Amsterdam, 13 March 1996//Knuth meets NTG members*

Abstract

On January 6th 1996, Kees van der Laan informed the NTG that Donald Knuth would be in Holland in March. Knuth was invited by the *Mathematisch Centrum* (MC, nowadays called *Centrum voor Wiskunde en Informatica*, CWI) because of CWI's 50th anniversary. Both Knuth and Mandelbrot were invited as speakers at the celebration.

The NTG noticed that this was an exceptional occasion to organize a special meeting with Knuth for all Dutch TEX and METAFONT users who would like to meet the Grand Wizard himself.

Fortunately Knuth accepted the NTG invitation and so a meeting was organized in 'De Rode Hoed' in Amsterdam on March 13th. About 35 people from all over the country and even from Belgium joined to meet Knuth.

Everything was recorded on both video and audio tape by Gerard van Nes. Christina Thiele volunteered to write this transcript.

Erik Frambach: Welcome, everyone. This is a very special meeting on the occasion that Mr. Donald Knuth is in Holland. The NTG thought it would be a good idea to take the opportunity and ask him if he would be willing to answer our questions about TEX, METAFONT, and anything else connected to the things we do with TEX. Luckily, he has agreed. So we are very happy to welcome Mr. Donald Knuth here — thank you for coming.

Tonight we have time to ask him any questions that we have long been waiting to pose to him. [laughter] I'm sure that all of you have many, many questions that you would like the Grand Wizard's opinion about. So, we could start now with questions.

Donald E. Knuth: I get to ask questions too! [laughter] Last Saturday I was in Prague and the Czechoslovak TEX users had a session something like this and you'll be glad to know that I saw quite a few copies of *4TEX* CD-ROMs at that meeting.

It's not my first time in Amsterdam. I was in Amsterdam in 1961, so it's only been 35 years, and probably less than 35 years till the next time. I guess they're tape recording these questions-and-answers to try to keep me honest, because they also did that

* This transcript was first published in *MAPS* **16**, 96.1, pages 33–44, and appears here by permission; *MAPS* is the journal of NTG, the Dutch language oriented TEX Users Group.

in Prague. So in case the same question comes up, you'll have to take the average of the two answers. [laughter]

Wietse Dol: Did you know that Barbara Beeton does that? She mails you and says "Tape everything."

Knuth: Yes, that's what they said in Prague too! [laughter] I think she's desperate for things to do, or maybe she just has a lot of questions. But before I open questions, let me say that one of the most interesting questions asked me in Prague was after the session. And I wish it would get into [the record]. The question was: how did I meet Duane Bibby, who did the illustrations for *The TEXbook* and *The METAFONTbook*? I always wanted people to know about that somehow.

Here's the story. I had the idea that after writing math books for many years, I wanted to have a book that had more weird — well, anyway, different — illustrations in it. Here I was writing a book about books — books have illustrations, so why shouldn't I have illustrations too. So, I wrote to an artist called Edward Gorey. Does anyone know ...

Frans Goddijn: Yup. *Amphigorey*. Beautiful.

Knuth: Yes, Edward Gorey. *Amphigorey*. He makes very morbid drawings but with a wonderful sense of humor. I had used several of his books with my children. I thought he would be a natural person. I wrote him two letters but he never responded. Then I wrote to a Japanese artist called Anno, Matsumasa Anno, who is really the logical successor to Escher. [...] Anno does what Escher did but in color, so I asked him if he would do it. He sent me back a nice letter, saying "I'm sorry I don't have time because I have so many other commitments, but here are five of my books full of pictures and if you want to use any of those, go ahead." I wanted personalized pictures.

Then I went to a party at Stanford where there was a lady who worked for a publisher. She'd just met a brilliant young artist who she'd just worked with. I invited him to come to my house, and we spent some time together and he's a wonderful person. Duane lives now up in northern California, about 4 hours' drive from my house, so I only went up there once to see him. He sometimes comes down to the San Francisco area on business. First we discussed the book and then he sent me a bunch of drawings and all kinds of sketches that he had. Originally, TEX was going to be a Roman, and he drew this man in a toga with olive branches in his head — which is why the lion has the olive branches now. But all of a sudden he started doing sketches of his cat, which really seemed to click, and pretty soon he had a draft of all 35 or whatever drawings, using a lion. Most of those eventually become the drawings in the book, and we adjusted half a dozen of the others. When I went to visit up at his house, I got to meet TEX the cat. He looks very much like the one you see in the book. So that's the story about Duane Bibby.

Erik: Thank you. Who would like to start with the first question? Please identify yourself when you ask one.

Piet van Oostrum: My name is Piet van Oostrum. You have this wonderful lion on *The TEXbook* the lionness on *The METAFONTbook*. What about baby lions?

Knuth: Oh, I see ... [laughter] Duane still does illustrations for special occasions. He's made illustrations for the Japanese translation of both *The TEXbook* and *The METAFONTbook*. He has TEX and META both dressed up in Japanese costumes. So now, if there happens to be some kind of an offspring that would come out of some other user, then, I imagine he would glad to help do it. But it would probably be a little bit of an illegitimate child, from my point of view. [laughter] I mean, I wouldn't take responsibility for anything those characters do. [laughter]

Piet: So what are your ideas about the offspring of TEX and METAFONT?

Knuth: Well, I think that no matter what system you have, there will be a way to improve it. If somebody wants to take the time to do a good, careful job with it, then as we learn more about typesetting, it will happen that something else will come along. I personally hope that I won't have to take time to learn a new system, because I have enough for my own needs. But I certainly never intended that my system would be the only tool that anybody would ever need for typesetting. I tried to make it as general as I could with a reasonably small program, and with what we knew and understood about typesetting at the time. So these other projects — I don't consider that they're a threat to me or anything. I hope that there will be some compatibility so that — I mean, I'd like to be immortal — so that the books I've written now could still be typeset 50 years from now without having to go through the files and edit stuff. I like the archival and machine-independent aspects of TEX especially, and I tried to set a model, a minimum standard of excellence for other people to follow.

Hans Hagen: But when you look in the future, ... you consider today's programming by a lot of people as an art, well a lot of art takes hundreds of years to be recognized as art. In about a hundred years there will be pretty different computers, the programming languages will be changed, the media on which we put all those things will be changed. Real programs and everything related to them, will they ever have a chance to become immortal, as you see it?

Knuth: Did you state your name? [laughter]

Hans Hagen: I'm Hans Hagen.

