TEX: A branch in desktop publishing evolution — Part 1

Barbara Beeton, Karl Berry, and David Walden

Abstract

Donald Knuth began the development of TEX in 1977 and had an initial version running in 1978, with the aim of typesetting mathematical documents with the highest quality, and with archival reproducibility far into the future. Its usage spread, and today the TEX system remains in use by a vibrant user community. However, the world of TEX has always been somewhat out of sync with the commercial desktop publishing systems that began to come out in the mid-1980s and now are now ubiquitous in the worlds of publishing and printing.

Keywords: TEX, METAFONT, Computer Modern typeface, desktop publishing, open-source software, user groups

This history of the TEX typesetting system comes in two parts. Part 1 is about the creation of TEX and how it began to spread beyond Stanford. Part 2 is about the impact TEX has had on the broader world and the evolution of the worldwide community of TEX users and developers. (To save on the page count, most of this history's large number of notes and references are posted online at `dw2.tug.org/dtptex/webnotes.pdf`) [this is a temporary location; if the paper is accepted we will find a permanent web location for the Web notes].)

Initial TEX history

The Knuth years

The story of the creation of TEX and METAFONT has been told before,[1,2,3*] but we recount it here for readers who are not close to the TEX world.*

In 1976, Donald Knuth had revised the second volume of his magnum opus, *The Art of Computer Programming* (TAOCP),* and had the galley proofs back from the publisher, Addison-Wesley. However, in the years since the original edition of the volume was published, Addison-Wesley had moved, for cost reasons, from using Monotype typesetting machines* operated by expert typesetters of mathematics and other technical material to using computer-based phototypesetting methods. To Knuth's eye, the new galley proofs were so poor that he despaired about continuing his TAOCP book series.[4]

Knuth says, "The genesis of TEX probably took place on February 1, 1977," when he first saw output from a high-resolution digital typesetting machine that could produce high quality output.* However, it seems he was already thinking about improving computer-based typesetting by the fall semester of 1976, as he gave an assignment in one of his Stanford classes (CS204) that included the problem of breaking paragraphs into lines.* By mid-February he had decided to spend his upcoming sabbatical year (1977–78 school year) working on digital typography. On May 5 he began the design

of TEX and a May 13 memo sketched the system's design.* He spent the next 45 days writing software to create digital letterforms and produced the 26 letters of the alphabet. In early July he returned to thinking about TEX and produced a revised "reasonably complete specification" of the typesetting language, and he then headed out of town for the better part of two months, leaving his students Frank Liang and Michael Plass at Stanford to do a prototype implementation of TEX.* Plass remembers* that they managed to implement enough of TEX to process "a subset of TEX input all the way to XGP [Xerox graphics printer] output... Knuth... seemed pleased with our efforts, and proceeded to re-code TEX himself." Knuth makes the point that Plass and Liang had prototyped only a fraction of the full TEX and that it was important that he code the complete program because he learned so much from doing the coding.

In 1978, Knuth also continued and finished typesetting the entire 700-page revised volume 2 of TAOCP in TEX, thus substantially achieving his original goal: to typeset his opus digitally as beautifully as the prior generation of hot-metal math-specialty human typesetters could; in particular, finding a good way to specify high quality typesetting of math.

However, several other things happened in 1978 that gave TEX a much wider audience and longer life. (a) In January Knuth gave the AMS Gibbs lecture at which he demonstrated the necessity for moving beyond the then state-of-the-art of computer-based typesetting, and he explained what TEX would do.[5] (b) In August other people began to use TEX at Stanford*; and immediately Guy Steele ported TEX from the Stanford AI Laboratory PDP-10 running the WAITS operating system to MIT's PDP-10 running ITS. (c) The American Mathematical Society (AMS) made a strong move to position itself to use TEX in its publishing activities.

Thus, over the next several years, Knuth (often with the help of others, as described in the next section) greatly extended the TEX project: he rewrote TEX for greater portability from the SAIL language of the Stanford AI Laboratory to a subset of standard Pascal; created the METAFONT program for designing type fonts, and developed the extensive Computer Modern set of digital fonts (we describe this more fully in the next section); and then reimplemented (and improved) TEX and METAFONT again, now using a "literate programming"[6] system called WEB, which he invented for the job.* With WEB, Knuth combined capabilities for writing documentation and writing code, with enhancements for both forms.*

To summarize, over the period 1977–1984, Knuth himself wrote two completely different implementations of each of TEX (called TEX78, written in SAIL, and TEX82, written in WEB), METAFONT (METAFONT79 in SAIL and METAFONT84 in WEB), and the Computer Modern fonts ("Almost" Modern and Computer Modern) using the experiences from his first implementation to make the second version as close to the ideal as he could. In each case, the second version is definitive. In 1986, Knuth's 5-volume set of books, *Computers and Typesetting*, was published using and (self-)documenting the complete TEX system,[7] and he considered his work done.* In Knuth's view, it was critical to personally do the three major activities of coding, using, and documenting TEX and METAFONT, or the final system would have been markedly inferior.

After 1986 Knuth made only one more major change to TEX: in 1989 he extended TEX from using 7-bit characters to allow arbitrary 8-bit characters as input (and he improved a few other things in passing).* Then in 1990 he announced that his work on TEX and METAFONT was done, and that he would "make no further changes except to correct extremely serious bugs."

Knuth put his code in the public domain so that it, and the underlying ideas, could be used by anyone for any purpose. While only Knuth will make changes to canonical TEX and METAFONT and Computer Modern for the rest of his life (and they would remain unchanged after that), anyone can make modified versions of TEX or any part thereof, or use his code and algorithms in any way, as

long as they called their work something other than TEX, METAFONT, or Computer Modern. He said, "Let us regard these as fixed points, which will give the same results 100 years from now as they do today." This announcement is worth reading for yourself.[8]

True to his word, since 1990 Knuth has only worked on TEX or METAFONT every few years, at ever-increasing intervals, to review bug reports accumulated in the interim.* For many years Barbara Beeton was the collector of bug reports (she called it being the "TEX entomologist"). She asked knowledgeable people to review each report to discern whether it is in fact a bug, a bug that has already been reported, or not a bug. Every few years, when Knuth was ready to look at possible bugs, Beeton sent her collection to him. Knuth is the sole arbiter of what is a bug and what will be left unchanged. In January 2017, Karl Berry took over the entomologist role.

Knuth's release number for TEX at the time he announced he was done was 3.1. He has released seven (slight) updates of TEX since version 3.1, documenting each time what the changes were. His most recent two TEX tuneups were in 2008 (version number 3.1415926) and 2014 (version number 3.14159265)* (the version number of each update has one more digit of pi). He has stated that at the time of his death, his then-current version should be relabeled "TeX, Version π, and any remaining 'bugs' will be permanent 'features'."*

TEX markup

The general approach of TEX is to use plain text files containing explicit markup to tell the TEX processor how to typeset a document. The markup uses commands (called "control sequences" in TEX jargon) and special characters; for example, `{\bf now}` causes text to be output in bold face from the `\bf` command to the closing brace (the close brace undoes the effect of the bold face command).* TEX has hundreds of such commands. Several of these commands allow macros to be defined which themselves can then be called as commands. Below is a trivial example of a TEX macro definition and call:

```
\def\Greeting#1{Hi #1! I hope you're well.}
```

The `#1` before the open brace indicates that the macro is called with one argument; the `#1` within the braces shows where the argument of the macro call is substituted into the definition. If the above macro is called with `\Greeting{Sally}`, it results in: `Hi Sally! I hope you're well.`

TEX as users see it (known as "plain TEX") is a combination of (hundreds of) primitive TEX commands and (hundreds of) macros defined using the primitives and other macros. The LATEX version of TEX (created by Leslie Lamport ca. 1983), which is vastly more popular than plain TEX, is defined in terms of many more (thousands of) such definitions.

A tiny but complete example LATEX document is shown in Figure 1. When processed by TEX, this example compiles into what is shown in Figure 2, appropriately placed on a page numbered 1 (the page number and margins are omitted from the figure).

Users of contemporary What-You-See-Is-What-You-Get word processing and typesetting systems such as Word and InDesign may think that TEX's use of markup rather than WYSIWYG is primitive. In a sense it is. TEX was created at the very beginning of the era of graphical user interfaces; WYSIWYG word processing and its underpinnings were a subject of software and hardware research and development, and there were no widespread commercial WYSIWYG word processors or typesetting systems yet. More importantly, Knuth was trying to typeset documents with TEX that were highly precise, were archival in terms of reproducibility far into the future, and used complicated symbology such as mathematics. Arguably, a markup approach is more useful for these conditions. One can do things with macro-based markup that are more difficult to do with

Figure 1: Example of LaTeX markup

```
\documentclass{article}
\title{A Small Example}
\author{A.U. Thor}
\date{December 1, 2017}

\begin{document}
\maketitle

\section{Introduction}
In this example, the type of document
is declared, the title content is specified,
the document itself is started, the title
content is inserted, there is some content
including some math, and the document ends.

\section{Example including math}
The quadratic formula
is $$-b \pm \sqrt{b^2 - 4ac} \over 2a$$

\end{document}
```

WYSIWYG. For example, the LaTeX markup in Figure 1 is entirely "logical"; the example output in Figure 2 could be altered in nearly any conceivable way (different fonts, different spacing, different ordering of elements, ...) without changing any of the input except for a different class name.

TeX's macro capability and how TeX is extended or built upon, using its macro capability, are explained in more detail in the Appendix (following Part 2).

Collaborative community

Our goal with this history is to report what happened and to avoid hagiography. However, as context for the rest of the history, we note here some of Donald Knuth's characteristics as we have observed them over the years.

Knuth comes from an academic research environment and takes advantage of it. He deeply researches a topic area and its history, builds on the work of others, and draws others into his work (and acknowledges both prior work and his collaborators in exceptional detail). His work is comprehensive within the limits of the task he sets for himself. He cares about the beauty (in his own eyes) of the implementation as well as the beauty of the result. He builds what he thinks he should (or wants to) build without regard for markets. There are mathematical and computer science aspects of his efforts. He finds a scrupulously detailed and often aesthetically pleasing way to publish or otherwise share his results and methods with the public.

We think of the above as being the "Knuthian" way. We have seen it, for instance, in his work on *The Art of Computer Programming* (a multi-volume analysis of fundamental algorithms throughout

Figure 2: Output from Figure 1 markup

A Small Example

A.U. Thor

December 1, 2017

1 Introduction

In this example, the document type is declared, the title content is specified, the document itself is started, the title content is inserted, there is some content including some math, and the document ends.

2 Example including math

The quadratic formula is

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

the field of computer science); *3:16 Bible Texts Illuminated* (a treatment of chapter 3 verse 16 in each book of the Bible, in collaboration with renowned calligraphers);* and *Fantasia Apocalyptica* (a long composition for pipe organ — in Knuth's words, a "somewhat literal translation of the Biblical book of Revelation into music").*

The history of the TEX project, collaboration, and community that we describe in the rest of this two-part history grew out of the Knuthian approach — a forty-year story of contribution and collaboration.

Stanford users and collaborators for TEX

Knuth's funding allowed him to have graduate assistants with whom he could collaborate on TEX.

We noted above that Frank Liang and Michael Plass wrote some TEX prototype code in the summer of 1977. Frank Liang also developed the original TEX hyphenation algorithm in the summer of 1977 in collaboration with Knuth; Liang's thesis research was completed in 1983* and included a major improvement to the original hyphenation algorithm, and Knuth adopted Liang's algorithm for TEX82. Knuth said that Liang's experience with the original algorithm "led him [Liang] to discover a much better way, which can adapt to all languages"[1] (all languages that are hyphenated).

After finishing his PhD at Stanford with Knuth as his advisor in 1978, Luis Trabb Pardo stayed on helping Knuth with TEX in the years 1978–1981. He was the associate director of the TEX project (Knuth called him "my right-hand man for TEX78"), mostly working with people that were obtaining TEX and looking into what could be done to print document proofs on various output devices. He was also a reader for the PhD theses of Michael Plass and Frank Liang, who both had Knuth as their thesis supervisor.

Knuth has stated, "One of the most challenging aspects of page layout is the problem of breaking paragraphs of text into individual lines. The fact that computers can do this task better than all but the most dedicated hand-compositors was one of my early motivations for developing the TEX system."[1] He understood this already when he left Liang and Plass to do their TEX prototyping in the summer of 1977. Michael Plass collaborated with Knuth in studying and documenting TEX's automatic line-breaking algorithm.*

In 1978 David Fuchs, then a graduate student at Stanford, came to work on the TEX project. He ported the SAIL version of TEX from the SAIL WAITS operating system to DEC's TOPS-20 operating system; conceived and developed the first version of device independent (DVI) output for TEX (more about this shortly); served as Trabb Pardo's successor as "right-hand man" for TEX82, the WEB implementation of TEX; and reported to the TEX community on TEX activities at Stanford until 1986,* as Knuth was winding down the TEX project. Knuth dedicated Volume B of *Computers & Typesetting*[7] "To David R. Fuchs: Who fixed everything."

John Hobby used TEX78 from the time he arrived at Stanford as a PhD candidate in 1980. He became involved with METAFONT79 and worked with Gu Guoan (whom Knuth brought from China as a visitor to Stanford) to develop a prototype set of Chinese characters. METAFONT's polynomial pens were an important part of his 1985 thesis (for which Knuth was thesis advisor).* Hobby contributed various other algorithms to the development of METAFONT, including METAFONT's representation of cubic spline (Bézier) curves. Hobby recalls that he and Knuth worked together on METAFONT in an unusual teacher-student collaboration in that Hobby mostly designed algorithms and Knuth wrote all the production code.* Knuth calls Hobby his "tireless co-conspirator throughout METAFONT's development", and he dedicated Volume D of *Computers & Typesetting*[7] "To John D. Hobby: Who improved everything." After leaving Stanford, Hobby developed MetaPost, building on many aspects of METAFONT while adding many capabilities, most critically PostScript output. (Knuth is an enthusiastic user of MetaPost for his diagrams in TAOCP and elsewhere.)

