iTEX — Document Formatting in an Ereader World

William Cheswick
AT&T Shannon Lab
Florham Park, NJ
Email: ches@research.att.com

***Abstract*— TEX and other traditional text layout markup languages are predicated on the assumption that the final output format would be known to the nanometer. Extensive computation and clever algorithms let us optimize the presentation for a high standard of quality, designed by artists who are experts at document layout.**

But ebooks are here, and the iPad sold millions of units in the first few months after its introduction. Book readers offer a new way to store and read documents, but they are a challenge to high quality text layout. Ebook users are accustomed to selecting reader orientation, typeface, and font size. We probably cannot run TEX over a document every time a reader shifts position in his chair.

iTEX is an experimental document bundle format and a free iPad application that present documents exactly as they were rendered by TEX. The bundle (a tar of a standardized directory structure) contains precomputed page images for portrait and landscape layouts, in standard and large type versions. I hope this may be a suitable standard to encourage similar applications on devices like the Nook or the many versions of the Kindle, included in the same bundle.

1. Introduction

> In these days of author self-publishing, we must not forget the value of professionals. *Don Knuth* [6]

In March 2010 I served on three program committees, reading and grading dozens of papers. March is also when my iPad arrived, and it would have been a handy device for the job.

But the papers were all in PDF, formatted for either US or European standard page sizes. Devices like the iPad and Kindle are not large enough to show an entire traditional page, unless the image is shrunk to uselessness or the user scans the text using the device as a sort of large magnifying glass. And the PDF layout comes in a single version, with no thought of portrait *vs.* landscape viewing.

Just-in-time (JIT) page layouts are common now, including HTML, EPUB, and other "reflowable" display engines. These page layouts are fine for most web pages and novels, but the authors of technical works usually need fine control, or at least final editorial veto, over the final appearance of the document. This is certainly not possible if the user is selecting his favorite typeface and font size while reading the document. And JIT readers do not render mathematics well, often serving up images of formulae in non-matching typefaces.

There are efforts to bring higher-quality layouts to JIT formats. For example, the EPUB 2.1 Working Group Charter seeks native support for mathematics [3]. They are far behind TEX, which is now over thirty years old and has a large corpus and user community.

Good text layout is hard, a job for professionals, and users shouldn't be making these decisions. The designer knows the size and resolution of the device, and which fonts are appropriate. The designer's decision is likely to be different for portrait and landscape viewing modes. And the authors should be able to see the document in its final form, before it is published.

It *is* reasonable for a user to require larger type sizes to remedy vision impairment and suboptimal reading milieu. Book designers are familiar with large type versions of books and newspapers, and with iTEX can provide that option.

The iTEX iPad application and bundle format presented here are an attempt to bring these ideas together into a reader and a simple document bundle format. The bundle is a standard tar file of a directory containing several files with specific names and formats. (See Appendix I for layout details.) The application can display portrait and landscape orientations of normal and large type versions of a document, formatted specifically for the iPad. It can display `.itex` documents retrieved from the web, or copied into the iPad from iTunes. There is no iPhone version envisioned: that form factor is simply too small for most technical documents. (The additions to the application would be minor, and I am willing to be persuaded that such support would be useful.) I hope the bundle format will be adopted and used for reading software on other devices.

The rest of the paper is laid out as follows: §2 discusses related work. The details of the iTEX implementation are covered in §3. A general discussion appears in §4 and future work (of which there is a lot) is in §5. Finally, §6 concludes, and some useful details are found in the Appendices.

2. Related Work

There are a growing number of reader applications for the iPad. Most support reflowable text from EPUB sources, and display PDF files. Unlike iTEX, they have large, expert development groups and are quite full-featured.

Apple's *iBooks* and the Kindle application are two major readers. *iBooks* will display purchases from the iTunes store (some are free), as well as PDF and EPUB texts. The latter are now available from their *Pages* program, as well. PDFs are displayed as a shrunken page and one can use a two-finger gesture to zoom in. PDFs formatted for the iPad portrait size are displayed correctly. Double-taps zoom in a bit. Landscape

