Revisiting WYSIWYG Paradigms for Authoring LaTeX

David Kastrup
Kriemhildstr. 15
44793 Bochum
Germany
David.Kastrup@t-online.de

Abstract

While the command-driven approach of TeX/LaTeX has shown its power and flexibility for a variety of purposes, the lack of immediate visual feedback often renders the authoring and reviewing process itself somewhat inconvenient for both beginners and experienced users. The idiosyncratic TeX syntax does not lend itself readily to proofreading and sustained composition where the input syntax differs considerably from the produced results.

A number of approaches trying to deal with this frequently perceived shortcoming will be illustrated. On the input manipulation side there are tools ranging from a token-based approach (Syntax Highlighting, X-Symbol) to complete editors using LaTeX mostly as a means of exporting documents (TeX$_{\text{MACS}}$, LyX). A different range of tools does not aim to provide differences in editing, but fast and convenient access to the output results, mostly in the form of a separate page-oriented preview (Whizzy-TeX, Instant Preview). While the performance of those implementations by now leaves little to be desired, a demand for a tighter coupling of source and preview for editing purposes remains. 'Source specials' provide one way for facilitating cross-navigation between source and previews. The preview-latex package (by the author) provides a much closer coupling by directly placing previews of small elements into the source buffer.

The Moving Finger writes; and, having writ,
Moves on: nor all thy Piety nor Wit
Shall lure it back to cancel half a Line,
Nor all thy Tears wash out a Word of it.

Omar Khayyám

The LaTeX/WYSIWYG clash

What is WYSIWYG? WYSIWYG, an acronym for "what you see is what you get", is really a marketing term applied to several different degrees of similarity between the input window and the typeset output from running a system.

In its strictest sense, it means that the typeset output will be identical to the screen display. Of course, this goal is ultimately impossible: print devices have different characteristics than the screens we are working on: different pixel resolutions, different gradation of colors and gray levels. A screen dump from a WYSIWYG type word processor will typically look awful compared to a regular printout.

So what are the things people have come to expect from the WYSIWYG moniker?

Similarity to print: The editing window is supposed to resemble the printed output.

Letter shapes: While pixel accuracy is not to be expected, one might at least be able to see the general shape of the letter. Display devices often offer considerably finer amounts of control for pixel intensity than printing devices do; this makes antialiasing feasible. Antialiasing tries to compensate for a lower spatial resolution by varying the brightness of pixels according to the amount of ink that would cover the pixel given higher resolutions. Antialiased letters tend to be more readable than their non-antialiased variants, but look rather blurry as the lack of spatial resolution itself is not overcome, only its impact on the local grayness level reduced.

Extended character sets: TeX and LaTeX itself typically use an input representation based on ASCII. While there are extensions in order to be able to access 8-bit character sets, most mathematical special characters and operators such as $\sum$ and $\int$ are entered as control sequences such as `\sum` and `\int`. WYSIWYG systems usually

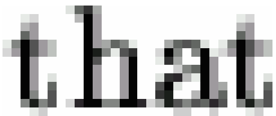


Figure 1: Antialiased TEX font.

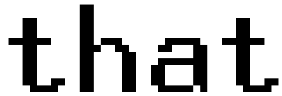


Figure 2: Typical screen font.

offer a more readable rendition of such characters, as well as convenient ways to enter them without knowing their names.

Non-text elements: The most important items to mention here would be mathematical formulas and tables. Those are distinguished by conveying information more directly than possible textual counterparts.

Line breaks: WYSIWYG word processors typically have the same line breaks in the printed output as they have on the screen.

Pagination: Page breaks and figure positioning are also common elements that WYSIWYG processors will apply to the edited source in the same manner as the resulting output.