Knuth: You're saying that it's pretty arrogant of us to assume that what we do now will last at all. Technology is changing so fast that we have absolutely no idea what people are going to think of next. One hundred years ago, physicists were saying there was nothing more to do in physics, except to get another decimal — a fifth decimal place for the fundamental constants — and then that would wrap up physics. So, there is no way to know about these things. But I do believe that once we have things in electronic form and we have mirror sites of them, there is a fair degree of immortality — whereas paper burns.

Do you know anything about this project called 'The Clock', being developed by Stewart Brand and his colleagues? He's the one who published the *Whole Earth Catalogue*. They have a bunch of people that are considering if they could build something that would last for a thousand years ... I don't want to go on too much more about that. I do hope that the stability of TEX will make it possible to reproduce the things we're doing now, later. And since it's fairly easy to do that, I think it will happen — unless there's a nuclear holocaust. Some mathematicians have this debate about the Platonic view ... does everything in mathematics exist and we're just discovering it, or are we actually creating mathematics? In some sense, once something gets put into bits, it's mathematics and therefore it exists forever, even if the human race dies out — it's there, but so what?

Erik: Who's next?

Marc van Leeuwen: If I could extend a bit on the previous questions. The stability of TEX itself, I could imagine, might be a stumbling block for development of new things exactly *because* it's so stable and everybody's already using it. So if something comes along that is just a bit better, then people will not tend to use that because it's not available everywhere, and there are all kinds of reasons to keep on using the old thing.

Knuth: I guess I said in Florida that people are still trying to use the old fonts that I'm still trying to stamp out from the world. Four years ago I redesigned the Greek lowercase delta and I made the arrowheads darker. I didn't change anything in the way TEX operates — all the dimensions and the characters' heights and widths stay exactly the same. But I did tune up a lot of the characters. Still I see lots of math journals are still using the old ones from four years ago, and I get letters and preprints from people with the old-style delta. I changed it because I just couldn't stand the old versions. [laughter] Now I've got home pages — if I ever have some errata to TEX or something I put them there: `http://www-cs-faculty.stanford.edu/~knuth`. This gets to my home page, and there's a reference saying, 'Important notice for all users of TEX', and that page says 'Look at the lowercase delta and if you have the wrong one, you die!' [laughter]

I understand that people have a reluctance to change from something that they've become accustomed to. I know of two main successors to TEX: one is ε-TEX and the other is NTS. ε-TEX is going to be apparently 100% compatible with TEX, so if somebody doesn't switch over to incompatible features, then they have a system that still works with old things. That will allow a gradual change-over. It'll take more space on a computer, of course, but that's not a big deal these days. The people who work on ε-TEX always sent me very reliable comments about TEX when they caught errors in my stuff, so I imagine they're going to be doing a careful job. So it'll be one of these things where you walk into a random installation of UNIX or whatever and you'll find ε-TeX there as the default, and you'll still have TEX. Then you also have certain other features that might be really important to you for your special applications.

Johannes Braams: You mentioned ε-TEX and NTS. But are you also aware of the Omega project?

Knuth: Oh, the Omega project? Yes, I'm hoping to use that myself for the authors' names in my *The Art of Computer Programming*. I've been collecting the names of Chinese, Japanese, Indian, Hebrew, Greek, Russian, Arabic authors and I want to typeset their names properly [laughter], not just in transliteration. I have some rudimentary software that will do this for proofing purposes, for getting my database going and for writing to people and saying, 'Is this your name?' With the Omega system, I'm hoping that it'll be accompanied by good fonts that will make it possible for me to do

this without a whole pile of work. Right now, to get the Arabic names, I have to use ArabTEX, to get the Hebrew names ... I had a terrible time trying to get Hebrew fonts on CTAN two weeks ago — I can tell you that whole story if you want to know ... I kept clicking on the different things and they would refer to files that didn't exist and README files that were four years out of date and inconsistent, so I couldn't find any Hebrew fonts. Maybe you have it on your CD ...

Johannes: I could certainly point you to someone who could help you with the Hebrew font — I know someone in Israel who's trying to do Hebrew support within the Babel system. And they do do typesetting in Israel with TEX.

Knuth: My own typesetting friend in Israel is Dan Berry, who unfortunately is fairly committed to troff. [laughter] I'm sure that I can get good Hebrew through Yannis and Omega. I sure hope UNICODE is going to arrive sooner rather than later; it's much better than the alternatives for much the reasons that Marc [van Leeuwen] mentioned. I haven't found a great enthusiasm in Japan for UNICODE, because they have a system that seems to work pretty well for them, so why change. Everytime I ask a Japanese for his name in UNICODE, he'll say, 'what's UNICODE? Here's my JIS name'. But the JIS characters don't include the Chinese codes, and in fact, my own name — I have a Chinese name — and my name in JIS isn't quite the same. There are two different unicode characters, one for the Japanese version and one for the Chinese.

In the back? Kees?

Kees van der Laan: I have a lot of questions of course. But I would like to start with some questions about METAFONT. The first one is: how come macro writing in TEX and METAFONT is so different?

Knuth: Why are macros in TEX and METAFONT so different? I didn't dare make TEX as extreme as METAFONT. These languages are of completely different design. METAFONT is in some ways an incredible programming language — it's object-oriented macros. You have macros in the middle of record structures.

The way I designed these languages is fairly simple to describe. Let's take TEX. I wrote down one night what I thought would be a good source file for *The Art of Computer Programming*. I took a look at Vol. 2, which I had to typeset. I started out on the first page, and when I got to any copy that looked very much like something I had already done I skipped that. Finally I had examples of all the different kinds of typesetting conventions that occur in Vol. 2. It totalled 5 printed pages — and you can even see these pages — exactly what my original test program was — in a paper by David Fuchs and myself, where we talked about optimum font caching.[1] In there, we gave an example and we show these 5 pages, which would illustrate what I wanted TEX to be able to do. I wrote out what I thought I would like to type — how my electronic file should look. And then, I said, OK, that's my input, and here's my output — how do I get from input to output? And for this, well, it looks like I need macros. [laughter]

Same thing for METAFONT. I went through my first draft of all the fonts that later became Computer Modern. I wrote actually in sail, an Algol-like compiler language, but SAIL had a macro ability, so I developed a few primitive macros in which I could say, 'pick up the pen', 'draw from point 1 to point 2', and things like that. These macros were compiled by the SAIL compiler into machine language, which would then draw the letters. I went though the entire alphabet, and by the end of the year, I had some 300 little programs, each one drawing a letter. Then I realized what kind of a language I would want to write in, to describe the letters. So one day, on a family camping trip — I was in the Grand Canyon with my wife and kids — I took an hour off, sat under a tree and wrote out the program for the letter A, in a language that I thought would be a good algebraic language, reflecting at a high level what I had been doing with pretty primitive low-level instructions in my SAIL programs. I did the letter B, too. Capital A and B, and then went back to the camping trip. These sheets of paper where I have my original programs are now in Stanford's archive — the program for the letter B was published in a Stanford library publication called *Imprint* last year. The woman who's in charge of rare books and manuscript collections at Stanford is quite interested in METAFONT so she wrote a little article about what they have.

