The Joy of TEX2PDF — Acrobatics with an Alternative to DVI Format

Petr Sojka, Han The Thanh, and Jiří Zlatuška

Abstract

This paper presents a discussion about generating Portable Document Format (PDF) directly from TEX source using a prototype TEX2PDF program. This is a derivative made from the TEX source which allows us to bypass DVI output generation, and to produce documents in Adobe PDF directly. Motivations for the TEX2PDF approach are discussed and further possible enhancements are outlined.

1 Motivation

> GO FORTH *now*
> *and create masterpieces*
> *of the publishing art!*
> *Don Knuth [19], p. 303.*

General acceptance of TEX for the publishing of technical documents has spread enormously during the last two decades. Since TEX's inception, however, new standards have emerged in the publishing world. SGML and LATEX for markup, PostScript and Portable Document Format as page description languages (PDL), are just a few of the buzzwords in the arena. Publishers are moving towards the art of creating *electronic* documents.

TEX's typesetting engine outputs its results in the device independent (DVI) page description format [9, **?**]. To avoid duplication, and to be backward compatible, various extensions to the DVI format have been used via the `\special` command. Do you need color? Use color supporting `\special`s. Do you need PostScript fragments in the `.dvi` file? Graphics in various formats? PDF fragments in the `.dvi` file? Hypertext? Document/object structure markup for an SGML driver? Every new application usually ends up as a new set of `\special`s, which are unfortunately, not yet standardized [27, 25].

Do you need portable object reuse in your `.dvi` file? Sound? Portable Multiple Master font parameters? No `\special`s for these are in sight.

As a result of all this, documents in DVI format are not really portable, as they usually contain a lot of `\special`s, and visual appearance depends on the device drivers available at the reader's site. These and similar problems and thoughts have led us to research on the possibility of generating portable electronic documents which will offer widest range of functionality from well established and widely used (LA)TEX sources.

This paper was awarded the UK TEX User Group's Cathy Booth memorial prize at TUG '96 in Dubna, Russia.

In section 2 we give an overview of current formats relevant to electronic document storage. In section 3 we discuss the current possibilities for producing PDF — a possible format of choice for electronic documents. We suggest a new approach by means of the TEX2PDF program in section 4 and, in section 5, its merits with respect to other approaches. We conclude with a discussion of object reuse in section 6, and future developments in section 7.

2 Formats for Electronic Document Delivery

2.1 DVI Format

A `.dvi` file is the standard output of a TEX run and is often used as a format for storage and exchange of typeset TEX documents.

DVI format is heavily (but not exclusively) used e.g. in the Los Alamos e-Print archive `http://xxx.lanl.gov/`. Several tens of thousands documents are available (typeset by autoTEXing scripts) from there. The disadvantage is that the documents are not 'self-embedded', which means that they rely on standardisation of font names and availability of fonts at the document consumer's site. Hypertext extensions to the DVI format have been accomplished by a set of HTML-like `\special`s defined by the HyperTEX project (`http://xxx.lanl.gov/hypertex/`) and special versions of previewers (`xhdvi`), `dvihps` and `ghostscript` (`ghosthview`) have been developed.

2.2 Portable Document Format

PDF [5] is a page description language derived from Adobe's PostScript language [2]. The design goals are:

- Rendering speed — algorithmic constructs were removed from the language.
- Portability — as a cross platform format, Acrobat Reader is available free of charge on major platforms.
- Compactness — the Lempel, Ziv, Welsh compression algorithm was licensed from UNISYS for maximum compression of files.[1] Multiple Master font technology, partial font downloading and built-in fonts in the Acrobat Reader lead to a minimum size for portable documents.
- WWW support — hypertext links to other documents on the Internet are allowed. PDF version 1.2 and Acrobat 3.0 (Amber) introduced a linearized arrangement of objects within PDF

[1] Latest news from Adobe says that ZIP compression has been added as well, leading to even better compression ratios.

documents, allowing for incremental downloading across the Internet.

- Extensibility — documents can be extended without losing the old version; notes (stickers) can be added to a document by the readers.
- Password protection — access to a document can be protected by a password.
- Object structure — allows for access to individual pages, with possibility of one-pass generation.
- Easy exchange — ASCII (7bit) PDF files can be generated for better portability and email exchange.

