with PDF. pdfTEX can import PDF figures, but it cannot handle PostScript figures, or support the wizardry of the pstricks package (`ftp://ctan.tug.org/tex-archive/graphics/pstricks/`).

No PDF viewers provide information about the properties (font, color, texture, metric, ...) of user-selected displayed text.

PDF disadvantages: no logical markup

The PDF format lacks begin/end markers for identifying words, lines, paragraphs, sections, ... This is a serious design flaw that TEX DVI and PostScript also share. UNIX troff at least outputs word and line markers. The reason that these boundaries are important is that some operations can reliably only be done on the formatted text, that is, the text that actually appears on the page image. Such operations include text extraction, cataloging and indexing, spell checking, grammar checking, and string searching. Attempting to do so on the input files is problematic: it is unreliable in the presence of macro expansion (such as in TEX files), and the job must be done differently for each possible document input format. It would be far better to perform these actions on the final typeset text in PDF form, so that the programming job could be done just *once* for *all* input formats.[2]

The begin/end marker lack is just a special case of a more general problem: *all* current page description languages (DVI, PCL, PDF, PostScript, ...) completely lose all logical markup that was present in the input. The PDF discussion lists again provide ample evidence that what users really need is a page description language in which all logical markup is *preserved*, allowing recovery of the input and reliable translation into *any* markup system. It is simply not the case that one can always go back to the original document: often, that document is no longer available, or is in a proprietary format that is no longer available or supported, or is not usable on the current platform.

The recent PDF version 1.3 [3, Section 8.4.3] has some logical structure facilities, and PDF version 1.4 [4] introduces the notion of 'Tagged PDF'. These may supply the needed features to preserve logical markup. One reviewer, however, expressed reservations at their complexity, and it remains to be seen whether PDF-producing applications will take advantage of them.

PDF disadvantages: design limitations

Cut-and-paste with Acrobat Reader is deficient: ligatures (fi, fl, ffi, ffl, ...) are lost, or corrupted, on every MacOS, UNIX, and Windows platform that I've used. xpdf does *not* have this problem.

While PDF viewers offer page selection for printing, only the now-dropped Acrobat Exchange viewer had the ability to clip out a rectangular region of a page and save it as a separate file, with the 'supercrop' toolbar item. That feature is poorly documented, hard to use, imposes an obnoxious minimum crop size, and requires installation of an additional plugin software component. Borrowing figures and text snippets from other documents is a common need in document preparation, so perhaps it is fear of copyright violation that discourages software developers from including the capability in PDF viewers.

Although the Acrobat product family is released in numbered versions for multiple platforms, the viewer features differ between platforms. For example, Acrobat Reader 4 on Apple MacOS has a page cropping feature that is absent from the same version on Microsoft Windows, and the toolbar and menus differ between the two versions. While the

[1] The TEX Live CD-ROM lacked space to include precompiled formats for ConTEXt, so you first have to build them, like this:

```
env TEXMFINI ..some-path../texmf/web2c:  \
  texexec --make en nl metafun
```

[2] The dvispell utility, announced by Daniel Taupin on the tex-euro mailing list on 29-Oct-2001, does something similar: it reconstructs text to be spell checked directly from the TEX DVI file.

differences are not major, they still require a certain amount of mental retooling for the human user.

In my view, such differences are simply poor software design and management. The window system interface, while platform-dependent, should be a relatively small portion of the PDF viewer code, most of which has the much more difficult task of dealing with complex PDF and font file formats. For example, in xpdf version 0.92, less than 10% of the code deals with the window system (as evidenced by inclusion of window-system-related header files), out of a total of 175,000 lines of C++ code (about nine times as much as either TEX or METAFONT have).

The color matching problem is still not satisfactorily solved, although other page description formats have the same problem. We have no technological way yet to guarantee that colors that the author used are very close to what the remote reader or printer gets.

Text searching, and page changing, in all current PDF viewers are vastly slower than those operations in a good text editor on a similarly-sized body of text on the same platform, and regular-expression pattern matching searches are unavailable. PDF viewer startup times are also far too long. Sometimes, performance gets worse instead of better: version 0.91 of xpdf introduced a much more powerful font rendering engine that has dramatically slowed that viewer. A test of viewing each page of this document showed that the new version runs two to fourteen times slower, depending on whether the file server and display are local or remote. Fortunately, the new rendering can be turned off with a command-line option, restoring performance to about the same as that of Acrobat Reader and gv.

The original PDF specification anointed 14 fonts as standard, requiring them to be supported by all viewers, and therefore, eliminating the need to store them in PDF files. When fonts are omitted from PDF files, their metrics are still stored, so that when the required font cannot be found, PDF viewers can substitute other fonts and obtain correct letter spacing, even though the letter shapes are wrong.

With the release of Acrobat version 4, the standard font set was abandoned, and some viewers changed their default fonts, so that displayed documents now look different. Unfortunately, Distiller does not give the user sufficient control to ensure that all fonts will be embedded or subsetted, so users may not be able to ensure the same appearance everywhere for their PDF files. This particular flaw has caused users of the U.S. National Science Foundation FastLane grant proposal process a huge amount of grief, since PDF files that do not include full embedding and subsetting are rejected.