So where's the clash? It turns out that some of those targets are not the best idea for screen-based editing and document creation in general, and some are at particular odds with the way TEX works:

Similarity to print: TEX is basically a programming language. Quite often complicated instructions lead to the final typesetting results. Editing those in a WYSIWYG manner is hardly possible. WYSIWYG tends to hide abominations in the input: while text processors such as Word offer various possibilities to structure your input (format templates, text styles and so on), this is not immediately apparent in the output. As a result, average users of such systems sometimes employ rather abominable means for the formatting of their texts. Not uncommon is the use of excessive amounts of single spaces for indentation and spacings, and hand adjustments that are not robust against changes of fonts and/or printer. TEX provides an easy way of introducing comments into the typesetting source (`%` introduces comment lines). Such comments have no good place in a true WYSIWYG display.

Letter Shapes: In connection with TEX, the most commonly employed fonts are the Computer Modern family by Knuth. Characteristic for these fonts is a delicate balance between various stem widths and hairlines that produces a 'closed' look of the letter shapes itself while still preserving the 'leading' characteristic of serifed letters in the overall grayness level. A typical example is lower-case "t" which has a closed bowl when viewed closely, but a distribution of stem widths that effectively makes the visual impact of the closing hairline diminish (compare figure 1 to the letters in the text of this document).

Screen resolution is inadequate to properly reflect those visual characteristics. For best legibility, fonts designed for computer screen usage are to be preferred.

Extended character sets: These can lead to an easily implementable improvement in legibility. While the application is straightforward, the benefits are limited.

Non-text elements: These benefit the most from WYSIWYG representations. TEX offers considerable power for mathematics, and many extension packages make use of its macro programming features in order to gain additional functionality. For that reason, adequate rendering of compositions like math formulas and tables necessitate considerable programming efforts in the editor in order to support them well. An appropriate display of those elements is a significant aid for developing a stream of thought, and for copy editing.

Line breaks and Pagination: TEX finds its line breaks by paragraph-global optimization, and employs a process of somewhat localized optimization for finding its page breaks. An insertion mechanism caters for determining the proper amounts of additional material (such as footnotes and figures) to attach to a page. All decisions are governed by the evaluation of various kinds of penalties, and several criteria spanning more than one line are employed (visual compatibility of spacing in subsequent lines, extra penalties for adjacent lines ending with hyphenated words and so on).

The non-local layout optimization of TEX is important to achieve the best typeset results. Maintaining this during text entry itself initially appeared infeasible primarily because of performance considerations (see later for examples where this has been mostly overcome). It also can be distracting: the resulting repeated extensive on-screen text rearrangements resulting from small changes in the source make it hard to keep focus on the actual editing location.

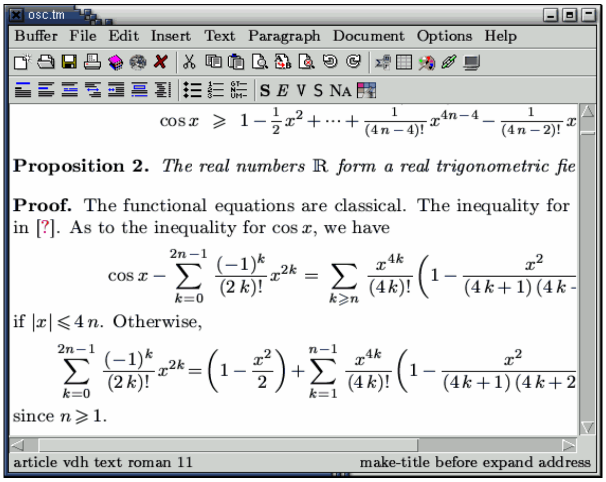


Figure 3: Screenshot from TeXmacs window.

Various systems bridging the gap

TeXmacs The word processor TeXmacs is about the clearest demonstration about what a full implementation of the WYSIWYG paradigm could look like for TeX. While TeXmacs does not make use of TeX itself nor offers convenient access to TeX packages and programming, it employs TeX typesetting algorithms and fonts (antialiased on screen) for its operation. The printed rendition of the pages is quite the same as the screen representation (though having your editing window paginated is optional). It does not support LaTeX as a native format, but exports to it (needing a special style file) and, considerably less reliably, also imports from it. Its keybindings are reminiscent of Emacs' keybindings, and it will interpret quite a few macro sequences introduced with backslash. The editor can be extended and customized in Guile, the GNU project's version of the Lisp-like Scheme language.

