Developing a Multi-Windowing Environment for Research Based on TEX

Michel Lavaud
C.N.R.S.
GREMI, Université d'Orléans,
45067 ORLÉANS Cedex (France)
Internet: lavaud@centre.univ-orleans.fr

Abstract

We have devised an experimental program, AST$_E$X, which provides an easy to use multi-window environment adapted to research work. It runs on any PC and is built on the top of a commercial all-in-one software (Framework). It endows it with scientific capabilities by coupling it to emTEX, Maple, Fortran and other scientific software. It allows for the easy modification of the structure of large multi-author documents and the performance of numerical and formal computations from the document. It adds a hypertext file manager, a preprocessor of LATEX structure, hypertext help and hypertext archiving of messages, among others. It can use the multitasking capabilities of Desqview or OS/2.

Many of the functions of AST$_E$X could be implemented also on the top of other existing software (commercial or public domain), provided they are endowed with an internal programming language which is powerful enough. We hope this could be done with GNU *emacs*.

Several commercial WYSIWYG scientific word-processors on PCs are now able to produce very nice output. Since the advantage of TEX is diminishing as concerns quality of output, some leading TEXperts have concluded that it is becoming too old, and have proposed creating a New Typesetting System from scratch, which would incorporate all aspects that are missing from TEX.

On the other hand, more and more scientists have access to international networks, and they are now using TEX as a language in the *linguistic* sense of the term, i.e., as a *means of communication.* This implies that TEX must remain stable in time as much as possible, for it to be able to fulfill this communication function.

We suggest that keeping TEX unchanged, as desired by many users, is not incompatible with building easy-to-use and powerful TEX-based software, as desired by TEXperts. This can be done by improving front ends and back ends to TEX and making them cooperate together via a multitasking OS.

In this article we describe a program, AST$_E$X, that we have written and that illustrates this point of view. It might provide — we hope — some guidelines for future developments in this direction.

Existing interfaces to TEX

In his article about the Future of TEX (Taylor, 1992), Philip Taylor described how painful it was to use TEX in the early eighties. Although he assured that TEX users enjoy this way of working... for those who do not, there are now several user-friendly public domain interfaces to TEX and related software, that make its use much easier! The first one is the AUCTEX Lisp package for GNU *emacs*. It is extremely powerful since it is based on the complete version of *emacs*; this requires 386-based PCs and big harddisks, and preferably OS/2 or Unix. Another interface that runs under OS/2 is TEXpert, by Johannes Martin.

For DOS users, there is the very nice interface TEXshell, by J. Schlegelmilch. Its latest version (2.5.2) is particularly useful for all users, since there is now on-line help on LATEX in English. It is also public domain, and very user-friendly. A new one, 4TEX (W. Dol, et al., 1993), appeared very recently. It uses the shareware programs 4DOS and QEDIT. It seems very nice too. Our interface AST$_E$X uses the commercial program Framework, and optionally Desqview or OS/2.

Finally, there are also several commercial scientific word-processors that are able to edit mathematical equations in WYSIWYG mode and are able to export them in TEX form (see Lavaud, 1991 and 1992 for some references).

This shows that, even on PCs, the Lion must not be afraid any more of the Mouse (Siebenmann, 1992)...

Motivations for writing AsTEX

In the early history of computer science, programs were written on sheets of paper by researchers, typed on punched cards by specialized typists, and submitted to the computer by an operator. When teletype terminals became available, all scientists began to type and run their computer programs themselves because this allowed them to gain much time, despite the fact that they did the work of three people.

Many researchers still write their scientific articles by hand, and have them typed by secretaries. This can take a very long time, especially for articles with many complicated formulae. It seems reasonable to expect that, if software adapted to research work becomes available, all scientists will also type their articles themselves, because this will allow them to gain much time, as with teletype terminals. In an earlier article (Lavaud, 1992), we argued that, for a software to be adapted to scientific work:

- It must allow the user to *display* and *modify* easily the structure of the document, to ensure that even very long multi-author documents will be coherent and logically organized;
- it must allow for the performance of everyday research tasks *from the document* (numerical and formal computations, management of the files created or received in the research process, etc.);
- it must be TEX-based.

