The LaTeX Legacy

2.09 And All That

© 2001 Chris Rowley
Open University, UK
c.a.rowley@open.ac.uk

ABSTRACT

The second edition of The Manual [23] begins: 'LaTeX is a system for typesetting documents. Its first widely available version, mysteriously numbered 2.09, appeared in 1985.'

It is too early for a complete critical assessment of the impact of LaTeX 2.09 because its world-wide effects on many aspects of many cultures, not least scientific publication, remain strong after 15 years—and that itself is significant in a technological world where a mere 15 months of fame can make *and* break an idea.

Therefore this paper provides simply a review and evaluation of the relationship between TeX, LaTeX and some of the major technical developments in the world of quality automated formatting since the publication of LaTeX 2.09 in 1985. It is is neither definitive nor comprehensive but I hope it is informative.

PRACTICAL

The primary immediate impact of the widespread distribution, by Leslie Lamport in alliance with Peter Gordon at Addison-Wesley, of version 2.09 of LaTeX in the mid-80s derived from its supreme importance to the use of Donald Knuth's TeX system [18], enabling the latter to spread rapidly beyond the community of North American mathematicians who had nourished its development from its birth as one of Don's 'personal productivity tools', created simply to ensure the rapid completion and typographic quality of *The Art of Computer Programming* [19]. A less direct but probably wider influence derives from its being the first widely used language for describing the logical structure of a large range of documents and hence its introducing the philosophy of logical design, as used by Brain Reid in Scribe [35]: when writing a document, you should be concerned with its *logical content*, not its *visual appearance*.

Back then, LaTeX was described variously as 'TeX for the masses' and 'Scribe liberated from inflexible formatting control.' It is not clear whether either of these was intended by Leslie as a design feature but it was certainly not his expectation to make, in either time or space, such a wide impact as he did. The availability of LaTeX was, even in the late 1980s, very wide compared with most non-commercial software at that time. The good news spread rapidly and by 1994 Leslie could write 'LaTeX is now extremely popular in the scientific and academic communities, and it is used extensively in industry.' But this level of ubiquity was still miniscule compared with the present day when it has become, for many professionals on every continent, a workhorse whose presence is as unremarkable and essential as the workstation on which it is used and when (probably as part of a Linux distribution) LaTeX sits ready-to-run on every desk top—if only supporting yet another used coffee mug!

We should not forget that LaTeX was not universally liked in its early days since it was, with justification, seen as importing too much of Scribe's inflexibility into the anarchic TeX world. Such feelings had many positive results with important and innovative ideas for improving on LaTeX being described and investigated; some of these even got built into highly functional and radical systems such as that of Michael Spivak's LAmSTeX [40].

1. INTRODUCTION

Such direct impacts of LaTeX 2.09 do not form my major subject here; rather I shall treat the the importance of Leslie's ideas and activities by a very selective description of some subsequent developments in high quality automated document processing. Leslie's work covered many aspects of the subject but here I shall concentrate on the visual formatting of logically described text documents rather than on other, equally important, parts of document processing that were also major components of the LaTeX system and of his interests. In particular there will be only passing references to a major aspect of the LaTeX story, one that was my only reason for ever starting down the road that led to this article: mathematical typesetting. I omitted this because developments in that area deserve a whole article to do them justice. One of Leslie's less quantifiable influences throughout the story I shall tell is that in addition to the other varied roots of their cognitive processes, since the mid-80s almost all the major developers, wizards, gurus and TeXnicians, except Don Knuth of course, surely owe much to themselves having been LaTeX users (and abusers).

Although a significant amount of the development since the late 1980s has built on TeX itself rather than on LaTeX, the widespread publication and use of LaTeX, especially outside North America, was a major factor pushing almost all of the subsequent technical progress within the TeX world and also a substantial influence on more general research in document science. Thus the history of such development over the last 10 years is the legacy of both Don and Leslie, even though neither has been more than tangentially involved during that period.

In addition to the particular nature and quality of the software, there are other areas in which the TeX support community [41] has been innovative and effective. For over a decade TeX was an archetypal example of the virtues and vices of the best producer-centred 'free software': it provided freely available, robust and

```
\begin{minipage}[c]{0.4\colwidth}
  Some vertically centered
  little paragraphs \ldots
```

```
$ \vcenter {
    \hsize 0.4\hsize
    Some vertically centered
    little paragraphs \ldots
```

Figure 1: Syntax compared: typical LaTeX (left); with (almost) equivalent basic TeX (right).

efficient functionality for those who understood and accepted DIY installation and maintenance where 'support' means support *for* the producers *by* the users, rather than vice versa. Over the last ten years the enterprising efforts of a large number of individuals, aided by user groups and other organisations, have completely changed the use and support of TeX-related software. Solid graft has given solid proof that open, non-commercial methods and infrastructure can be used not only to produce effective, robust software, but also to provide it with reliable user and technical support. The software maintenance system for standard LaTeX [24] is an easily accessible example of such totally user-centred software support—one that was, until recently, pleasantly distanced from the politics, bureaucracy and legalistic trappings of the various movements and sects now named 'Open Source', etc.

2. TeX AND LaTeX

From its inception, the development of LaTeX and related software has been completely intertwined with that of TeX itself. This section therefore sets the context by presenting some thoughts and perspectives on the design of these two systems and their interconnection. A more detailed but still brief history of TeX and LaTeX can be found in Chapter 1 of *The LaTeX Companion* [12], the remainder of which contains a large amount of relevant information including the definitive user documentation of the mid-90s versions of many parts of the LaTeX system including many extension packages.

Although there are many legitimate doubts about the continued utility of some aspects of the fundamental models and design of TeX as a text formatting engine, it was then and remains now (the beginning of the millennium) the only mature, widely available, programmable and highly flexible typesetting engine. Thus for Leslie then as for me now, it is the only choice as the foundation of any practical automated system for high quality formatting.

One reason for the production and distribution of LaTeX, illustrated in Figure 1, was (misquoting Leslie's preface to the original manual [22]) to provide 'a family sedan' as an alternative to TeX's 'highly tuned racing car.' More explicitly he wrote 'LaTeX adds to TeX a collection of commands that simplify typesetting by letting the user concentrate on the structure of the text rather than on formatting commands' and I have therefore deliberately chosen an example in Figure 1 which illustrates that LaTeX also introduced many formatting commands.