For document versions exported to LaTeX, it is possible to include direct code passages that are passed on verbatim, and that may differ from code that is used when the document is printed natively.

On a 200 Mhz system, TeXmacs appeared to respond somewhat sluggishly. While most people used to LaTeX may get reasonably well along with TeXmacs, accessing the power of LaTeX and document exchange with other LaTeX users will be somewhat problematic.

Personally, I have found the 'concertina effect' distracting: constant reformatting during text entry causes the line spacing to shrink until the line gets wrapped differently again. While the immediate feedback from keystroke to final output is beneficial for administrating the final touches to a document, the constant fervor with which paragraphs get adjusted and shifted while single letters are being typed is about as useful and convenient as constantly busy window cleaners on a construction site.

LyX Similar to TeXmacs, LyX is a complete word processor. In contrast to TeXmacs, however, LyX promotes the WYSIWYM (What You See is What You Mean) buzzphrase instead of WYSIWYG. In earlier versions of LyX, this was mostly a euphemism for 'what you see is somewhat reminiscent of what you will get', but in more recent versions a lot of work has been invested to make the display indeed show additional information about the underlying structure of the document. Since LyX is not bound to have the input's appearance match the output, it can generously apply colors for outlining structural details, and use push buttons and similar graphic elements impacting the editing window layout for indicating footnotes and cross references.

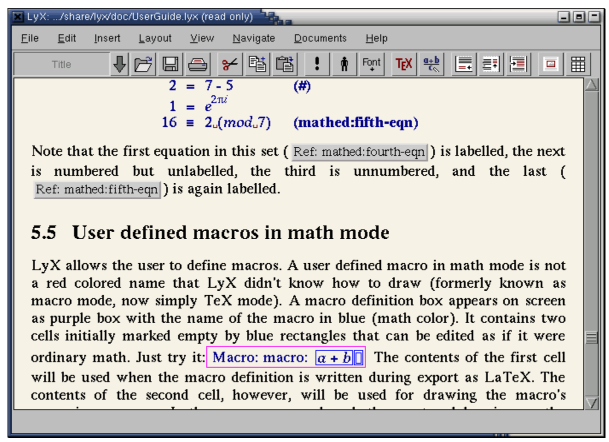


Figure 4: Typical LyX display.

While font changes and the like are reflected in the screen font selection of LyX, screen and print fonts are not the same. The screen fonts are configurable as part of the user interface, the print fonts are part of the document. It must be noted that LyX's screen display, particularly with regard to mathematics, does not seem to be optimized for readability. LyX uses LaTeX for its print typesetting, employing ordinarily available class and style files. Naturally, LaTeX export is unproblematic; importing it, however, is not without its difficulties. LyX uses its own formats for saving files; as with TeXmacs, sustained document interchange with other authors using pure LaTeX is infeasible.

It is possible to embed LaTeX code into documents even where LyX does not cater for it specifically: when LyX is unable to convert LaTeX phrases into its own format, it retains them as "Evil Red Text" (ERT) which is passed without modification into the exported LaTeX code. It is a declared goal of the LyX developers to eventually obliterate most of ERT by letting LyX natively support more and more LaTeX constructs. Considering the extent covered by LaTeX (a lot of which does not seriously gain usability by a separate screen representation), this means a continuing drain of resources. In particular, things like LaTeX3 will need much work to accommodate.

One of the strengths of LyX is its math editor, quite similar to that of TeX$_{\text{MACS}}$. LyX has the same update-per-keystroke policy that, in connection with justification, leads to the concertina effect of shrinking and expanding lines during normal text insertion.