That program again implied that I wanted some macros to go with it. But these needed to be much more structured than the macros of TEX. It had to be that when I said, 'z 1 prime', this would actually be equivalent to '(x 1 prime, y 1 prime)' and I wanted to be able to write, 'z 1 prime' without any delimiters. It turned out that in order to have a high-level language that would feel natural to

[1] *ACM Transactions on Programming Languages and Systems* **7** (1985), 74.

me writing the program, it had to look completely different from TEX. So TEX and METAFONT share a common format for error messages and certain other data structures inside, but otherwise, they're quite different systems because, in order to have a good high-level language, I don't want to have to waste time writing parentheses, brackets, commas, and other delimiters.

Kees: It's a nice introduction to my second question: [laughter] For the future of MetaPost, which allows mark-up of pictures, with `.eps` as the result, what is your attitude to 2.5d for MetaPost and METAFONT? For example, adding a triple as an analogy of the paired data structure?

Knuth: MetaPost already has a data structure for triples because of color. So RGB are actually triples of numbers.

Kees: Yes, but the triple as a data point in space?

Knuth: Ah, I see. I did write METAFONT in a way that has hooks in it so that it can be easily extended; [for example], if you want to draw 3-dimensional pictures, for perspective and projective geometry instead of affine geometry. The program itself for METAFONT was written so that it could easily be changed by people who wanted to have a system that goes beyond the basics. I always wanted the systems that I would make widely available would be be able to handle 99% of all applications that I knew. But I always felt there were going to be special applications where the easiest thing would be to change the program, and not write a macro.

I tried to make the programs so that they would have logical structure and it would be easy to throw in new features. This hasn't happened anywhere near as often as I thought because people were more interested, I think, in inter-changeability of what they do; once you have your own program, then other people don't have it. Still, if I were a large publisher, and I were to get special projects — some encyclopaedia, some new edition of the Bible, things like that — I would certainly think that the right thing to do would be to hire a good programmer and make a special computer system just for this project. At least, that was my idea about the way people would do it. It seems that hasn't happened very much, although in Brno I met a student who is well along on producing Acrobat format directly in TEX, by changing the code. And the Omega system that you mentioned, that's 150,000 lines of change files. [laughter] I built in hooks so that every time TEX outputs a page, it could come to a *whatsit* node and a *whatsit* node could be something that was completely different in each version of TEX. So, when the program sees a *whatsit* node, it calls a special routine saying, 'how do I typeset this *whatsit* node?' It'll look at the sub-type and the sub-type might be another sub-type put in as a demo or it might be a brand-new sub-type.

Similar hooks are in the METAFONT program. If people have extra time when they're not browsing the Web [laughter], I recommend as a great recreation to read the program for METAFONT. Some parts of it are pretty rough going and I hope that nobody ever finds a bug there because I'd hate to have to look at them. [laughter] But those are the rasterization routines, the things that actually fill in the pixels. There are many other things in that program — the linear equation solver that it has and the data structure abilities ... lots of beautiful algorithms are in there — to take square roots in fixed point, and the intersection of two curves, and so on. METAFONT is full of little programs that were great fun to write and that I think are useful and interesting in their own right. I think when John Hobby wrote MetaPost, he enjoyed it, because he could add his own nice little programs to the ones that are already there.

I'm a big fan of MetaPost for technical illustrations. I don't know anything that's near as good, so I'm doing all the illustrations of *The Art of Computer Programming* in MetaPost. Also, the technical papers I've written are going to be published in a series of eight volumes by Cambridge University Press, and all the illustrations, except the photographs, are going to be MetaPosted. The first volume of these eight was the book *Literate Programming*; the second volume is going to come out this summer and is going to be called *Selected Papers in Computer Science*. It reprints a dozen or 15 papers that I wrote for general audiences, not for specialists in computer science, but in *Scientific American* or *Science* magazine and things like that. The third volume will be about digital typography, and it'll reprint all my articles in *TUGboat* and things about TEX. What do you think, by the way — should I publish in that third volume the memo that I wrote to myself the first night, when I designed TEX? I put it in a computer file and it's in the archives, but I've never shown it to anyone. [round of "of course!" and "sure" and laughter from the audience] Maybe it'd sell more books [more laughter].

Frans Goddijn: You need to put it on your home page and we can then decide —

Knuth: No, no. That way we'd never sell the books. [laughter] Not that I'm a mercenary type

of person, of course. It's in a file called TEX — well 'teks', actually. I have to admit I pronounced it 'teks' for a month or two — I was thinking of 'technical texts', though. `tex.one` was the name of the file and it would make interesting reading probably, someday.

And your name is?

Jan Kardan: In this company I will probably ask a very heretic question, but a little heresy makes a lot of fun — talking about METAFONT. There are probably many type foundries now [that] crank out lots of good-quality fonts and kerning tables. It's not clear whether PostScript or True Type will survive. Do you think that METAFONT will survive text fonts? Not talking about the math fonts.

Knuth: I don't think the extra capabilities of METAFONT have proved to be necessary for good-quality type fonts, although I think that you can still make better-quality type fonts with it. Designers find it difficult to think as a computer person does, in the sense that when people in the computer business automate something, trying to make the computer do something, it's natural for us to have parameters and say that we're going to try to solve more than one problem. We try to solve a whole variety of problems based on the parameters that people set. But it's much easier if people gave us only a single problem with a single parameter, then we could have the computers do exactly the prescribed thing. Computer scientists have become accustomed to thinking of how we would change behavior as conditions change, but designers aren't at all accustomed to this. They are much happier if the boss says one month, "Give me a roman font," and the next month, "Give me a bold font." It's much more difficult to say, "Show me how you would draw something no matter how heavy I want the letters to be." METAFONT provides a way to solve that problem and to draw characters with parameters, but it's a rare designer who's comfortable with that notion. They can do multiple master fonts by making multiple drawings and then matching up points between the drawings and having the computer interpolate. The multiple master fonts in PostScript allow up to four parameters, and almost all of them have only one or two parameters. The most I know of is two; probably others have gone all the way to four. But then they have to provide drawings for all the extreme points of these parameters.