PDF files can be embedded directly in an HTML page using the HTML `<EMBED>` tag [1]. These are becoming more and more popular in the WWW world, as they render faithfully what the author saw (modulo color rendering and resolution of an end-user's display).

2.3 SGML

Roll on SGML, and real document storage.
Not just this strange PDF thing
which traps the visuals like an insect in amber ...
James Robertson on `comp.text.pdf`

SGML is a widely accepted international standard (ISO 8879) [12, 6, 3] for document markup. It is the format of choice for document storage chosen by many publishers [23, 7, 4]. It is a language for describing markup, aimed at long-term storage, but not at visual layout. As TEX's typesetting engine is still the state-of-the-art, the perspective of typesetting of SGML documents via LATEX3 with a TEX based engine is a viable option.

3 Current Possibilities for Producing PDF from TEX

If PDF is required as the end format, with currently available programs one has to generate PostScript from a `.dvi` file and then to 'distill' (using Adobe's Distiller program) the result to PDF. Some comments and suggestions on how to create PDF files from TEX are collected in [17]. Problems with configuring fonts are described in [28] and [8].

4 The Name of the Game

There still are countless important issues
to be studied, relating especially to the many
classes of documents that go far beyond
what I ever intended TEX to handle.
Don Knuth [21], p. 640

Motivated by a note by Don Knuth to one of the authors (private communication, 1994), who mentioned he expected people would attempt to create derivations from TEX suitable for, e.g., outputting PostScript instead of DVI, a project for creating PDF files directly from the TEX source has been attempted [14], introducing the possibility of creating either DVI or PDF output. The working name of this game is TEX2PDF. An example of the TEX source taking advantage of the new possibilities is shown in figure 1 and the resulting document as viewed with Adobe Acrobat Reader is shown in figure 2 on page 250.

4.1 New primitives

New primitives have been introduced in TEX2PDF in order to allow for more straightforward use of hypertext features from within TEX-like source. Most of their parameters are taken implicitly from the context of use in TEX terms, which simplifies their use considerably. We do not specify the full syntax here, because it is not yet fully stable.

`\pdfoutput` changes TEX2PDF behaviour from DVI-producing mode to PDF-producing one.

`\pdfannottext` takes an argument which specifies the text of an annotation to be created at the current position.

`\pdfannotlink`, `\pdfendlink` allows the user to specify hypertext links with all of the link attributes available in the PDF specification. An integer argument is used as a key to the corresponding anchor. If no link border has been specified, it is computed for all boxes between `\pdfannotlink` and `\pdfendlink`, so the link will automatically become multiline if a line break occurs in between.

`\pdfoutline` allows for the generation of bookmarks; bookmarks can be hierarchically structured.

`\pdfdestxyz`, `\pdfdestfit`, `\pdfdestfith`, `\pdfdestfitv` provide specification of various types of anchors with zooming and fitting possibilities.

`\pdfdestfitr`, `\pdfendfitr` specify the position of anchor corners. In this case, the anchor area is computed from the corners.

4.2 Font handling

Font handling in TEX2PDF is currently limited to Type1 fonts only. Metric information is extracted from the `pfb` file. Font name mapping is handled using an auxiliary font mapping configuration file introducing the list of fonts available, together with the information on the type of font embedding and its usage.

Virtual fonts [18] are supported in TEX2PDF. As they are in fact part of `.dvi` files, they have to be unfolded before PDF is output, as in today's DVI drivers.

4.3 Compression

Compression is allowed in the PDF specification, and several types of compression filters can be used; JPEG compression for color graphics, LZW and ZIP compression for text and graphics, and CCITT Group, Run Length and LZW compression for monochrome images.

As the LZW compression algorithm is licensed by UNISYS, we cannot distribute TEX2PDF with LZW support, but we used it for testing runs to compare TEX2PDF with Distiller (see table on the following page). However, the even more effective ZIP compression will be available in PDF version 1.2, avoiding the need for LZW compression in TEX2PDF, and the patent problems. The test figures show that TEX2PDF generated an even more compact PDF file than Adobe Distiller on standard text files.

4.4 Graphics

`\special`s are not yet handled by TEX2PDF. As most of the graphics included in TEX documents are PostScript and TIFF, at least support for the PostScript to PDF and TIFF to PDF conversion will have to be included in the future.