The first point implies that a document cannot be just a sequence of characters typed inside one or a few windows; it must be a *tree* whose leaves are windows that contain coherent blocks of information of various nature (paragraph of text, worksheet of numerical results, database, computer program, numerical result, e-mail, illustration, etc.).

Overview of AsTEX's possibilities

We enumerate here the main possibilities of AsTEX. Some are developed in more detail below. Others are detailed in Lavaud, 1991 and 1992 and in references therein.

- **Hypertext file manager**:
  - Immediate access to thousands of files through hierarchy of explicit titles.
  - Easy modification of the structure of very big multi-author documents.
- **Scientific computations**:
  - Numerical (e.g., Fortran): compilation / execution run directly from text of the document.
  - Formal (e.g., Maple): results automatically included in text, worksheets, databases.
  - Live links to data files.
  - Interdependent worksheets.
- **Scientific text processing**:
  - Mathematical and chemical formulas displayed with a single keystroke.
  - Preprocessor of LATEX structure.
  - Cut and paste from hypertext help into the document.
  - Automatic generation of environments.
  - Creation of LATEX tables from worksheets or databases of formulas.
- **Tool Box**:
  - External DOS (and UNIX) tasks can be run from a customizable Tool Box.
- **Electronic mail**:
  - Hypertext archiving of messages.
  - Automatic extraction of messages from files issued from discussion lists.
  - Local archiving of information about ftp and archie servers to speed up connections.
- **Hypertext help** for AsTEX, LATEX, emTEX, Ghostscript, graphs used in physics ...

The hypertext file manager of AsTEX

Writing a multi-author scientific book from scratch, (or collating results of a research team regularly over several years) is not only the matter of typing text with a scientific word-processor. When you create with colleagues a document that will at the end have several hundred pages, and that will make reference to hundreds of files (articles, chapters of book, numerical results, computer programs, input and output data, electronic illustrations, electronic mail...), it is very important to be able to navigate easily, in a *logical* way, in the document and in the files that are related to it, during the whole process of its creation. You need a file manager which allows you to classify and archive your files in a *structured* way, so that you can retrieve them easily, with a few keystrokes, regardless of who created them.

Usual file managers. With usual file managers (those of Framework 3, of Windows 3.1, etc.), you access a file from its *physical location* on disks: you have first to remember on which disk it is, then in which directory. Then, you have to scroll among a set of files, most of which are not pertinent to your document. Moreover, as names of files and directories are limited to eight characters (with MS/DOS),

they are not very explicit in general, and it may be very difficult to retrieve a file that was created or received a long time ago (see Figure 1). Even worse, files created by colleagues and pertaining to the document may have been moved without their alerting you.

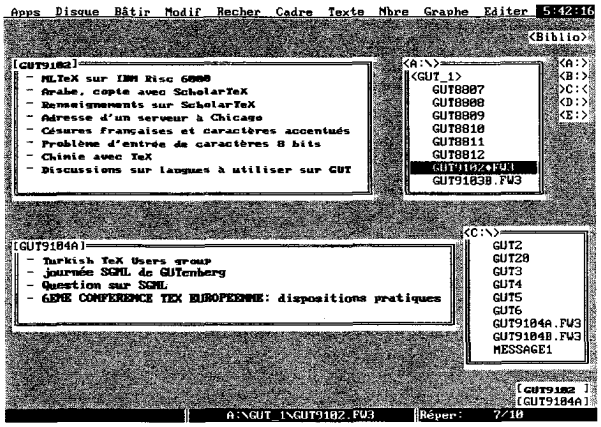

Figure 1: File retrieval with the file manager of Framework 3, illustrated with e-mail received from the GUTenberg discussion list (French TEX Users Group): names of archive files are not explicit, files are scattered over several disks.