Worse than that, in the light of its future influence, LaTeX introduced, or at least crystallised, the inherent ambiguity of many parts of a document description. For example, is 'a list' no more than a document element whose content is partitioned into 'items'? ... or is 'a list' a part of the text of the document that consists of 'items' and should be formatted as paragraphs using a paragraph style that distinguishes that text from the surrounding text and enables individual items to be labelled?

The design of LaTeX deliberately provides no clear answers to such questions and Leslie's ambiguity has propagated right across The Web where the HTML language [48] could be used, as intended by its founders, to describe the logical structure of a document whereas it is in practice rapidly becoming the standard language for describing the 'formatted form of a web document', the logical structure being described in XML [46] or even LaTeX itself. Whatever the inconsistencies of the philosophy, the syntactical consistency and power of the document interface is a defining characteristic of LaTeX. Behind the interfaces, Leslie was equally creative as he introduced and exemplified lots of neat tricks and monumental edifices of TeX programming and other ground-breaking examples of practical software design, including what is still the basis of the only complete non-trivial float-positioning algorithm that has ever been fully implemented.

In the design of LaTeX Leslie deliberately allowed the underlying TeX engine to act directly on the majority of the textual matter. In typical text processing systems of that era, including TeX, the primary methods for handling document text (the input character strings) are as follows: each input token sent to the system is treated as a complex imperative command. In such systems a 'character in the text of a document', typically a keyboard event or a token in an input buffer, is not simply destined to invoke the creation of an 'element in a string' in an 'object of class *text-stuff*', such a 'string' eventually being processed by some other module of the system, or even by external applications.

In the design of TeX the 'complex command' normally invoked by each such text character is 'typeset me right now!' Contrast this with a common current paradigm, such as the XSL model [45], in which, right down to the individual 'character objects', the whole document is represented and manipulated as an explicit hierarchical object structure (in the jargon, a DOM). To 'format the document' explicit methods must then be invoked to act on the whole or parts of this hierarchy.

TeX was designed in this imperative paradigm because this leads to a highly efficient (in both time and space) machine, despite 'typesetting' being for TeX a relatively sophisticated computational process involving, primarily but not exclusively, the optimisation of glyph choice and positioning over whole paragraphs as controlled by a highly configurable dynamic programming algorithm. However, because this 'typesetting process' has been highly optimised for speed, doing anything that is not available within this monolithic process (as defined by TeX's design) is both difficult to implement and noticeably inefficient in use. Such processes are central to quality typesetting and are especially important in the typesetting of languages other than US English. They include the modification of important subprocesses such as choice of glyph (as for ligatures) and of their size and positioning; the hyphenation and justification (H&J) subsystem is another example.

In the same sprit the LaTeX command **\addvspace** immediately adds some space to TeX's current vertical list. Such immediate and direct construction of formatted material from input token lists has now become a burden to the creative design of typesetting software for a wide variety of languages, scripts and document types.

The story I present here of TeX/LaTeX-inspired development since 1985 has two substantive motifs: globalisation of the typography and interaction with the typography. In the subsequent two sections I shall consider the impact of each of these on TeX-related development but first I must devote a section to document science by explaining further some principles of software modelling for quality formatting.

3. DUMMY'S GUIDE TO QUALITY

Here therefore is a summary of how I understand two of the major constituents of a model for quality typography engines; here the 'quality' of a typographic artefact means its 'fitness to purpose', which is most often not simple conformance to some arbitrary visual aesthetic. I hope that it provides enough information to make sense of the descriptions and analyses later in this paper. Note that it all relates solely to *visual* formatting; none of this makes sense for such important activities as 'audio formatting.'

3.1 Text into paragraphs

This section is aimed primarily at conscience-raising amongst fellow English-speaking computer scientists; it therefore somewhat simplifies reality in places but I hope it is not too abbreviated.

The subtask of producing a formatted paragraph that visually represents some textual material, in the form of a bounded sequence of text characters, comprises these interacting processes.

1. For each suitable subsequence of input characters (in English normally a whole word together with its immediately contiguous punctuation characters), finding its potential representations (PWRs) by a collection of relatively positioned glyphs.
2. Finding potential relative positions for the elements of each sequence of PWRs thus generated (in English normally 'simple H&J').
3. Choosing one formatting of the paragraph from amongst all the above.

These processes all require information and, depending on the design requirements of the document, they may require considerable computing power for sophisticated selection and optimisation techniques. The automated treatment of the LaTeX logo (as in Section 2) within the headings of this document is a good example of what can go wrong when the complexity of this process is not understood or the necessary information is unavailable.

One class of essential resources is font information; exactly how much information is needed about individual glyphs varies considerably with the requirements of the script being typeset and with the design requirements, but it can be a lot more than just the nominal rectangle occupied by the glyph and it is needed for all of the often large number of potential glyphs. It is, however, generally accepted that the typesetting process does not require information about how a particular glyph in a particular font is rendered; thus a large proportion of the content of, for example, a PostScript font file need not be accessed by this process.

At the later stage of describing in detail how to print a formatted document, rendering information for the glyphs is needed (in English, for efficiency of implementation, information about nominal glyph widths is also normally used at this stage). However, this information is needed only for the typically small number of glyphs actually used in the document. One practical problem that is often encountered at this stage is ensuring that the glyph rendering information is accessible by the application that actually renders the document on screen or on paper. There are two good solutions to this: to embed all the necessary information in the document or to have universally accessible font resources; unfortunately, both of these conflict with typical licenses for commercial fonts.

Before any such choices can be made it is essential that the system knows the availability and typographic characteristics of all potentially useful fonts and also precisely what glyphs are provided by each font. The glyph collection provided by a font is commonly referred to as that font's 'encoding' since it is often specified via an array structure; these are still most often of size 256 (8-bit fonts) but in modern fonts standards this can be 65536 (16-bit fonts).

3.2 Text, graphics and interaction

It has for some time been accepted that models for formatting documents should treat text glyphs simply as specialised graphics since these both reduce finally to instructions on how to treat each individual pixel on the output medium. Nowadays, when those pixels are on an interactive screen, it is also likely that in the model of that screen, they must have properties other than their visibility. For example, they have attributes that give them the power to change the representation of the screen pointer or that define the consequence of a pointer click on that pixel.