4.5 Implementation

The implementation of of TEX2PDF is realized as a `web` change file to the latest TEX source [20]. This implies that TEX2PDF is as portable as TEX itself is. Karl Berry's `web2c` package has been used for the development and for producing a running UNIX version. We expect easy recompilation on any UNIX platform.

5 Pros and Cons

> *I was constantly bombarded by ideas for extensions, and I was constantly turning a deaf ear to everything that did not fit well with TEX as I conceived it at the time.*
> *Don Knuth [21], p. 640*

To compare TEX2PDF with the other methods of producing a hypertext PDF document from a TEX file, we did several testing runs. They were done on a Sun Sparc 10 under the Solaris 2.4 operating system. Measurements were done using the `time` program (CPU times are listed). We used `tex.tex`, generated from the TEX source (`tex.web`) file, as the testing document. For the hypertext version we used a slightly changed version of `webmac.tex` (see `http://www.cstug.cz/~thanh/tex2pdf`).

In both time and size comparisons TEX2PDF beats its competitors (see tables on the following page). This is mainly due to the absence of intermediate DVI and PostScript formats in TEX2PDF, allowing for better PDF optimisation. TEX2PDF is slightly slowed down by `pfb` file parsing.

The users familiar with the (`emacs` + TEX + `xhdvi` (+ `ghostscript`)) suite of programs might want to switch to (`emacs` + TEX2PDF + `xpdf`), thus speeding up the document debugging cycle considerably.

TEX2PDF is written in `web` so that its source blends naturally with the source of TEX the program. The obvious benefit is absolute compatibility with TEX proper; the actual code which drives the typesetting engine is that of Don Knuth (modulo `whatsits` use for the hypertext primitives added in TEX2PDF). While this conformance to TEX source greatly benefits from Don's appreciation of stability, it makes the implementor's life more difficult in the world where PDF still evolves. It is also hard to debug TEX2PDF without incremental compilation. When we come to add implementation of `\special` commands, maintenance will become tough.

The changes introduced in new versions of PDF are motivated by achieving better performance when handling Acrobat documents, and so TEX2PDF is bound to have the PDF-generating modules modified or rewritten so that maximum benefit of the features supported by PDF technology can be used. The fact that PDF specification has been made public is crucial to success of this approach.

The TEX2PDF approach is naturally backward compatible with TEX — in fact, if PDF output is not switched on, it can still generate DVI output identical to that of TEX. Just by redefining some cross-referencing macros, the new hypertext features of TEX2PDF can be instantly used even without modifying the markup of old LATEX documents.

6 Object Reuse

> *Using well-designed formats results in LATEX source that clearly reflects the document structure.*
> *T. V. Raman [24]*

With PDF, there is the possibility of taking advantage of the object structure and manipulation specified within a PDF file to store elements of document structure (higher level document model) in the PDF file generated by the application (TEX2PDF). Some work has been already done in this direction by defining Encapsulated PDF (EPDF) blocks and

Program(s)	Time without compression	Time with compression (LZW)
TEX2PDF (α-test version)	1:57	2:38
TEX + dvips 5.58 + Adobe Distiller 2.1	6:34 (1:33+0:18+4:43)	6:56 (1:33+0:18+5:05)
TEX + dvips 5.58 + Aladdin Ghostscript 4.0	40:23 (1:33+0:18+38:32)	not applicable

Table 1: Speed comparison of several ways of producing PDF file (`tex.pdf`) from a TEX file (`tex.tex`)

Program(s)	without LZW compression	with LZW compression	without compression and PDF file gzipped
TEX2PDF (α-test version)	8 063 658	3 086 545	1 906 184
TEX + dvips 5.58 + Adobe Distiller 2.1	10 530 967	4 387 232	2 115 827
TEX + dvips 5.58 + Aladdin Ghostscript 4.0	16 908 552	not applicable	

Table 2: Size comparison of several ways of producing PDF file (`tex.pdf`) from a TEX file (`tex.tex`)

their reuse [26]. This format, however, is not supported or used by a wide variety of applications.

The logical structure of a document model is also urgently needed in applications like AsTeR [24], which *reads* LATEX documents using a speech synthesizer. Developing an application that is capable of reading aloud enriched PDF files might become possible.