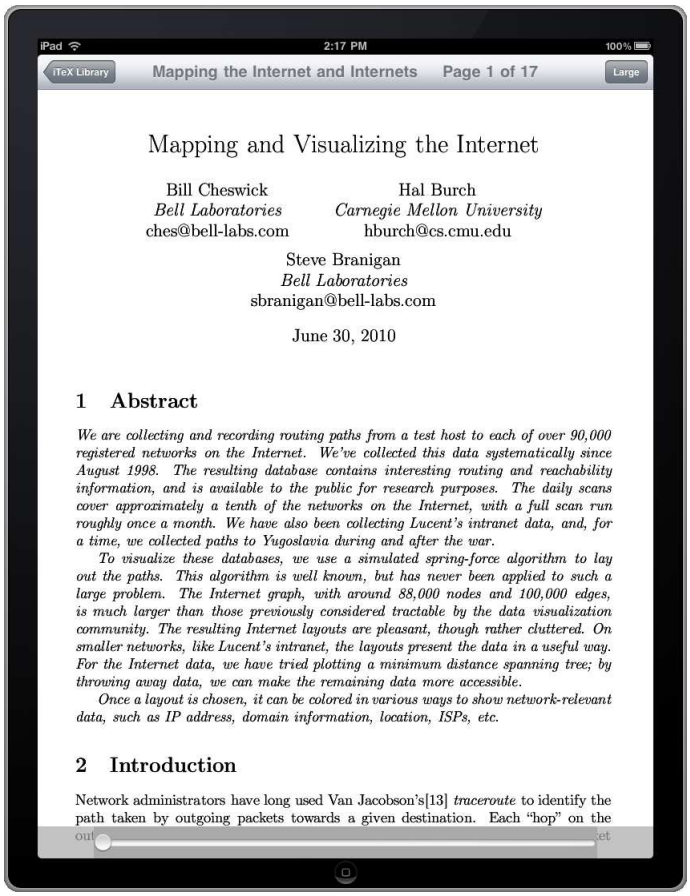

Fig. 1. Portrait view of a document page, with navigation bar and page selection slider.

display shows the portrait image, letterboxed, *i.e.* it is much smaller. *iBooks* uses page numbers, with different document lengths for the two orientations, which is also what iTEX does. It is not entirely clear to me how the page correspondence between portrait and landscape is accomplished, but it comes out about right.

The Kindle normally displays books purchased from Amazon's store. There is also a conversion service for downloading PDF documents for a small charge. PDFs can be loaded into the Kindle through a USB device without charge. Page position on the Kindle is shown as a percentage of progress through the document. Rotation of the iPad preserves the first word on the page.

The *Stanza* application offers access to a number of booksellers as well as open sources like Project Gutenberg. The portrait/landscape positions correlate well. Page numbers are confusing, and progress percentage is also shown.

arXiver offers access to the arXiV of scientific preprints [4]. It displays PDF versions of the documents, computed for page-size display and reduced on the iPad.

Kaveh Bazargan anticipated this iTEX work by a year [1]. At River Valley Technologies they process the original "author" TEX into XML, and then have an engine that automatically processes the XML into "slave" TEX and then PDF. These latter steps are amenable to implementation in an iPhone, and they demonstrate a nice result. The images are rapidly recomputed on-the-fly for orientation changes. This is quite a *tour de force*, but it seems like a lot of extra work, and the translation to a simpler TEX may induce errors.This seems a bit unwieldy to me, but perhaps they are on the right track.

This work should ot be confused with Don Knuth's vaporware, i(whistle)TEX [7].

3. Implementation

A. Where to Process the TEX?

iTEX's goals raise some interesting design decisions. In particular, where and when should TEX be run, and what exactly should the reader application do?

One possibility is to feed the document to the iPad and implement TEX/LATEX in its full glory there. That certainly makes all the information available to the reader, and the document is compact. But the TEX system is large, and the iPad is meant to be a simple device, mostly dealing with user interface issues. The CPU is fast, but not that fast, and there are battery life concerns when using a lot of CPU power.

Also, the author doesn't get to see the final result without an iPad. And document generation often involves a number of other utilities. In our firewalls book [2] we use *gv*, *bibtex*, *grap*, *pic*, *sed*, *grep*, *awk*, *convert*, and about a dozen scripts