Ignacio Zabala was another of Knuth's PhD students (his thesis was also relevant to the TEX project). In April of 1980, Zabala (with Trabb Pardo) completed a conversion of Knuth's implementation of TEX in the SAIL language to Pascal, to greatly increase portability of the TEX system. This conversion was done with the DOC system (also implemented by Zabala) that was a predecessor of Knuth's WEB system of literate programming. Zabala's Pascal implementation continued to track changes to the SAIL implementation until TEX82 became available.*

While at Texas A&M, Tomas Rokicki created the initial version of what became the Web2C system that is used today to compile TEX. Rokicki started working for the TEX project in the summer of 1985, before his first term as a PhD student at Stanford. There he designed and implemented the PK font format and tools, a more compact form for the bitmaps output by METAFONT than the format previously used.

In other parts of this history, we will mention other people at Stanford who helped Knuth develop TEX.*

TEX leaves Stanford

While Knuth's initial idea for TEX was merely to be a tool for him and his secretary to use to produce *The Art of Computer Programming*, his work was too finished and too capable even in its initial implementation to remain very long just for personal use; plus, his inclination was to share the results of what he was doing and reasons for doing it.

Making TEX portable

As mentioned above, Knuth spent much of his 1977–78 sabbatical year getting TEX working on the Stanford AI Laboratory PDP-10 computer running the WAITS operating system, and able to drive SAIL's XGP printer. By the summer of 1978 he had TEX basically working, was trying to get a manual for TEX finished,[9] and was fixing bugs in TEX but already trying to deflect requests for new features.[2] This original implementation of TEX had five components: TEXSEM.SAI to handle the semantics; TEXSYN.SAI to handle the syntax; TEXSYS.SAI, to do the processing; TEXEXT.SAI for possible extensions; and TEXOUT.SAI to produce XGP output.*

There were three technical dimensions for making TEX portable: getting the code for TEX to run on other computers; making TEX able to output to other printers, and making TEX operate precisely the same when running on different computers.

While Knuth was still finishing the initial implementation of TEX in the summer of 1978, Guy Steele (at Stanford for the summer from MIT) began porting TEX to the MIT AI Lab PDP-10 computer running the ITS operating system and made TEX at MIT work correctly with the AI Lab printer. (MIT and Stanford were both connected to the ARPANET which permitted file transfer and remote login to computer systems.) Steele also gave Knuth many changes for his draft TEX manual and convinced Knuth to add a few new commands to TEX. Steele made his changes in the TEXEXT.SAI module. TEX was already running at MIT by August although Steele did a little more work for the MIT printer in September–October when he was back at MIT. Usage of TEX at MIT spread quickly as the port reached completion, with a user group being formed at MIT almost immediately.**

Around 1980 Knuth asked David Fuchs to write a new printer driver (namely, TEXVER.SAI) to replace TEXOUT.SAI (which drove the XGP) to handle a new Versatec printer they were getting. Fuchs noted that writing a new driver module in order for each new printer to work with TEX was a bad idea and instead created a device independent (DVI) page description language, with its module, TEXDVI, being an optional replacement for the TEXOUT.SAI module.* From then on, drivers for different printers could be written using the DVI output* without having to worry about properly linking into the TEX executable itself.

As Knuth discovered that more and more people saw TEX as being useful to them, he then set about (in the Knuthian way) making it as portable as possible at all levels.

By early 1980, as mentioned above, the SAIL implementation of TEX had been reimplemented in Pascal by Ignacio Zabala, so TEX could be compiled to run on computers without the SAIL programming language. This was called PTEX. As additional changes were made to the SAIL version of TEX, they were also moved over to PTEX (this included adding a transcription of the TEXDVI module from Fuchs). There were two versions of PTEX. The first version merely copied the initial SAIL version of TEX in using PDP-10 floating point arithmetic; however, different computers used different standards for floating point arithmetic. The second version of PTEX implemented machine independent arithmetic within TEX, so TEX output would be exactly the same on different computers regardless of what form of floating point arithmetic they used.*

At this point, Knuth began redoing TEX and METAFONT using his WEB development system to replace the DOC system used by Zabala while maintaining Pascal as the base implementation language. Knuth scrupulously avoided any aspect of Pascal that was either machine dependent (e.g., floating point, as mentioned above), or too idiosyncratic (e.g., pointers); this maximizing of source code portability greatly eased future work on TEX. The resulting definitive TEX82 version of TEX included many other improvements based on user experience with TEX78 and feedback.*

Starting with PTEX and continuing with the WEB version of TEX, many non-Stanford people were able to join the TEX porting effort.

AMS involvement

The AMS had heard rumors about what Knuth was doing with typesetting, and people from there paid a visit to Stanford where Knuth showed them what he was doing, along with some mockups of how TEX could be used for journals as well as for books. Possibly influenced by that visit but also surely based on his computer science work with the analysis of algorithms (e.g., the TAOCP volumes), Knuth was invited to give the prestigious AMS Gibbs lecture in January 1978. Rather than talk about "about the glories of computer science" (as he was expected to do),[1*] Knuth used the example of the *Transactions of the American Mathematical Society* to illustrate the poorer quality of typesetting since the AMS had ceased "to use traditional methods of typesetting," i.e., Monotype (hot metal type). He then described the possibilities for improvement using computer-based typesetting and sketched his work with TEX.[5*]

The response was enthusiastic, confirming Knuth's view that there was a need for a TEX-like system in the math world. In particular, Richard Palais, the chairman of the AMS Board of Trustees, pushed AMS's adoption of TEX, sending a delegation to Stanford for a month to learn TEX and to bring it back to the AMS in Providence. Two of that group were Barbara Beeton, who was to learn TEX and install it at the AMS, and Michael Spivak. (Beeton believes that the AMS management overestimated how finished TEX was as a production product.) In the fall of 1979, David Fuchs visited AMS and set up TEX on a DECSystem 20, and the organization started using TEX for administrative publications (member list, sales catalog).

Michael Spivak, who was known to be a clear writer of mathematics, was commissioned by the AMS to write a tutorial guide to TEX, which Spivak titled *The Joy of TEX*.* The AMS was also looking to commission someone to create additional macros on top of plain TEX, to make TEX even more suitable for writing papers containing advanced math. Things evolved such that Michael Spivak also received the commission for this software, which became AMSTEX.[10*] With AMSTEX developed and tested, AMS began accepting TEX input for its *Proceedings* and *Transactions*, with the first published issue happening in 1984.

The AMS also co-published, with Digital Press in 1979, another relevant book — *TEX and METAFONT: New Directions in Typesetting* by Knuth.[11] Gordon Bell remembers* that Digital Equipment Corporation was starting Digital Press around that time. Bell, a legendary computer designer and then a vice president of Digital, had met with Knuth about computer standards for floating point arithmetic, and thinks he heard of the TEX effort then. Bell passed the news of TEX to Heidi Mason who contacted Knuth about publishing with them. The 1979 book contains a copy of Knuth's Gibbs lecture and, slightly updated, his Stanford reports on the SAIL versions of TEX[9] and METAFONT[12] (Bell wrote the foreword to the book).

In the early 1980s, the AMS commissioned Hermann Zapf to create a new typeface called AMS Euler,* to be done in collaboration with Donald Knuth using METAFONT79. The new type design had the goal of being more like how mathematicians handwrite math.* The project was monitored by an AMS "font committee" reporting to the AMS trustees. (The project is discussed more in and around Table 1.)

According to Gaudeul's 2003 research paper in *TUGboat*[13]; the AMS support of TEX was part of a deliberate plan to "become involved in helping to develop a document preparation system, instead of waiting for a commercial system to be provided to them." They wanted "a system that was compatible with most hardware, simple, flexible, and cheap; it was to run on mainstream

computers." It seems that the AMS was anticipating mathematicians preparing their own papers on the time-shared computers of the time.

The AMS endorsement of TEX in the above ways (and others) undoubtedly led to faster general acceptance of and enthusiasm for TEX. In particular, the AMS was significantly involved in establishing the TEX Users Group and supported the first issues of *TUGboat*, the main publication of the users group.

TEX Users Group

The formation of the TEX Users Group (TUG) was an important step in TEX's becoming widely popular and in TEX development activity eventually becoming independent of Stanford. We will touch on the early days of TUG here. Part 2 of this history will discuss the later evolution of TUG and expansion of the user community to include "local user groups" for various national/language communities.

In addition to assisting Knuth with TEX developments (e.g., TEX-in-Pascal and device independent output) as discussed earlier in this section, Knuth's collaborators at Stanford were dealing with distributing TEX to the people who had learned about it, answering questions as people moved PTEX to different computers, and so on. This could not go on forever; a non-Stanford structure was needed.

We are not quite sure of the steps leading up to it, but the first meeting of the TEX Users Group was held at Stanford on February 22, 1980.* David Fuchs and Richard Palais suspect the core invitation list for this first meeting came out of Stanford. (However, inside the back cover of *TEX and METAFONT: New Directions in Typesetting* were two postage-paid business-reply postcards to be returned to the AMS, "if you have an interest in participating in a TEX users' group"; and various informal invitations were doubtless made). About 50 people attended the meeting. People were told how to get the files for TEX with or without ARPANET access. Knuth reported on the status of TEX (stability, interchangeability, and a common version were more important than different people trying to add "missing" features) and of METAFONT (more than a dozen users); he also answered questions (e.g., TEX should be applicable to non-math scientific documents, but he discouraged non-math use until the AMS gathered more experience). The coming Pascal version of TEX was described. There were reports on some macro packages that users had developed and experiences porting to particular machines. Finally, a steering committee for the group was elected. This was described in the first issue (October 1980) of *TUGboat*, the user group's publication,* which also contains introductory material on TEX and half a column on what TUG would be doing in the near future — being a clearinghouse for information on TEX and helping get the Pascal version of TEX into the field. (As implied, another of the early decisions of TUG was to create *TUGboat*.)

In its early years, TUG's (and therefore *TUGboat*'s) efforts were involved with spreading the word about TEX, getting TEX running on many different computers and operating systems (including on the early "small" computers),* getting TEX output supported on a variety of printers (in those pre-PostScript days), and sharing instances of the use of TEX. Also over these years, Knuth and others at Stanford decreased their involvement in TEX distribution and support, while TUG members and others increased their involvement, with the AMS continuing to help. For instance, in addition to ARPANET file transfers or TEX on magnetic tapes or from Stanford, there was (for several years) a"Unix TEX" tape organized by Richard Furuta and Pierre MacKay at the University of Washington, a DEC VMS tape (for a shorter time), and distributions on diskettes for Windows (e.g., from Jon Radel, and the commercial PC TEX).

As TUG evolved, *TUGboat*'s contents also evolved.[14] *TUGboat* has served as a TUG (and the broader TEX world) newsletter about projects, events, and people. It has simultaneously provided

tutorial material for all levels of TEX practitioners, a forum for new ideas to be suggested and experiments to be described, and a place for major new developments in the TEX world to be permanently documented.*

TUGboat has never primarily been a journal of pure academic scholarship. Its articles are always reviewed by knowledgeable people (often the editors themselves) but do not generally undergo anonymous refereeing. Nonetheless, *TUGboat* has served the typical role of a scientific or engineering journal in allowing participants in the field to learn about and build on (or create alternatives to) the work of others. Furthermore, it has played a role beyond TEX, regularly dealing with non-TEX issues of typography, design, document preparation and display.* As of the end of 2017, *TUGboat* has published 120 issues (38 yearly volumes) with an average of about 370 pages per year.

Type design collaboration

We mentioned METAFONT and Knuth's work with type design in the first section, but we primarily focused on TEX: TEX is more widely known, used, and appreciated than METAFONT; and, in the view of many, TEX more satisfactorily solved the problem TEX was meant to take on than is the case for METAFONT. METAFONT's approach to type design was and remains unique among all type design tools, for better or worse.

Knuth started working on digital letterforms as soon as he had a draft description of TEX, before he had implemented TEX. In so doing he began to learn the subtleties and intricacies of type design and began building a computer-based tool to aid in creating digital fonts. He has said it took him seven years before he was satisfied with his work on type design (typography practitioners have said that this is a typical length of time for someone to become a competent type designer).QWnotetypography practitioners have saidFor example, Charles Bigelow said this: Note on Typeface Protection, *TUGboat*, vol. 7, no. 3, 1986, pp. 146–151, `typographypractitionershavesaid`

Digital type

Digital type ultimately involves a rectangular array of pixels that represent any character — letters, numbers, and so on. One could thus create a character of digital type by explicitly defining an array of 0s and 1s saying which pixels are off and on; such creation of bitmaps (by humans) was common at the time TEX and METAFONT were developed.

These days, however, the typical approach is to specify the positions of various key locations in characters (for instance, the top and bottom positions of a letter I, or a few key points along the curve of a letter C), and then have a mathematical function that smoothly specifies the outline of a character or paths among key locations. These mathematical functions then must be rasterized — turned into discrete pixels — for a particular device for the character to be displayed or printed.

METAFONT takes this mathematical specification a step further: a character is defined not just by bare points and curves, but by a generalized computer program, written in the METAFONT language. This programming approach allows for, among many other things, shared subroutines to draw common elements of characters, such as serifs.