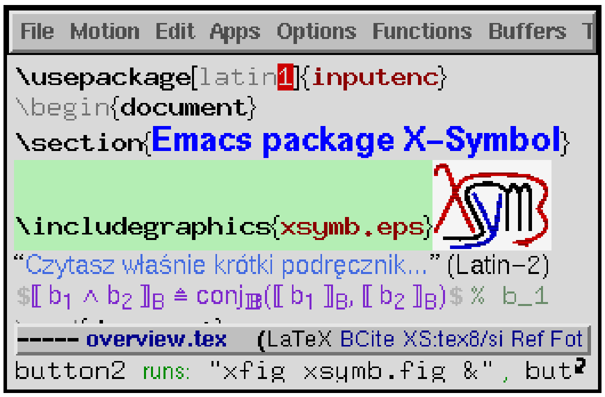


Figure 5: X-Symbol at large.

X-Symbol X-Symbol is the first system presented here falling short of providing a complete text processing system 'merely' for the sake of introducing WYSIWYG into TeX authoring. It is not an accident that, like all solutions presented here not coming with their own editor, it runs under the extensible editor Emacs (or its variant XEmacs). Emacs is

- intended for editing source texts
- free software
- a powerful editing environment
- easily extensible with a highly interactive extension language (Emacs Lisp)
- available for a large number of different operating systems

Since TeX is a multi-platform free software system with an editable plain text source language, Emacs and TeX are matched rather well. The most commonly employed package for Emacs providing an extensive editing and runtime environment for TeX and LaTeX is called AUC TeX.

So what does X-Symbol do? It pretty much exclusively deals with the 'extended character set' angle of WYSIWYG. It will display the likes of `\sum` and `\leq` as "$\sum$" and "$\leq$" and will also cater for accented letters in text, such as `\"a` being displayed as "ä". Its operation is limited to unique short control sequences. Some of the employed fonts come with X-Symbol and fit in with the normal monospaced screen fonts of Emacs. This means, for example, that operators like $\int$ will look rather small on-screen. As an added convenience, it has special fonts that it can use in connection with syntax highlighting to make super- and subscripts appear with smaller letters and appropriately displaced. While it will automatically convert control sequences into its special characters as soon as they are typed, it also offers a few other methods of directly accessing those characters.

X-Symbol actually replaces the characters in question in the source text buffer, and just converts them back into their TeX equivalents when writing out files. There are several potential problems which can lead to files being permanently changed:

- If for some bug the text does not get converted to TeX readable form when written out, you'll end up with basically illegible garbage. Some effort is needed to get control sequences back.
- The combination of reading file in under control of X-Symbol and writing it out again is not guaranteed to be unique. This is particularly a problem in verbatim-like settings. Recent versions of X-Symbol behave much more predictably and cautiously than previous ones, however.

Since the internal presentation of the text is changed to extended characters, this means that searching for those control sequences with the usual search functions of Emacs becomes awkward to infeasible.

One part of the value that X-Symbol provides to the user is an editor-level replacement for LaTeX's inputenc package (for plain TeX, this could be even more important). Using X-Symbol is a particularly convenient option when your text would otherwise mandate switching between input encodings in a single document. In that case, a coherent editor display would be hard to achieve. With X-Symbol, the characters from the 'foreign' encoding are expressed in appropriate control sequences when saving, obliterating this particular problem.

But the most important addition of X-Symbol is probably the support for numerous ways of inputting the characters it caters for, ranging from keyboard shortcuts to the 'grid' (a large menu with all of the available characters displayed in an editing buffer) and entry via the menubar. Apart from the grid, those techniques are not really WYSIWYG-related.

X-Symbol is mainly an input prettifier, converter and accelerator: it does not provide for any previewing facilities. It also does not cater for any more complicated compositions, like formulas. In that respect it does a lot less than systems like LyX. On the other hand, it provides no obstacles or inconveniences with regard to accessing the full power of LaTeX, and it will work also with SGML or HTML as well as plain TeX.