In spite of this limited use of parameters, what's available commercially is quite beautiful, as far as readability is concerned, although it doesn't really provide the quality that you guys had in the Netherlands in the 17th century. What's the man's name, the great punch cutter at Enschedé — he made 4.5, 5pt up to 16pt, and each letter was designed for its size, and fonts had a nice uniform appearance. This wouldn't have happened at all with the Type1 fonts. There were two guys who did most of the punch cutting for Enschedé and others in the 18th century: One of them, Fleischman, was a genius for really beautiful letters; the other, Rosart, was just good at making lots and lots of letters. [laughter][2] [...] They were fun. Rosart would make all kinds of highly decorated alphabets and things like that. I have a big coffee-table book that gives examples of all the fonts from Enschedé, which was translated into English by Matthew Carter's father. Anyway, in this book, *Typefoundries in the Netherlands*, you can look at these typefaces and weep.[3]

Still, on a laser printer, we get pretty good fonts now, and therefore it looks like there won't be that many professional type designers using METAFONT. Pandora was a good design by a genuine graphic artist. METAFONT has turned out to be wonderful for making ordered designs and special-purpose things for geometry. There's now this really neat system in Poland where they have TEX and METAFONT in a closed loop — TEX outputs something and then METAFONT draws a character and if that doesn't fit, TEX says, 'go back and try it again'. Jackowski and Ryćko understand TEX and METAFONT, and the programs are well documented and can do these things. So METAFONT isn't going to disappear for that reason; but it's never going to be taught in high school.

Frans Goddijn: My name is Frans Goddijn and I have one question with some sub-questions [laughter]; I'd like to ask the sub-questions first. What I'm wondering — and this may have been asked often before — is whether you would consider, in retrospect, what you have created [to be] an art or a tool? And the reason I ask is — when I hear you speak with so much passion for type fonts and the beautiful algorithms that you put into METAFONT that you would like to point people to and the recognition that you get from people who understand that — but, there is a vast majority of users who just got

[2] Johann Michael Fleischman, 1701–1768; Jacques-François Rosart, 1714–1777.

[3] *Typefoundries in the Netherlands from the Fifteenth to the Nineteenth Centuries*, by Charles Enschedé, translated by Harry Carter (Haarlem: Stichting Museum Enschedé, 1978), 477 pp. This magnificent book was composed by hand and printed by letterpress to commemorate the 275th anniversary of Joh. Enschedé en Zonen.

TEX from some server, never realized who created it, and use it to typeset not always very pretty documents. [laughter] They do that in a very crude way and don't care less. You froze TEX at a certain point, allowing other people to build around it. I was wondering how such a thing would feel to a father — are you father of a piece of art that other people use as a tool, or is it a child that you have frozen in its development, that will never grow up ... there are so many questions ... if you just go back to the art vs. tool idea, and your feelings about that.

Knuth: Obviously, if I write something that has a lot of power to do many different things, it'll be possible to make it do awful things. I just came from the *Rijksmuseum*, where they have an exhibit called "The Age of Ugliness". It was a whole bunch of fancy silver bowls from the late 19th century .... When you say an art, I'm not sure I understand exactly what you mean. To me, art is used in two quite different senses, most often nowadays in the sense of fine art, while art, originally, *Kunst*, was anything that was not natural — so we have the word *artificial*, something that is made by people instead of by nature. The Greek word is *techne*. [laughter] But then you refer to a tool as something that is maybe just a device that is the fastest way to get from here to there but maybe you don't care about elegance ... But what I think you mean when you talk about art is the aesthetics — something about beauty and something with a little bit of love in it. With TEX, my idea was to make it possible to produce works that you are proud of; I assumed that people can enjoy actually spending a little extra time making the results better. I didn't expect that the whole world would be doing this. [laughter]

Incidentally, I can't understand the mentality of a person who writes graffiti on a beautiful building although I can see why drawing is fun. Why would you want to scrawl something — some kind of animal instinct of territory might account for it, I suppose, but it's really impossible for me to conceive of such actions.

When it comes to matters of aesthetics, you can't dictate taste. You can't say that your idea of beauty is going to match anyone else's idea of beauty. But I did want to have a tool where we could reach the highest levels of beauty according to our own tastes. I didn't allow people to have letterspacing very easily, but I tried to make everything else easy. [laughter] .... Of course, I originally designed TEX just for myself, for *The Art of Computer Programming*; I thought my secretary and I were going to be the only users. And it wasn't until later that I was convinced that I should make it more general and so on. But I did want a tool for myself by which I could produce books that would make me feel good after spending almost all my life writing those books.

I started writing *The Art of Computer Programming* when I was 24 years old and I still have 20 years of work to do on it. That's a lot of time. I don't want to write those books if they're going to come out looking awful. I wanted a way to make it possible [to produce good-looking books]. Originally, when computers started out, they knew only numbers, digits. The 19th-century computers could print tables. Then we had computers that could do numbers and letters, but only on a teletype machine; so you had some capital letters and a 32-character set. But then, after I graduated from college, we got ... let me see, I was probably ten years out of college before we could do lowercase letters on a computer. You know, the PASCAL language, when it came out, it used all uppercase letters — there was never any consideration that there would be more than 64 characters in a computer's repertoire. Finally, we were beginning to see in the middle 70s that computers could actually do lowercase letters, and produce something that looked a little bit readable, a little bit like books. Wow! [laughter]

Then there was this development of typographic software starting at MIT in 1960 and going through 4 or 5 generations, leading to troff and EQN, where there was even mathematics being typeset. In 1977 I therefore knew an existence theorem: It was possible to typeset something that looked almost like good mathematics. EQN was being used in physics journals and experience showed that secretaries could learn how to do it. So I thought, "Why not go all the way to the end, to convergence?" What I wanted to do with TEX was not to be a little refinement over troff and the other things, but I was saying now, "Let me try to go to the best typography that's ever been achieved by mankind" Except for the illuminated gold leaf type of lettering, I wanted to at least — when it came to black and white printing — I wanted to match the best conventions that had been achieved. Computer typesetting had gone through this lengthy development, getting a little better and a little better. It was time to say, "Well, let's jump to the end now." Of course, I didn't think this would be an activity that everybody would want to do. But there were enough people that would care about trying to get as much quality as possible, that they could be — well, that's why I finally made TEX more available. The American Math Society were the first people, nearly the first people who

convinced me that I should make the system do more than I originally intended.

