AcroTEX: Acrobat and TEX Team Up

D.P. Story
Department of Mathematics and Computer Science
The University of Akron
Akron, OH 44325
dpstory@uakron.edu
http://www.math.uakron.edu/~dpstory/

Abstract

Adobe's Acrobat (PDF) and Donald Knuth's TEX system make a powerful team for putting mathematics on the internet. For the educator, this team, called "AcroTEX," is the poor man's multimedia software company.

Though TEX was implemented before the rise in popularity of the World Wide Web, PostScript code written to the `.dvi` file, using TEX's `\special`s, can be used to enhance an electronic document created from a TEX source by introducing such elements as color, hypertext links, form features, sounds, and even video clips. These special features are achieved by inserting 'pdfmarks' into the output file. The Adobe Distiller, in turn, interprets these pdfmarks and translates them into the appropriate element as it writes the PDF document. TEX is, therefore, well suited for creating PDF files, especially technical material. With the aid of its very powerful macro facility and the ability to position material very precisely on a page, these electronic enhancements can be created and placed in an exact and automated way.

This paper explores the capabilities of AcroTEX and the contents of the AcroTEX web site (`www.math.uakron.edu/~dpstory/acrotex.html`); examples include tutorials, an electronic grading system, mathematical games, and technical articles.

Introduction

In this paper, I will describe how TEX, the outstanding mathematical typesetting engine, and the Portable Document Format (PDF) of Adobe Systems, first introduced in the Acrobat suite of software, form a team called "team AcroTEX",[1] and how this team has opened up a world of possibilities to people who are interested in electronic education.

From the perspective of an educator, this paper is an exposition of the natural *implications* of combining TEX and PDF; the exposition covers genesis (first thoughts), creation (of tutorials), problems and solutions, educational games, technical articles, and new macro packages for readers who may want to develop on-line educational materials themselves.

Here, AcroTEX refers to a process of producing PDF documents from a TEX source. A PDF document is a compact, self-contained file format which preserves the page layout of the authoring application. This makes a PDF file suitable for distribution over the Web.

AcroTEX also refers to a web site:

`www.math.uakron.edu/~dpstory/acrotex.html`

The documents referenced in this paper can be accessed at the AcroTEX web site, a site primarily dedicated to mathematical education. All files at this site, save only some start-up HTML pages, are in PDF format.[2]

Genesis

The original concept was to create a series of electronic tutorials covering some of the topics of the first semester of a standard course of calculus as offered in many colleges and universities in the U.S. The design goals of the tutorials included typeset quality on-screen mathematics, cross-referencing using hypertext links, and on-screen color.

Typeset quality mathematics and a presence of limited finances implies TEX. Of the freeware and

[1] A proposed alternative is TEXrobat, but this sounds too mechanical.

[2] Readers are available for virtually every platform; see `www.adobe.com/prodindex/acrobat/readstep.html`.

commercial TEX systems available at the time, late 1994, only the Y&Y TEX system offered coloring of fonts and a hypertext feature within its .dvi previewer, DVIWindo. A small site license from Y&Y was purchased using a start-up grant from the Educational Research and Development Office at the University of Akron.

The calculus tutorial, called "e-Calculus", was written and put on the department intranet as a collection of .dvi files. The students would come into the computer lab to review the materials using Y&Y's DVIWindo. The system worked well; however, the students were reluctant to spend the long periods of time that would be necessary to read the tutorial in the computer lab.

This reluctance prompted me to consider the internet. For many reasons, the .dvi file format is not a suitable format for the tutorials on the Web. Adobe's PDF seemed to be a natural choice: the typeset quality of the material, page layout, and color all would be preserved.

Fortunately, Y&Y was well positioned to convert its .dvi files to .pdf files. Their dvi-to-ps driver, dvipsone, automatically converted the hypertext links that DVIWindo understood to hypertext links that the Acrobat Reader understood. As a result, "e-Calculus" was ported to the internet with very little trouble.[3] This was the beginning of the AcroTEX web site.

The site has since grown in size to include other tutorials, mathematical games, a demonstration of forms processing using the PDF forms format FDF, several technical articles on TEX, LATEX and PDF, and a couple of macro packages (web/exerquiz).

The tutorials

The writing of the "e-Calculus" tutorial was simple enough and will not be discussed here; another tutorial, entitled "An Algebra Review in Ten Lessons" (mpt_home.html), has since been written in response to the needs of the incoming students at the University of Akron.