See the illustration in figure 6(b) for a more typical illustration of the visual buffer effect, this time in connection with preview-latex.

preview-latex preview-latex also is an Emacs add-on package. In contrast to X-Symbol, it is concerned exclusively with the display aspect of WYSIWYG. It does not change the text of the edited buffers at all, and has no impact on the format of external files. While the aim of X-Symbol is to replace single control sequences with letters and symbols matched to the screen fonts, preview-latex interchanges the source text display of whole compositions (formulas, section headers, figures, included graphics) with a proper antialiased preview obtained by running the LaTeX passage in question through LaTeX, Dvips and Ghostscript. A single call of LaTeX (potentially with a pre-dumped format) can provide previews for the entire document, and Ghostscript will be used as a rendering daemon working from a single PostScript file, processing those images currently on-screen with priority. Future versions will mostly bypass Dvips and Ghostscript, but update speed is already quite workable even on modest hardware. Another future option will be to replace LaTeX and Dvips with PDFLaTeX.

The previews are made to match the default screen font in background and foreground color, as well as in scale. Other than that, they are identical to actual print previews. Since preview-latex does not know about compositions' inner structure, it will just provide mundane source text display while you are editing them, switching back into graphical preview mode when you indicate you are finished.

What elements in a text are actually considered previewable compositions is determined by an external LaTeX style. Its operation can easily be customized by placing declarative commands in the document preamble or a separate configuration file.

The underpinnings of preview-latex are not particular to the Emacs editor: similar functionality is being implemented for LyX, currently restricted to its math mode.

Whizzy-TeX Whizzy-TeX is one of a number of systems that focus on automatic fast updating of a print preview in a separate window. It is (surprise, surprise) an Emacs package and best complemented with the previewer Active-DVI (written in Objective CAML) from the same author, since that previewer can switch to the correct page and location without flashing when the DVI file changes. While you can use Whizzy-TeX also with XDvi, the update action is less smooth.

Whizzy-TeX is a preview system that continually tracks cursor movements and text changes, and in case of a change, reruns LaTeX from a recent point where it has made LaTeX dump its state into a format file. That way, the DVI updates occur quite fast, and it becomes feasible to play around with stuff influencing typesetting decisions.

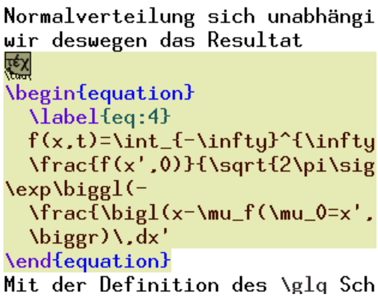
Take a look at subfigure 7(e) for an illustration of Whizzy-TeX in connection with Active-DVI.

ActiveTeX/Instant Preview ActiveTeX's core is a constantly running TeX process called the TeX daemon, which typesets pages on demand. A separate program then processes individual pages from the resulting DVI file as they get produced.

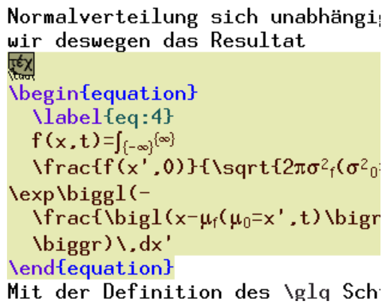
The Instant Preview package for Emacs will use this for keystroke level updates of a TeX buffer. Since TeX does not get restarted, the material processed in this manner should mostly be stateless so that repetitive runs work well. For that reason, the system is mostly unsuitable for LaTeX. The principal author of the system uses XDvi for the display; it might well be that Whizzy-TeX's Active-DVI could provide a smoother update action and avoid flicker.

The main aspect of ActiveTeX is raw speed on low hardware. Apart from that, it offers little if any advantage over Whizzy-TeX and serves similar goals, while being less well-suited for LaTeX.