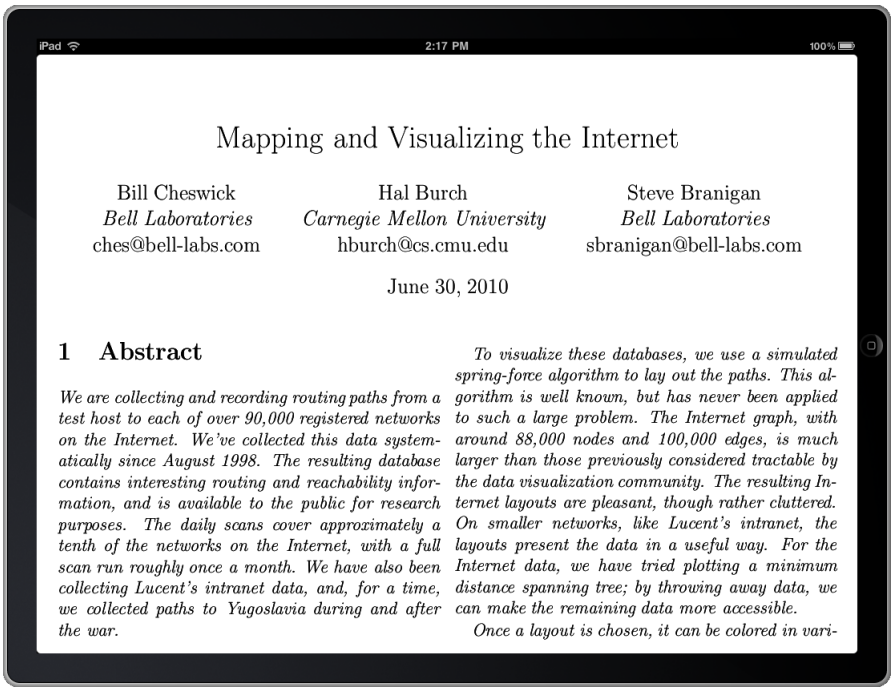

Fig. 2. Landscape view of a document page, with large text size selected.

for fixing the glossary, acronyms, generating the index, etc. These would not be available on the iPad: it is not a software development machine.

A second approach is to download the DVI files to the reader, and perform the rendering step there. This seems more promising: the layout work has been done, the DVI file is easily compressed, and all that is left is the font generation and placement. One could even use the resident Apple fonts, which would add a consistent feel to the application with Apple's efforts towards a uniform user experience.

It does mean that special fonts will have to be computed and loaded. In normal TEX processing, this may involve Metafont calls and libraries of precomputed font information. This seems like a lot to have to load into the iPad, and a source of reader disappointment if the proper fonts are missing.

It also means that files needed by DVI `\special` commands have to be downloaded somehow as well, and the iPad/DVI processor has to keep up with new specials.

For these reasons I chose to generate the images at the source, where the user is already accustomed to formatting his documents. The interface is much cleaner, which simplifies the application considerably. (It also would make redaction foolproof, since there is no hidden text.) But this approach does discard information that a good reader could use, so it will have to be augmented in the future (see §5).

B. Page image production

The production of quality page images from DVI would seem easy, but I ran into problems. When open source's thousand flowers bloom, which one do you pick?

I started with *dvi2bitmap*, but that turned out to be fairly rudimentary, creating output reminiscent of DECwriter printers. Greyscale and anti-aliasing were not available, nor was processing of most important `\special` commands.

Dvips post-processed with *gv* to generate PNG files worked fairly well, and could have been made to work. But there was a lot of mechanism there, and it was slow and fairly unwieldy. It does support all the PostScript-related `\special` commands, and may ultimately be the process of choice.

Finally I found *dvipng*, and it works well, creating the anti-aliased characters that rival Apple's own on the iPad. (The *dvi2bitmap man* page should have a pointer to *dvipng*.) *Dvipng* does have some bugs—one of my PostScript images causes it to crash, for example—but it will be the basis for a *dviitex* (see below). The program *genitex* is a crude prototype script that generates an iTEX bundle from many LATEX files. See `http://www.cheswick.com/itex` for details.

C. The iPad application

The iPad application is a pretty standard reader. It has a library screen (currently unsortable) and traditional navigation bars to change screens. Pages are turned with a flick of the finger, and a tap on the document display shows or hides the page slider at the bottom and navigation bar at the top. A navigation bar entry selects different versions, such as large type or others that may be supplied. Figures 1 and 2 display portrait and landscape displays of a version of this paper.

The app displays bundles downloaded into the app or by a web browser. Any file with an extension of `.itex` or `.iTeX` will be scanned and installed in the iTEX library on the iPad. Malformed bundles are skipped, with an error message.

The iTEX library is displayed when the app is started.

Bundles may be downloaded, replaced, or deleted with the usual iPad gestures. iTEX documents are keyed with the title and author: identical title/author pairs are deemed to be the same document, and only one is stored in the library.