Beyond the technical discussions of calculus or algebra that any paper document would provide — though the discussions are more verbose than would normally be seen on paper — these tutorials try to take advantage of the electronic medium.

For finding information within the tutorials, hypertext versions of tables of contents, bookmarks (a feature of PDF), cross-references, and indexes — both local (for the file being viewed) and global (for the entire calculus/algebra set of tutorials) — are all provided.

The tutorials include in-line examples and exercises *à la Knuth*: that is, in the source file, the questions and solutions are kept together; however, solutions are linked to the questions by hypertext. The syntax is as follows:

```
\exercise < Some exercise question. >
\answer < The answer or solution. >
  ⋮    ⋮    ⋮    ⋮    ⋮    ⋮    ⋮
\endexercise
```

The macro \exercise sets the hypertext link, \answer sets the target, a named destination, for the link and starts a verbatim write macro. All material between \answer and \endexercise is written to the file \jobname.ans, which is later included at the end of the main file. There is an \example macro that behaves in the same way to handle examples.

This method of handling exercises and examples allows the posing of the question within the body of the text, but the solution does not appear (and take up screen space) unless the reader wants to see it. This allows for a clean, clear, more orderly presentation and discussion of topics.

By the way, proofs of theorems are handled the same way. The theorem is stated and a hypertext link is given to the proof. The main text then continues with a discussion of the meaning of the theorem, followed by examples and exercises.

Another feature of the tutorials is user interaction. User interaction with the document comes in the form of multiple-choice questions or quizzes. Click on the chosen response to obtain instant feedback in the form of humorous congratulations — or an error message.

TEX and PDF

Macro packages. When I first started this project in 1994, I decided to use $\mathcal{AMS}$-TEX 2.1 as the basic macro package. At the time, I had a rather slow computer with very little RAM. I found that LATEX was very slow in loading and took a long time to process a file; however, $\mathcal{AMS}$-TEX 2.1 on the same system loaded quickly and ran acceptably fast. The consequences of that decision meant: (1) I must write all my own macros for page formatting, cross-referencing, tables of contents, indexes, color inclusion, hypertext linking, etc., and (2) I must read *The TEXbook* not once, not twice, but many times. Other books studied and found to be very useful are the ones by Salomon (1995) and Eijkhout (1992).

[3] File: e-calculus.html. All files mentioned in this paper are located at www.math.uakron.edu/~dpstory.

If I were starting over today, I would probably use LaTeX and Sebastian Rahtz' `hyperref` package. LaTeX comes with many of the features that I had to struggle with to include in my own system; `hyperref` provides many of the hypertext features that I regularly use in my tutorials.

However, I have no regrets. At the time I began this project in late 1994, `hyperref` was still in its infancy anyway. One thing is certain, since I wrote the macros myself, *if something goes wrong, I know immediately where the problem is, and how to fix it.* Problem turnaround time is much shorter: I don't have to complain (report bugs) to the package author, then wait for the fix to arrive.

Creating PDF. There are essentially three methods for creating an *interactive* PDF document from a TeX source:

1. Convert the `.dvi` output file to a PostScript file using a `dvi`-to-PostScript driver such as dvipsone or dvips, then use the Acrobat Distiller to convert the PostScript file to a PDF file.[4]
2. Convert the `.dvi` file to PDF, bypassing the PostScript step, by using either dvipdf (Lesenko, 1996) or dvipdfm[5] by Mark A. Wicks.
3. Convert the `.tex` source file directly to a PDF file by using PDFTeX,[6] a program that is the ongoing labor of Hàn Thế Thành (Sojka, Thanh, and Zlatuška, 1996).

Figure 1 depicts the various paths from a `.tex` source file to a `.pdf` file.

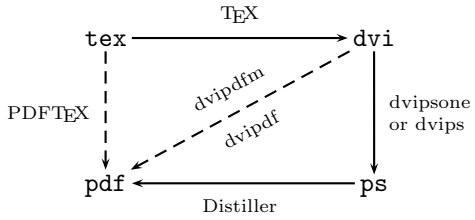

Figure 1: From TeX to PDF

Use Type 1 fonts. One last point must be made before leaving this topic. To create a *quality* PDF document from a TeX source it is necessary to use Type 1 fonts. The traditional font used by many freeware TeX systems is the bitmap or `pk` font; these fonts look choppy and jagged when incorporated into a PDF document and viewed on screen (though they do print decently).

Quality Type 1 CM fonts have been made available by a consortium of Bluesky Research, Y&Y, AMS, SIAM, IBM, and Elsevier. The freeware, shareware, and commercial TeX systems now come with Type 1 fonts. For an author wanting to publish on the Web using PDF, every effort must be made to reconfigure their TeX system to use these quality fonts.