Andries Lenstra: Why didn't you start from troff? It was completely inappropriate?

Knuth: Yes, yes. You see, troff was patched on top of ... I mean, there was a whole system, it was a fifth generation, each of which was a patch on another one. So it was time to scrap it and start all over again: "Here's what the language should be, so let's design some good data structures for it." Not "Let's try to be compatible." I had the advantage that I was not at Bell Labs, where I wouldn't be hurting anybody's feelings by saying, "Let's throw it all away." [laughter] It was impossible for the people at Bell Labs to do such a thing — it wouldn't be nice. But it occurred to me that now that we had proof that this goal was possible, I should start over, and rethink how I could get from input to output, so the program could be much more unified, much smaller, and would also work. I mean, troff was collapsing all the time. A lot of the earliest users of TEX had been frustrated by troff breaking over and over again, so it had gotten unwieldy. But it had also proved that there was light at the end of the tunnel.

I also had to scrap TEX, you know, and start over again; after five years, I decided that it would be best to go back and re-do the program. But it would have been very hard to do that if my friend in the next office had done it. [laughter] So, I just have this philosophy that there will be always some people who are more interested in quality than others, and I wanted to make TEX good for them. I don't see any good way to make it impossible to make a bad document, unless you have only a system with a small menu of options; that's good for a large class of users, to make a system that's so simple that you can't possible do anything ugly in it.

Erik: I think it's time for a coffee break now — we'll take five or ten minutes.

Knuth: Johannes, you had a question that you had to ask, so let's get that over with. [laughter]

Johannes Braams: It's about typesetting. What is your opinion about the skyline model of typesetting? In TEX, you talk about boxes: each letter is inside a box, and we glue boxes together to a line, and the line itself is inside a box, and each line is viewed as a box and the boxes are fitted together to form a paragraph. The skyline model tries to go a little bit further than the rigid box and line, and tries to take into account that some of the descenders in the upper line and the high parts in the lower line don't overlap, so that you could actually have lines much tighter together — especially in math typesetting, that could be an advantage.

Knuth: Hmmm, I guess you're talking about general principles of computer graphics where you have rectangles inside a picture, instead of having the rectangles grouped only inside of a rectangle. ... This certainly would be a major change in all the data structures of TEX. You could go to a quad tree structure or something like that. All the things that people use to solve hidden-line problems and do rendering, to find out what's in front of something else, and all the algorithms they use to make movies like *Toy Story*. It would be most valuable, I imagine, for catching unusual cases in math formulas.

I have two feelings about these things. One is that I like to see people extending the things that computers can do automatically. People learn a lot when they try to do this. The whole field of artificial intelligence has been one of the areas that has had greatest spin-offs to computer science because they've tried to solve very hard problems. Especially in the early days, they came up with methods that turned out to be useful in many other parts of computer science. So, it's my feeling that when people are working on more ambitious goals, they develop powerful techniques that often have very relevant spin-offs. Even so, after they've solved that problem, they're going to think of something else which will be another refinement and so on — they'll never have a situation where they're going to automatically create the most beautiful document. There's going to be a time when you can look at the output and see that you can still improve it. Designers of the most automatic systems would be well advised to at least still leave a chance for somebody to move something up and down and fake out their automatic algorithm.

The philosophy that I had when I did TEX was that I would try to have a system that did 99% of everything automatically, and then I would look at what remained and I would kludge the rest. But kludging it is one way to say it; another way of saying it is "Tidy up the rest," or "Dot the i's and cross the t's." My feeling is that this non-automatic part gives me a little extra pride that I have put the spit and polish on the final product, that I know I did it. If it occurs a lot, then it's a nuisance and I'm wasting time. But if I can really limit this to 1% — if I've spent 30 hours writing a paper and it takes me only another 15 minutes to clean up, then I'm

happy to do another 15 minutes at the end. It's a small little extra that gives me a chance to celebrate the fact that I've finished the paper.

The spacing that TEX does worst right now, in my experience, is with respect to square root signs being a little too tight, with the operand either too close to the radical sign or too close to the bar line or both; I find that I'm most often fiddling with that. I've adopted in the book *Concrete Math* and also in *The Art of Computer Programming* now, the convention where in the math formula I put an @-sign where I want one math unit of extra space. The @-sign is then defined to have a math code of hexadecimal 8000, which means that this will invoke in math mode and the @-sign will be regarded as a macro that adds one math unit of space. So I'll type 'square root of' '@-sign' 'log of n' [laughter], because otherwise the space before 'el' is a little bit too tight. Now maybe even this skyline model wouldn't know that 'el' was too tight, maybe it would. But it's cases like that ...

The most common case really is where I have something like 'x squared over 3', where you have a simple superscript and then a slash, and then the denominator. There's almost always too much space before the slash. And this is true, I find, in all the books that I used to think were typeset perfectly by hand [laughter], but now I'm sensitive to this. Now I go through, typically with emacs, and look for all occurrences of something with a one-character exponent followed by a slash, and most of those look better with a negative thinspace before the slash. It would be nicer if I didn't have to do that. But still, it's a small thing for me.

Would the skyline model help me much? Sometimes I run into cases where I'll add another word to the answer to an exercise in order to avoid a clash between lines. Here, the lines are actually not getting spread apart too far, but they're so close together that the 'subscript k less-than-or-equal to n' will clash with a left parenthesis in the next line. And I don't want the type to be quite so close together there. Now, if I had been smarter, I would have designed my $\leq$-sign to have a diagonal stroke under the $<$ instead of a horizontal bar, and I wouldn't have had those clashes — too late for that now. [laughter]

Kees?

Kees: May I ask you a question about your attitude to mark-up in general? And let me illustrate it by first telling a story. When we started with using TEX etc., we mean actually we start with LATEX — I mean, that is the effect in Holland. And then I looked at the products of the mark-up and I did not like it. And then I was wondering, what is your attitude to that? I'm sorry to say so, I paged through *The TEXbook* file `texbook.tex` and I looked at all the things in there and then I thought, "Well, I have some idea of what your ideas are of mark-up." And when you explained about METAFONT and all those things not in there, which you have implicit — am I wrong if I summarize this, that you adhere to something like minimal mark-up?