Our suggestions for further work could lead to primitives which allow handling of PDF *objects* stored in the trailer of a PDF file indirectly. At least three primitives are foreseen:

`\setpdfbox` typesets its argument and stores the result as a PDF object. The reference to that object will stay in the internal register accessible by `\lastpdfbox`.

`\lastpdfbox` returns the reference to the last stored object by `\setpdfbox`.

`\usepdfbox` This primitive puts a *reference* to an object into the output stream.

7 Future Work

> *Few claim to know what will be the preferred electronic format a century from now, but I'm willing to go out on a limb and assert that it will be none of TEX, PostScript, PDF, Microsoft Word, nor any other format currently in existence.*
> *Paul Ginsparg [11]*

TEX2PDF is currently under development and is available to beta testers only. We do not guarantee that the input syntax will remain unchanged. Support for object reuse, graphics and OpenType (TrueType) fonts when the PDF specification 1.2 comes out may be added.

For testing purposes, a `tex2pdf` option for the `hyperref` package [16] will be written, using the hypertext possibilities of TEX2PDF directly. This will allow using TEX2PDF for re-typesetting of LATEX documents just by loading with `hyperref` package with the `tex2pdf` option in the document preamble.

Support for the full usage of Multiple Master technology remains to be added, possibly in the combination with METAFONT [15, 13]. Extensions of the paragraph breaking algorithm [22] to take advantage of Multiple Master fonts with a variable width axis (but constant grayness) to help justification (`\emergencyfontwidthstretch`) is another possible direction of future work.

Acknowledgements

The support of the TUG '96 bursary committee is acknowledged, having allowed presentation of a preliminary version of this paper at the TUG '96 conference in Dubna, Russia.

References

[1] Adobe. Adobe Acrobat 3.0 beta. `http://www.adobe.com/acrobat/3beta/main.html`, 1996.

[2] Adobe Systems. *PostScript Language Reference Manual*. Addison-Wesley, Reading, MA, USA, 1985.

[3] American National Standards Institute and International Organization for Standardization. *Information processing: Text and Office Systems: Standard Generalized Markup Language (SGML)*. American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, 1985.

[4] Association of American Publishers. *Association of American Publishers Electronic Manuscript Series Standard for Electronic Manuscript Preparation and Markup: an SGML Application Conforming to International Standard ISO 8879–Standard Generalized Markup Language. Version 2.0 Dublin, Ohio: Available from the Electronic Publishing Special Interest Group, c1987.* Association of American Publishers, Dublin, OH, USA, 1987.

[5] Tim Bienz, Richard Cohn, and James R. Meehan. *Portable Document Format Reference Manual, Version 1.1.* Addison-Wesley, Reading, MA, USA, 1996.

[6] Steven J. DeRose and David G. Durand. *Making Hypermedia Work.* Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands, 1994.

[7] Andrew E. Dobrowolski. Typesetting SGML Documents using TEX. *TUGboat*, 12(3):409–414, December 1991.

[8] Emerge, Inc. TEX and PDF: Solving Font Problems. `http://www.emrg.com/texpdf.html`, 1996.

[9] David Fuchs. The Format of TEX's DVI Files. *TUGboat*, 1(1):17, October 1980.

[10] David Fuchs. The Format of TEX's DVI Files. *TUGboat*, 3(2):14, October 1982.

[11] Paul Ginsparg. Winners and Losers in the Global Research Village. `http://xxx.lanl.gov/blurb/pg96unesco.html`, February 1996.

[12] Charles F. Goldfarb and Yuri Rubinsky. *The SGML Handbook.* Clarendon Press, Oxford, UK, 1990.

[13] Michel Goossens, Sebastian Rahtz, and Robin Fairbairns. Using Adobe Type 1 Multiple Master Fonts with TEX. *TUGboat*, 16(3):253–258, June 1995.

[14] Han The Thanh. Portable Document Format and Typesetting System TEX (in Czech). Master's thesis, Masaryk University, Brno, April 1996.

[15] Yannis Haralambous. Parametrization of Postscript Fonts through METAFONT—an Alternative to Adobe Multiple Master Fonts. *Electronic Publishing*, 6(3):145–157, April 1994.

[16] Yannis Haralambous and Sebastian Rahtz. LATEX, Hypertext and PDF, or the Entry of TEX into the World of Hypertext. *TUGboat*, 16(2):162–173, June 1995.