Problems with multi-file systems

In this section, problems and issues associated with maintaining a large multi-file system are discussed. Hopefully, there will be enough detail to help readers better manage their own systems.

The use of a Make utility. The tutorials consist of a large number of files that are undergoing constant revision. A make utility[7] is used to help maintain this system of files.

A script file listing file dependencies was developed for each of the two tutorials, "e-Calculus" and "Algebra Review". (Actually, two sets of scripts are maintained: one for the system of `.dvi` files and the other for the system of `.pdf` files.) The make utility reads the script and initiates the programed action, perhaps calling the TeX compiler or the `dvi`-to-`ps` converter.

To create PDF files, for example, the `make` utility, as signaled by the controlling script file, calls the `dvi`-to-`ps` converter (dvipsone in my case) for each `.dvi` that needs to be updated, which in turn dumps the PostScript output into "Watched Folders". The Acrobat Distiller is activated and distills the files in these watched folders automatically and places them in an `out` folder. A batch file is then invoked to move the new `.pdf` files into their proper location.

The behavior of the `make` can be controlled by way of command-line switches of the make utility. As a result, only the files that have changed can be updated or the whole system of files can be re-TeXed and (optionally) re-distilled.

The `make` utility has been very helpful with the problem of trying to keep all files up-to-date. Updating the whole system of files is a matter of starting the `make` utility twice, once to TeX all files, then again to create the corresponding PostScript files dropped into the watched folders. The distiller does the rest.

Macro option switches. Each file belonging to one of the tutorials contains a table of contents, an

[4] This method is the process referred to as 'AcroTeX'; it is the only one that uses the Acrobat Distiller.

[5] Information and download are available at `http://odo.kettering.edu/dvipdfm/`.

[6] See `www.tug.org/applications/pdftex/` for more information and download links.

[7] The make utility that came with Microsoft Macro Assembler 5.0 is used.

index, and cross-document hypertext jumps. To further complicate matters, a collection of files covering a common topic, such as "Integration", shares the same table of contents (and same collection of bookmarks). When a new section is introduced or a new cross-document link is created, the supporting auxiliary files (and there are eight of them) must be properly updated and synchronized.

LATEX has a more or less automated system of updating the auxiliary files. Usually, LATEXing three times does the job. However, the macros developed for the multi-file system of tutorials are not nearly as automated but still are quite effective.

The system of macros that I have developed has "option switches" to update any cross-references between files, update the tables of contents, or update the indexes.

Below is a verbatim listing of (a portion of) the preamble of one of my "e-Calculus" files.

```
\documentstyle{tutorial}
\LocalOptions={}
\InstallOptions
```

The `\LocalOptions` token list allows local control of what auxiliary information is written. For example, after some changes in the file, this option can be changed to read

```
\LocalOptions={\CompileTOC}
```

The updated table of contents information will now be written to the file `\jobname.toc`.

In addition to local control of a file, global control of a collection of files is needed as well. In this case, the master macro style file, `tutorial.sty`, is opened and a token list, `\GlobalOptions`, is modified.

For example, the procedure for updating the entire "e-Calculus" tutorial is as follows. First, all local options are turned off and a `\GlobalOptions` token list is modified to perform each of the following tasks in turn: write to the appropriate auxiliary file (1) all table of contents entries, (2) all cross-document labels, and (3) all index entries. The "e-Calculus" source files are then TEXed (with the aid of the make utility) with each of these three tasks set. Finally, MakeIndex is run, the system is re-TEXed again. The entire multi-file system of PDF files can now be uploaded to the AcroTEX web site.

Write the document state. The tutorial system consists of a series of related articles. Files are kept to a size of around 500K in order to minimize the download time yet maximize the content.

Sometimes a single article is spread over several files. In this case, it is desired to have correct numbering of sections, examples, exercises, figures, and so on. For this purpose I wrote a macro called `\WriteDocumentStateTo{<filename>}`. The macro is placed at the end of a file and its purpose is to write the values of several count registers to the file called `<filename>.sts`. Here is a sample listing of one of the `.sts` files:

```
\secno=2 \subsecno=6 \resultno=2 \exno=43
\exmplno=14 \tagno=11 \figureno=2
```

The file `<filename>.sts` is then input by the file `<filename>.tex`, which sets the ongoing count of each of the listed registers.