An example program, the METAFONT code for a letter A, is shown in Figure 3, along with a so-called "smoke proof" of it, showing the positions of the points defined, the outline of the drawn character, and the box in which the character is typeset (thus showing the space inserted to the left and right of the character itself).

The METAFONT program interprets these character definitions and outputs a bitmap for each; the user must specify the assumed output resolution for this rasterization.

Knuth learns about (digital) type design

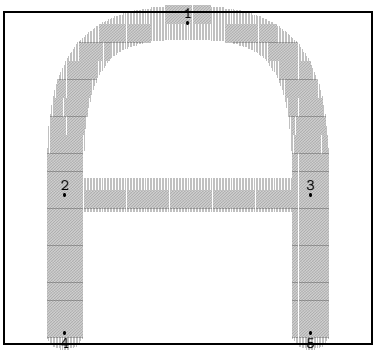
Knuth started his type design effort by researching the font used for the last well-printed volume of TAOCP, which was Monotype Modern 8A. This thus became the basis for Knuth's Computer Modern design. His wife, Jill, took 35mm photos of the printed letters, and they blew up the images by projecting them down a long hallway where Knuth traced them onto a big piece of paper. Quickly reaching the limits of what could be done with the photographic enlargements (distortion from the projector was significant, for instance), he studied books on the history and practice of type design and carefully inspected many typefaces. He then "began to plan for a unified design in which all the shapes would change gradually as the overall specifications of an alphabet were varied." He worked on that software using the data he had sketched from the wall and began to produce machine drawn letters, in June 1977.* In July he took a previously planned trip to China while Liang and Plass worked on prototype code for TEX.

Arriving back at Stanford, Knuth developed TEX's basic text processing without math, and then began to develop fonts for TEX to use. Beginning in the spring of 1977 a prototype set of subroutines and macros was written to develop the first iteration of what become Knuth's Computer Modern fonts; characters in this proto-METAFONT were defined directly in SAIL. After several hundred characters had been designed, Knuth developed an interpretable language called METAFONT with which to express character shapes more easily and precisely, and had a set for testing by March 1978.

After working on fonts for some months, Knuth began to read about how mathematics was typeset; and from the (conflicting) ideas in the books, he "was able to put together a picture of how a decent job of mathematical typesetting could be done." After giving the Gibbs lecture in early 1978, he spent four months developing the math processing parts of TEX, and had TEX running in the fall of 1978.[9]

Figure 3: METAFONT code to draw the letter A in the font used for the logo METAFONT, all code written by Knuth; the comments (starting with %) are editorial. The drawn character is shown graphically to the right, with points labeled. The `leftstemloc` and other variables not defined here are specified globally, ensuring consistency throughout the font.

```
beginlogochar("A",15); % width w = 15 units
  % x positions of the points:
x1=.5w;
x2=x4=leftstemloc;
x3=x5=w-x2;
  % y positions:
top y1=h+o;
y2=y3=barheight;
bot y4=bot y5=-o;
  % draw the straight lines and curved top:
draw z4--z2--z3--z5;
super_half(2,1,3);
labels(1,2,3,4,5);
endchar;
```



Then, based on his initial experiences with type design, Knuth reimplemented the METAFONT font generation system in 1979, coding it himself in SAIL.[12] As noted above, John Hobby researched and devised algorithms that Knuth implemented. Knuth completed the first prototype of Computer Modern in January 1980, being largely self taught.

He continued to read about font design and the history of letterforms, and he talked to people in the industry, for instance at Linotype. As sketched in the next subsection, Knuth was introduced to or came in contact with experts in type design, often through the digital typography program at Stanford that Knuth initiated with the students aiming for a Master's degree and with various experts spending periods of time at Stanford involved with the program.[15] These designers critiqued the fonts in the Computer Modern typeface Knuth was developing, and he improved his designs based on their comments.

By 1981 Knuth had hundreds of specimen sheets that were critiqued by Matthew Carter in England and Charles Bigelow and Kris Holmes in the United States. Richard Southall came to Stanford during April 1982, and he and Knuth made many changes; Southall was especially influential with the sans serif variant of Computer Modern. After these years of continuing study and refinement, in 1984 Knuth finished a substantially new METAFONT language and system, incorporating ideas especially from Zapf, Bigelow, and Southall. Perhaps the most significant change in METAFONT84 was that it was based on outlines, where the inside and outside of a path could be defined separately and explicitly, instead of solely on pen strokes. The Computer Modern fonts were improved again for the new METAFONT, benefitting from critiques by Southall, Carter, and N.N. Billawala.*

In the course of the TEX project, Knuth obtained an Alphatype CRS typesetting machine, which had an extremely high resolution. With the Alphatype, his improved fonts, and TEX, Knuth was able to generate a good looking high resolution set of pages for the revision of Volume 2 of TAOCP which he sent to Addison-Wesley where the pages were photographed for offset printing. The book was published in 1981, five years after Knuth had received the proof copy of the book that he considered unacceptable and four years after he started work on TEX.

METAFONT and the Stanford digital typography program

If it is not clear already, METAFONT is both a language processor (a program of some 27,000 lines) and a programming language (Figure 3). The METAFONT processor compiles routines written in the METAFONT language to produce characters for digital fonts.

As mentioned above, a number of noted calligraphers and font designers collaborated with Knuth at Stanford during the development of TEX and METAFONT and the Computer Modern fonts. Much of this effort took place as part of the Stanford digital typography program.

Charles Bigelow first met Knuth in 1980 when John Seybold (of the *Seybold Report on Publishing*) organized a small workshop at Stanford involving people from several organizations to explore and discuss TEX and METAFONT with the thought, "to introduce them to a wider audience and to encourage support."* Seybold and his son Jonathan had previously done consulting work with SRI and "had been introduced to Donald through our SRI contacts. We were fascinated by what Donald was doing and thought it deserved wider attention."**

In 1982 Bigelow moved to Stanford from the Rhode Island School of Design, bringing some of his type design students with him. At Stanford he was an "associate professor (teaching)" of digital typography and was associated with both the studio art department and the computer science department. Bigelow led a Knuth-initiated Master's program in digital typography at Stanford. He was on Stanford's faculty for thirteen years and taught type design, typography, and the history and theory of writing.

The content of the following block-quoted material came to us from Charles Bigelow. It is not literally a quote but rather informal email text from Bigelow* edited, often paraphrased, and reordered by us.

> Bigelow reports that they "tried to tailor the courses to what students needed and to what we thought digital typography should become. Because it was all new, there was no standard curriculum. We made it up as we went along." The design students, who did not have a background in computing, took a basic computer science programming course. The students also took some courses in art and design. Mostly they were free to choose what interested them, because the program was "exploring and creating a new field". The group did not know where new ideas would come from or what would be needed to develop them, apart from a basic grasp of both computer science and typography.
>
> Bigelow taught courses, a notable example being "grammatology",* and he taught a course that covered basics of type history, letterforms, typography, and type design. (His partner, Kris Holmes, taught an occasional evening course on calligraphy.) Later, Bigelow merged several of the subjects into a one-quarter course called "Concepts of Text" that he taught every year and which combined some theory of writing systems with some history of typography with some technology with some linguistics.*
>
> There was no typesetting course per se, but Stanford's computer science department had Alto computers with graphical user interfaces, and the students did some typography projects on those and liked the WYSIWYG interface. There were lessons on TEX and on METAFONT; the latter was a major focus because Knuth had gotten TEX to a stage that he could say was nearly finished and was now actively developing METAFONT, "so the students were in the METAFONT orbit."
>
> At Stanford, in the early 1980s, Knuth would meet with interested students and colleagues at lunch to discuss a wide range of questions and problems that came out of his research. He called it the "METAFONT for lunch bunch".* There was recurring discussion about whether there was an optimal class of mathematical curves that would render the best shapes while offering the most natural and intuitive manipulations by designers. (Polygonal approximations and circular arcs were rejected, for reasons described below. Knuth preferred cubic polynomials, perhaps as a matter of mathematical aesthetics, and eventually settled on curves based on based on Bézier cubics, a case of Bernshteĭn polynomials.* Vaughan Pratt, a Stanford computer science colleague and former student of Knuth, proposed another approach — generalized conic splines. Pratt described his approach to conic splines to the METAFONT group and later published a series of papers on the subject, beginning with "Techniques for conic splines", SIGGRAPH, 1985.)
>
> From 1982 onward, the "METAFONT for lunch bunch" met once a week. They included Knuth's graduate students John Hobby, David Fuchs, and Scott Kim, occasionally Howard Trickey, and sometimes others; in 1982–83, the design students usually attending were Daniel Mills and Carol Twombly, and then in 1983–84, Cleo Huggins and David Siegel. Lynn Ruggles, a PhD student from UMass Amherst, came in 1982.* Knuth was the final arbiter of what would be done with METAFONT, but the discussions were wide ranging, from the best mathematical form for curves to the perceptual quality of different curves, to the user interface.
>
> Issues with type design tools were also hot topics. The consensus was that outline fonts were the way to go, that polygonal and circular arc outlines were not good enough perceptually or mathematically, that general conics — proposed by Vaughan Pratt — were

> good and were somewhat more comprehensible to designers than cubic curves, but cubic curves of the Bézier–Bernshteĭn parametric form were mathematically Don's eventual favorites. All the designers wanted a visual graphical interface, but Don liked the keyboarded programming-like interface.

During this time and sometimes as part of the digital typography program, Knuth has acknowledged the help and influence of font designer Matthew Carter, though he was at Stanford for only a short time. Knuth has also noted Gerard Unger, who spent February 1985 at Stanford. We have already mentioned Charles Bigelow and Kris Holmes. Richard Southall also worked close by and visited Stanford for several periods of time and influenced Knuth.*

In August 1983, Bigelow arranged the first academic conference on digital type design, "The Computer and the Hand in Type Design", held at Stanford.** Also, a number of interesting reports came out of the program.*

In another important learning step, for several years members of the Stanford digital typography program collaborated with Knuth and Hermann Zapf on a commission to design the Euler typeface for the AMS (Table 1).*

Ultimately, however, the digital typography program didn't continue. Bigelow notes that although the program was begun under auspices of both the computer science and studio art departments, the studio art department was not completely on board and wouldn't award MFAs to the students or spend a faculty position on Bigelow. Knuth thus had to obtain outside funding and arranged for the digital typography students to get MS degrees through the computer science department. Bigelow notes, "The difference between an MFA and an MS might have made a difference if they had been interested in teaching careers, but they were all eagerly hired by high-tech firms, especially Adobe, and did very well, most retiring early."

Bigelow continues, "Without the art department connection, the program could not continue to attract visually talented design students, so the program would be [no] more than a minor sideline of computer science. Then Knuth went into early retirement around 1989–90 to go back to working on TAOCP and fully retiring a few years later, and that ended the program." Bigelow stayed on a Stanford teaching courses for a few more years, but was mostly consulting to industry by this time.

Bigelow suggests that for a brief time they had not only the best program in digital typography but also the most visionary. "There was not much written *about* the program, as we were too busy doing research, organizing, teaching, advising, . . . "

Knuth remembers that the digital typography program produced "more than a dozen students who have a masters degree in typography, basically, and they've been extremely important in the industry since then."* In particular, the METAFONT class which Knuth, Bigelow and Southall taught* was important to Knuth. He also remembers, "that was a big point in the development of METAFONT because I was implementing the features of METAFONT one week before they were introduced by the class." The font language finally stabilized in 1984, the manual for METAFONT was written, and then he had to do the final revision of the fonts, to make them look acceptable. "Let's say I finished that in 1985, and so finally, after eight years, I was able to bring my typography project to a conclusion. It was supposed to be a one year project for my sabbatical year."

Funding the development project

At a May 2017 desktop publishing pioneers meeting at the Computer History Museum, Knuth discussed funding sources for his work on TEX and METAFONT at Stanford.* He explained that he had never been very good at "understanding money", but things at Stanford did have to be paid for.