[17] Berthold K. P. Horn. Acrobat PDF from TEX. `http://www.YandY.com/pdf_from.pdf`, 1996.

[18] Donald Knuth. Virtual Fonts: More Fun for Grand Wizards. *TUGboat*, 11(1):13–23, April 1990.

[19] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, Reading, MA, USA, 1986.

[20] Donald E. Knuth. *TEX: The Program*, volume B of *Computers and Typesetting.* Addison-Wesley, Reading, MA, USA, 1986.

[21] Donald E. Knuth. The Errors of TEX. *Software–Practice and Experience*, 19(7):607–685, 1989.

[22] Donald E. Knuth and Michael F. Plass. Breaking Paragraphs into Lines. *Software–Practice and Experience*, 11:1119–1184, 1981.

[23] Sebastian P. Q. Rahtz. Another Look at LATEX to SGML Conversion. *TUGboat*, 16(3):162–173, September 1995.

[24] T. V. Raman. An Audio View of TEX Documents. *TUGboat*, 13(3):372–379, October 1992.

[25] Tomas G. Rokicki. A Proposed Standard for Specials. *TUGboat*, 16(4):395–401, December 1995.

[26] Philip N. Smith. Block-Base Formatting with Encapsulated PDF. Technical Report NOTTCS-TR-95-1, Department of Computer Science, University of Nottingham, January 1995. `http://www.ep.cs.nott.ac.uk/~pns/pdfcorner/complete.pdf`.

[27] Mike Sofka. Dvi Driver Implementation and Standardization Issues. Available as `http://www.rpi.edu/~sofkam/DVI/dvi.html`, 1996–.

[28] Kendall Whitehouse. Creating Quality Adobe PDF Files from TEX with Dvips. `http://www.adobe.com/supportservice/custsupport/SOLUTIONS/2d7a.htm`, 1996.

◇ Petr Sojka, Han The Thanh, and Jiří Zlatuška
Faculty of Informatics
Masaryk University Brno
Burešova 20, 602 00 Brno
Czech Republic
Internet: `sojka, thanh, zlatuska@informatics.muni.cz`

```
\hsize 3in
\baselineskip 13pt
\pdfoutput=1              % we will produce PDF instead of DVI
\pdfannottext
    open                  % optional specification if the text annotation is implicitly opened
    {The text annotation} % the text itself
\def\BL{\pdfannotlink
    depth 3pt height 8pt  % optional specification for link size
    1                     % key of destination
    border 0 0 1          % optional specification for link border
}
\def\EL{\pdfendlink}
\pdfoutline
    1                     % key of destination
    0                     % number of sub-entries of this item
    {The outline entry}   % Text of this item
\pdfdestxyz
    1                     % key of this destination
    zoom 2                % optional zoom factor
%\pdfdestfit  1 or %\pdfdestfith 1 or %\pdfdestfitv 1
%\pdfdestfitr 1 ... \pdfendfitr

This is \TeX, a document compiler intended to produce typesetting of
high quality.  The PASCAL program that follows is the definition of
\TeX82, a standard version of \TeX\ that is designed to be highly
portable so that identical output will be obtainable on a great
variety of computers.

The main purpose of the following program is to explain the algorithms
of \TeX\ as clearly as possible. \BL As a result, the program will not
necessarily be very efficient when a particular PASCAL compiler has
translated it into a particular machine language.\EL\ However, the
program has been written so that it can be tuned to run efficiently in
a wide variety of operating environments by making comparatively few
changes. Such flexibility is possible because the documentation that
follows is written in the WEB language, which is at a higher level
than PASCAL; the preprocessing step that converts WEB to PASCAL is
able to introduce most of the necessary refinements.  Semi-automatic
translation to other languages is also feasible, because the program
below does not make extensive use of features that are peculiar to
PASCAL.

A large piece of software like \TeX\ has inherent complexity that cannot
be reduced below a certain level of difficulty, although each individual
part is fairly simple by itself. The WEB language is intended to make
the algorithms as readable as possible, by reflecting the way the
individual program pieces fit together and by providing the
cross-references that connect different parts. Detailed comments about
what is going on, and about why things were done in certain ways, have
been liberally sprinkled throughout the program.  These comments explain
features of the implementation, but they rarely attempt to explain the
\TeX\ language itself, since the reader is supposed to be familiar with
{\sl The \TeX book}.
\bye
```