Notice that the count register `\pageno` is not transmitted to the next file. This is because, at the time the system was designed, page numbers were determined to be of little importance in a multi-file system of tutorials!

No page numbers. Of course, TEX and PDF both maintain physical page numbers. In the tutorials, they are not printed on the electronic page and, with one exception, not referred to at all.

Because the tutorials were designed specifically for on-screen reading — not for the printed page — all cross-references can be made using hypertext links to named destinations. There is no need to write "see the definition of continuity on page 106"; it suffices to write "see the definition of continuity", where continuity is a hypertext link. (In the tutorial, links are color-coded; for this paper publication, they are underlined but do not work.)

Multi-file indexes. That one exception is in the creation of indexes. Page numbers in the index are used only as a visual reference of how far an index entry lies in from the beginning of the file.

Another problem with indexes in a multi-file system is that a given keyword, say "Euclid", might be referenced in several different files — perhaps, in an article on functions, on limits, on continuity, on differentiation, or on integration. How can the index give an *hint as to the* context *of the reference*?

Looking at the index in "e-Calculus" you would see the following entry under "Euclid":

```
Euclid, c1l:12, c1i:205
```

Each entry has an *index descriptor* followed by a colon and a page number. The index descriptor describes the file the reference lies in; for example, `c1l:12` indicates that the word "Euclid" was used on page 12 in the calculus 1 file on limits (`c1l`). The page numbers are underlined and hypertext-linked to the indicated page in the appropriate file.

The technique of manipulating the MakeIndex utility to obtain this index descriptor prefixed to the

page numbers is a little sneaky and should, perhaps, be left for another occasion.

Mathematical games

Games stimulate interest in mathematics, promote creativity, and provide a resource of class projects. All of the games listed below suggest that TEX and PDF can be used to create simple games that are inexpensive to build and challenging, educational, and informative to play.

My interest in games was stimulated by Gary Cosimini of Adobe. He created a game board in PDF that fascinated me; I wondered how he did it and set out to duplicate and expand on his game. To construct my own version of the game from a TEX source, I determined that I needed a greater understanding of the `pdfmark` operator. My reading and experimentation resulted in a Jeopardy-like game called "Algeboard" covering some topics in elementary algebra.

Studying the *Pdfmark Reference Manual* (Bienz and Staas, 1996) and the *PDF Reference Manual* (Bienz et al., 1996) gave me greater understanding of how to construct the game. With TEX's macro feature and its ability to place objects very precisely, I was able to stack form fields one on top of the other, and control their hidden attributes. When the user clicks on one of the game squares, the visible form field becomes hidden and another becomes visible. This is the way some of the effects are accomplished in the game.

"The Giants of Calculus", a two-column game of matching, was another challenge in coordinating layered form fields to get the desired effects.

Later, when Adobe Acrobat 3.0 came out with JavaScript, the Algeboard game was revised and JavaScript was used to keep track of the score. Thus was born "Algeboard/JS". This gave me some early experience in using JavaScript.

The experience with JavaScript helped when I attempted to create the more complicated "PDF Flash Cards for Kids", an electronic variation on the flash cards children use to practice their arithmetic skills of addition, subtraction, multiplication, and division.

All of the above-mentioned games can be downloaded from the AcroTEX web site.

Technical documentation

One of the goals of the AcroTEX site is to try to encourage other educators to use TEX (or LATEX) and the Portable Document Format to put mathematics (or any technical material) that might be of benefit to students on the internet. To that end, I've written several technical articles that describe how to insert hypertext links and form annotations into a PDF document, and how to use LATEX to create a quality interactive document in PDF.

About `Pdfmarks`. A `pdfmark` operator is an extension to the PostScript language that is read and understood by the Acrobat Distiller. The `pdfmark` is used to insert PDF-related features such as hypertext and form annotations. Primary documentation on `pdfmark`s is from Bienz and Staas (1996); an excellent on-line reference to `pdfmark`s is the "pdfmark Primer", one of the chapters from Thomas Merz' fine book, *Web Publishing with Acrobat/PDF* (1998).

For the TEX programmer wanting to create hypertext links or to insert form elements into a document, the appropriate code can be inserted into the `.dvi` file by using raw PostScript `\special`s. This code is then passed on to the `.ps` file by `dvi`-to-`ps` converters such as dvipsone or dvips.

For LATEX users, Sebastian Rahtz' `hyperref` package performs all these tasks wonderfully. Still, there are certain special effects that `hyperref` does not include; in this case, knowledge of `pdfmark` programming is essential.