Table 1: Timeline for the development of the Euler typeface (derived from Siegel's monograph[14])

- In 1980 Zapf and Knuth began a collaboration to develop the Euler typeface, under a commission from the AMS, with Zapf doing the design and Knuth using METAFONT to do the digital work.
- Zapf sent paper designs to Knuth who had Scott Kim try to turn the designs into METAFONT penstrokes. Because the strokes didn't have well defined center lines, each letter created new programming challenges.
- In February 1980, Knuth demonstrated the first experimental METAFONT system to Hermann Zapf, and they worked together intensively for two weeks improving many of the character shapes, Knuth learning many things he continued to use.
- In 1981, the AMS commissioned Charles Bigelow to report on the feasibility of using METAFONT to produce mathematical characters ("Evaluation of METAFONT as a production tool"). In the report, he said METAFONT had potential but needed improvements to achieve production font standards. In particular, he suggested input from a graphics tablet specifying contours only.
- Bigelow and four students came to Stanford in 1982 to start the Digital Typography Group.
- Students Dan Mills and Carol Twombly experimented with Bigelow's idea of specifying two one-pixel-wide boundaries for each stroke. (This produced better shapes but was not "meta".) They then measured (literally) locations and directions of points on the boundaries and fed these to METAFONT in a way that allowed METAFONT to draw the character. This allowed a person to create one character in a few hours.
- Zapf came to Stanford in February 1983 for two weeks to review the work of Twombly and Mills.
- After Zapf's visit, production began using the outline method. Building on an approach from Lynn Ruggles, Siegel wrote a program to take data from a high resolution digitizing tablet and convert it to METAFONT input format. Using these tools, Siegel could produce 12 to 15 characters a night. (The program was demonstrated at the ATypI seminar at Stanford in August 1983.)
- Next the characters were tried on the Computer Science department's Alphatype CRS and corrected again based on informal notes by Zapf.
- In October 1983, David Fuchs made TEX work with the Alphatype, high resolution characters were printed, and these were sent to Zapf for critique in January 1984. Zapf had minor corrections for 60 of the 500 characters, and, in a later review, half a dozen more corrections. Some of the problems were that the proofs looked fine but at actual type size the letters were not quite right. With several months of work, John Hobby wrote a program that processed the font files (with new commands that Siegel had to add) and the letters got better in lower resolutions.
- The first batch of fonts was sent to the AMS in September 1984.
- A new METAFONT language and system was developed in 1984 incorporating ideas from Zapf, Bigelow, and Southall.
- A decision was made to convert the Euler fonts to the new METAFONT. Another program by John Hobby helped Siegel with this conversion.
- In August of 1985, new results were sent to Zapf, who made a few corrections, and then gave his overall approval.
- The Euler typeface was delivered to the AMS in September 1985, three years after the Digital Typography Group took on converting Zapf's designs to METAFONT — 484 characters total.*

It was different than starting a business. He had NSF grants and Office of Naval Research contracts to study algorithms, for which he needed to say what he was spending money for under the grant or contract terms. To inform NSF that he was doing (lots of) work on TEX, he noted in a report that, "by the way, I have some software that was used to do the formatting to help me prepare these papers" [on his algorithms work]. In this way he was able to have graduate assistants to help him with TEX and METAFONT. He also had a couple of tens of thousands of dollars in funding from IBM which had no restrictions on use. He also had a private donor who gave $200,000 to help get Charles Bigelow to Stanford.

Eventually the NSF said that, while the TEX project was doing good work, it was not an appropriate area for NSF to fund and thus introduced Knuth to Charles Smith of the System Development Foundation.* As Knuth explained it, the Systems Development Corporation had money remaining after its work on the missile shield was done at the end of the 1950s. This money was turned over to the Foundation to be distributed for "good works". The Foundation had already given a million dollars to Stanford's music department. Smith asked Knuth what he needed to finish the project; Knuth gave him a number, but Smith said, "you have to ask for more." (On video 64 of note 2, Knuth says that System Development Foundation provided $1 million in funding so he could finish his typography work and get back to TAOCP.)

Thus, all in all, Knuth had money to bring in distinguished visitors to Stanford and to have about a dozen graduate students. Many of these later contributed to desktop publishing beyond the TEX project.

While Stanford liked to license technology developed there, Knuth was "adamant that TEX stay in the public domain because [of the] big need in his under-served community"; he "refused to seek any intellectual property." Knuth believed people were being held back by not having something like TEX, and he "wanted to help people who were willing to put more into formatting to produce beautiful output."

Key references

[1]Donald E. Knuth, *Digital Typography*, CSLI Publications, Stanford, CA, 1999.

[2]Videos at `webofstories.com/play/donald.knuth/50` through `webofstories.com/play/donald.knuth/70`

[3]Nelson Beebe, 25 Years of TEX and METAFONT: Looking Back and Looking Forward, *TUGboat*, vol. 25, no. 1, pp. 7–30, `tug.org/TUGboat/tb25-1/beebe-2003keynote.pdf`

[4]Barbara Beeton and Richard Palais, Communication of Mathematics with TEX, *Visible Language*, issue 50-2, August 2016, pp. 40–51, `tug.org/pubs/vislang-16/article.pdf`

[5]Donald E. Knuth, Mathematical Typography, *Bulletin of the American Mathematical Society* (new series), vol. 1, no. 2, March 1979, pp. 337–372, `tinyurl.com/knuth-gibbs`

[6]Donald E. Knuth, *Literate Programming*, CSLI Publications, Stanford, CA, 1992.

[7]Donald E. Knuth, *Computers and Typesetting*, Addison-Wesley, 1986: Volume A, The TEXbook; Volume B, TEX: The Program; Volume C, The METAFONTbook; Volume D, METAFONT: The Program; Volume E, Computer Modern Typefaces. Volumes A and C are exceptionally detailed manuals. Volumes and B, D, and E are unique examples of computer program documentation: B and D are complete program listings intended to be read as beautiful programs and literature, and E is the complete METAFONT source code with annotations for the Computer Modern typeface.

[8]Donald E. Knuth, The Future of TEX and METAFONT, *TUGboat*, vol. 11, no. 4, p. 489, `tug.org/TUGboat/tb11-4/tb30knut.pdf`

[9]Donald E. Knuth, *TAU EPSILON CHI: A System for Technical Text*, Stanford Computer Science Department Report No. STAN-CS-78-675, September 1978.

[10]User's Guide to AMSTEX, `mirror.ctan.org/macros/amstex/doc/amsguide.pdf`

[11]Donald E. Knuth, *TEX and METAFONT: New Directions in Typesetting*, Digital Press, Bedford, MA, 1979.

[12]Donald E. Knuth, *METAFONT: A System for Alphabet Design*, Stanford University. Computer Science Department Report STAN-CS-79-762, 1979.

[13]Alexandre Gaudeul, The (LA)TEX project: A case study of open source software, *TUGboat*, vol. 24, no. 1, 2003, pp. 132–145, `tug.org/TUGboat/tb24-1/gaudeul.pdf`

[14]*TUGboat*'s 100 issues — Basic statistics and random gleanings, *TUGboat*, vol. 32, no. 1, pp. 17–22, `tug.org/TUGboat/tb32-1/tb100walden.pdf`

[15]See the introductory material of volumes C, D, and E of *Computers and Typesetting*.[7]

Biographies

Barbara Beeton is employed by the AMS and been involved with TEX since 1978. She has been a TUG board member since TUG's founding in 1980 and has been editor of *TUGboat* since 1983. **Karl Berry** has worked with TEX since 1982, is production editor of *TUGboat*, a past president of TUG, and has been a key person in a variety of aspects of the TEX and TEX community infrastructure. **David Walden** studies and writes about computing and digital typography history, including doing oral history interviews (75 from the TEX world and the others from the more general computing world).

TEX: A branch in desktop publishing evolution — Part 2

Barbara Beeton, Karl Berry, and David Walden

Abstract

Donald Knuth began the development of TEX in 1977 and had an initial version running in 1978, with the aim of typesetting mathematical documents with the highest quality, and with archival reproducibility far into the future. Its usage spread, and today the TEX system remains in use by a vibrant user community. However, the world of TEX has always been somewhat out of sync with the commercial desktop publishing systems that began to come out in the mid-1980s and now are now ubiquitous in the worlds of publishing and printing.

Keywords: TEX, METAFONT, Computer Modern typeface, desktop publishing, open-source software, user groups

Part 1 of this history was about the creation of TEX and how it began to spread beyond Stanford. Part 2 is about the impact TEX has had on the broader world, the expansion of TEX-based and TEX-related technology, and development of a worldwide community of TEX users and developers following the lead of Knuth's original collaboration model.

In Part 1 we were primarily talking about TEX as developed by Knuth. In Part 2 we sometimes speak of (LA)TEX, meaning TEX or LATEX but mostly LATEX. We also often say TEX when we mean TEX and everything that has been built on top of and around TEX; we hope the distinction between the TEX program itself and these extended meanings is clear from the context.

(To save on the page count, most of this history's large number of notes and references are posted online at `dw2.tug.org/dtptex/webnotes.pdf`) [this is a temporary location; if the paper is accepted we will find a permanent web location for the Web notes].)

Reflections on the impact of TEX

Technology and social context

At the time of TEX's initial development, phototypesetting had substantially replaced typesetting with metal type, and consequently the quality of published books and journals had decreased.* The commercial hardware technology of desktop publishing (personal computers, laser printers) was still in the future. The software used at the time for commercial typesetting (e.g., from Atex for newspapers or Science Typographers, Inc., used by the AMS) and for technical reports (e.g., Pub, nroff/troff) was not yet WYSIWYG — which was still in its initial stages of creation at the Xerox PARC laboratory. Digital type and type design (e.g., Ikarus developed by Peter Karow of the URW type foundry) was breaking new ground but not widespread. Word processing on a dedicated machine had recently become popular, but the widely popular word processing software systems (e.g., WordStar, WordPerfect, Word) were still in the future. The popularity of, for instance, the

Wang systems and then WordStar, were indicative of a building (if still latent) market demand for people other than professional typesetters to be able to do their own document formatting.

The technology and social context at Stanford was that Knuth had students who could help him; they had state-of-the-art computer and display technology (and were close to Xerox PARC with its own state-of-the art technology, and also close to the stream of Silicon Valley innovations); and Knuth could take a sabbatical year or later reduce teaching responsibilities to concentrate on computer typesetting. Knuth had deep mathematics *and* computer science understanding that were relevant to digital typesetting. He cared about typographical beauty and wanted to create a system able to do the "best" typesetting (not just "good enough" typesetting), and he also wanted to build a system that was stable and archival ("to create systems that would be independent of changes in printing technology as much as possible."[1]). He was an academic who still knew how to program complex computer software systems that worked efficiently in practical use. Knuth was uninterested in personal financial gain from his typesetting work. He believed in collaboration and that advances in computer-based typesetting had previously been held back because of competing commercial interests. He had the connections and reputation to successfully get help from typography experts and attract potential collaborators and users in (especially) the mathematics and computer science communities. Knuth's reputation allowed him to get funding for the Stanford digital typesetting activities while keeping the technology essentially in the public domain.

The rest of this section considers the impact of TEX in terms of first its strengths (and areas of influence), and then its weakness (including misconceptions). There is considerable overlap throughout, so items may fit in more than one category.

Areas of strength or influence

Breakthrough application. Knuth sought and to a considerable extent *achieved* a level of typesetting quality that was driven by researching and matching the best hot type and engraving typesetting, rather than just doing the (supposed) "best" a computer could do at the time, as was necessary for companies needing quick and reliable financial returns. TEX was an early example of doing good digital typesetting that non-TEX people knew about. That TEX had influence is clear from its fast adoption within the mathematics community and beyond. Knuth, being a perfectionist, significantly revised TEX more than once; documented it comprehensively, including publishing the source code and detailed logs of his rigorous debugging efforts; and, after a certain point, he paid small rewards for people finding new bugs in the program (what commercial company pays users for finding a bug?). Not only was TEX a breakthrough in its application area; it was also an example of a highly perfected instance of computer programming.

Math markup. TEX was originally developed by Knuth to produce a "beautiful" book with lots of mathematics in it.[2] Perhaps the most important legacy of TEX is the language used for its math input. It's the language that mathematicians became and are still used to, and it continues to be widely used today, both in TEX and in various other systems (e.g., MathJax, JavaScript for browsers, wikis, and even somewhat by Microsoft Word*).

Use in publishing. Over the decades since (LA)TEX initially came on the scene, many books and journals, especially for math and science, have been published using TEX or a TEX-based system, especially LATEX. Many publishers still use TEX although perhaps not as many as once did. Because most distributions of TEX are free, TEX can be adapted to and disproportionately used in less developed countries with their smaller markets.*

Innovative algorithms. Several of the algorithms developed for TEX have been used in other word processing and desktop publishing systems, e.g., those for line breaking, hyphenation, and the box-and-glue model.*

The line-breaking algorithm uses the dynamic-programming ("total fit") mathematical optimization method to improve the look of pages by avoiding ugly line breaks.**

The hyphenation algorithm is driven by tables of statistically more likely patterns for valid breaks within words, and the algorithms and tables do a surprisingly good job of finding good hyphenation points and typically require only small tables of exceptions. Maintaining and using the program to process language dictionaries to produce the tables has become an international collaboration that spans many languages and applications in many text processing systems.*

Knuth's boxes-and-glue model also has had impact beyond TEX. This is a surprisingly simple but powerful idea for dealing with words, paragraphs, pictures, and pages in a more or less common way, considering each type of element to be a box with glue being a stretchable thing (distance) between adjacent boxes.*

The so-called *hz* micro-typesetting method of Hermann Zapf improves line-breaking further, lessening the need for end-of-line hyphens and making the right margin of justified text look better. It has been fully implemented in pdfTEX and LuaTEX (discussed below). (The *hz* approach also exists in some form in Adobe's InDesign.)*

The combination of above four approaches (three of them from the 1970s) still appear to be, fundamentally, at the state of the art for creating good looking columns of text.

Open-source community advancing typography. TEX was created in an era when much software was freely shared, and this has remained the primary approach in the TEX world over the past 40 years. In this sense, TEX was an "open-source" project (years before the term was coined) and continues to be a major open-source project. The worldwide TEX community with its publications, and later websites, became and remains a center of digital typography R&D and technology exchange.

The TEX community's vast collection of materials related to TEX, the Comprehensive TEX Archive Network (CTAN — `ctan.org`), had at least naming impact on the equally vast sites of two freely available major programming languages — the Comprehensive Perl Archive Network (CPAN) and the Comprehensive R Archive Network (CRAN).

The evolution of the TEX community will be addressed in more detail in the next section.

Impact of students. The Stanford students of Knuth and Bigelow, beginning in the late 1970s and continuing well into the 1980s, provided a significant addition to the burgeoning desktop publishing and greater typography industry, e.g., at Adobe, in the Word group at Microsoft, Frame Technology, and others.

All in all, TEX arguably has had broader direct effect on the world than Knuth's work in analysis of algorithms (TAOCP).