Figure 1: Example of new hypertext primitives added in the TEX2PDF source file

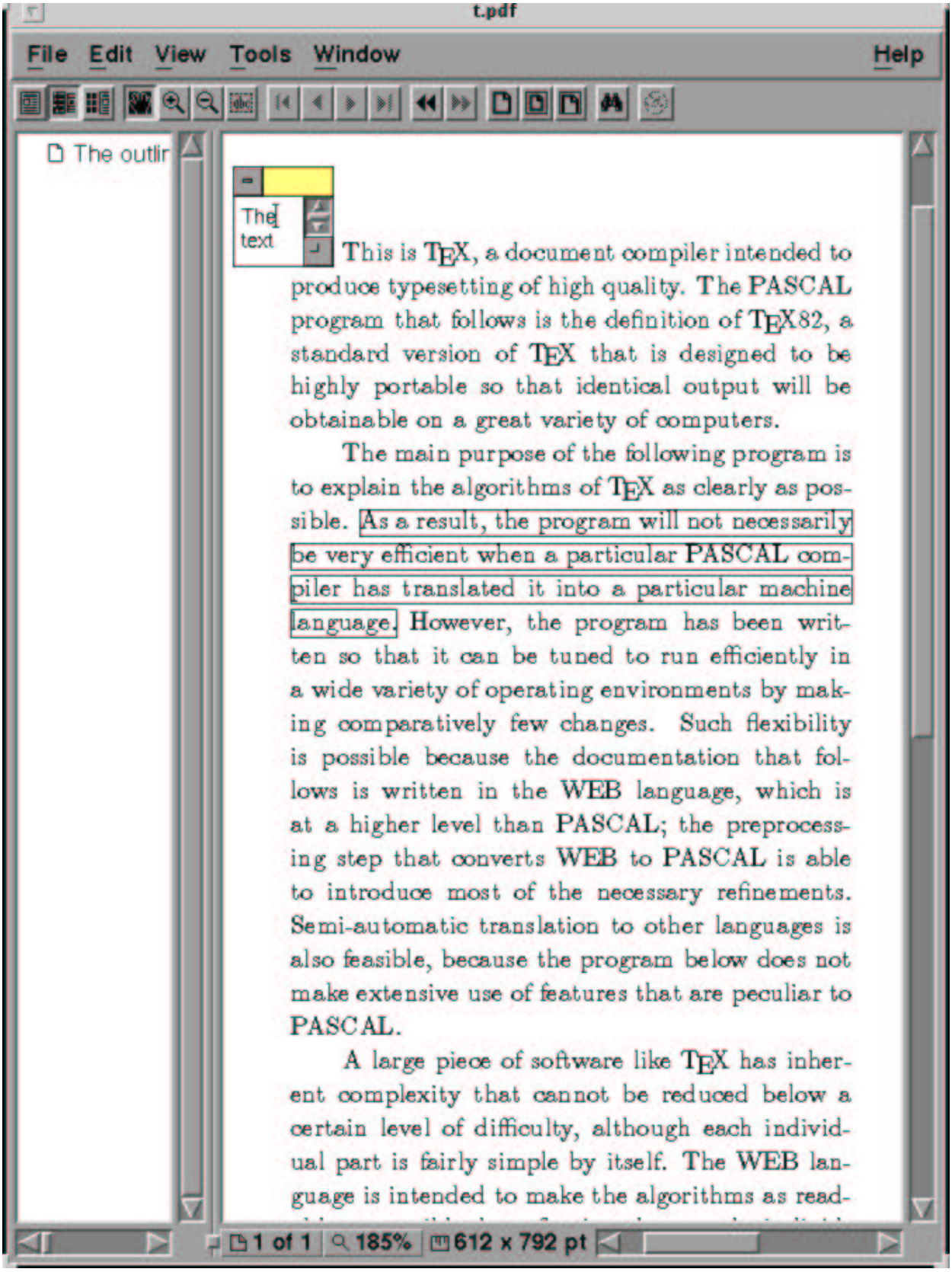

Figure 2: Result of TEX2PDF source in Fig. 1 viewed in Acrobat Reader