The file manager of AsTEX. For AsTEX, a document is a *set of files related logically* and accessible from one of them (the master file) by loading into linked windows. The set of files has a *tree structure*, and each file is itself a tree whose leaves are *objects* of various nature (linked windows, texts, databases, worksheets, graphics, computer programs...). The files may be on different media and be created/modified by several people on a network.

With the hypertext file manager of AsTEX, a file is accessed from its *logical location* in the document, not from its physical location on disk. Each file related to the document is retrieved from a hierarchy of *explicit titles*. This way of accessing files has many advantages, among which (see Lavaud, 1991 for more details):

- The way to retrieve a file from the document *remains unchanged* when the file is moved physically to another place for some reason (the directory is too crowded, the local hard disk is full,...).
- Only the files *pertinent to the document* are displayed and accessible from it. The files that are unrelated are not displayed.
- *Modifying the structure* of a document is very easy and very fast, because files are reorganized logically in the document, not physically on disk(s).
- A file can be accessed from several documents, in different ways, i.e., with different hierarchies of titles.
- Data (e.g., computer programs, numerical results, etc.) can be accessed as *live links* (i.e., the latest version of the data file is automatically loaded) or stored as *backups* in parent files.
- One has *several levels of backup* for linked files.
- You are automatically informed of new files added to the document by colleagues, without them having to tell you.

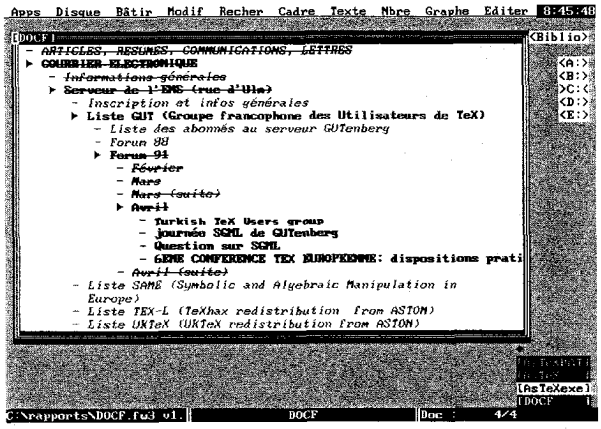

Figure 2: File retrieval with the hypertext file manager of AsTEX, illustrated with the same example as in Figure 1. e-mail is retrieved from a hierarchy of titles.

Word processing with AsTEX

The general philosophy of AsTEX is to display interactively only *global formatting* of text, and to use LATEX commands for *local formatting*. These are considered as encapsulated in small blocks of information that are stored into linked files. AsTEX deals with the organization of these blocks through organization of linked windows on screen, and it allows the author to forget completely about local formatting commands. He is just reminded of the contents of the blocks through their titles, and he can concentrate on the important part of his work, that is on the *logical connection* of the various components of his document and on the *scientific computations* that are related to it.

WYSIWYG or not WYSIWYG? That is the question! When speaking of WYSIWYG scientific editors, one thinks automatically of interactive editing of mathematical formulas. In an earlier article (Lavaud, 1992), we explained that there are many interactive equation editors able to *export* mathematical equations in TEX, but that, to be really useful, they ought to be able also to *import* such equations and, more generally, any TEX or LATEX file. And this is much more difficult, since this means that the equation editor must contain almost all the capabilities of the TEX compiler.

With ASTEX, mathematical equations and most local formatting commands are supposed to be written in native TEX. Some simple local commands that Framework is able to display, such as italics, bold, indices and exponents, can be translated automatically by ASTEX. It provides also on-line hypertext help and a multi-level assistance in typing LATEX code, in particular by generating automatically environments from a hierarchical menu (see Lavaud, 1992).

Previewing portions of text. In the absence of a satisfactory WYSIWYG editor able to import TEX and LATEX files, a good front end to TEX must be able at least to preview any portion of text with one or a few keystrokes (Siebenmann, 1992). This has been possible for quite a long time with *emacs*. This is possible also with ASTEX (see Figure 3). It is further possible to preview the text contained in a selected subset of windows, as appears in Figure 4.