Weaknesses and misconceptions

While TEX and METAFONT were significant breakthroughs in computer-based typesetting created in the late 1970s and still relevant today, TEX had and has some clear areas of weakness.

In terms of the word processing and desktop-publishing markets (which came about after TEX), TEX clearly has the disadvantages of not being WYSIWYG, not directly handling popular type formats, and being essentially unknown to the consumer markets and largely unknown to the professional typography market.

Because TEX was created long before today's popular font formats and because TEX's own font formats were deeply embedded in the TEX implementation, TEX has always been playing catchup as other font formats became popular. TEX originally had only its own font formats; when PostScript came on the scene, a post-processor to TEX (called dvips) became available to convert TEX's device-independent DVI output to PostScript, and work had to be done to connect the Type 1 fonts used in PostScript to the TEX system. After PDF was invented, eventually the TEX program was modified to produce PDF output directly (pdfTEX), with the advanced line breaking features of so-called micro-typography also added, as mentioned above.

The TrueType and OpenType formats, which have essentially replaced Type 1 fonts in the rest of the digital world, are usually still converted to Type 1 fonts for use with TEX and pdfTEX. Considerably later (ca. 2007), another modification of the TEX engine, XƎTEX, was created that reads these font formats directly, but XƎTEX lacks the full micro-typesetting capability that was implemented in pdfTEX. LuaTEX, another extended TEX engine created around the same time, can also handle TrueType and OpenType directly, and does keep the full micro-typesetting capability. (Besides supporting the modern font formats, another principal reason for the creation of XƎTEX and LuaTEX was to natively support Unicode input, notably UTF-8. LuaTEX also has great extensibility capabilities via an embedded Lua language interpreter. We will say more about LuaTEX towards the end of this history.)

METAFONT, Knuth's type-design tool companion to TEX never became particularly popular; Charles Bigelow has given a lengthy discussion of this.[3] The early fonts Knuth produced were not very good but soon got to be adequate, and were ultimately as sophisticated as any other digital type. Adobe's John Warnock also took issue with Knuth's approach to rasterizing fonts in a "Knowledge@Wharton" interview.* Knuth's Computer Modern is not especially pleasing to many people, nor has the Euler typeface developed for the AMS by Zapf and Knuth become a popular font.

Nevertheless, as TEX's default typeface, Computer Modern has been used by millions of TEX users. The Computer Modern typeface *is* appreciated for being an unusually large set of fonts and for having a relatively complete set of math symbols. Also, over the years members of the TEX community have designed and coded significant extensions to Computer Modern — font support for eastern European languages, Cyrillic, African, Vietnamese, and so on (`ctan.org/topic/font-mf`). For a time, TEX likely supported a much wider array of languages and scripts considerably earlier than any other system did, with generally excellent quality. Members of the TEX community have also worked on making a vast number of popular and interesting fonts, created outside the TEX world, available for use with TEX.

Knuth's `WEB` system for literate programming of software development and documentation also never caught on very much in the larger computer world. `WEB` remains today as the implementation source for TEX, METAFONT, and related early programs.

There are other areas that some people see as weakness for TEX but others don't see as disadvantageous, and there some areas where most people in the desktop publishing world are unaware of TEX's breadth of use.

Hard to use. That TEX is not WYSIWYG is part of the reason TEX is viewed as hard to use. Not only does TEX markup look cryptic and hard to the uninitiated, using TEX markup requires learning a separate editor (e.g., Emacs, WinEdt) or another kind of front-end (e.g., LyX, Scientific Word). However, for people who do use "TEX and friends" (e.g., LATEX, XƎTEX, LuaTEX, etc.), the TEX systems can be easier to use and more productive than WYSIWYG systems. For instance, having figures appear at sensible places within pages of text can be easier in LATEX than in Word; you can

have powerful search and editing capabilities in a separate editor; TEX naturally and automatically updates numbering and maintains cross-references when parts of documents are moved around in the drafting process; and so forth.*

When difficulty is encountered with TEX (either in trying to do something quite normal or to do something quite unusual and sophisticated), the TEX community is more easily tapped for help (e.g., from `tex.stackexchange.com`, online repositories of decades of user group journals, etc.) than one finds with some of the commercial word processing and desktop publishing systems.

Users of many TEX systems also have to deal with many fewer user interface changes, instances of new operating system releases obsoleting old versions of the text processing system, and new versions of source file formats. (Source files, at least for TEX and LATEX, are essentially totally portable among different users of each system, and what compiled once can be and has been compiled successfully again years or decades later with the now-current TEX distribution.)

The TEX systems have great capacity for being modified or extended by users to gain efficiency in specifying the markup. In particular, TEX's markup approach is also useful for being embedded in larger workflows, and (LA)TEX is seen as an adjunct to various non-TEX systems (e.g., R, SAS, WordPress) or formats (e.g., XML). On the negative side, many users who appreciate TEX's capability for modification or extension view its macro-language approach to such changes as a "disaster". However, that hasn't prevented massive numbers of augmentations to TEX from being created.

Math only. Many people believe that the TEX systems are used only for math and perhaps "math adjacent" fields (e.g., economics). However, the TEX systems are in fact used by anyone who wants the advantages sketched above (e.g., power, beauty, stability, low cost) of TEX over word processors. To list a few areas: the rest of the sciences, the humanities, linguistics, law, for business back ends, catalogs and schedules, etc. A look at such "unexpected" uses reported in *TUGboat* is stunning in its breadth.

Non-commercial. Another misconception is that TEX is not commercial. Knuth put TEX in the public domain in a way that allowed commercial companies to package it for particular markets or subsets of users, and a variety of (typically rather small) companies have sold TEX commercially over the years, starting in the mid-1980s. These days, however, TEX is so well packaged by a collaboration of the user groups that commercial versions are less and less differentiated from free versions. On the other hand, a relatively new commercial use of TEX is from the companies Overleaf* and ShareLaTeX (which announced they were merging in mid-2017) which run web servers to process LATEX files, thus relieving users (or perhaps university departments) of maintaining local TEX distributions and perhaps allowing users to create LATEX-based documents from phones and tablets that don't have TEX running on them. The Overleaf system includes a WYSIWYG front end to LATEX. (The companies claim many hundreds of thousands of users and thousands of institutions use their systems for millions of projects. It is perhaps doubtful that they have that much ongoing use beyond initial trials, but it is an indicator of a new area of TEX expansion.)

Front-ends and editors for LATEX is a place where a few small companies are making money by supplying a product supporting the TEX world.* Some fonts have been sold that specifically augment TEX systems. There are typesetting services that turn manuscripts into (perhaps ready-to-print) TEX for publishers or individuals. (Many more examples can be seen in the advertising section of *TUGboat* throughout the years.*)

Evolution of the user community

In Part 1 of this history, we sketched the development of TEX and METAFONT, starting in 1977, by Knuth, his graduate students, and others, through Knuth's definitive releases of TEX82 and METAFONT84, and his final significant addition to TEX, expansion to 8-bit characters, in 1989.

In the approximately 35 years since TEX82 and METAFONT84, TEX and its derivatives and expansions have been used by millions of users. We speculate that a reason for TEX longevity is the Knuth-TEX "business model" of keeping strong control of TEX itself (providing long term stability and consistency[4]) while completely revealing its internals for the edification of all, and providing hooks such that practically anything could be (and has been) built on top of basic TEX. This "business model" provided the foundation for a world wide and enduring community of developers, "resellers", and users.

Another unusual aspect of Knuthian TEX is that it had an endpoint. Knuth finished his software development and went back to writing TAOCP. Arguably, this has benefitted future TEX developments by being a stable reference point, as Knuth intended. Much of the software built on top of TEX is also highly stable. Unlike commercial companies that need to change things regularly to keep selling new versions, the TEX community of users and developers try to keep things working as is, with new capabilities not obsoleting prior capabilities.

There is no single reason why many developers have worked together and individually on TEX-related projects for years or decades adapting TEX to the ever changing world. Some need a new typesetting capability for themselves; some get gratification from serving the community; for some "hacking TEX" is a fascinating hobby; some are on a mission to keep a beautiful system alive. Whatever the reason, the Knuth model of openness and stability has allowed those attracted to working on TEX and the TEX infrastructure to dig in and make their contributions.

The rest of this section sketches the continuing, post-Knuth, evolution of TEX and the TEX community — the larger part of the history of the development of TEX and TEX's role in desktop publishing.

Continued expansion

By 1983–84, TEX was fully operational and running on many different computers with different operating systems. Also, LATEX had come on the scene in 1983, making TEX easier to use,* as had BIBTEX for systematic handling of bibliographies and references for documents. As with TEX, these were available for free, which surely contributed to their popularity.*

Throughout the rest of the 1980s, use of TEX continued to grow. A variety of distributions, including commercial distributions for personal computers, became available to make TEX easier to install and begin to use, and were supplying other capabilities such as alternative math fonts.

Also, by this time, the cultural move was underway of authors being willing or wanting to do their own composition rather than having secretaries type manuscripts (although some secretaries were also trained to use (LA)TEX); and various technical publishers liked receiving manuscripts in (LA)TEX and created macro packages to format a (LA)TEX document in the publisher's print style for a journal or book. TEX produced higher-quality output than many publishers' systems, and saved some effort in converting manuscripts into the publisher's style. Macro packages of thesis styles were also created at universities.

By the end of the 1980s, TUG membership was approaching 4,000 members and additional TEX user groups, primarily focused on languages other than English, began to form — for example, a

Dutch group in 1988 (De Nederlandstalige TEX Gebruikersgroep, NTG) and a German group in 1989 (Deutschsprachige Anwendervereinigung TEX e.V., DANTE). The number of local user groups continued to expand throughout the 1990s.** These groups typically formed to accomplish the tasks necessary for having good support in TEX for their languages: hyphenation patterns, special macros to make input convenient, extending the Computer Modern fonts, translating embedded English words (such as "Chapter") in the LATEX macros, and writing documentation in their own language. Eventually there would be about 25 such "local" user groups; nowadays, half a dozen or so actively carry on more typical user group activities — newsletters or journals, semi-yearly or yearly conferences, and so on. (See `tug.org/usergroups.html`.)

The user groups, taken together, supported the task of information dissemination and help requests primarily through various mailing lists and the Usenet newsgroup `comp.text.tex` (still active today). As the 1980s ended, CTAN (the Comprehensive TEX Archive Network, mentioned above) was created as a central repository of TEX-related software—packages, programs, fonts, documentation, and more. It was mirrored by dozens of sites around the world, providing reasonable access to most TEX users. CTAN remains the clearinghouse for the TEX world today.

Competition

In the second half of the 1980s, Microsoft Word (Word for Windows by 1990) and the graphically oriented desktop publishing systems had built momentum and undoubtedly began converting users from (LA)TEX. The paradigm for word processing and desktop publishing was strongly moving to WYSIWYG; and markup of plain text files, which had been in widespread use in word processing, came to be popularly viewed as hopelessly complex. Moreover, Word was beginning to become a de facto standard for submissions to book and journal publishers, preferred by many authors and publishers.

In addition to Word impacting TEX use, a variety of output formats that were not natural to TEX had perhaps even more effect, such as PDF, HTML, XML, and EPUB. While there had been a path from TEX's DVI output to PostScript since 1985–86 (Tomas Rokicki's `dvips` DVI post-processor), by 1993 PDF was also on the scene and getting from DVI to PDF required another post-processor (e.g., ps2pdf or Acrobat Distiller). In other words TEX was a further step more awkward to use compared with text processors that could go directly to PDF.

Also from 1993 to 1997, Mosaic and then Netscape had become serious rivals to email lists as a source of information, and providing free searchable information immediately; these browsers also helped make the World Wide Web widely popular. This was the beginning of the "information-wants-to-be-free" movement, the decline in interest in print publications which had long provided support to professional societies (as had *TUGboat* for TUG) and the demand for open-access journals.

The rise of WYSIWYG word processors and typesetting systems, the ubiquity of Word, and so many things being available on the Web meant that membership in the TEX user groups provided less and less tangible benefit (and the TEX "products" themselves had already been available for free). Thus, generally speaking, memberships in the various user groups began a slow decline over time, and probably the much larger group of non-user-group-member TEX users also declined.

Adapting to the digital world

Regardless of the decline in TEX user group membership (and the concerns it caused), members of the TEX community continued on with new developments. A few major examples (among hundreds

or thousands of other more or less important developments):

- The Web2C program was developed (1985) to maintain TEX's portability as the Pascal language went out of style.
- The TEX engine was extended by the Japanese company ASCII Media Works (ca. 1987) to support Japanese typesetting, first as jTEX, later pTEX and other derivatives.*
- An improved version of LATEX, LATEX2ϵ, was created (1989).
- TEX distributions oriented to specific operating systems were developed — teTEX for Unix (1994), MiKTEX for Windows (1996), gwTEX for Mac (2001).
- The PSTricks package was created (1997) that allowed specification of PostScript programs* from LATEX and provided an alternative to drawing programs from outside the TEX world (e.g., Adobe Illustrator).
- The Beamer class was added to LATEX (2003), providing another, especially powerful, slide-presentation capability to the TEX community.
- The XETEX (2004) and LuaTEX (2007) programs were created; these are extensions of TEX, directly supporting Unicode input and use of OpenType, TrueType, and related font formats.

Of particular note, in 1996 pdfTEX was released, thus bringing TEX into the contemporary digital world, and arguably saving TEX from extinction. Also in 1996, the first TEX Live software CD was produced, providing a large set of (LA)TEX macros, packages, and fonts, and in time adding other formats. These surely helped sustain TEX use and perhaps motivated some users and developers to continue to support the user groups. With ever increasing Internet bandwidth, the user group journals got on the Web, but typically didn't substantively become more graphical or interactive as allowed by non-print publication. (An example of a non-LATEX format of TEX in the TEX Live collection is ConTEXt,* an advanced TEX-based typesetting system developed by Hans Hagen for use in his own business and distributed freely to the TEX community — it is especially oriented to interfacing between TEX and XML.)