Such 'interaction properties' of the graphical elements of a formatted document are so similar to the classical graphical properties such as colour or gray-level that a model of visual document formatting should treat them identically. To internalise this concept, compare a hypertext link in the text of a browser document with the use of a graphical technique to highlight the linked text. Such a paradigm has significant consequences for the models and interface languages used by quality document formatting software and, although it is not yet a common design feature of languages or software, it has influenced the development of the Scalable Vector Graphics (SVG) language [50].

Thus text, graphics and interaction, whilst each having their own peculiarities at the implementation level, are unified in the model that lies behind the views expressed in this paper. As all attempts to implement such integrated formatters have illustrated, uniform provision of font resources is vital to successful solutions. Such a model should be considered as part of an overall model for the process of document design and formatting developed from that of 'interacting modular formatters' introduced by Frank Mittelbach and myself in 1992 [29] which was based on an analysis of the fundamentals of professional book design undertaken within a conceptual framework introduced by the Reading (UK) school of typographic design [39].

4. GLOBALISATION

If the impact of LaTeX (or 'TeX for the rest of us') as limited to the English-formatting world far exceeded Leslie's earlier expectations, the welcome it received from the rest of the globe was deafening: here at last was some ex-USA document processing software that merited investment of national and international effort. LaTeX 2.09 was (deliberately) not globalised but it was globalisable: moreover it came with documentation worth translating because of its clear structure and straightforward language.

The world-wide availability of LaTeX therfore quickly increased international interest in TeX and its potential for typesetting a range of scripts and languages; but there had already been started two important adaptations of TeX, neither of which has had a major effect on more recent extensions of TeX in this area. One was JTeX, a modified version that works exclusively with a particular 16-bit text encoding used in Japan; the other was MLTeX (see below).

Don had understood well many of the principles and techniques needed to support multi-lingual typesetting but he was unable to implement them within his time constraints and his over-riding design goal of production-quality software that could be widely distributed for use on the commonly available platforms of the early 1980s. Note that the 1990 version (TeX 3) makes only small extensions with little concession to the increased desk-top computing power by then available. These extensions are largely language-related, going a small way towards full support for 8-bit font (glyph) encodings whilst almost achieving excellent support for hyphenation of multiple (natural) languages within a single paragraph. The 'almost' is there since Don did not resolve a fundamental design flaw which still prevents TeX from being able to guarantee correct hyphenation when two very different font encodings are used for language fragments within that one paragraph. I mention this particular problem not only because its consequences have recently occupied a noticeable proportion of my energy but because it is an archetypal example of why working with TeX at a deep technical level leads frequently to frustrating challenges.

Perhaps this explains why the only significant earlier work on changing some details of the implementation of TeX in order to make it a viable system for typesetting text in a variety languages other than US English was Michael Ferguson's MLTeX [11]. This extension of TeX dates from 1987 but only 10 years later did it become politically feasible for MLTeX to be incorporated as an option into major TeX sources; ironically by then its technical features were no longer essential to the globalisation effort but it is still well supported by some, most noticeably Francophone, users.

Please note that, throughout this discussion, the word 'language' does not refer exclusively to the variety of natural languages an dialects across the universe; it also has a wider meaning. For typography 'language' covers a lot more than just the choice of 'characters that make up words' since many important distinctions derive from other cultural differences that affect typographic traditions. Thus important typographic differences are not necessarily in line with national groupings but arise from different types of documents or publishing communities.

4.1 Multi-lingual use of standard TeX

In the late 1980s the tendency for national and language groups to enhance the local value of TeX and LaTeX spread rapidly across Europe, notable are those for Polish, Czech, German and French. This was not enough for NTG, the users group in The Netherlands (Europe's printing house) who not only fixed Dutch but went on to pioneer more widely applicable efforts that encompassed the needs of multi-lingual documents. Another European initiative resulted in the 'Cork encoding' needed to bring order to the use of TeX 3's 8-bit capabilities; it is named after the Irish location of its origin.

One feature of LaTeX needing attention was that many US words, such as 'Part' were used in generated-text and, despite the impression given by Leslie, most of these character sequences were explicitly embedded in arbitrary parts of the code. Johannes Braams [7] wrote as follows in connection with this aspect of the Dutch (NTG) group's project [8]:

> This [Section 5.1.4 of The Manual [22]] looked rather promising to me, so I had a look at the style files to find out how other [strings such] as "Figure" might be redefined. It was then that I found out that **\@chapapp** is the *only* string defined this way, whereas all others are hard-wired ... .

Fixing LaTeX seemed to be proving as difficult as the then current myths suggested, but the courage of the pioneers such as Johannes, Hubert Partl and Joachim Schrod triumphed over such pessimism. Out of these efforts and two pivotal conferences (Exeter UK, 1988 and Karlsruhe Germany, 1989) emerged International LaTeX [37] and the idea of more general support for using a wide variety of languages and for switching between them.

The resultant development by Johannes Braams, based on some technical insights from Bernd Raichle, concentrated on two further aspects of multi-lingual documents:
– dynamic access to all the necessary hyphenation rules;
– dynamic support for a range of keyboard input methods.

The result was Babel [7], 'a multilingual style-option system for use with LaTeX's standard document styles.'

It is that 'dynamic' requirement that makes Johannes' achievements into such a difficult task. Whilst setting TeX up for a just one non-English language can be far from trivial, providing support for the use of more than one language in the same document increases enormously the complexity of design and coding needed to coordinate TeX's many internal mechanisms. But Johannes had added to this the requirement to support, in addition to Don Knuth's quirky use of regular characters such as **$** or **%**, the arbitrary range of traditional or specialised keyboard input techniques needed for 'multi-lingual typing.' Other perceived dangers of this work are captured well by this further quote from Johannes' documentation for the Babel system [7]: 'Although Leslie Lamport has stated ... that one should not try and write *one* document-style option to be used with *all* the standard document styles of LaTeX, that is exactly what I have done with [Babel].'

