E-books and Graphics with LaTeXml

Deyan Ginev[1], Bruce R. Miller[2], and Silviu Oprea[3]

[1] Computer Science, Jacobs University Bremen, Germany
[2] National Institute of Standards and Technology, Gaithersburg, MD, USA
[3] University of Oxford, Oxford, UK.

Abstract. Marked by the highlights of native generation of EPUB E-books and Tikz support for creating SVG images, we present an annual report of LaTeXML development in 2013. LaTeXML provides a reimplementation of the TeX parser, geared towards preserving macro semantics; it supports an array of output formats, notably HTML5, EPUB, XHTML and its own TeX-near XML.
Other highlights include enhancing performance when used inside high-throughput build systems, via incorporating a native ZIP archive workflow, as well as a simplified installation procedure that now allows to deploy LaTeXML as a cloud service. To this end, we also introduce an official plugin-based scheme for publishing new features that go beyond the core scope of LaTeXML, such as web services or unconventional post-processors.
The software suite has now migrated to GitHub and we welcome forks and patches from the wider FLOSS community.

1 Introduction

Another busy year of LaTeXML development has gone by; while we've not completely accomplished all the tasks we'd hoped for, we've finished others including some we hadn't originally planned.

- finished move to GitHub, fork us and send patches!
- Project is alive and stronger than ever (do "gitstats /path/to/git-checkout-of/LaTeXML html; firefox html/index.html"). We have had a steady increase in commits - 200 in 2010, 400 in 2011, 600 in 2012 and 800 in 2013.
- We have a brand new CPAN-friendly, plugin-based contribution model for fancy features
- added support for EPUB (discussion on mobi?)
- added support for Tikz (dsicsussion on XYPic and other formats?)
- native ZIP conversion workflow, with arXiv's algorithm for determining main TeX files.
- ? one-pass conversion to HTML (with Bibliography)
- ? Additional commitment to quality control in development (perltidy, perlcritic)
- cloud-friendly - now possible to install locally with cpanm on cloud platform such as Heroku (I will try that today!)

2 Reorganization

We have reorganized both our code development and our code base. In the first sense, we have moved our repository to GitHub, see `https://github.com/brucemiller/LaTeXML` where you can more conveniently browse our code, or check out the latest version. We have also ported our Trac ticket database to GitHub's Issues, so you can also report bug and request features from the same place.

Along with the move to github came more opportunities to share code and development which called for clearer code standards. We have made a commitment to code quality and formatting by adopting `perltidy` and `perlcritic` policies which are automatically enforced by `git` mechanisms.

In the second sense, we have reorganized the code to more clearly separate the code related to the separate phases of processing, but allow better connection and code sharing for when more complex processing is called for, such as carrying a single TEX source file through the full processing to HTML, or even ePub (see 3). In particular, it provides better support for the Daemon mode of usage, with a separate web-service module available.

This reorganization positions us to develop an plugin architecture that will allow modular extensions that cover both new LATEX styles and bindings, but also include enhanced postprocessing for more sophisticated applications such as sTEX.

And some tentative plugins are already available?

3 E-books

The newest version of ePub, version 3, incorporates MathML to represent mathematics. LATEXML is already generating HTML, with embedded MathML, and allows that output to be split into multiple pages as specified by the user. Additionally, it provides for zipping up the resulting directory into a `zip` archive. With appropriate rearrangment of the peices, and the addition of a Manifest of the correct structure, we have the basics needed to generate ePub documents.

That sounds too easy, doesn't it?

4 Graphics

Before we turn our attention to graphics, a brief digression may be in order. There are two main approaches used to generate HTML from TEX. The first approach, exemplified by `tex4ht`, uses the TEX engine itself by redefining certain commands to drop `\special` data into the `dvi` output file. Instead of `dvips`, a special `dvi` processor then deciphers the `dvi` and `\specials` to infer and construct the appropriate HTML. In the second approach, used by LATEXML, a program is developed which emulates TEX for the most part but interprets some macros (called "Constructors" in LATEXML) specially, so that it produces XML directly.

The first approach has the advantage of at least allowing the processing of arbitrary TEX and LATEX packages, whether or not the resulting HTML ... The challenge is in the TEX programming necessary to insert the `\specials`, generating valid HTML, and in whether sufficient semantic structure can be recovered from the `dvi`.

The second approach has the advantage of having more direct control of the generated output. It is easier, though not trivial, to extend to new XML structures.

The challenge is to emulate TEX sufficiently to process complex packages. Furthermore, LATEXML: uses an intermediate XML format which preserves the semantic structure; is fundamentally XML aware, so it produces valid XML; A feature of LATEXML bindings is that macro control sequences can be defined to be "Constructors" which directly construct the XML representation of their content.

In either approach, LATEX packages that define macros with semantic intent must be dealt with individually or else the semantics will be lost.

Within that context, we were skeptical when Michael Kohlhase initially posed the proposition: Was LATEXML's engine good enough to implementing the `tikz` package? Presumably any semantics implied by tikz markup isn't so critical? The package is so large and complex, not to mention its development so fast-moving, that creating LATEXML-specific bindings for all its commands impractical. However, it is designed with a relatively small driver layer in mind, and even has a `tex4ht` driver for producing SVG!

The main tasks, then, were to implement LATEXML bindings for that driver and improve LATEXML's engine to cope with the sophisticated TEX macro usage in the higher layers of `pgf` and `tikz`.

Ultimately, we succeeded beyond our expectations. Although the results are not perfect, LATEXML now successfully processes 3/4 of the first page of `tikz` examples on the `http:texamples.net` site, generating valid HTML5, with text and MathML combined. In contrast, `tex4ht` succeeds on slightly more than half the examples, often producing invalid markup, and doesn't support MathML embedded in the `SVG`. It must be admitted, however, that LATEXML is *very* slow at processing `tikz` markup!

In the process, we have further improved the fidelity of the TEX emulation, introduced a (currently very rudimentary) mechanism for estimating the size of displayed objects and exercised the integration of both MathML and `SVG` into HTML. These improvements are beneficial even outside the graphics

Areas needing further work are `tikz`' matrix structure which currently clashes with LATEXML's handling of alignments; inaccuracies of LATEXML's sizing of objects; and, of course, examples involving other exotic packages not yet known to LATEXML.

We are hoping to leverage this experience and apply it to supporting the `xy` packages, another popular and powerful system. It seems to have a less well-defined driver layer and we are in the early stages of discovering a small set of macros that could serve that function. We've had some preliminary success, however.

We already have minimal support, for the `pstricks` package, but with its Postscript oriented design, it is more time consuming to develop further bindings.

5 Outlook

Daemon is merged,

References