For a few years around the year 2000, the TEX Live software CDs (later DVDs) became at least as important a benefit as journals as a reason for people to join the various user groups. (For some time the CD/DVD has been known as the TEX Collection and has included MacTEX (for Mac OSX including TEX Live, proTEXt (for Windows based on MiKTEX), TEX Live (for Linux, Unix, and Windows), and a yearly snapshot of the CTAN archive). The project has always been carried out by a group of contributors, and there is a yearly poll of how many copies each user group needs to send to their members.) However, increasingly the TEX world has used the availability of the Internet to make more prior benefits of membership freely available, for example, free downloads of TEX Live (`tug.org/texlive/`) and *TUGboat* issues being open access from one year after publication.

The Internet, email, web servers, software version control systems, and other capabilities of the digital age also have benefits (while decreasing the tangible value of user group membership). They allow better coordination of development efforts across TEX user groups and individuals working on various development, infrastructure maintenance, and user support activities. CTAN, TEX Live, the `tug.org` web site itself, and many other development projects are examples of this. They also allow for different styles of membership administration, e.g., online voting for officers, paying for membership and user group store items online, and providing electronic-only subscriptions to user group publications (i.e., eliminating some printing and mailing costs).

The future

TEX has now existed for 40 years. During this period many commercial word processing and desktop publishing systems have come and gone. While the TEX systems have never been relevant in the minds of users of the well known commercial word processing and desktop publishing programs and the number of users of TEX is surely a minuscule number compared to the couple of billion users of Word, it would not be surprising if the number of TEX users (mostly individuals doing their own typesetting) can be measured on the same scale with the number of users of a commercial desktop publishing system such as InDesign (which is used by many professional typesetters). We also note that TEX (and the various derivatives of TEX) is one of two remaining widely used page layout systems of the type of RUNOFF, the earliest important such system,* where the user explicitly types markup into a plain text editor (the other is groff).

We see continuing demand for TEX for many years.[5] The TEX world has powerful free tools producing high quality typographic output. This will long be attractive to some people. For instance, the ArXiv system is helping increase popularity of TEX; many researchers from a variety of fields publish their results at `arxiv.org` (approaching 100,000 articles per year) and thus most get accustomed to using LATEX.

TEX documents are in plain text files that are compiled (in the sense a C language program is compiled). This allows TEX-based typesetting to be part of automated work flows perhaps more easily than Word or InDesign document formatting can be; and this capability will remain useful to various publishers and individuals.

Another TEX niche is among users who like to be able to open the box of products and possibly make their own contributions to a product or at least adapt it to their own needs.

Moreover, the user groups provide places (physical and virtual) to meet and exchange ideas for people who have interests in or needs for typography, font design, and a powerful typesetting system. In this sense, the TEX user groups are more like professional societies than computer product user groups. In another sense, the user groups and community of TEX developers are rather analogous to a small multi-national company, with multiple product lines and branch offices for various national markets, and worldwide user support capabilities; except this "company" is primarily made up of volunteers and there is not much possibility of top-down planning and assignment of resources.

For all of the above reasons, users of TEX and related technologies are a somewhat self-selecting community, and the user groups and community development and maintenance projects are aimed at serving this community.

We foresee the use and development of the TEX systems continuing for many years for these users for whom TEX is the "right" system and for developers who enjoy working in the TEX world. The international contributions and collaboration started by Knuth 40 years ago will continue within this self-selected TEX community.

TEX is a successful and continuing example of what is now generally known as open-source software development.[6] Supporting a large community of users are many (slowly changing over time) volunteers who put in time developing new capabilities and maintaining the existing capabilities of the TEX world. The (more or less ad hoc or informal) communication that goes on among various subsets of volunteers is critical.

The TEX systems range from exceptionally stable ((LA)TEX) to being regularly updated with significant new developments (LuaTEX). Throughout that spectrum, users are constantly extending the systems to their own more or less sophisticated needs, as has always been the practice in the TEX world. The support community and capabilities remain extensive and vibrant. And most TEX

distributions continue to be available for free.

LuaTEX is perhaps the most recent fundamentally new (and still ongoing) development in the TEX world. It has addressed several traditional complaints about TEX: as mentioned above, it (along with XETEX) natively support Unicode (UTF-8) input, and TrueType and OpenType fonts; LuaTEX also provides embedded scripting closely integrated with TEX in the Lua programming language (traditional TEX macros are still available); and many of TEX's internal variables and algorithms are available to the user to inspect, override, and modify. The LuaTEX project is a good example of a long term project, addressing perceived needs, driven by a few individuals who have dedicated much time over the years for personal reasons and for an interest in serving the community (and occasionally funded by user groups or other charitable sources of money to keep a programmer working).*

As mentioned, membership in a TEX user group no longer provides much tangible benefit. Many people, especially younger ones, are no longer interested in paper journals. Forums such as `tex.stackexchange.com` and others not associated with user groups have become the principal electronic help mechanism, largely supplanting mailing lists and `comp.text.tex`. Computing historian David Grier has said, "... the Internet has become the new user group, and that's where you go for a lot of your support."* Probably significant numbers of people are now joining the user groups or renewing memberships only to be part of the club and support the organizations and TEX more generally rather than to get particular benefits from their membership.

The longstanding TEX user groups probably have another decade or two of life until their memberships decline too far; and even when one (for instance TUG) eventually becomes financially unviable (e.g., TUG being unable to support a print journal and a paid employee), it could continue as a volunteer organization: the current volunteer labor (or replacements for it) could put out an electronic-only journal, organize meetings, operate CTAN, and release the TEX software collection. In fact, given the state of online support and open-source cooperation, the TEX community could survive without the formal user groups, while the various volunteer-staffed activities continue more or less unchanged. The real end for TEX users will come when appropriately capable people are no longer interested in providing the volunteer labor to continue TEX development and maintenance.

However, in recent decades math, TEX's original domain, has taken a new step in importance to the world at large. Math is used "for everything" these days: analytics in sports, polling in politics, evidence-based medicine, "big data" in marketing and business planning, etc. Today, 40 years after the creation of TEX and with TEX continuing to function as an outstanding way to produce documents including math, perhaps TEX has new potential outside its current self-selected community.

Appendix: How TEX is extended

The levels of TEX

As mentioned earlier, plain TEX consists of hundreds of primitive commands, implemented in the TEX program itself, and hundreds more macros. The primitive commands alone provide the fundamental input, output, and typesetting functionality but at too low a level to be convenient for a user to specify the typesetting of a document. The macros turn that base engine into a system that knows the conventions of typesetting. An important group of primitive TEX commands is those for defining macros and the mechanisms for calling the new macros.

Knuth deliberately allowed TEX to be augmented by users. There are many ways this can be

done. For instance, one can define new macros to augment plain TEX or redefine existing macros to change the operation of plain TEX. (Eplain is an example of a collection of macros* that augment plain TEX, for instance, to add cross-referencing by labels.) Knuth also put hooks into TEX to allow the system to work with other programs that TEX doesn't know about (and, largely, that came into existence many years after TEX was finished); for instance, images in various formats can pass through TEX, and TEX can read and write external files (other than the normal input and output).

LATEX, created originally by Leslie Lamport, is a set of macro definitions that reside on top of TEX's primitive commands, while reproducing many of the macro definitions available in plain TEX. LATEX is probably the most common form in which people use TEX, and thousands of macro packages have been written to reside on top of (LA)TEX or to modify the operation of (LA)TEX, for example, the `url` package that handles the unusual characters in URLs and tries to do sensible line breaking of them, and the `fancyhdr` package that supports nearly arbitrary page header and footer conventions. The previously mentioned AMS-TEX version of TEX is also implemented with macros (on top of plain TEX), as is the later created AMS-LATEX (on top of LATEX). A user can also add his or her own macros on top of everything mentioned — including making changes to the existing macros.

The TEX ecosystem includes various other levels: `tug.org/levels.html` identifies large collections of TEX-related software, free from the user groups or from a dedicated individual or from commercial companies; front-ends or editors that provide a development environment, provide sophisticated editing of TEX markup, or provide a graphical user interface; and extended TEX engines that provide new basic functionality at the primitive level. Various packages and engines also allow TEX to be connected to various high level languages instead of the user being forced to program with macro definitions and calls — although some interaction with macros is all but inescapable.

The sustained effort of Hàn Thế Thành to directly produce PDF files from TEX (rather than by conversions from DVI to PostScript to PDF)* is an example of how one project dealt with the various levels of TEX. It is also a notable example of how one person became motivated to undertake a significant project and how other people collaborated on it over time, rather analogous to the original TEX project, albeit on a much smaller scale.*

TEX macro processors

Let's turn now to some details about TEX's macro capability. The TEX macro processor is enormously powerful and flexible, in its unique way, and a comprehensively documented piece of software.[7,8,9*] Massive programs have been (and continue to be) written in this macro language. For example, as mentioned above, LATEX is implemented entirely with TEX macros.

Historically, there has been an interesting set of pressures around TEX's macro capability. Originally Donald Knuth included only enough macro capability to implement his typesetting interface. However, he was persuaded ("kicking and screaming"*) by early users to expand that macro capability. That expanded capability allowed users to construct nearly any logic they wanted on top of TEX (although often such add-on logic was awkward to code using macro-type string manipulations). On the one hand, TEX and its macro-implemented derivatives have always been very popular and there have been non-stop macro-based additions for over 30 years. On the other hand, users then and now despair at how annoying coding using macros is, moan about "why Knuth couldn't have included a real programming language within TEX", and otherwise cast aspersions on TEX's macro capability.

Knuth has made the point that he was designing a typesetting system that he didn't want to make too fancy, i.e., by including a high level language. He has also noted that when he was designing TEX he created some primitive typesetting operations and then created a set of macros for the more

complete typesetting environment he wanted. He expanded the original macro capability when fellow Stanford professor Terry Winograd wanted to do some fancier things with macros. Knuth's idea was that TEX and its macro capability provided a facility with which different people could develop their own typesetting user interfaces, and this has happened to a large extent, e.g., LATEX, ConTEXt, etc.

The TEX macro capability can do more or less anything any other powerful macro processor can do, but it cannot be used independent of TEX. TEX also has a way of defining a macro call to have a much more free-form format, and with some programming (or use of an appropriate TEX macro package) macro call arguments can be specified with attribute-name/value pairs. There are explicit commands in TEX creating local or global definitions, as well as various other definition variations, such as delayed definition and delayed execution of macro calls. TEX has a rich rather than minimal set of conditional and arithmetic capabilities (some related only to position in typesetting a page). There are also ways to pass information between macros and, more generally, to hold things to be used later during long, complicated sequences of evaluation and computation. These capabilities allow big programs to be written in the macro language; and so for debugging purposes, TEX also has a capability to trace the flow of macro definition and execution.

TEX has another unusual capability that is sometimes used with macros, although it is a capability more closely related to lexical analysis than to macro definitions and calls; this is the TEX "category code" feature. TEX turns its input sequence of characters into a list of tokens. A token is either a single character with a category code (catcode) or a control sequence. For instance (using an example from Knuth's *The TEXbook*), the input "`{\hskip␣36␣pt}`" is tokenized into

```
{ hskip 3 6 ␣ p t }
```

where the opening brace has the category code 1 (begin group), `hskip` has no catcode because it is a control sequence, `3` and `6` have catcodes of 12 (other), the space after `36` is catcode 10 (space), `p` and `t` have catcodes of 11 (letter), and the closing brace is catcode 2 (end group). (There are 16 such category codes in all.)

The next step in the TEX engine decides what to do with these different tokens, e.g., put the numbers together into a numeric value and the letters together into a unit of measurement, execute the primitive `hskip` command, and so on. The backslash has a catcode of zero indicating an escape character, thus telling TEX that the following letters (five in this case) form a control sequence; the space after the command name delimits the end of the name and doesn't become a token.

TEX has the capability of arbitrarily changing which character(s) have which catcode(s). For instance, the dollar sign (by default having catcode 3, indicating a shift to math mode) could be made a "letter" (catcode 11). This capability is routinely used in TEX program libraries to define macro names internal to a program in the library which users of the library cannot use when (normally) defining their own macro names. In short, the entire input language of TEX can be changed.

The typeface design system, METAFONT, that Knuth developed in parallel with TEX, has a more powerful macro capability in some ways, and significantly different, as people writing mathematical character definitions have vastly different needs than people typesetting documents.

One final place for macros in the TEX world: when Knuth created his literate programming system `WEB`, which uses Pascal for the code, he included a macro capability at the `WEB` level, partly to get around some of Pascal's shortcomings for systems programming; when literate programming systems targeting C were later developed, C's own macro processor could be used.

Prior literature

We see the topics of this history as being in two worlds of literature: the world of digital typesetting and desktop publishing, where TEX resides; the world of user groups and/or open source* communities, where the TEX user groups and community of users reside.

An interesting place to start studying the history of digital typography is a recent book chapter by Jacques André.[10]

Regarding the literature on the history of word processing and desktop publishing, interesting starting points are the *Annals* special issue on word processing[11] and Pamela Pfiffner's *Inside the Publishing Revolution.*[12] The other papers in these desktop publishing special issues are another good source. James Cortada places desktop publishing in the context of computing in the print media industries.[13]

The literature of computer user groups is not particularly extensive. An early famous example of a computer user group is IBM Share.[14,15] Other mainframe and minicomputer vendors also had user groups.[16] (Digital Equipment's DECUS user group meetings included presentations on LATEX and there was a style file creating the meeting proceedings using LATEX.)