The primary development phase of Babel was complete by 1995 and since then it has come into almost universal use with standard LaTeX installations; the system is still being developed and maintained by Johannes in cooperation with various people around the world. Until recently its predominant use was with the Roman script for European languages but since 1995 practical extensions into Cyrillic have been developed from substantial work by Olga Lapko, Vladimir Volovich and Werner Lemberg. This has opened the way to experiments with other scripts, such as Greek and Indic, but it is not clear that substantial useful progress can be made in this direction within the constraints of standard TeX.

4.2 Extending TeX

Whilst Babel moves rapidly towards the provision of a uniform interface to all aspects of low-level language support that can reasonably be provided when using the standard TeX engine, meeting within the TeX tradition these needs for more fundamental changes to support 'all the languages of the globe' is the current heroic aim of The Omega Project [32]. This goal of overcoming some of the practical limitations of TeX is being achieved by using, as Don Knuth encourages everyone to do, the TeX code as a basis for new systems that are *not called* TeX. Omega is an extension of TeX that has been developed since the early 1990s by John Plaice (University of New South Wales, Australia) and Yannis Haralambous (Atelier Fluxus Virus, France).

The slogan 'feel the width!' introduces the most well-known feature of Omega: that the representations of all characters and glyphs (together with other data slots) are 16-bit wide compared with TeX's 8-bit standard; such widening is in line with modern standards for text files [42]. Whilst such long overdue implementation updates are very welcome, the strategic importance of Omega lies in its new approaches to the central and very challenging problem of setting type: the choice and positioning of glyphs to represent an input character sequence within a given logical and visual context. The currently released version provides two important aids to implementing this process.

The first is that Omega does not, by default, unthinkingly and rapidly typeset sequences of characters as would TeX; instead it is designed to easily store sequences of 'type-settable characters' (including white-space) in a buffer so that they can be processed further as character strings strings before being delivered to the 'rapid typesetting process.' This provides for the first time in a typesetting system a programmable interface to 'near-typesetting character manipulation.' It can be used, for example, to define the complex contextual analysis needed for such processes as correct ligature choice and diacritic placement, or character cluster building as required for scripts such as Arabic, Indic, Hebrew or Khmer.

The second is really a collection of more radical extensions that are part of ongoing development work to provide comprehensive support for typesetting all known scripts and writing systems; the current version has a lot of support for nested multi-directional typesetting with a potential 16 writing modes (or 64, depending on who is counting).

John and Yannis' innovations also encompass the data structures needed by a typesetting engine. The most obvious of these are the font resources, the potentially large amount of information needed about available fonts and the glyphs they can render. In order to illustrate and utilise Omega's quality and versatility, Yannis has designed a 'Unicode-based font' whose visual design is based on Times/Helvetica; it will contain all the glyphs needed for a large range of alphabetic scripts including the following: Latin, Greek, Cyrillic, Armenian, Georgian, Hebrew, Arabic, Syriac, the Indic scripts, Thai, Khmer. In addition, the font information resources native to TeX have been extended to support such features as the graphical justification of Arabic and other cursive scripts. However, Omega is still restricted to using bespoke TeX-like font information formats and encoding systems rather than directly accessing the latest font resource formats such as OpenType [4].

There will be several spin-offs from this globalisation process: one example is robust support for more complex paragraph structures, including line decorations for change bars or line numbering. The developers [33] have written further that:

> These extensions not only make it a lot easier for TeX users to cope with multiple or complex languages, like Arabic, Indic, Khmer, Chinese, Japanese or Korean, in one document, but will also form the basis for future developments in other areas, such as native color support and hypertext features.

This last sentence takes us forward into areas of document formatting, graphics and active regions, that form the subjects of the next section, so I must here mention my hope that these further developments will be informed by, inter alia, the research outcomes and software developments I shall describe later in this paper.

Finally I can reveal that this year has seen the start of research into the full integration of standard LaTeX with an extended and stable future Omega. At present this integration is via 'a LaTeX format adapted to the special features of Omega' called Lambda: thus Leslie's family sedan can start its global adventure with all the equipment needed for navigating terrains far removed from the California freeways.

4.3 Extending LaTeX

Returning to the late 1980s, a few more of us were ignoring the consensus that 'LaTeX is a black box' (or Chamber of Secrets? [34]). In my case this led to a direct assault on the kernel, making adaptations needed for an in-house publishing system usable by a range of professionals to produce a high throughput of high-quality printed materials for supported, home-based higher education in the pre-internet era. All of this was, of course, required 'yesterday' so was barely suited to wider usage.

Fortunately others were more altruistic, distributing their efforts in the form of add-ons to LaTeX. Most active of these were Frank Mittelbach and Rainer Schöpf working in Mainz Germany, the city made famous by the somewhat earlier innovations in printing technology of Johannes Gutenberg [10]. Just like the ubiquitous British clothing store, M&S rapidly became well known in the LaTeX world for 'good quality foundation-ware.' One of their major projects formed an important part of the globalisation of LaTeX: this was a complete replacement for LaTeX's interface to font resources, The New Font Selection Scheme (NFSS). This introduced an abstract syntax for specifying font resources that had much in common with that now widely used for system-independent font specification in modern web standards.

TeX itself supports none of the higher-level abstractions needed to access information about fonts. These had presumably not been needed by Don Knuth or the AMS for their early applications, where font information was always accessed simply via an arbitrary identifier that must be set to point directly to an explicitly named system file. Leslie had added much of the missing indirection layers and data structures but his method was found not to scale well as the number of fonts and sizes increased. Another indication of Don's earlier assessment of the needs in the area of glyphs and fonts are contained in his indication ([18] Volume A, p. 153) that providing simultaneous access to 4000 distinct glyphs at each size was unnecessarily generous, even for mathematical typesetting. This has long been acknowledged to be a considerable underestimate of the needs of technical notation, as future versions of the ISO-10646 (Unicode) character standard [42] will demonstrate.

Frank and Rainer's 1989 work enabled LaTeX to use the very limited capacity of contemporary TeX implementations to efficiently access a large number of fonts. The very substantial labour and the wealth of innovation they had put into this work must not be undervalued but perhaps its real importance at the time was not its intrinsic utility but its wider effect: it provided conclusive proof that the impressive efforts of Leslie and others during the 1980s had spoken nowhere near the last word on the exploitation of TeX's programming features—they had opened our eyes to the future potential of TeX and LaTeX. Whilst this breakthrough was motivated by the immediate need to expand the use of TeX as intended by Don for high quality mathematical typesetting, it also had major benefits for the support of multi-lingual documents since they also tend to need a wide variety of fonts.