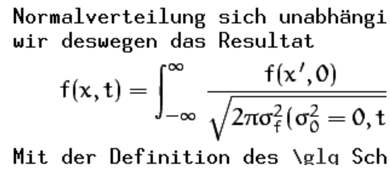
Source Specials Source specials are an editor/previewer coupling tool that works by placing special marks into the produced dvi file that indicate the source location where the dvi file results originated from. These can be either inserted with a special LaTeX style, or automatically by most newer TeX implementations. In combination with support in both editor and previewer, one can implement forward search (the position in the editor gets automatically tracked in the preview window) and reverse search (clicking into the preview window will relocate the cursor in the editor window to the corresponding source line). This is exclusively a cross-navigational tool. It serves no actual WYSIWYG



(a) Opened equation

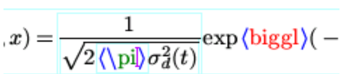


(b) The same, with X-Symbol

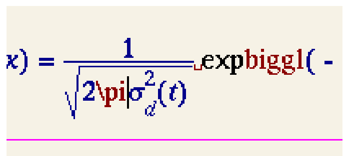


(c) The same, closed

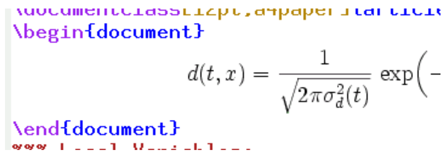
Figure 6: Using preview-latex with and without X-Symbol.



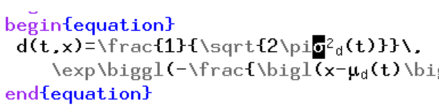
(a) TeXMACS



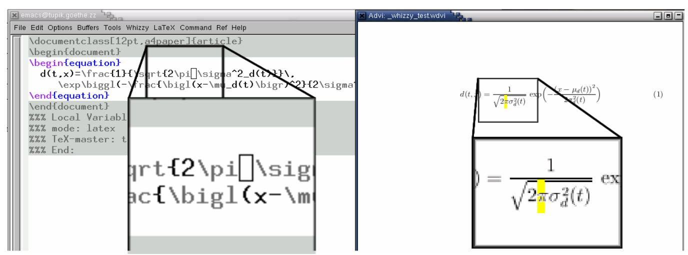
(b) LyX



(c) preview-latex



(d) X-Symbol



(e) Whizzy-TeX

Figure 7: Detail views.

functionality. We mention it here because it illustrates the perceived need for a closer coupling between editing window and preview.

Summary

Figure 7 shows screen shots of how the various systems treat a formula in display, and an overall summary can be found in table 1. Those WYSIWYG systems that try to provide a more customary input experience suffer from the handicaps of

- having to implement a complete editing environment on their own.
- needing to be able to interpret all of the supported constructs by themselves instead of letting LaTeX do the work. In that way, only a selected subset of LaTeX can be supported properly and efficiently, and the developers of both TeXMACS and LyX have chosen to employ a native format different from LaTeX for working purposes, which makes accessing LaTeX as an external interchange format largely unfeasible. Take a look at figure 7 for how the unknown delimiter size specifiers are treated by those first two systems.
- Needing to always show a coherent editing display during operation. This makes it infeasible to actually generate the display with LaTeX.

Systems with keystroke level reformatting and justification in the input window some users find distracting for continuous text entry. This sort of operation costs considerable performance on slower systems, and it renders the canvas unquiet. Repercussions to larger areas are counterproductive in creation mode, while desirable for admistering final touches to a document. While Whizzy-TeX also does keystroke update, it does so in a separate area which is less of a distraction but has the disadvantage of requiring a different focus of attention in case you actually need to look at the typeset contexts. preview-latex economizes updates (only done

Table 1: Ratings of available tools.