PDF version 1.4 [4] introduced a transparency feature, something that is completely absent from the PostScript imaging model of opaque paint. It is uncertain how such documents will be converted back to PostScript. So far, the transparency feature is little used, because most software cannot yet produce it. Its omission from PostScript, along with support for 3-D coordinates (and 4-D homogeneous coordinates), are the major flaws in that language that prevent PostScript from serving as a universal output format for modern computer graphics.

PDF disadvantages: bugs

In order to simplify, or compress, complex PostScript files that use language features (see [1, Appendix H.2.4]) that prevent their inclusion in other documents as Encapsulated PostScript figures, it can be helpful to convert such files to PDF, and then back to PostScript.

Unfortunately, Distiller has an automatic page rotation feature that is beyond user control, even though there is an option for it. I posted an example to the PDF developers list showing two small PostScript files differing only by a single comment: one was rotated by Distiller, and the other was not. This has to be a bug, and it completely prevents automated PostScript → PDF → PostScript cleanup of collections of figure files. ps2pdf does *not* have this problem.

Several PDF producers incorrectly rename subsetted fonts, causing the UniqueID problem discussed in the *TEX Font Panel* article elsewhere in these proceedings.

One audience member reported that the Hewlett-Packard 4550N has problems with some PDF files that other HP printer models with PostScript Level 3 support do not have. This may perhaps be traced to the lag between specification and software: the former should always come first.

Some PDF viewers incorrectly handle fonts with characters in positions $0 \ldots 31$ or $128 \ldots 159$.

Despite the fact that Adobe's own co-founder, and chief architect of PostScript, showed over twenty years ago how to use monitor gray scale for effective display of fonts [17], Acrobat Reader does a completely unacceptable job of displaying PDF files that use bitmap fonts (see `http://www.math.utah.edu/~beebe/fonts/outline-vs-bitmap-fonts.html` for further discussion, and visual comparisons). There is *no excuse* for this! PostScript and PDF are capable of handling several different font formats, and

PDF viewers should be able to perform equally well with all of them, subject to bitmap resolution in the original font data. By contrast, Paul Vojta's xdvi (ftp://ctan.tug.org/tex-archive/dviware/xdvi/) does an excellent job with bitmap fonts.

When Adobe announced Acrobat version 5.0 in August 2000, there was soon a flurry of correspondence on the PDF user and developer mailing lists about problems with the new release. These appeared serious enough, and were shown to impact PDF files produced from TEX documents, that we chose not to install the new version on local desktops at my site, even though we are licensed to do so. A year later, these problems are still under investigation, and the available version remains at 5.0. This seems to indicate quality control problems that should not be present in commercial offerings (even though they are rampant in the desktop software industry).

PDF files may contain metadata, such as text annotations, bookmarks, hypertext links, and so on, which must be coded in either Unicode, or in a superset of 7-bit ASCII called PDFDocEncoding, defined in [3, Table D.1, p. 551]. The encoding defines 194 characters, which should be the same in all applications. However, while writing software to convert text using standard TEX accents to PDFDocEncoding, Hans Hagen discovered platform differences for at least these characters: Ł, Œ, Š, Ÿ, Ž, Ø, and their lowercase companions. Such deviations are simply inexcusable when the specification is so clear.

Further reading

The three editions of the PDF Reference Manual [7, 3, 4] define the PDF language.

Thomas Merz's books [11, 10, 12] are a good source of information about PDF, and PDF forms. They are also typographically interesting, having been typeset with colored ink.

There are a few other books on PDF that I've recorded, but not yet seen [5, 6, 13, 14, 15, 18].

Finally, an extensive bibliography of publications about PDF and PostScript is available in the TEX Users Group bibliography archive at http://www.math.utah.edu/pub/tex/bib.

Acknowledgements

Thanks go to my fellow panel members, my wife Margaret, and L. Peter Deutsch of Aladdin Software, for many helpful comments on, and historical background for, drafts of this article.

References

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley, Reading, MA, USA, second edition, 1990. ISBN 0-201-18127-4. viii + 764 pp. LCCN QA76.73.P67 P67 1990.

[2] Adobe Systems Incorporated. *Programming the Display PostScript System with X.* Addison-Wesley, Reading, MA, USA, 1993. ISBN 0-201-62203-3. LCCN QA76.73.P67 D57 1993. US$29.95.

[3] Adobe Systems Incorporated. *PDF reference: Adobe portable document format, version 1.3.* Addison-Wesley, Reading, MA, USA, second edition, 2000. ISBN 0-201-61588-6. xvi + 679 pp. LCCN QA76.76.T49 P38 2000. US$49.95. http://partners.adobe.com/asn/developer/acrosdk/DOCS/PDFRef.pdf.

[4] Adobe Systems Incorporated. *PDF reference: Adobe portable document format, version 1.4.* Addison-Wesley, Reading, MA, USA, third edition, 2001. ISBN 0-201-75839-3. US$54.95.