As a prize for all their efforts, which included a steady stream of bug reports (and fixes) for Leslie, by 1991 they had 'been permitted' to take over the technical support and maintenance of LaTeX. One of their first acts was to consolidate International LaTeX as part of the kernel of the system, 'according to the standard developed in Europe.' Very soon version 2.09 was formally frozen and, although the change-log entries continue for a few months into 1992, plans for its demise as a supported system were already far advanced.

5. INTERACTION

Since I argued above that, for a formatter, interaction is essentially graphics, this section starts with some history of TeX and graphics. Extensive practical details of the systems mentioned here can be found in *The LaTeX Graphics Companion* [13].

5.1 Graphics and TeX

One eternal problem with graphics is the unlimitable variety of specification methods and file formats. I shall finesse this by limiting this description to PostScript [2] and its description-only derivative PDF [3]. The starting point is that TeX provides no concept graphics above the level of positioned, aligned rectangles of black pixels, not even coloured text or grey rectangles.

Linking TeX with PostScript is natural because they are both powerful imperative languages with similar models of a formatted page and neither necessarily makes any assumptions about the capabilities of the hardware or operating systems in use. This may be inefficient in a single-OS world but it is essential to more general document portability. Moreover the linking mechanism provided by TeX works well for them: it is the TeX `\special` node.

A 'TeX special' is a device for attaching a character string to a precise graphical point in the output formatted document. Their use requires great care but is tremendously powerful when the output is to be interpreted as PostScript since the 'location point of a special' can be naturally and easily identified with the 'current point' of the PostScript machine. This power is well illustrated by Timothy van Zandt's PSTricks package [51]. The 'killer-app' that was to turn TeX+PostScript into *the* full-featured typesetting system for the 90s was Tom Rokiki's dvips [36], which takes the formatted output of TeX with embedded specials and interprets it as pure, compact and efficient PostScript ready for powering your printer.

Another product of Don Knuth's typesetting enthusiasm has also been harnessed to the PostScript wagon: John Hobby's METAPOST is derived from METAFONT ([18] Volume C), used by Don and a handful of enthusiasts to support the programming of glyph creation. Since glyphs are graphics, it seemed natural, at least to programmers, to divert METAFONT into outputting the premier language for graphics programming. METAPOST needed also to contain a few extensions since, for example, to METAFONT as to TeX rendering is a black-and-white matter.

METAPOST itself, like early versions of the SVG work, does not implement a fully integrated model for text and graphics but John provided it with an intelligent interface to TeX+dvips so that the latter can, given effective access to font resources, produce typeset text which is then incorporated into the PostScript graphics.

METAPOST provides an advanced tool for producing PostScript figures incorporating TeX-typeset text without any need to stare at a screen and mess with a mouse. Thus it is a great tool for non-visual 2-dimensional geometers (aka mathematicians) but at present it has only this declarative/imperative programming interface without even, I believe, any simple GUI add-ons.

Still under the dictum that 'document specification equals programming' Hans Hagen of Prgama-ADE [14] has used a range of tools including METAPOST to show us the exciting future of on-line, fully active technical publishing based on ConTeXt and MathML (an XML vocabulary for representing mathematics [49]).

For our purposes the important ways in which Adobe's more recent product, The Portable Document Format (PDF), differs from PostScript is not the removal of programming constructs but the addition of the ability to specify, albeit in a somewhat limited and ad hoc manner, both interactivity as part of the formatted document and additional screen features of the document, such as navigation aids, that lie beyond its constituent pages. The similarity of the two languages was exploited by Mark Wicks in taking dvips as a basis for the development of dvipdfm [44]: this interprets TeX with embedded specials directly as PDF, avoiding the benefits and costs of Adobe's distillation process via PDFMarks.

5.2 Extending TeX

In 1995 Hàn Thế Thành, following suggestions by Peter Sojka and Jiří Zlatuška, started a project at Masaryk University, Czech Republic which he describes thus [16]:

> 'This research originated in modifying the TeX program to produce PDF output directly from TeX source without passing through the intermediate steps of DVI (the original TeX output format) .... This means providing capabilities that are needed for generating PDF files, like font downloading, graphics inclusion, etc.[38].'

Due, for example, to the existence of tools such as dvipdfm, integrated software such as pdfTeX is not essential to the production of PDF files from LaTeX source. However, such other methods involve post-processing the pages output by a standard TeX in order to generate a PDF document. Moreover, since PDF's model of text formatting is essentially the same as TeX's there is no gain in not writing PDF directly; and it is certainly natural for a document formatter to write directly in a language such as PDF.

However, the real power of the pdfTeX design is not that it writes PDF files but that it adds valuable access to formatting information and data structures that are completely internal to standard TeX, which throws away crucial facts, important to the integration of text and graphics, about the typesetting it has done when it writes out the formatted form of the document (even to a PDF file).

The other potential power of the pdfTeX approach is that it allows independent direct generation of all PDF objects, possibly even without producing *any* typeset pages! Such software will become increasingly important for the production of 'active documents' for which the output requirements of document formatters move beyond producing only sequences of pages (including long ones that are scrolled past a screen window).

As with other extensions of TeX, the syntax invented to access the new features is irredeemably Knuthian: using, within a text-based system, completely undelimited keywords as the syntax to specify command parameter values! The mixture of TeX code with directly embedded bits of explicit PDF gives typical pdfTeX documents a discomforting 'look and feel.' It appears that the description of such interactive documents is in need of a 'family sedan version' if this is not too wild a dream.

Thành's research has also highlighted the unsettled relationship between the simple TeX model of a formatted document, as no more than a collection of pages each of which contains just positioned glyphs and rules, and PDF's richer, object-based description of a document in which the page sequence object is just one component and within those page objects a formatted page can contain a relatively large range of atomic graphical elements. At the implementation level further complications arise from the need to interface both with PDF's internal object reference system and with TeX's array references.