As Laurent Siebenmann has emphasized, the mechanism is very simple, but it seems very underused. For example, the question "*How can I transform my Wordperfect files to TEX or LATEX*" is asked very often on the net. Now, with Wordperfect, mathematical equations are typed in the `eqn` language and are debugged exactly as indicated in Figure 3. So, instead of trying to transcode from `eqn` to TEX, it would be much more efficient to write and debug equations directly in TEX. A program, written in the programming language of Wordperfect, that would implement the above mechanism would certainly solve many problems. More generally, this mechanism could be implemented very easily into many word-processors, so that files in native TEX could be typed and debugged from these word-processors, instead of being translated by an external program, so that users accustomed to a given word-processor can take advantage of TEX from within their favorite editor. So, although the mechanism is fairly trivial, let us describe it in some detail for the PC.

The editor needs only to be able to save a selected portion of text into a file, shell to DOS and run an external program. A prolog and a trailer have to be added to the selected text: this can be done either inside the editor, if it is able to concatenate chains, or during the shell by adding `\input prolog` and `\input trailer` at the beginning and at the end of the file containing the text, with the DOS `copy` instruction.

This results in previewing a portion of text by switching from source text in full screen to the previewer in full screen, and back again to the editor. A more elaborate way is to display code and result simultaneously as in Figure 3. This is obtained by coupling the preceding mechanism to the multitasking properties of Desqview or OS/2. With Desqview for example, instead of running the previewer directly during DOS shell, you have to run the shareware utility *dvexec*, telling it to create a child window and to run the previewer in it. This is done by running a batch program containing a line such as:

```
dvexec c:\dv\tp-pif.dvp
```

where `tp-pif.dvp` is a file created by Desqview, which contains the parameters to run the TEX previewer.

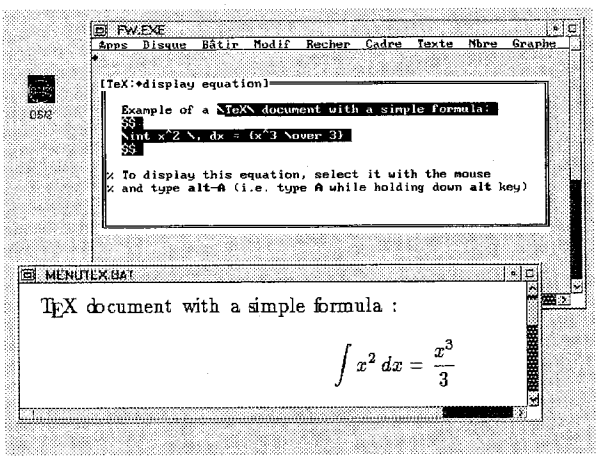

Figure 3: Displaying a portion of text with ASTEX: the text selected with the mouse or the key arrows, is displayed automatically in a child window by typing a single keystroke (`alt-A`).

The prolog and trailer attached to the current document have to be stored somewhere, in external files or inside separate windows, according to the capabilities of the editor used. The prolog must contain all the necessary definitions. For example, if we want to preview the chemical formula:

```
$$
\ethene{$CH_2 OH$}{$R^2$}{$R^3$}{$R^4$}
$$
```

with the ChemTEX package, the prolog must contain at least the instructions:

```
\documentstyle[chemtex]{article}
\begin{document}
```

The prolog must also contain personal macros that are used in the document. The trailer must contain at least the `\end{document}` instruction for LATEX, or `\bye` for TEX.

With AsTEX, since many individual documents may be stored inside the master document, each may have a special prolog/trailer. These are stored in windows that immediately precede/follow the subtree that contains the text of the document.

Preprocessor of LATEX structure. When you write a long document, you have to modify its structure very often. Local commands, such as `\it`, `\indent`, mathematical formulas, etc. remain unchanged. But global commands such as `\chapter` or `\section` must usually be modified: for example, if a long section is becoming too big and has to be transformed into a chapter, all `\subsection` commands must be transformed into `\section`, etc.... This may require modifying many LATEX sectioning commands, which may be very error-prone if these are scattered among several files.