The electronic article, "Pdfmarks: Links and Forms" (Story, 1998a), was written not long after I had constructed the games just described. I had made an in-depth study of the `pdfmark` operator and thought it might be a useful to write about what I had learned as a way of organizing the information in my own mind.

The article describes the basic, the advanced, and the more subtle techniques of creating hypertext links and form annotations for PDF. Written from the perspective of a TEX user, the article is interactive and highly informative. The many extensive and detailed examples contained in the article use TEX primitives and macros; TEX users can copy and paste code swatches into their own source document.

Authoring PDF documents using LATEX. In the past few years, I've received several inquiries from educators about how to construct good online tutorials using LATEX. Not a regular user of LATEX myself, I really didn't know. Ultimately, I got interested in learning LATEX, and in understanding how to use Sebastian Rahtz' `hyperref` package. As a result, I wrote the article, "Using LATEX to Create Quality PDF Documents for the WWW" (Story,

1998b). In this article, I tried to describe all the elements that go into the creation of a visually attractive, interactive PDF document using LaTeX, with special emphasis on how to use the `hyperref` package.

The `Web` and `Exerquiz` packages

The goals of these two packages are: (1) to create an attractive, easy-on-the-eye page layout suitable for the WWW, and (2) to make it very easy (for educators) to create interactive exercises and quizzes in the PDF format. Both packages are available in `webeq.html`.

The `web` package (for LaTeX) is a set of macros that establishes a page layout for a (PDF) document that is meant to be read on-screen and *not* meant to be printed. The package also redefines the table of contents to a web style and defines optional navigational aids. The package has options for use with dvipsone, dvips, and PDFTEX.

The `exerquiz` package defines three new environments:

- the `exercise` environment creates on-line exercises; solutions are hyperlinked to the question;
- the `shortquiz` environment is used to construct multiple-choice quizzes (with or without solutions) with immediate feedback in the form of "Right" and "Wrong" message boxes appearing on-screen;
- the `quiz` environment is used to write longer multiple-choice quizzes that are graded by JavaScript.

The `exerquiz` package is somewhat independent of the `web` package; it can be used with any of the standard LaTeX classes and works correctly, for example, with `pdfscreen`,[8] a screen design package written for PDFTEX by C.V. Radhakrishnan.

It should be remarked that both packages have been extensively tested using the commercial TEX system by Y&Y, which uses dvipsone, and the MikTEX system for Win95/NT, which includes dvips and PDFTEX.

Final remarks

It is surprising what can be done with TEX and PDF. In the past, when I thought of TEX, I thought of a compiler and a collection of macros that could produce an outstanding typeset-quality article on paper. TEX, however, is capable of more than just black and white. When teamed with Adobe Acrobat and the Portable Document Format, TEX can produce colorful documents with the richness of user interaction.

TEX and PDF are a natural for putting educational material, especially technical material, on the internet. It is hoped that the use of "team AcroTEX" will continue to grow in the educational and TEX communities.

References

Bienz, Tim, R. Cohn, and J. Meehan. "Portable Document Format Reference Manual". Version 1.2, Adobe Systems, Inc., Mountain View, CA, 1996.

Bienz, Tim and G. Staas. "pdfmark Reference Manual". Technical Note 5150, Adobe Systems, Inc., Mountain View, CA, 1996.

Eijkhout, Victor. *TEX by Topic: A TEXnician's Reference*. Addison-Wesley, Reading, MA, 1992.

Goossens, Michel, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, MA, second edition, 1994.

Goossens, Michel, S. Rahtz, and F. Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TEX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, 1997.

Lesenko, Sergey. "The DVIPDF Program". *TUGboat* **17**(3), 252–254, 1996.

Merz, Thomas. *Web Publishing with Acrobat/PDF*. Springer-Verlag Berlin, 1998. Chapter 6, "pdfmark Primer", is available on-line from: `www.ifconnection.de/~tm/pdfmark/`.

Salomon, David. *The Advanced TEXbook*. Springer-Verlag, Berlin, 1995.

Sojka, Petr, H. T. Thanh, and J. Zlatuška. "The joy of TEX2PDF — Acrobatics with an alternative to DVI format". *TUGboat* **17**(3), 244–251, 1996.

Story, D.P. "Pdfmarks: Links and Forms". On-line documentation: `lnk_forms.html`, 1998a.

Story, D.P. "Using LaTeX to Create Quality PDF Documents for the WWW". On-line documentation: `latx2pdf.html`, 1998b.

[8] CTAN: `macros/latex/contrib/supported/pdfscreen`.