Knuth: Yes. For example, when I am reading Edsger Dijkstra's books, every time I get to a section where it says 'End of Comment', it strikes me as redundant. And I always think, "Oh, yes, this is Edsger's style." When I wrote a paper for his 60th birthday, I said at the end, "Acknowledgment, I want to thank Edsger for such-and-such," and 'End of Acknowledgment'. [laughter][4] But that's the only time in my life I'll ever do that. Maybe I'm an illogical person, but apparently half the people using `html` now type only the `p` at the beginning of a paragraph, and the other half type only a `/p` at the end of a paragraph. [laughter] Hardly anybody uses both, according to what my spies tell me. And I don't know what the heck these systems actually do with the unbracketed material. When I write `html`, I'm scrupulous with my mark-up. If you look at my home pages — I'll pay you $2.56 if you find any case where I started something and didn't close it with the right tag. I tried to be very careful in that, and to indent everything very well, and so on. But I found it a terrible nuisance, because it's not the way I think.

I think a high-level language, to me, is something that should reflect its structure in some visual way but not necessarily explicitly; so that, when I know the conventions, we can suppress some things. Parentheses are one such convention and mathematics got a lot better when people invented other notations like operator precedence that we can see structure without spelling it out in too much detail. A mathematician spends a lot of time choosing notations for things, and one of the things we try to avoid in mathematics is double subscripts. I read one French PhD thesis where the author had five levels of subscripts [laughter] — he kept painting himself into a trap. He started out with a set x_1 through x_n, so then when he talked of a subset, it had to be x sub-i_1 through x sub-i_k, and then he wanted to talk of a subset of this, so then he had a theorem that says, let a sub-b sub-c ... and so on. [laughter]

[4] *Beauty is Our Business* (Springer, 1990), 242.

I try to choose notations that give me the economy of thought at a high level.

That's why I probably didn't believe in a great deal of mark-up in *The TEXbook*; I would begin typewriter type and end typewriter type for sections by saing `\begintt` and `\endtt`. I would also delimit the lines and when I'm presenting parts of the plain TEX macros, `\beginlines` and `\endlines` — those macros are in the file, since it's very important to me to see how that works. But in other cases, I've left [things] as simple as possible, for me to see visually the beginning and end of stuff. It's something also like problem solving — sometimes, if I've solved a problem and I'm not worried about it anymore, I forget to tell anybody else the solution. I was always a very bad committee chairman because I'm not very good at finishing that last ending line, I guess. Still, with `html`, the document was short and I decided that my home pages were going to be used by many different kinds of browsing software so I had better be very rigorous.

While I was developing TEX, I attended one of the meetings of the committee that designed SGML and had a very good discussion with Charlie Goldfarb and the other people on the committee — we only had that one meeting near Stanford. Certainly I appreciate the fact that this structure makes it possible to build other kinds of programs around what you have. The more structure you have in a document, the easier it is to make a database that includes things about it, and knows what's going on. I never objected to it; I just always felt that in order to maximize my efficiency, I didn't want to mess around with full mark-up unless I had to.

X: SGML allows minimizations; that's why the end-paragraph is not necessary. So that's one of the reasons why it's so difficult sometimes. You have a formalization to minimize.

Knuth: But LATEX doesn't allow it.

Johannes: But we do have some books, however, permitting omitted end-tags in LATEX3, but that's not far enough along.

Knuth: Well, talk to him. [laughter] I don't need a special editor for `html` — people are hyping fancy things where you can click on a tool and it'll put in the start and end tag together — but when I wrote my files, I did make up a simple emacs macro that would take whatever tag I just typed and create the end-tag. All it had to do was search back till it found a less-than sign and then copy that string twice and put a slash in front of it, so I used that all the time — it was easy.

Johannes: Quite different type of question now, from someone who'd like to ask here: literally, he writes, "Why is the height of the minus sign in the cm symbol font the same as the height of the cmr plus sign?"

Knuth: Ah. A lot of people are wondering about that one. Where you have 'a minus c' or you say 'x sub minus' or something, why is it that the height and depth are greater than the actual shape of the minus sign?. In fact, it's not just the plus and minus, it's also the $+$, $-$, $\pm$, *MetaPost*, $\oplus$, $\ominus$, $\otimes$, $\oslash$, $\times$ and $\div$ — if you look at the code for these, there is a **beginarithchar** macro that begins all of the arithmetic characters in the font, guaranteeing that they will have the same size.

Johannes: But it doesn't say why.

Knuth: That's right — it doesn't say why. And the reason is that early on, I wanted certain things to line up the same. For example, if you had

$$\sqrt{x+y}+\sqrt{x-y},$$

I wanted the square root signs to be place in the same way. Otherwise, you would get

$$\sqrt{x+y}+\sqrt{x-y}.$$

And so there are many other cases where you have formulas where there's a plus sign in one part of a formula and a minus sign in the other part, and for consistency of spacing, it ought to look symmetrical. There are other cases, I readily admit, where you have only a minus sign — you never have a similar thing with a plus sign, and you wonder why there's extra space left there. So I say `\smash minus` [laughter] in those cases.

Johannes: The particular application, why this question was asked — Michael Downes from the AMS —

Knuth: Yes, Michael Downes, he has more experience than any of us in this room; he's the chief typesetter of most of the mathematics in the world.

Johannes: He has a problem properly attaching a superscript on top of the `\rightarrowfill` ...

Knuth: The `\rightarrowfill`? OK ... The `\rightarrowfill` is this thing that makes a right arrow of any desired length, and then he wants to put a superscript on this. What's the macro for building that up? I haven't used that page in a long ... [laughter][5] The `\rightarrowfill` is made up of minus signs and so probably if I had known Michael ... known about that in the old days, I would have changed the plain TEX macros so that

[5] Knuth was trying to remember `\buildrel`; see *The TEXbook*, p. 437.

it would not use the height of the minus sign in the `\rightarrowfill` operator [...][6] Anyway, I've now told you the reason why it's there for the other ones.

Johannes: Another question, which is about multiple languages. There's a problem when you have one paragraph where you have different languages.

Knuth: Yes, the `\lccode` changes. This is the ...

Johannes: And I've been told that inside one paragraph you can only use one hyphenation table, which is the one which is active at the end of the paragraph. So, switching hyphenation tables inside paragraphs. Suppose, for example, you have a paragraph with English text, with a German quote inside it, the German quote being several lines long.

Knuth: I know that TEX will properly keep track of which hyphenation table to use. The glitch, the mistake, that I didn't anticipate is if the two languages have different `\lccode` mappings — so that each has a different idea of which characters are lowercase. When you hyphenate, you need to hyphenate an uppercase word the same as an lowercase word, so TEX uses the `\lccode` of a character to convert every letter into the lowercase code of that letter. I didn't anticipate that people might, for different languages, have a different mapping from uppercase to lowercase. And so it's that mapping that, at the end of a paragraph, applies to all the languages in the paragraph. But otherwise, TEX is careful to keep track of what language you have.