With AsTEX, you do not have to modify sectioning commands because you never write any of them. Any modification in the structure of the document is made within Framework, and sectioning commands are automatically generated by AsTEX from the tree of the Framework document.

Implementing some functionalities of SGML parsers. SGML is the ISO standard for document description. It is designed specifically to enable text interchange (van Herwijnen, 1990). Although SGML is not very well adapted to everyday research work, many of its ideas are very important and of general scope, and can be implemented fruitfully into TEX-based software. For example, an important function of SGML parsers is to ensure that a document has no chapter inside a section. This possibility is not forbidden by LATEX: if we want to write "Hello everybody!" in large letters in the middle of a paragraph, it is possible to do it by including the instruction:

```
\section*{Hello everybody!}
```

With AsTEX, it is impossible to create an ill-structured document, for two reasons:

1. Sectioning commands are generated automatically from the tree of the document (cf. preceding section);
2. Writing sectioning commands in text is inhibited: if we type `\section` in the text, it is automatically erased, and the message "*Instruction \section forbidden*" is displayed on the screen.

Therefore AsTEX plays the role, at the front end level, that is fulfilled by SGML parsers at the back end level.

Exporting a LATEX document. Let us consider the document of Figure 4 (which corresponds to the article by Lavaud, 1991):

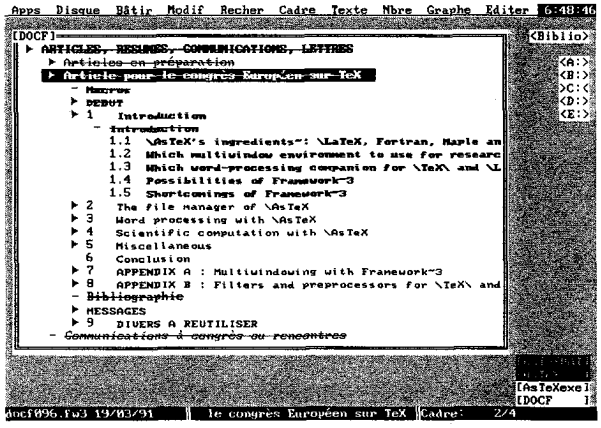

Figure 4: Example of an article, in the Table of Contents mode.

When AsTEX is asked to create a LATEX file from it, it generates the document of Figure 5. We see that AsTEX does not blindly export the whole document: a lot of windows have been eliminated. When the document is displayed as in Figure 4 (title of first section in boldface, other titles in normal characters), it is configured for debugging only the first section of the article.

To illustrate further the great flexibility of AsTEX in creating LATEX documents, let us just indicate that the master document of Figure 4 manages several thousand files; nevertheless when going from the state of Figure 4 to Figure 5, AsTEX has exported only a small part of one file linked to the document; and it could have exported as easily small selected (non-consecutive) parts of several files.

The Tool Box of AsTEX

AsTEX contains a Tool Box that automates access to general internal or external resources, independent of the current document. *Internal resources* are, for example, agenda, alarm, frequently used databases...). *External resources* are in general batch programs, to run external PC programs in a customized environment or to send UNIX requests to the server. All resources are accessible from the

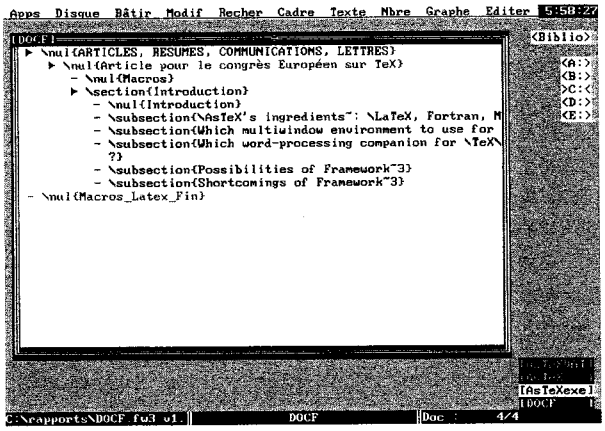

Figure 5: Document with LaTeX sectioning commands, generated by AsTeX from the document of Figure 4.