	TeXmacs	LyX	X-Symbol	preview-latex	Whizzy-TeX
Portability	−	−	−	0	+
Source preservation	−	−	+	++	++
Keystroke updates	yes	yes	yes	no	yes
Uses editing window	yes	yes	yes	yes	no
Edit responsiveness	−	0	+	N/A	N/A
Edit accuracy	++	0	−	+	N/A
Preview speed	++	0	N/A	+	+
Edit screen estate	+	+	+	+	−
Preview estate	+	−	N/A	+	−
Cross-navigation	++	−	−	+	0

Description of categories:

Portability	How easy will it be to transfer this system's mode of operation to other editors and editing platforms?
Source preservation	How faithfully will existing LaTeX document source be preserved when editing?
Keystroke updates	Does the system perform its updates on a per keystroke level?
Uses editing window	Does the system work within the editing window?
Edit reponsiveness	How fast does the system process keystrokes while editing? N/A if the system has no influence on keystroke processing time.
Edit accuracy	How close to the typeset result is the representation in the editing window?
Preview speed	How fast will a true preview be available?
Edit screen estate	During normal operation of the system, how much screen real estate is needed?
Preview estate	If in need of a true preview, how much screen estate will be needed? N/A for packages which don't have a default way of previewing.
Cross-Navigation	How easily can we establish the correlation between a true preview and the corresponding source position?

Description of ratings:

++	very good
+	good
0	fair
−	unfavorable
N/A	not applicable

on request, which is very easy), screen real estate and focus, at the cost of not providing any preview of the current object you are editing. This can be improved somewhat by employing X-Symbol which offers additional input convenience.

The upcoming combination of LyX's math editor with preview-latex-like functionality is an interesting development. While experienced LaTeX users might prefer linear text entry, the combination of easy access to both readably composed input as well as perfectly typeset LaTeX code will be a great help for advanced users exploring beyond the LaTeX constructs still fully featured by the LyX editor.

On the input side of WYSIWYG, a reasonable compromise under Emacs is provided by X-Symbol. On the preview side, once you leave the specialized editor area (where TeXmacs offers the most integrated approach), the preview-latex paradigm appears most useful for general implementation in editing systems: use of LaTeX for the typesetting ensures high accuracy while yielding full and unencumbered access to the full power available from LaTeX. For the kind of syntactical units that preview-latex processes, the lack of a per-keystroke update policy (unique among the presented tools) is in practice an advantage since it allows the user to compose the unit without distraction and commit it only when it is actually ready to be run through LaTeX. For interactive changes in connection with adjusting page layout material, Whizzy-TeX provides the user with fast updates. The added screen estate, and the separate preview area render it less optimal for other tasks in copy editing and document creation. That Whizzy-TeX has two competing screen locations of interest becomes apparent in figure 7(e): no other screen shot required the use of magnification glasses.

An advantage of Whizzy-TeX is that it provides a preview framework which can easily be embedded into different editors: it should be conceivable to adapt it to provide an alternative previewer for LyX, for example.

Future developments

The complete text editing environments will continue to support more LaTeX constructs. Performance increases will not have much of a further impact on the usability of currently available approaches. The most interesting approaches to watch may be hybrid ones which may make their way into non-Emacs based editing solutions eventually. As an example, LyX will come with functionality similar to preview-latex in its next major version (presumably 1.3.0), at first just for previewing math (due to technical reasons).

Prospective embeddable TeX components (Ωlibraries, DVI-rendering daemons) might make impact on the operation of display engines. The ligaturing and compositing mechanisms that constitute TeX's backend might make a good object for integration into existing word processors in a manner similar to TeXMACS. Perhaps drop-in equation creation components based on TeX code might become more prevalent in free software systems eventually.

Availability

Here is where the packages can be found on the Internet (the leading `http://` has been omitted):

TeXMACS	`www.texmacs.org`
LyX	`www.lyx.org`
X-Symbol	`x-symbol.sourceforge.net`
preview-latex	`preview-latex.sourceforge.net`
Whizzy-TeX	`pauillac.inria.fr/whizzytex`
Active-DVI	`pauillac.inria.fr/advi`
ActiveTeX	`www.activetex.org`
Emacs	`www.gnu.org/software/emacs`
XEmacs	`www.xemacs.org`
Src Specials	`xdvi.sourceforge.net/inverse-search.html`
Ω	`omega.cse.unsw.edu.au`

All of the described packages are released under the GNU General Public License and are thus free software (LyX is released under a modified version in order to allow linking with the XForms library).