4. Discussion

iTEX requires the generation of a fair number of final documents. In the simplest case, a technical document is likely to be generated for 8.5x11 inch format (or A4), plus two versions for the iPad. If a large type version is generated, that is two more LaTeX runs. As more devices are supported, perhaps including a "retina" version of the iPad someday, the iTEX bundle will grow to considerable size.

I don't consider this size to be a real problem. Download times will increase, though probably not significantly for WiFi and hardwired downloads. The iPad app uses only the relevant versions and discards the rest after download. If size becomes a problem, separate iTEX bundles could be offered for differing devices.

More important is the problem of generating multiple documents of varying format from the same source. There is trouble enough chasing down all the TEX nits (like overfull hboxes) when the final version of a document is ready to go. iTEX offers many more opportunities for these, since one is now generating a page-size version of the document, plus four versions just for the iPad. For really serious typographers, this requires human judgement for each version, and it is possible that the problem is over-constrained, that there is no single LaTeX source file that will look good in all versions. The page designer may have to cut TEX a little extra slack in the layout settings of some of the versions for author usability and automation schemes. It remains to be seen if the automated results work well enough.

Page numbers present a problem. In the current iTEX implementation, a document can have four different page lengths. When one changes orientation or type size, what is the appropriate page number to switch to? At present iTEX estimates based on the percentage distance through the document, which seems to work well enough for the reader.

But how do we convey document position to someone else? "There is a typo at the bottom of page 4" doesn't work unless we refer to the device, device version, orientation, and text size version as well.

It would be nice to have a URL format that gives word- or at least paragraph-position. TEX could collect this information by marking each word and passing the information on in the DVI file for processing.

iTEX has been available in the Apple app store since February 2011. There have been quite a number of downloads from non-English speaking countries. This raises the point that iTEX documents and the app have almost no linkage to the English language. Other than file names (UTF-8 is processed) and error messages, and the order in which pages are presented, iTEX has no need to know what language is presented, since they are just a series of images. It might be useful to supply an option in the iTEX bundle indicating that the document creator wishes the document to be displayed starting at the "right" end.

Apple does provide an easy means for internationalizing text in the error messages, and I should implement that.

5. Future Work

A. Improving the Application

The iTEX iPad application is an implementation of a simple reader, the product of my first Objective C project. While it has the basic features of a reader and is quite usable as such, it could use a fair amount of polish.

At present, the iTEX bundle stores a version of the document as a sequence of PNG images in a directory. These files could be replaced by a single PDF for each version and orientation. The next version will support this. More importantly, there is a lot of semantic data available during TEX processing that is or could be captured in the DVI file and provided to the application in a separate file, one per page, regardless of PDF/PNG choice.

For example, one cannot zoom into the precomputed text in iTEX. If the designer did his job correctly, it shouldn't be necessary, and large type versions should help those with vision impairments. But images and graphs certainly should be selectable, and viewed at arbitrary zoom levels. I expect to include EPS versions of these, and let the reader access them.

Other features are needed: words, footnotes, bibliographic references, glossary entries, and URLs should be active and tap-able. Selected text should be paste-able elsewhere on the iPad. No search function is available at this time. And it would be nice to have a note-taking and commentary feature, something that would have been very handy while I was reviewing papers.

The page images that iPad handles contain none of this information—it will have to be supplied separately. It will contain a list of rectangles for each word on the page and properties for each rectangle, and should be easily processed by the application each time the screen is touched. The precise format and contents of this information needs to be determined. I plan to modify *dvipng* into a new program, *dviitex*, to provide this data.

It might be useful to compute even more versions of a document, perhaps with extra-large font. This adds to the bundle length and generation time, but might be worthwhile for some reason. iTEX places no specific limit on the number and names of versions available, although the navigation bar software will eventually have too much to handle nicely.

B. New Devices

Since the iPhone 4 doubled its screen resolution over previous versions (the "retina" version), the same is likely to happen on some future version of the iPad. Document designers will care—the current iPad has only 132 DPI and hence needs low-resolution fonts like Lucida Bright. The iPad application will need modifications to deal with this.

If the iTEX idea is successful, there ought to be applications for the Kindle, and other reading devices. These devices have

other challenges for the document style designers, such as monochrome-only displays with limited grey scale support.