The ability to write out such riches brings responsibilities as much as opportunities. In contrast with TeX, where the production of illegal formatted output is considered a bug, pdfTeX allows somewhat arbitrary output and at present there is little documentation of the PDF structures that it is supposed to produce. Is it perhaps necessary for a system whose primary task is to produce PDF files to make stronger checks on the validity of its output PDF? (Note that 'validity of PDF' should probably not be equivalent to 'accepted by some version of some Adobe product.')

6. AFTER 2.09

But where has LaTeX got to? Was there life after 2.09? Although all the above development was done in a LaTeX world, much of it does not require LaTeX; in particular, much pdfTeX development was done within the framework of Hans Hagen's ConTeXt [14], the comprehensive in-house TeX-based document processing system of the Dutch company Pragma-ADE who specialise in the production of multi-use documents for legal, business and education purposes—his work is a splendid example of what to use when LaTeX just won't hack it.

In one sense too much had been happening in and around LaTeX: under the hood of Leslie's 'family sedan' many mechanics had been labouring to add such goodies as supercharged, turbo-injection, multi-valved engines and much 'look-no-thought' automation.

6.1 Why a new LaTeX?

Thus the announcement in 1994 of the new standard LaTeX, christened LaTeX 2_ε, explains its existence thus:

> Over the years many extensions have been developed for

LaTeX. This is, of course, a sure sign of its continuing popularity but it has had one unfortunate result: incompatible LaTeX formats came into use at different sites. Thus, to process documents from various places, a site maintainer was forced to keep LaTeX (with and without NFSS), SLiTeX, AMS-LaTeX, and so on. In addition, when looking at a source file it was not always clear for which format the document was written.

To put an end to this unsatisfactory situation a new release of LaTeX was produced. It brings all such extensions back under a single format and thus prevents the proliferation of mutually incompatible dialects of LaTeX 2.09.

The development of this new standard LaTeX and its maintenance system was started in 1993 by the LaTeX3 Project Team which was then comprised of Frank Mittelbach, Rainer Schöpf, Chris Rowley, Johannes Braams, Michael Downes, David Carlisle and Alan Jeffrey, with some encouragement and gentle bullying from Leslie. Although the major changes to the kernel and the standard document classes (styles in 2.09) were completed by 1994, substantial extra support for coloured typography, generic graphics and fine positioning control were added later, largely by David Carlisle. Access to fonts for the new system incorporated work by Mark Purtill on extensions of NFSS to better support variable font encodings and scalable fonts.

Although the original reason was consolidation of the wide range of models carrying the LaTeX marque, what emerged was a substantially more powerful system with a controlled extension mechanism (via LaTeX packages) and a solid technical support and maintenance methodology. It provides robustness via standardisation and maintainability, of both the code base and the support systems. This system remains the current standard LaTeX and it has fulfilled most of the goals for 'a new LaTeX for the 21st Century', as we had envisaged them at the turn of the previous decade. The specific claims for the new system were: 'This version has better support for fonts, graphics and colour, and will be actively maintained by the LaTeX3 Project Team. Upgrades will be issued every six months, in June and December.' The details of how this was achieved, and the resulting subsystems that enabled the claims to be substantially attained, form a revealing study in Distributed Software Support since the core work was done in at least five countries whilst, as is illustrated by the bugs database [25], the total number of active contributors to the technical support effort is high.

Although the kernel has shown a little feature creep, the package system together with the clear development guidelines and the legal framework of the LaTeX Project Public License [27] have enabled LaTeX to remain almost completely stable whilst a very large number of workers have, as we are happy to acknowledge, extended the available functionality [24]. The major developments of the base system are listed in the regular issues of LaTeX News [26] but of even more crucial importance to the continuing relevance and popularity of LaTeX is the diverse collection of contributed packages. The success of the package system for non-kernel extensions is demonstrated by the enthusiasm of these contributors—many thanks to all them! It can be easily appreciated by means of the highly accessible and stable Comprehensive TeX Archive Network [1], a treasure trove of all things related to TeX for which we are extremely grateful to its developers and maintainers.

The provision of services and tools for such a highly distributed maintenance and support system was itself a major intellectual challenge since many standard working methods and software tools for these tasks assume that your colleagues are in the next room, not the next continent—and back then, e-mail and ftp were the only reliable means of communication. The technical innovations and the personalities of everyone involved were both essential to creating this example of the friendly face of open software maintenance but Alan Jeffreys and Rainer Schöpf deserve special mention for 'fixing everything.' A vital part of this system that is barely visible is the regression testing system and suite; this was devised and set up by Frank and Rainer with Daniel Flipo and has proved itself countless times in the never-ending battle of the bugs.

Perhaps the most worrying deficiency of the current LaTeX 2_ε kernel is that it contains little support for the needs of even the simplest of interactions such as hypertext links. The widespread use of PDF for formatted interactive and multiple-use documents has amplified this problem; and the access that pdfTeX has given to the advanced features of PDF has shown the need for some fundamental additions to the LaTeX kernel to support interactivity. The work of Sebastian Rahtz and Heiko Oberdiek, inspired by work done for the Los Alamos Pre-print Archive [6], on the hyperref package [31, 30] shows the dangers and difficulties of adding such functionality via a package on top of the current kernel.

6.2 A newer LaTeX?

History is not a totally ordered subject and our story now links back to the early 1990s when Frank Mittelbach, inspired by enthusiastic technical input from Denis Duchier and Leslie's ideas about improving LaTeX's user interfaces, was working hard on prototype designs and implementations of some very ambitious LaTeX3 project plans to use TeX's seriously inadequate programming interface to code a completely new document processing system (the LaTeX3 of myth and legend). Since 1997 some of these plans have been further developed by Frank and David Carlisle [24] and Frank has taken on the immense task of carrying forward the principles and methodology of standard LaTeX, adding to it declarative object-like interfaces and the underlying parameter handling technology. At present most of this is solidly based on the original paradigm of using TeX directly to process a stream of imperative typesetting commands but we are experimenting with other models for handling some aspects of building vertical lists and we are analysing the possibilities for working with an extended version of TeX (probably based on Omega) that will perhaps better support contemporary models of text handling.