The TEX user groups, which deal with TEX and the systems derived from or built on TEX, are somewhat different than the user groups of the commercial desktop publishing systems (e.g., for InDesign). Those commercial user groups are typically about how to use a system. While the TEX user groups and community provide user help as the commercial groups do, they are also "open source" development projects, expanding TEX in many ways (although the open-source term came into use more than 15 years after the first TEX user group was founded). With regard to TEX specifically, Gaudeul did an extensive study on TEX as an open source effort.* The papers by Gaudeul include many references to the academic literature of open source software, for example, see section 6.2 of Gaudeul's 2003 article in *TUGboat*.[6] In this sense, the TEX world is also a lot like the Unix/Linux community.[17]

As with many popular applications (e.g., InDesign), many books have been written about using (LA)TEX for various levels of users — from those wanting only an introductory tutorial, to those who want a comprehensive reference book, to those who want to know how TEX works on the inside. TEX is not exceptional in this domain; anyone wanting more information about TEX books and journals might start by clicking on links at `tug.org`.

Key References

[1]Donald E. Knuth, Remarks to Celebrate the Publication of *Computers & Typesetting*, *TUGboat*, vol. 7, no. 2, 1986, pp. 95–98, `tug.org/TUGboat/tb07-2/tb15knut.pdf`

[2]Barbara Beeton and Richard Palais, Communication of Mathematics with TEX, *Visible Language*, issue 50-2, August 2016, pp. 40–51, `tug.org/pubs/vislang-16/article.pdf`

[3]Yue Wang, Interview with Charles Bigelow, *TUGboat*, vol. 34, no. 2, `tug.org/TUGboat/tb34-2/tb107bigelow-wang.pdf`

[4]Donald E. Knuth, The Future of TEX and METAFONT, *TUGboat*, vol. 11, no. 4, 1990, p. 489, `tug.org/TUGboat/tb11-4/tb30knut.pdf`

[5]Jim Hefferon and Karl Berry, The TEX Family in 2009, *Notices of the AMS*, vol. 56, no. 3, March 2009, pp. 348–354, `tug.org/pubs/notices-09`

[6]A. Gaudeul extensively studied the TEX world through 2003 as an example of an open-source software community: Alexandre Gaudeul, The (LA)TEX project: A case study of open source software, *TUGboat*, vol. 24, no. 1, 2003, pp. 132-145, `tug.org/TUGboat/tb24-1/gaudeul.pdf`

[7]Donald Knuth, *The TEXbook*, Addison-Wesley, 1986, particularly chapter 20.

[8]Donald Knuth, *Computers & Typesetting, Volume B, TEX: The Program*, Addison-Wesley, 1986.

[9]Donald E. Knuth, *Digital Typography*, Center for the Study of Language and Information, Stanford University, 1999.

[10]Histoire technique des fontes numériques, chapter 5, volume 2 of *Histoire de l'Ecriture Typographique — le XXième siècle*, tome II/II, de 1950 à 2000, editorial direction by Jacques André; Charles Bigelow has written an English summary of this chapter — Review and summaries: *The History of Typographic Writing—The 20th century*, Volume 2 (ch. 1–5), *TUGboat*, vol. 38, no. 2, 2017, pp. 274–279, `tug.org/TUGboat/tb38-2/tb119bigelow.pdf`

[11]*IEEE Annals of the History of Computing*, vol. 28, no. 4, October–December, 2006 (see also `computer.org/web/computingnow/annals/extras/wordvol28n4`).

[12]Pamela Pfiffner, *Inside the Publishing Revolution: The Adobe Story*, Peachpit Press, Berkeley, CA, 2003.

[13]James W. Cortada, *The Digital Hand, Volume 2: How Computers Changed the Word of American Financial, Telecommunications, Media, and Entertainment industries*, Oxford University Press, 2006.

[14]Paul Armer, SHARE — A Eulogy to Cooperative Effort, *IEEE Annals of the History of Computing*, vol. 2, no. 2, 1980, pp. 122–129.

[15]Atsushi Akera, Voluntarism and the Fruits of Collaboration: The IBM User Group, Share, *Technology and Culture*, vol. 42, no. 4, October 2001, pp. 710–736.

[16]Herbert S. Bright, Computer User Groups, *IEEE Annals of the History of Computing*, vol. 10, no. 3, 1990, pp. 56–61.

[17]Peter Salus, *A Quarter Century of UNIX*, Addison-Wesley, 1994.

Additional notes and references are at `dw2.tug.org/dtptex/webnotes.pdf`

Acknowledgements

We appreciate the great efforts and enthusiasm over 40 years of the developers and users of TEX and its derivatives, starting with Donald Knuth. The following people answered questions over the past several years as we researched this history: Atsushi Akera, Gordon Bell, Charles Bigelow, David Brock, James Cortada, Dimitrios Filippou, David Fuchs, Burt Grad, David Grier, Hàn Thế Thành, Ferdy Hanssen, David Hemmendinger, John Hobby, Jerzy Ludwichowski, James Mason, Paul McJones, Doug McKenna, Richard Palais, Massimo Petrozzi, Norbert Preining, Lynne Price, Arthur Reutenauer, Jonathan Seybold, Christian Schenk, Petr Sojka, Guy Steele, Tommie Usdin, Boris Veytsman, Herbert Voß, Gerben Wierda, Joseph Wright, and Jeffrey Yost. We apologize if we have forgotten anyone. Finally, we thank the special issue editors for their encouragement and guidance.

Biographies

Barbara Beeton is employed by the AMS and been involved with TEX since 1978. She has been a TUG board member since TUG's founding in 1980 and has been editor of *TUGboat* since 1983. **Karl Berry** has worked with TEX since 1982, is production editor of *TUGboat*, a past president of TUG, and has been a key person in a variety of aspects of the TEX and TEX community infrastructure. **David Walden** studies and writes about computing and digital typography history, including doing oral history interviews (75 from the TEX world and the others from the more general computing world).

Web notes

In the following Web notes, the number at the beginning of a note is a page number; words or a topic from that printed text page then appear indicating the position on the page to which the note or reference relates; then comes the note or reference itself.

[***Editorial note: If the papers are accepted, we will convert the page numbers from being the numbers in the manuscripts to being the numbers of the printed pages.***]

From Part 1

1 **"has been told before"** Steve Ditlea, Rewriting the Bible in 0's and 1's, *MIT Technology Review*, September 1, 1999, `tinyurl.com/bible0s1s`

1 **"not close to the TEX world"** TEX originally meant "technical text": Donald E. Knuth, *TAU EPSILON CHI: A System for Technical Text*, Stanford Computer Science Department Report No. STAN-CS-78-675, September 1978. Later TEX became Tau Epsilon Chi with the logo TEX, as TEX is an abbreviation of $\tau\epsilon\chi\nu\eta$, Greek for both "art" and "craft".

1 **TAOCP** Knuth's TAOCP website, `cs.stanford.edu/~uno/taocp.html`

1 **Monotype** Fred Williams, The Monotype Story, spring 1984, `tinyurl.com/williams-monotype`

1 **"genesis of TEX"** The Errors of TEX, chapter 10 of Knuth's *Literate Programming* CSLI Publications, Stanford, CA, 1992, `tug.org/texlive/devsrc/Master/texmf-dist/doc/generic/knuth`

1 **"breaking paragraphs into lines"** TUG Interview Corner, interview of Michael Plass, 2009-12-20, `tug.org/interviews/plass.html`

1 **"May 13 memo sketched the system's design"** `TEXDR.AFT`, Chapter 24 of *Digital Typography* (see note above); or click on "[1,DEK]" at `saildart.org/DEK`

1 **"prototype implementationof TEX"** TUG Interview Corner, interview of Franklin Liang, 2009-12-20, `tug.org/interviews/liang.html`

1 **"Plass remembers"** Plass interview (see note above).

2 **"In August other people began to use TEX"** Email exchange with Guy Steele, June 2017.

2 **"invented for the job"** Chapter 7, How to Read a `WEB`, of Knuth's *Literate Programming*, CSLI Publications, Stanford, CA, 1992.

2 **"With `WEB`, Knuth combined capabilities"** For more on `WEB`, see, for example, a description by Knuth at `literateprogramming.com/knuthweb.pdf`; and a significant-sized program in `WEB`, also by Knuth, but not related to TEX, at `literateprogramming.com/adventure.pdf`

2 **"he considered his work done"** Donald Knuth, Remarks to Celebrate the Publication of *Computers & Typesetting*, *TUGboat*, vol. 7, no. 2, pp. 95–98, `tug.org/TUGboat/tb07-2/tb15knut.pdf`

2 **"allow arbitrary 8-bit characters"** Donald E. Knuth, The New Versions of TEX and METAFONT, *TUGboat*, vol. 10, no. 3, pp. 325-328, `tug.org/TUGboat/tb10-3/tb25knut.pdf`

3 **"review bug reports accumulated in the interim"** See "Errata" at `www-cs-faculty.stanford.edu/~knuth/abcde.html`.

3 **"most recent two TEX tuneups"** Knuth's so-called "tune-up" reports for his two most recent reviews are readily available: Donald Knuth, The TEX tuneup of 2008, *TUGboat*, vol. 29, no. 1, pp. 233–238, `tug.org/TUGboat/tb29-2/tb92knut.pdf`; Donald Knuth, The TEX tuneup of 2014, *TUGboat*, vol. 35, no. 1, pp. 5–8, `tug.org/TUGboat/tb35-1/tb109knut.pdf`. They too are worth reading as an example of the care and careful explication Knuth puts into merely fixing a rare bug.

3 **"will be permanent 'features'"** METAFONT has its own sequence of version numbers — increasingly precise approximations of e.

3 **close brace** One use of braces in TEX is to indicate scope in the programming language sense.

4 **3:16 book** Donald Knuth, *3:16 Bible Texts Illuminated*, A-R Editions, Madison, WI, 1990.

4 **Fantasia Apocalyptica** `www-cs-faculty.stanford.edu/~knuth/fant.html`

5 "Liang's thesis research was completed" Franklin Mark Liang, Word Hy-phen-a-tion by Com-put-er, Stanford University PhD thesis, August 1983, `tug.org/docs/liang/`

5 "TEX's automatic line-breaking algorithm" This algorithm was described in a classic paper by Plass and Knuth: Donald E. Knuth and Michael F. Plass, Breaking Paragraphs into Lines. The paper was originally published in 1981 in *Software — Practice and Experience*. It is reprinted in Knuth's *Digital Typography*, pp. 67–155. It also was part of Plass's 1981 PhD thesis: Michale F. Plass, Optimal Pagination Techniques for Automatic Typesetting Systems, Stanford University PhD thesis, June 1981, `tug.org/docs/plass/plass-thesis.pdf`

6 "reported to the TEX community" `tug.org/TUGboat/Contents/listauthor.html#Fuchs,David`

6 "part of his 1985 thesis" John Douglas Hobby, Digitized Brush Trajectories, Stanford dissertation, August, 1985, `ect.bell-labs.com/who/hobby/thesis.pdf`

6 "Hobby mostly designed the algorithms and Knuth wrote all the code" TUG Interview Corner, Interview of John Hobby, `tug.org/interviews/hobby.html`

6 Zambala's Pascal implementation `tug.org/TUGboat/Contents/listauthor.html#Zabala,Ignacio`

6 "people at Stanford who helped Knuth" To the above list of students who helped Knuth, one can also add faculty member emeritus Arthur Samuel who also found ways to support Knuth in his TEX efforts. For instance, see his introduction to TEX: `i.stanford.edu/pub/cstr/reports/cs/tr/83/985/CS-TR-83-985.pdf`; also check *TUGboat* for writings by Samuel, `tug.org/TUGboat/Contents/listauthor.html#Samuel,Arthur`

7 .SAI files Click on "[TEX,DEK]" at `saildart.org/DEK`

7 Steele port to MIT In addition to our 2017 email exchange (see note above), Guy Steele provided us with copies of his SAIL emails from July 28 to September 8, 1978, and his MIT emails from September 2 to October 28.

7 Ports to PDP-10s Other ports of TEX to PDP-10s are noted in Nelson H. F. Beebe, The design of TEX and METAFONT: A retrospective, presented at the Practical TEX conference of 2005, `tug.org/tugboat/tb26-1/beebe.pdf`

7 "TEXDVI, being optional replacement for" TUG Interview Corner, Interview of David Fuchs, `tug.org/interviews/fuchs.html`

7 "could be written using DVI output" David Fuchs, The Format of TEX's DVI Files, *TUGboat*, vol. 1, no. 1, pp. 17–19, `tug.org/TUGboat/tb01-1/tb01fuchs.pdf`

7 "regardless of form of floating point arithmetic" Nelson H. F. Beebe, Extending TEX and METAFONT with floating-point arithmetic, *TUGboat*, vol. 28, no. 3, 2003, pp. 319–328, `tug.org/TUGboat/tb28-3/tb90beebe.pdf`

7 "based on user experience with TEX" METAFONT went straight from SAIL to `WEB`.

8 "Rather than talk about" Richard Palais email, 2017-11-30.

8 "sketched his work with TEX" `webofstories.com/play/donald.knuth/61`

8 Joy of TEX Michael Spivak, *The Joy of TEX*, American Mathematical Society, Providence, RI, 1986.

8 Michael Spivak In addition to his AMSTEX work, Spivak early on also designed the MathTime professional fonts, based on and for use with the Times font or to replace the Computer Modern math fonts; these were made available via the PCTEX company.