C. TEX library access

Adrian Petrescu has been thinking along similar lines, for the Kindle [8]. But he has suggested making a browser for arXiV, with a server that translates the arXiV text and downloads it into the device. This is a terrific idea, and I am planning to implement an arXiV browser window in the iTEX application and put up such a translator (with appropriate security, of course). I hope our two efforts can merge. The automated translator *genitex* mentioned in §3 could use some help from people with good LATEX hacking skills.

I understand that the Gutenberg Project may also be converting to LATEX. If so, a similar browser would be appropriate for their library as well.

6. Conclusion

iTEX was a useful project to help get me up to speed on programming iPhones and the iPad. After several months of intensive programming, I have a fair grasp of the Objective C language and many of the Apple libraries and UI design ideas. It was the first radically different language I have learned in twenty-five years.

But I think iTEX may fill a need, and perhaps will form the basis for a useful documentation publication format. I am prepared to work hard on supporting that effort, if there is sufficient interest. If not, I have other applications in mind, and this project was fun.

Acknowledgements

Without Dave Kormann's help, I never would have made a dent in Objective C: the documentation just isn't that clear.

Adrian Petrescu mentioned ideas about the Kindle and especially the arXiV access. I hope we can work together on this.

My thanks also to Barbara Beeton, Boris Veytsman, Will Robertson, Alexis Shaw, and to several other helpful folks who offered suggestions and comments at the TEX Users Group Meeting in San Francisco in June 2010. Barbara Beeton and Brian Clapper provided editing help as well.

7. Availability

The iTEX app is available for free in the iTunes store. Additional information, sample iTEX bundles, and source code are available at `http://www.cheswick.com/itex`.

References

[1] Kaveh Bazargan; *TEX as an eBook reader*, TUGboat 30:2 (2009), 272-273.

[2] William Cheswick, Steve Bellovin, and Avi Rubin. *Firewalls and Internet Security; Repelling the Wily Hacker. (Second edition).* Addison Wesley Longman, 2003.

[3] EPUB 2.1 Working Group Charter-DRAFT 0.11, (7 May 2010), item 9. Retrieved from `http://idpf.org/idpf_groups/epub21wg/IDPF-EPUB-WG-Charter-5-7-2010.html`

[4] Paul Ginsparg. First steps towards electronic research communication. *Computers in Physics*, 8(4): 390–396, 1994.

[5] Brian Kernighan and Rob Pike. it The Unix Programming Environment, Addison Wesley, 1984.

[6] Donald E. Knuth, Tracy L. Larrabee, and Paul M. Roberts. *Mathematical Writing*, page 14. Mathematical Association of America, Washington, D.C., 1989. ISBN 0-88385-063-X.

[7] Donald E. Knuth. An Earthshaking Announcement. TUGboat 31 (2): 121124. ISSN 0896-3207

[8] Adrian Petrescu. Personal communication.

Appendix I
iTEX bundle format

An iTeX bundle is a file with the extension .itex that is actually an uncompressed POSIX tar file. (These are quite easy to unpack on the iPad.)

The root directory contains:

`Title`	A file whose first line contains the document title, in UTF-8. The rest of the file is ignored.
`Author`	Similarly the authors on the first line, UTF-8.
dir	One or more directories containing device-dependent versions of the document.

Each directory *dir* is named for the document type it contains. The names may be one of:

`iPad`	original iPad
`iPad3`	"retina" version of iPad
`iPhone`	original iPhone
`iPhone4`	"retina" version of the iPhone
`Kindle`	original Kindle
`Kindle2`	newer Kindle
`xoom`	Motorola android Xoom

At this time, the iTEX app only installs and uses the `iPad` directory. Non-Apple names are tentative, to be chosen based on display details for the various devices.

Each of these device directories contains a subdirectory for each version of the document. The version names are displayed on the top of the document display, and may have any short name. By convention:

`normal`	the default document version
`large`	the large-type version.

Neither the names nor the number of versions is enforced, but the app may have trouble displaying them usefully if there are too many, or they are too long. All these names honor UTF-8 format.

Each version contains two directories, `p` and `l` for the portrait and landscape orientations of the version of the document. At present, each of these directories contains a series of page images as PNG files of the form `%04d.png`, starting with `0001.png` as generated by *dvipng*. In future versions of the iTEX app there will also be page-related data in files named `0001.dat`, `0002.dat`, etc. in these directories. The multiple PNG files will be replaceable by a single PDF file. There will also be EPS files for images in the document, and perhaps other related files.