[5] Ted Alspach and Jennifer Alspach. *PDF with Acrobat 4.* Peachpit Press, Inc., 1085 Keith Avenue, Berkeley, CA 94708, USA, 1999. ISBN 0-201-35461-6. 240 pp. US$17.99. http://www.amazon.com/exec/obidos/ASIN/0201354616/thepdfzone/002-7978061-6024210.

[6] Mattias Andersson et al. *PDF Printing and Publishing: the next revolution after Gutenberg.* Micro Publishing Press, Torrance, CA, USA, 1997. ISBN 0-941845-22-2. vii + 198 pp.

[7] Tim Bienz and Richard Cohn. *Portable Document Format Reference Manual.* Addison-Wesley, Reading, MA, USA, 1993. ISBN 0-201-62628-4. xii + 214 pp. LCCN QA76.9.F5P67 1993. US$24.95.

[8] James Gosling, David S. H. Rosenthal, and Michelle Arden. *The NeWS Book: an introduction to the Network/extensible Window System.* Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1989. ISBN 0-387-96915-2. vi + 235 pp. LCCN QA76.76.W56 A731 1989.

[9] David A. Holzgang. *Display PostScript Programming.* Addison-Wesley, Reading, MA, USA, 1990. ISBN 0-201-51814-7. x + 406 pp. LCCN QA76.73.P67 H63 1990.

[10] Thomas Merz. *PostScript and Acrobat/PDF: applications, troubleshooting, and cross-platform publishing.* Springer-Verlag, Berlin,

Germany / Heidelberg, Germany / London, UK / etc., 1997. ISBN 3-540-60854-0. xiii + 418 pp. LCCN QA76.73.P67M4713 1997. US$69.50.

[11] Thomas Merz. *Mit Acrobat ins World Wide Web: Effiziente Erstellung von PDF-Dateien und ihre Einbindung ins Web.* dpunkt Verlag, Ringstraße 19, 69115 Heidelberg, Germany, 1997. ISBN 3-9804943-1-4. 225 pp. DM 69,00; ATS 504,00; CHF 61,00. Includes CD-ROM. Also available in a revised edition in an English translation [12].

[12] Thomas Merz. *Web Publishing with Acrobat/PDF.* Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1998. ISBN 3-540-63762-1. xi + 234 pp. LCCN TK5105.888.M47 1998. Includes CD-ROM. Revised and extended English translation of the original German edition, [11].

[13] Bruce (Chauncey Bruce) Page and Diana Holm. *Web publishing with Adobe Acrobat and PDF.* John Wiley and Sons, New York, NY, USA; London, UK; Sydney, Australia, 1996. ISBN 0-471-14948-9. xx + 363 pp. LCCN Z286.E43P34 1996. US$35.96.

[14] Frank Romano. *Acrobat PDF and Workflow In-Detail.* P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 2000. ISBN 0-13-088948-2. ca. 522 pp. US$39.99. http://www.phptr.com/ptrbooks/ptr_0130889482.html.

[15] Frank Romano, Melbert B. Cary Jr., Mattias Andersson, William Eisley, Amie Howard, and Mark Witkowski. *PDF Printing and Publishing: The Next Generation After Gutenberg: The definitive guide to creating and using Adobe Acrobat 3.0 files.* Agfa Division, Bayer Corporation, 100 Challenger Road, Ridgefield Park, NJ 07660, 1997. US$27.95. http://www.AgfaHome.com/publications/dcp8text.html; http://www.amazon.com/exec/obidos/ASIN/0941845222/qid%3D905955098/002-1104103-2057202.

[16] Philip Taylor and Jiří Zlatuška. The $\mathcal{NTS}$ project: from conception to birth [Abstract]. *TUGboat*, 21(3):304, September 2000. ISSN 0896-3207.

[17] J. E. Warnock. The display of characters using gray level sample arrays. *Computer Graphics*, 14(3):302–307, July 1980. CODEN CGRADI. ISSN 0097-8930.

[18] Mark Witkowski. *The PDF Bible: The Complete Guide to Adobe Acrobat 3.0.* GATF-Press, 200 Deer Run Road, Sewickley, PA 15143-2600, USA. Telephone 412-741-6860, Fax 412-741-2311, Toll Free 1-800-662-3916, 1998. ISBN 0-941845-23-0. 438 pp. http://www.bookbuyer.com/aisles/titles/834607.htm; http://www.gatf.lm.com/98-10.HTM.

[19] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kaufmann Publishers, San Francisco, CA, USA, second edition, 1999. ISBN 1-55860-570-3. xxxi + 519 pp. LCCN TA1637 .W58 1994. US$54.95. http://www.mkp.com/books_catalog/1-55860-570-3.asp; ftp://munnari.oz.au:/pub/mg;http://www.cs.mu.oz.au/mg/; http://www.cs.mu.oz.au/~alistair/arith_coder/; ftp://ftp.math.utah.edu/pub/mg/; http://www.math.utah.edu/pub/mg/; ftp://ftp.math.utah.edu/pub/mg/mg-1.3x/bibsearch-1.02.tar.gz; http://www.math.utah.edu/pub/mg/mg-1.3x/bibsearch-1.02.tar.gz.