Such a system will have two major advantages over anything else that will emerge in the next 10 years to support fully automated document processing: it will efficiently provide high-quality formatting of a large range of elements in very complex documents of arbitrary size; it will be robust in both use and maintenance and hence will contain the potential to be in widespread use for at least a further 15 years. It should also provide some integration with the fast growing world of XML documents—LaTeX for e-commerce? The current aim of this work is to make available some immediately usable interfaces and extensions that will work with standard LaTeX. As more functionality is added it will become necessary to assess the likelihood that this path will lead directly to a more powerful yet robust and maintainable system.

7. THE FUTURE

Having built up the background it is disappointing that time does not allow me to write much under this head; but a very encouraging boost has recently been given to research in automated typography through the recent Ph.D. dissertation from Hàn Thế Thành [16]. This is the first doctoral work on micro-typography in 'the legacy' since that of Knuth's own students [21] and it is one of only a handful on the analysis and development of the automation of quality typography in the last two decades. In it Thánh analyses some TeX-based implementations of some new font technologies and glyph choice mechanisms introduced by Hermann Zapf in the early 1990s [52]. The commercial development of his techniques has now been taken up by Adobe Systems and they have influenced the capabilities of the InDesign product.

I am not going to attempt to answer the question 'Whither TEX?' rather I shall suggest that TEX should form only a portion of the basis for further research and development of automated typography in the context of current activities and influences.

Most obvious amongst these influences is the rapid spread of XML as a common standard for information encoding. This implies that, for the formatting-independent parts of document processing, the utility and limits of 'XSL Transformation' technology [47] must be investigated since it certainly provides in this area suitable tools that are far more effective than anything programmed in TEX. The relationship between 'XML Style' [45], developed originally to control rapid, real-time formatting of browsable web documents but now extended to the wider variety of instant outputs needed for e-business systems, and 'LATEX Style' is close in concept but with significant design (both typographic and software) differences. We need to determine whether and how these two perspectives should be unified or be encouraged to tread separate paths.

The approach of most TEX-related research must be contrasted with that of the document engineers, whose recent mission statement (**`http://www.documentengineering.org`**) exemplifies an unreconstructed and somewhat limited vision of what a document processing system can be and hence of what is the proper compass of automated document processing. Non-WYSIWYG document formatting is considered by their orthodoxy to be almost exclusively an isolated, static transformation from a logical to a visual description of a document. By contrast, good information design requires the use of several dynamicly configurable, collaborative formatters. Many aspects of typographic quality are therefore dismissed by them as being for ever incompatible with automated document formatting. This leads me to suggest that we urgently need to do some document science so as to understand and guide the work of these engineers: it is not wise to engage engineers to build even primitive bridges before completing an elementary investigation of the gravitational constraints and available materials!

Many of the current projects using TEX as the basis for experimental developments in support of such document science are exciting and important but, as described by myself and Frank Mittelbach [28, 29], the glittering prizes for automated typography are unlikely to lie at the end of this line of development. Don and Leslie have both consistently expressed to me their surprise that TEX has not yet evolved into, or been displaced by, something clearly superior. I hope very sincerely that this historical review will inspire research projects in automated typography that will preserve and enhance the principles incorporated by Don and Leslie in their pioneering work, but without further undignified prolongment of the near immortality of their particular software designs and implementations.

8. ACKNOWLEDGEMENTS

Were I too do the job properly, this section would be by far the longest. The list of people and organisations deserving appreciation goes far beyond those whose names I have mentioned elsewhere.

Thus I will start by simply thanking, on behalf of the whole current and future scientific and publishing communities, all the Good Guys for the quality and quantity of their contributions to the work (both that described here and much else) and to the infra-structure that supports the production and dissemination of that work. Many of the items referenced contain their own long lists of acknowledgements; I have not attempted even a rough estimate of the cardinality of the set of people in the transitive closure of that process! What remains is largely personal thanks to individuals, many of whom supplied direct assistance in putting together this paper.

First there are three people who made it all possible for so many of us: Don Knuth for TEX and METAFONT in their literate forms and for his uncompromising dedication to these and many other ideals; Leslie Lamport for a production quality TEX-based document processing system and for being persuaded that the whole world needed it; and Barbara Beeton for nurturing the TEX community and TUGboat [5]. Two organisations must also be included here: The American Mathematical Society and Addison-Wesley.

Then there are the people I have worked closely with: The LATEX3 Project Team, past and present, including Robin Fairbairns; and Sebastian Rahtz, whose practical contributions to document processing are diverse, manifold and legendary, with his dry enthusiasm best summed up as 'never mind the bugs, feel the existence!'

The more theoretical leanings of the paper also owe much to the opportunities for lengthy and often heated discussions about typography and document science with anyone who would listen but particularly, over many years, with Frank Mittelbach who, very much in Leslie's tradition, will never let any sloppy thinking get by. In this context I shall also single out Joachim Schrod, Michael Downes, David Carlisle, Bernd Raichle and Donald Arsenau.

I should like to be able to thank the referees and editors for help in improving this paper ... but they do not appear to have existed so I do not need to explain that all the remaining errors are mine, or my spell-checker's or (is it possible?) bugs in LATEX.

9. BIBLIOGRAPHIC NOTES

As can be seen from the list of references below, much of the relevant material exists only as on-line documents or as the software systems themselves. Most of the software described here is available from The Comprehensive TEX Archive Network [1] (denoted **`CTAN:`** in the bibliography).

Except where otherwise indicated in the text, the developments described above are extensively documented in The Communications of the TEX Users Group [5] (denoted TUGBoat in this bibliography); this also contains many related technical announcements and papers. The TUG web-site (**`http://www.tug.org`**) contains or points to a large amount of useful material including information about local TEX user groups.

It would be good if this bibliography could contain more references to literate descriptions of the design and implementation of this software but too often this is impossible as the quality of the technical documentation does not follow Donald Knuth's [20] example of making it match the high quality of the software itself. A notable exception to this Peter Breitenlohners's contribution to the technical part of the e-TEX documentation [9].

10. REFERENCES

[1] CTAN Team. The Comprehensive TEX Archive. `http://www.tex.ac.uk/tex-archive` (denoted **CTAN:** below).

[2] Adobe Systems. PostScript Language Reference 3rd ed. Addison-Wesley, 1999.

[3] Adobe Systems. PDF Reference version 1.3 2nd ed. Addison-Wesley, 2001.

[4] Adobe Type Library. OpenType fonts. `http://www.adobe.com/type/opentype`.