8 "Gordon Bell remembers" Email of 2017-08-03.

8 "AMS commissioned Hermann Zapt" Zapf mentions this in his own life story: `linotype.com/1494/the-lifestory-of-hermann-zapf.html`

8 " goal of being more like how mathematicians handwrite" *Digital Typography* (see note above), chapter 17 — a reprint of a paper co-authored by Knuth and Zapf.

9 "first meeting of the TEX Users Group" Palais interview, `tug.org/interviews/palais.html`; Beeton interview, `tug.org/interviews/beeton.html`; Fuchs interview, `tug.org/interviews/fuchs.html`

9 "first issue (October 1980) of *TUGboat*" `tug.org/TUGboat/Contents/contents1-1.html`

9 "getting TEX running on many different computers" See the categories Output Devices, Site Reports, and "small" TeX at `tug.org/TUGboat/Contents/listkeyword.html`

9 **"developments in the TEX world would be permanently documented"** *TUGboat* was originally subtitled as *The TEX Users Group Newsletter*; as of 1988 its subtitle became *The Communications of the TEX Users Group*.

10 **"*TUGboat* has served the typical role"** In parallel with *TUGboat*, TUG published 13 issues of the *TEX and TUGboat News* from 1991–1995 and 20 issues of *The PracTEX Journal* between 2005 and 2012. The news function of the former was merged into *TUGboat*; the latter's goal was to publish only practical articles, where *TUGboat* has a spectrum of articles.

11 **"began to produce machine drawn letters"** Pages 64–72 of Donald E. Knuth, *Companion to the Papers of Donald Knuth*, Center for the Study of Language and Information, Stanford, CA, 2012.

12 **"fonts were improved again"** As described in this and the next subsection, Knuth redid his font design software and improved his set of fonts several times: the first version of the fonts was called Almost Modern; a second, never released, version was called Better Modern.

12 **"wider audience and to encourage support"** Jonathan Seybold email of 2017-09-03.

12 **"thought it deserved wider attention"** Barbara Beeton was also at Stanford in March 1980 (the visit timed so she could also attend the Seybold-organized seminar) "to learn how to program METAFONT and to create a prototype Cyrillic font for use in *Math Reviews*." The Seybold-organized meeting was also where she first met Bigelow.

12 **"thought it deserved wider attention"** Bigelow notes (email of 2017-08-31), "The commercial typesetting systems guys [at the seminar] all said that TEX was too complicated and slow to be commercially acceptable. Of course, they are all gone now."

12 **"informal email text from Bigelow"** Emails of June 6 and August 31, 2017. A parallel version of the story is in his interview in *TUGboat*, `tug.org/TUGboat/tb34-2/tb107bigelow-wang.pdf`

13 **"a notable example being grammatology"** Which was based on an older notion of the study of letters, from the book *A Study of Writing: An introduction to the study of grammatology* by I.J. Gelb.

13 **"'Concepts of Text' course"** The syllabus for that course is available.

13 **"METAFONT for lunch bunch"** Before Bigelow got to Stanford, Knuth had a TEX-for-lunch-bunch.

13 **Bernshteĭn polynomials** `en.wikipedia.org/wiki/Bernstein_polynomial`

13 **Lynn Ruggles** In 1983, Lynn Ruggles, a graduate student at Stanford, compiled a catalog of different approaches to digital type tools. ("Letterform Design Systems" by Lynn Ruggles, Stanford Technical Report No. STAN-CS-83-971), `i.stanford.edu/pub/cstr/reports/cs/tr/83/971/CS-TR-83-971.pdf`

14 **Southall also influenced Knuth** `tug.org/TUGboat/tb36-2/tb113southall.pdf`, `i.stanford.edu/pub/cstr/reports/cs/tr/85/1074/CS-TR-85-1074.pdf`

14 **"first academic conference"** `visiblelanguagejournal.com/issue/73`

14 **"first academic conference"** At the time, Bigelow was leading the committee on letterform education and research of ATypI (Association Typographique Internationale).

14 **"number of interesting reports came out"** Pijush K. Ghosh on Indian scripts, `i.stanford.edu/pub/cstr/reports/cs/tr/83/965/CS-TR-83-965.pdf`; John Hobby and Gu Guoan on a Chinese Meta-Font, `i.stanford.edu/pub/cstr/reports/cs/tr/83/974/CS-TR-83-974.pdf`; A formal approach to letter form design by Ghosh and Bigelow `i.stanford.edu/pub/cstr/reports/cs/tr/83/966/CS-TR-83-966.pdf`; Letterform Design Systems by Lynn Ruggles (cited above); *The Euler Project at Stanford*, a monograph by David R. Siegel, 1985.

14 **"commission to design the Euler typeface"** Stanford library guide to the Euler project archive, `pdf.oac.cdlib.org/pdf/stanford/uarc/sc0362.pdf`

14 **Table 1** While this was the end of work on Euler by Knuth and the Stanford group, in 2008 Hermann Zapf collaborated with Hans Hagen, Taco Hoekwater, and Volker RW Schaa on a "reshaping" of many of the letterforms (`tug.org/TUGboat/tb29-2/tb92hagen-euler.pdf`). For this update, Metafont was abandoned and the character outlines were manipulated directly in a font editor. This is the version of Euler currently distributed by AMS and included in TEX distributions.

14 **"Knuth remembers that the digital typography program"** A summary of the TEX project, video #70[2]

14 **"the METAFONT class which Knuth"** Donald E. Knuth, A Course in METAFONT Programming, *TUGboat*, vol. 5, no. 2, pp. 105–118, `tug.org/TUGboat/tb05-2/tb10knut.pdf`

14 **"May 2017 desktop publishing pioneers meeting"** [***Editorial note: We need to replace the following with the URL of the transcript of the video when the transcript is available.***]Last DTP meeting video, minutes 26 to 38.

16 **System Development Foundation** `oac.cdlib.org/findaid/ark:/13030/tf429003m4/`

From Part 2

1 **"TEX's initial development"** Donald E. Knuth, *Digital Typography*, CSLI Publications, Stanford, CA, 1999.

2 **"in various other systems"** `tinyurl.com/ms-blog-use-of-tex`

2 **"disproportionately used in less developed countries"** Glyn Moody, *Rebel Code: The Inside Story of Linux and the Open Source Revolutions*, Perseus Publishing, 2001, p 317.

2 **"algorithms developed for TEX"** TEX has so much computer science in it that Victor Eijkhout built a university computer science course around TEX: Victor Eijkhout, *The Computer Science of TEX and LATEX*, based on CS 594, fall 2004, University of Tennessee, `pages.tacc.utexas.edu/~eijkhout/Articles/TeXLaTeXcourse.pdf`; no doubt other word processors and desktop publishing systems also had lots of embedded computer science — lexing, parsing, semantic interpretation, optimization of searches, etc. — but their source files may not have been so thoroughly documented or may not be available for study.

3 **"line-break algorithm uses"** Donald E. Knuth and Michael F. Plass, Breaking Paragraphs into Lines, reprinted in Knuth's *Digital Typography*, CSLI Publications, Stanford, CA, pp. 67–155; `TEXDR.AFT`, Chapter 24 of *Digital Typography*.

3 **dynamic-programming** TEX's line-breaking algorithm is routinely used as an example in computer science algorithms courses in explaining dynamic programming.

3 **international collaboration on hyphenation** See `hyphenation.org`; also Mojca Miklavec and Arthur Reutenauer, Hyphenation in TEX and elsewhere, past and future, *TUGboat*, vol. 37, no. 2, 2016, pp. 209–213, `tug.org/TUGboat/tb37-2/tb116miklavec.pdf`

3 **"Knuth's boxes-and-glue model"** Nelson Beebe, Using boxes and glue in TEX and LATEX, `math.utah.edu/~beebe/reports/2009/boxes.pdf`

3 **"*hz* micro-typesetting method"** We know the details of Hàn Thế Thành's pdfTEX development, but not of the development in InDesign. Both implementations got help from Zapf or his work: `wikipedia.org/wiki/Hermann_Zapf`; Hàn Thế Thành email of 2017-09-10; Hàn Thế Thành, An Experience from a Digitization Project, `cahiers.gutenberg.eu.org/cg-bin/article/CG_1998___28-29_197_0.pdf`; Hàn Thế Thành, Micro-typographic extensions to the TEX typesetting system, dissertation, Masaryk University Brno, Faculty of Informatics, October 2000, reprinted in *TUGboat*, vol. 21, no. 4, December 2000, pp. 317-434, `tug.org/TUGboat/tb21-4/tb69thanh.pdf`

4 **John Warnock interview** Knowledge@Wharton, Adobe Co-founder, John Warnock on Competitive Advantages of Aesthetics and the "Right" Technology, January 20, 2010, `tinyurl.com/wharton-warnock`

4 **"easier to use and more productive than WYSIWYG systems"** This two-part paper, including the figures and the Webnotes, was composed and revised with LATEX before being converted to Word for submission to the journal's prepress process.

5 **Overleaf and ShareLaTeX** `overleaf.com`

5 **"a few small companies are making money"** The Wikipedia article on "comparison of TEX editors" (accessed in November 201) lists 49 editors for (LA)TEX, with eight of them requiring payment. Most of the 49 editors work at the source code level; five claim a mix of source-code and WYSIWYG editing; two claim to be WYSIWYG; four claim What-You-See-Is-What-You-Mean editing. (LyX, for instance, lets one edit graphically with menu commands to declare what lines of text are, e.g., title line, in-line equation, etc.; and the program then turns the text into a LATEX document out of view of the user.)

5 **"advertising section of *TUGboat*"** `tug.org/TUGboat/Contents/listkeyword.html#CatTAGAdvertisements`

6 **"LATEX had come on the scene in 1983"** One brief overview of LATEX is Open Source Documentation Software: An Overview, Shubhashree Savant and Sonal Sarnaik, International Conference on Advances in

Information Technology and Management ICAIM, 2016, research.ijcaonline.org/icaim2016/number1/icaim201635.pdf

6 "surely contributed to their popularity" In *TUGboat* LaTeX creator Leslie Lamport said, "I don't think TeX and LaTeX would have become popular had they not been free. Indeed, I think most users would have been happier with Scribe. Had Scribe been free and had it continued to be supported, I suspect it would have won out over TeX. On the other hand, I think it would have been supplanted more quickly by Word than TeX has been."; *TUGboat*, vol. 22, no. 1/2, 2001, pp. 20–22, tug.org/TUGboat/tb22-1-2/tb70lamp.pdf

6 "local user groups continued to expand" Eric Frambach, TeX user groups worldwide — what's cooking?, *MAPS*, Autumn 2003, pp. 6–9, ntg.nl/maps/29/03.pdf

6 "local user groups continued to expand" Christina A. L. Thiele, The Future of TeX and TUG, *TUGboat*, vol. 14, no. 3, 1993, pp. 162–166, tug.org/TUGboat/tb14-3/tb40thiele-future.pdf

8 jTeX and pTeX ctan.org/pkg/ptex, https://ctan.org/pkg/uptex; also Haruhiko Okumura, pTeX and Japanese Typesetting, *The Asian Journal of TeX*, vol. 2, no. 1, April 2008, pp. 43–51, ajt.ktug.org/2008/0201okumura.pdf

8 "specification of PostScript programs" While PostScript is often thought of in terms of text fonts, it can specify any sort of drawing.

8 "in the TeX Live collection" ConTeXt is also frequently distributed independently of the TeX Live collection.

8 RUNOFF J. H. Saltzer, TYPSET and RUNOFF, Memorandum editor and type-out commands, MIT Project MAC, MAC-M-193, November 6, 1964, web.mit.edu/Saltzer/www/publications/CC-244.html

10 LuaTeX project luatex.org

10 "David Grier has said" Page 11 in the transcript of the Computer History Museum "PC Software Workshop: Marketing and Sales," recorded May 6, 2004, CHM reference number X4621.2008.

10 "Eplain is an example" Originally created by Karl Berry, tug.org/eplain

11 "sustained effort of Hàn Thế Thành" Interview of Hàn Thế Thành, tug.org/interviews/thanh.html

11 "notable example of how" Hàn Thế Thành, The PDFTeX Program, *Cahiers GUTenberg*, no. 28–29, 1998, pp. 197–210, cahiers.gutenberg.eu.org/cg-bin/article/CG_1998___28-29_197_0.pdf

11 "comprehensively documented piece of software" Victor Eijkhout, *TEX by Topic*, Addison-Wesley, 1991, particularly chapters 11–14, ctan.org/pkg/texbytopic

11 "kicking and screaming" Peter Seibel, *Coders at Work*, Apress, 2009, p. 597.

12 "world of users groups" en.wikipedia.org/wiki/History_of_free_and_open-source_software

13 "Gaudeul did an extensive study pn TeX" Alexandre Gaudeul, The (LA)TeX project: A case study in open-source software, working paper, September 17, 2004 (quite different text than in the 2003 *TUGboat* publication); Alexandre Gaudeul, Competition between open-source and proprietary software: the (LA)TeX case study, working paper, January 6, 2005; Alex Gaudeul, Do Open Source Developers Respond to Competition? The (LA)TeX Case Study, March 27, 2006 (a different paper than the next paper with the same name); Alexia Gaudeul, Do Open Source Developers Respond to Competition? The (LA)TeX Case Study, working paper, March 2007, perhaps a preprint of a paper of the same name in *Review of Network Economics*, vol. 6, no. 2, June 2007, pp. 239–263.