And by the way, there's a file called `tex82.bug`. Go to the CTAN archives, and find subdirectory `/knuth`, and under that `/errata`, and that's where this is. At the end of `tex82.bug` this particular error about `\lccode` is mentioned as being something that's an oversight that's too late to fix.

Marc van Leeuwen: Why is it too late to fix? It would conflict with other things?

Knuth: Yes. So that people are already using these things in lots of documents, and it's very hard to change. In fact, I don't see any way to fix it. [laughter] I would say that when you are faced with a situation where you're doing multiple languages with multiple `\lccode`s, this is a good reason to write your own version of TEX.

Andries Lenstra: Could I ask a question? Happily enough, I'm not the first person to mention LATEX, so I may mention it now. There's a situation that often arises when people try to write a PhD thesis where they want to change LATEX code because they think they know better about things of beauty or typography, and unhappily enough they are not experts on LATEX, so they don't succeed or they succeed badly. In general, people who know about typography can't write beautiful LATEX code or other forms of code, and vice versa — people who know how to write these forms of code, are no experts on typography. What do you think of the endeavors in the past to bring the two worlds together, for instance, as Victor Eijkhout has tried to do with his `lollipop` format, a machine to create other formats. I would have thought that it would have had a big success but the opposite seems to be the truth. What do you think of it?

Knuth: I'm not familiar with the details of `lollipop`. I suppose that was based on a famous quotation from Alan Perlis, who said that, "If somebody tells you that he wants a programming language which will only do the right thing, give him a lollipop."

Andries: Yes.

Knuth: I'm sure that the lollipop effort was instructive and worthwhile, but I don't know the details so I can't answer in great detail on this. Probably the type designers didn't find the language easy to learn. I do think that we're having much more communication now, as every month goes by, between the people that know about type and the people that know about macros. It's just a matter of time as we wait for these waves to continue moving — we're nowhere near a convergent stage, where TEX has reached its natural boundary and the type designers have reached their natural boundary. They're still moving toward each other. I don't think it's like a hyperbolic geometry, where they never will get together.

The main difficulty of course is that TEX is free, and so a lot of people will say, "Well, how could it be any good, if you're not charging money for it?" A lot of the people in the type design community would only work in things where there's money behind it; money proves to them that it's worth talking to people. So it just takes a little while till they see some good examples, which will make them more open for these discussions. And that's happening all the time in different countries.

In the Czech Republic I was quite delighted to learn that the new encyclopaedia in Czech, which is the first one for many years, is being done with TEX. And not only that, it's being done with a very high budget. They made this decision because they tried all the other systems and were disgusted with

[6] In fact, the `\leftarrowfill` and `\rightarrowfill` macros now omit the height and depth of the minus, in `plain.tex` version 3.14159 (March 1995).

them. They had good results with TEX. Many other commercial publishers are using it too because they talk to their friends at the big publishing houses. This will, I think, be solved with time. And products like `lollipop` are very worthwhile in the meanwhile to facilitate this. It takes time to bring different communities together. I think the financial factor is definitive for a lot of people.

Piet van Oostrum: I don't know if you have ever looked into the LATEX code inside, but if you look into that, you get the impression that TEX is not the most appropriate programming language to design such a large system. Did you ever think of TEX being used to program such large systems and if not, would you think of giving it a better programming language?

Knuth: In some sense I put in many of the programming features kicking and screaming, and I'll try to explain the background. I know how Leslie went about writing LATEX
Dash first he would write the algorithms out in a high-level programming language, with **while**'s and **if-then**'s and so on, and then he would pretty much mechanically convert this to TEX macros. If I had suspected that such a style was going to be the most common use of TEX, I probably would have worried a lot in those days. Now, computers are so fast that I don't worry so much about the running time, because it still seems to go zip!

In the 70s, I had a negative reaction to systems that tried to be all things to all people. Every system I looked at had its own universal Turing machine built into it somehow, and everybody's was a little different from everybody else's. So I thought, "Well, I'm not going to design a programming language; I wanted to have just a typesetting language." Little by little, I needed more features and so the programming constructs grew. Guy Steele began lobbying for more capabilities early on, and I put many such things into the second version of TEX, TEX82, because of his urging. That made it possible to calculate prime numbers as well as do complicated things with page layout and figure placements. But the reason I didn't introduce programming features at first was because, as a programmer, I was tired of having to learn ten different almost-the-same programming languages for every system I looked at; I was going to try to avoid that. Later, I realised that it was sort of inevitable, but I tried to keep it as close to the paradigm of TEX as a character-by-character macro language as I could. As I said before, I was expecting that the really special applications would be done by changing things in the machine language code. But people didn't do that, they wanted to put low-level things in at a higher level.

Piet: What do you think, for example, of something like building in a programming language which is, from a software engineering point of view, easier to use?

Knuth: It would be nice if there were a well-understood standard for an interpretive programming language inside of an arbitrary application. Take regular expressions — I define UNIX as "30 definitions of regular expressions living under one roof." [laughter] Every part of UNIX has a slightly different regular expression. Now, if there were a universal simple interpretive language that was common to other systems, naturally I would have latched onto that right away.

Piet: The Free Software Foundation is trying to do that and Sun is trying to do it and Microsoft is trying to ...

Knuth: The Free Software Foundation is trying actually to include also the solutions of Sun and Microsoft. In other words, to make all of the conventions work simultaneously as much as possible. And that conflicts with my own style, where I've tried to have unity rather than diversity ... I didn't go for ten ways to do one thing. C++ is similar — when the committee would say, "Well, we could do it this way or this way," they did both. I hadn't gone that route in my system, because it is messy. But I admit that the messy way is the best that can presently be realized in practice.

Marc van Leeuwen: I have a question about literate programming. I know you must be very fond of it, if I understand your interviews —

Knuth: Yes, I'm so fond of it that I could ... well ... OK. [laughter] You know, I'm really so fond of literate programming, it's one of the greatest joys of my life, just doing it.

Marc: My question was that obviously it's not nearly as popular as TEX is, and, what's more, there isn't much coherence in the world of literate programming. There are a dozen different systems being used — some people favor this, some people favor that — and this worries me a bit. I too am very fond of this style of programming, but I would like to see it being used much more.