[5] Barbara Beeton (Editor-in-Chief). The Communications of the TEX Users Group, vols 6–22. 1985–2001 `http://www.tug.org/TUGboat/tugboat.html`.

[6] Tanmoy Bhattacharya, David Carlisle, Mark Doyle, Paul Ginsparg, Alan Jeffrey, Hiroshi Kubo, Kasper Peeters, Sebastian Rahtz and Arthur Smith (Japanese translation by Kazuhito Ohya). HyperTeX FAQ. 1996–2000 `http://xxx.lanl.gov/hypertex`.

[7] Johannes Braams. The Babel system. **CTAN:**`macros/latex/required/babel`.

[8] Johannes Braams, Victor Eijkhout and Nico Poppelier. *The development of national LATEX styles*. TUGBoat [5] 10(3), 401–406, 1989.

[9] Peter Breitenlohner et al. The e-TEX manual. **CTAN:**`systems/e-tex/v2/etex_man.pdf`.

[10] Martin Davies. The Gutenberg Bible. The British Library Board, 1996.

[11] Michael Ferguson. MultiLingual TEX. 1987 (now an option in the standard Web2C implementation).

[12] Michel Goossens, Frank Mittelbach and Alexander Samarin. The LATEX Companion. Addison-Wesley, 1994.

[13] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. The LATEX Graphics Companion. Addison-Wesley, 1997.

[14] Hans Hagen. Official public CONTEXT distribution. `http://www.pragma-ade.com`.

[15] Hàn Thế Thành. pdfTEX. **CTAN:**`systems/pdftex`.

[16] Hàn Thế Thành. Micro-typographic extensions to the TEX typesetting system. PhD dissertation, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 2000 `http://www.fi.muni.cz/~thanh/download/thesis.pdf`.

[17] John Hobby. The METAPOST System. **CTAN:**`graphics/metapost`.

[18] Donald Knuth. Computers and typesetting, vols A–E. Addison-Wesley, 1986.

[19] Donald Knuth. The Art of Computer Programming, vols 1–3. Addison-Wesley, 1998.

[20] Donald Knuth. Digital Typography. CSLI Publications, 1999.

[21] Donald Knuth and Michael Plass. *Breaking paragraphs into lines*. Software Practice and Experience 11(11), 1119–1184, 1981.

[22] Leslie Lamport. LATEX: A Document Preparation System, 1st ed. Addison-Wesley, 1986.

[23] Leslie Lamport. LATEX: A Document Preparation System, 2nd ed. Addison-Wesley, 1994.

[24] LATEX3 Project Team. Standard and experimental LATEX. `http://www.latex-project.org`.

[25] LATEX3 Project Team. LATEX bugs database. `http://www.latex-project.org`.

[26] LATEX3 Project Team. LATEX News 1–14. 1994–2001 `http://www.latex-project.org/latex2e.html`.

[27] LATEX3 Project Team. LATEX Project Public Licence. `http://www.latex-project.org/lppl.html`.

[28] Frank Mittelbach. E-TEX: *Guidelines for future* TEX. TUGBoat [5] 11(3), 337–345, 1990.

[29] Frank Mittelbach and Chris Rowley. *The pursuit of quality—how can automated typesetting achieve the highest standards of craft typography?* In Vanoirbeek and Coray [43], 260–273.

[30] Heiko Oberdiek. PDF information and navigation elements with hyperref, pdfTEX, and thumbpdf. **CTAN:**`macros/latex/contrib/supported/hyperref/doc/paper.pdf`.

[31] Heiko Oberdiek and Sebastian Rahtz. Hypertext marks in LATEX: the hyperref package. **CTAN:**`macros/latex/contrib/supported/hyperref/manual.pdf`.

[32] John Plaice and Yannis Haralambous. The Ω system. **CTAN:**`systems/omega`.

[33] John Plaice and Yannis Haralambous. *Methods for Processing Languages with Omega*. Proceedings of the International Symposium on Multilingual Information Processing, Tsukuba Japan. 1997.

[34] J.K. Rawlings. Harry Potter and the Chamber of Secrets. Bloomsbury, 1998.

[35] Brian Reid. Scribe Document Production System User Manual. Unilogic Ltd., 1984.

[36] Tom Rokiki. Dvips: A DVI-to-PostScript Translator. 1997 **CTAN:**`dviware/dvips`.

[37] Joachim Schrod. *International LATEX is ready to use*. TUGBoat [5] 11(1), 87–90, 1990.

[38] Petr Sojka, Hàn Thế Thành and Jiří Zlatuška. *The joy of TEX2PDF—Acrobatics with an alternative to DVI format*. TUGBoat [5] 17(3), 244–251, 1996.

[39] Richard Southall. *Presentation rules and rules of composition in the formatting of complex text*. In Vanoirbeek and Coray [43], 275–290.

[40] Michael Spivak. LAmS-TEX. **CTAN:**`macros/lamstex`.

[41] THe TEX Users' Group. Just what is TEX?. `http://www.tug.org/whatis.html`.

[42] The Unicode Consortium. The Unicode Standard version 3.0. 2000 `http://www.unicode.org/unicode/uni2book`.

[43] C. Vanoirbeek and G. Coray, eds. Electronic Publishing '92. Cambridge University Press, 1992.

[44] Mark Wicks. The Dvipdfm User's Manual. **CTAN:**`dviware/dvipdfm/dvipdfm.pdf`.

[45] W3C. Cascading Style sheets and the EXtensible Stylesheet Language Formatting Objects. `http://www.w3.org/Style`.

[46] W3C. EXtensible Markup Language. `http://www.w3.org/XML`.

[47] W3C. EXtensible Stylesheet Language. `http://www.w3.org/Style/XSL`.

[48] W3C. HyperText Markup Language. `http://www.w3.org/MarkUp`.

[49] W3C. Mathematical Markup Language. `http://www.w3.org/Math`.

[50] W3C. Scalable Vector Graphics. `http://www.w3.org/Graphics/SVG`.

[51] Timothy van Zandt. PSTricks. **CTAN:**`graphics/pstricks`.

[52] Hermann Zapf. *About micro-typography and the hz-program*. Electronic Publishing: Origination, Dissemination, and Design 6(3), 283–288, 1993.