Tool Box exactly as any item in a hierarchical system of menu. They are activated by pointing to an explicit title instead of typing the name of a program.

Porting AsTeX to other software

AsTeX has been developed on the top of Framework because, at the time when the project began (1990), this commercial program was the most suited to our purpose, while public domain editors available on PCs were not powerful enough. In particular, *emacs* was not available in its complete version.

Many functions of AsTeX can be implemented in other software, either PD or commercial, provided it has a powerful enough programming language. We enumerate the main possibilities offered by Framework that are used by AsTeX, to indicate the prerequisites for such a porting.

Why use Framework?

- Framework offers a *hierarchical* multiwindowing system for the three basic applications:
  - editor of text,
  - spreadsheet,
  - database manager.
- It has a very powerful programming language, which allows us to program very complex applications. This language is identical in all applications (when using specialized programs, you have to learn several different languages).
- It is ideally complementary to TEX: each application is much more rudimentary than a specialized program (e.g., the spreadsheet module as compared to Excel), but most possibilities that are missing are added by its coupling with TEX, and many more are added that may not exist in the specialized program. For example, mathematical formulas cannot be written in cells of Excel, while this can be done with Framework + TEX (of course, this could also be done by coupling Excel to TEX).
- Telecommunications can be done in a window, with the possibility to cut and paste text to and from other windows containing text, worksheet or database.
- The three basic applications run much faster than an equivalent set of programs under Windows 3.1, and the multi-windowing system for the basic applications, combined with multi-windowing facilities of Desqview or OS/2, is much more powerful than that which can be obtained with Windows 3.1.

Framework also has several other advantages: it runs on any PC, occupies only about 2 Mbytes on disk, and it has some interesting built-in possibilities (spell-checker, synonyms, mailing, etc.).

Porting AsTeX to GNU *emacs*. Our dearest wish would now be to port AsTeX to GNU *emacs*. Indeed, the complete version of *emacs*, with its Lisp-like programming language, has been ported to OS/2. This is still a limitation, because this requires a PC386 and large disks, but hardware prices are going down very fast and older models will disappear soon.

Porting AsTeX to *emacs* would be desirable for many reasons. First, it is public domain, well-supported and widely used. Second, since Framework is a commercial program, some of its shortcomings cannot be solved. For example, although Framework is mouse-based, no control of the mouse is provided by its programming language. Since the code of Framework is not public domain, this makes programming the use of the mouse with AsTeX very difficult. Many other problems cannot be solved neatly for the same reason. For example, we could only forbid typing `\section {}` but not `\section{}`, because only the space character is considered as an end of word, in the automatic substitution function of Framework. This fairly stupid limitation could be solved in a few lines of code with a public domain editor.

Conclusion

We have proved, by building the program AsTeX, that it is possible not only to make the use of TEX and related software easy on low-cost PCs, but also to build a powerful multi-window environment that is adapted to scientific research and based on TEX.

For AS$\TeX$ to be useful in practice (not only as a model), it ought now to be ported to other more widely used commercial software and above all to public domain editors, in particular to *emacs*.

Bibliography

Dol, Wietse, Eric Frambach, and Maarten van der Vlek, *MAPS 93.1*, pages 53 - 57, 1993.

van Herwijnen, Eric, *Practical SGML*, Kluwer, 1990.

Lavaud, Michel, *EuroTEX'91 Conference Proceedings*, pages 93 - 116, 1991.

Lavaud, Michel, *EuroTEX'92 Conference Proceedings*, pages 307 - 330, 1992. Reprinted in *MAPS 93.1*.

Siebenmann, Laurent, *EuroTEX'92 Conference Proceedings*, pages 43 - 52, 1992.

Taylor, Philip, *EuroTEX'92 Conference Proceedings*, pages 235 - 254, 1992. Reprinted in *MAPS 93.1*.