Knuth: Literate programming is so much better than any other style of programming it's hard to imagine why the world doesn't convert to it. I think that Jon Bentley put his finger on the reason and it was something like this: There aren't that many

people in the world who are good programmers and there aren't that many people in the world who are good writers, and here we are expecting them to be both. That overstates the case but it touches the key point. I think that everyone who's looked at literate programming agrees that it's a really good way to go, but they aren't convinced that ordinary students can do it. Some experiments at Texas are proving otherwise, and I've had a smaller-scale experience at Stanford. It's a hypertext way of programming and I imagine that with better hypertext systems that we're seeing now and people becoming so familiar with the Web, we're going to get a variety of new incompatible systems that will support literate programming. Hopefully somebody with time and talent, and taste, will put together a system of literate programming that is so charming it will captivate a lot of people. I believe that the potential is there, and it's just waiting for the right person to make that happen.

Marc: I think one problem might be that if you compare your programs with the average program that people write, there just aren't nearly as many interesting algorithms in the average program, so literate programming doesn't add too much to a program which is very dull by itself.

Knuth: Well, thank you for your comment. But maybe sometimes I make a non-interesting algorithm interesting just by putting in a joke here or there. I've taken programs that I got from Sun Microsystems, for example, and as an exercise, spent the afternoon converting them to a literate form. There weren't any exciting algorithms in there, but still, you could look at the final program and it was better — it had better error diagnostics, better organisation, it corrected a few bugs. I don't have time to go over to Sun and show them this, and say, "Why don't you rewrite your operating system?" [laughter] but I know that it would be much better. So all I ever published was the very simple rewrite of the wc word count routine in UNIX, which is not at all an exciting algorithm, but as a demo of how it could be done.[7]

My approach to literate programming isn't the only one, of course, and in the recent book by a group at Princeton, *A Retargetable C Compiler*, Chris Fraser and Dave Hanson used a variety of literate programming to describe their C compiler. Other books are coming out now that are using some flavors of literate programming. I was talking to someone at Microsoft who said that he thought literate programming was on the rise, and I said, "Does that mean the next version of Windows is going to be all done in literate programming?" "Well, no, not exactly" ... [laughter] The people who've experienced literate programming will never go back, and they'll probably gan influence gradually. The companies that use it are going to sell more products than their competitors, so pretty soon this will happen. I imagine that there are about ten thousand users of literate programming and a million users of TEX, so it's a factor of a hundred.

Marc: Do you think it still has to develop? I get the impression that with so many tools around, that it's not yet mature. The idea is mature, but the implementation still has to ...

Knuth: Yeah, that's true. There's great need for programming environments based on this idea. It's not at all easy to create these environments and to have the power to promote them and maybe the support to do it in a way that wouldn't make it too expensive or too hard for people to install. The most ideal thing would be if the Free Software Foundation were to adopt it, or something like that, or some of the people they work with. Actually, [Richard] Stallman [of the FSF] designed a variant of literate programming that he uses, and he has it well integrated with TEX, in his own style. He hasn't put it in too many of his programs, but he has a version. It's one of these things that needs, as you say, to mature.

Marc: Do you believe it should go in the direction of integrated systems, where you really have all the facilities you need in one system? Because I think the tendency is more towards very minimalistic systems that do not do any pretty printing because that gets you into too much trouble when you're switching programming languages. So it really boils down to something which is very flexible but not very convenient for someone to use.

Knuth: One programming language is good enough for me so I'm not the right person to ask. But then, I guess, for the *The Art of Computer Programming*, for the next twenty years, I'm pretty sure that `CWEB` is going to be as good as anything I need. I'll write programs for Mathematica and I'll write some programs for MetaPost; I could develop or use literate programming for those programs, but I don't think I will. I don't write so many lines that I would gain a great deal ... although I would get a better program afterwards. Unless somebody already presents me with a good system for it, I won't go ahead with `MathWeb` or `MPWeb`. But with

[7] D.E. Knuth, *Literate Programming* (Stanford, 1991), 341–348, based on a prototype by Klaus Guntermann and Joachim Schrod, *TUGboat* **7** (1986), 135–137.

CWEB, I'm going to write an average of five programs a week for the foreseeable future, and there, my productivity is infinitely faster when I do it with literate programming.

One other thought flashed in my mind as I was talking just now ... I wrote a paper last year, I think it was, about mini-indexes for literate programs[8] and here I was trying to show what sort of programming environment would help me. In the listings for TEX the program and METAFONT, and also for the Stanford Graphbase, on the righthand page of each two-page spread, there's an index to all the identifiers used on that page and where they were declared. My paper explains the system I used to get those indexes, and this kind of functionality would also be needed in any hypertext system. These minimalistic systems are attractive primarily because a good programmer can write them in a couple of days, understand them and use them, and get a lot of mileage out of them. Once somebody writes a good hypertext system for literate programming, I think that'll attract a lot of people—a system that doesn't crash, and has a familiar user interface because it's like other hypertext systems that we're already using. The time for that will be ripe in about two years.

Erik: It's half past nine now and I think we'll have to stop here. I want to thank our special guest, Donald Knuth, for the time here. I think we've all learned a lot now. We're very happy that you were able to be here. Thank you very much.

Knuth: I really appreciate all the work you did to get this special room here on rather short notice. [applause]

Erik: Also, thank you to Elsevier Science, who helped, in the person of Simon Pepping; and your English colleague, Sebastian Rahtz, who is not here, although I expected him. But he paid for the coffee and tea, so thanks. There's of course a little present that we have for you. I hope you like it! [He presents a book about Dutch art called '*De Stijl*'.]

Knuth: Yes, ... the Dutch type designer, Gerard Unger, came to Stanford for three weeks, he and his wife Marjan, and they talked about things like this to our type designers. They also related fashion of clothes and furniture and architecture to type styles as well. This is great. This was done with TEX?

Erik: I don't think so ... as we are in Holland now ... [laughter] [He also presents a pair of wooden tulips]

Knuth: A nice gift for my wife.

Erik: And of course a copy of the *EuroTEX'95*. [He presents the proceedings]

Knuth: Oh!! I thought you'd never ... [laughter] Yes, I was looking at this last week in the Czech Republic, so thank you everyone.

Erik: What is your opinion about the fonts we used?

Knuth: I think it's ... oh, you used the Computer Modern Brights. Yes, the only complaint I had was that the kerning in the word 'TEX' itself could be tuned a little bit.[9] It's quite attractive—thank you very much.

[8] *Software Concepts and Tools* **15** (1994), 2–11.

[9] "The TEX logo in various fonts," *TUGboat* **7** (1986), 101.