TUG 2010 Panel: Don Knuth & Stanford TeX Project members

David Walden, moderator

[This transcript has been lightly edited. The session is available on video at `http://river-valley.tv/tug-2010-panel`.]

Karl Berry. I will briefly introduce our panel moderator, Dave Walden, who as you know handles the Interview Corner and all kinds of interviews . . . so I asked him to handle this one too.

Dave Walden (moderator). Thank you, Karl. I was really pleased to be invited by Karl to chair this panel, for three reasons.

First, despite the fact I've lived in Boston for 46 years, I grew up in the San Francisco Bay Area and graduated in mathematics from San Francisco State. So, it's a real pleasure to be back at a math meeting in San Francisco. This is probably the first time I've thought about math in San Francisco in 46 years. It's good to be home.

Second, I've always admired Donald Knuth. I bought his *Art of Computer Programming* in the late 60s and used it in my daily work. When volumes 2 and 3 came out, I immediately bought them, and we used those in our daily work. More recently, when I decided to stop using Word and go to some kind of a text processing system that didn't have hidden proprietary undocumented markup, I chose TeX because I admired Don Knuth and I thought I'd like to try something that he created.

And, of course, the third reason is that that brought me in contact with this community, and through interviews and so on with everyone on this panel. So, third, it's a real pleasure today to get to get to meet everyone here today in person.

With that I'd like to introduce the panel members. I'll first mention Don. It's a cliche to say, "he needs no introduction", but with this group and this man, he needs no introduction. I'm sure each of us knows of several things in his massive set of accomplishments in a variety of areas. And, in fact, his publisher conveniently gave us this list [holds up advertising page from CSLI] of nine different books of his collected works in different areas, and that does not include *The Art of Computer Programming* which has a different publisher.

So I'll go through the rest of the panel in alphabetical order, and I will introduce you to them saying a word or two about their TeX accomplishments. Naturally, they have had full careers in other areas and have done many other things, and I commend the interview series to you. And the couple of you [panelists] who have not yet participated — it's time.

[At this point the moderator introduced David Fuchs, John Hobby, Frank Liang, Oren Patashnik, Michael Plass, Tom Rokicki, Luis Trabb Pardo, Howard Trickey, and Joe Weening using essentially the descriptions of the "TUG 2010 Conference Report" on page 117 of this issue.]

Regarding the format of this panel, Don asked me to say, "I am going to have a half hour to myself later. This is the only time the rest of the panelists get to talk, so please focus as many questions as you can on the rest of the panel" — said Don, and I say that to you [members of the audience]. What I'd like to do is go through the panel one-by-one. I won't do a strict rotation but let's have a question for each panel member before we go to a more open format. And, panelists, I encourage you, if you have something to add to an answer of a fellow panelist, please chip in. I think that hearing different sides of these historical stories is often interesting both because it elaborates on the stories or sometimes it shows some conflict in the stories.

May I have the first question from the audience. Oh, one more thing, unless we have a mike in the audience, please say the question loud enough for me to hear and I'll then repeat the question for Kaveh's videotape.

Unknown. The question for Frank is, "How did you discover your hyphenation algorithm?"

Frank Liang. Well, I was assigned this problem as a thesis in 1978, I believe. As I have mentioned in my thesis, there was an initial suggestion to use a kind of statistical algorithm which looked at two letters — well, actually four letters — surrounding a potential break point, and then you were supposed to make tables using two letters before, the middle two letters, and the last two letters, and then combine these tables somehow to do hyphenation. So I started experimenting with some word lists, and I quickly found that that wasn't sufficient; you needed more context. One example I mentioned in my thesis is that sometimes a letter seven letters away from the hyphen point can alter the breakpoint.

Anyway, I'm playing with word lists a lot and came upon the idea that just patterns of letters was a very simple way. Because I had started with just these two letter digrams, it was natural to extend that to longer letter sequences. And then through a long process of evolution I came up with patterns and then with the rules and exceptions. Don actually came up with the idea of assigning the numbers and having them at one of my thesis review meetings. I sort of had the idea of having rules and exceptions, and he said, “Oh, you could assign numbers to them.” So that’s part of the answer.

One of the hardest parts of the whole thing was acquiring a suitable word list. I didn’t have access to every database, and there weren’t that many at the time. I got a copy of the Merriam-Webster dictionary that had hyphenation points. But upon looking through it, there were many errors and lots of typos and things that just weren’t quite right. So I had to hand edit the dictionary, and that took about three months. So that is where most of the work was actually.

Moderator. Any of the other panelists have a comment on this?

Howard. And the tries ... ?

Frank continues. The tries — that sort of came up separately, after the pattern idea. As I was playing around with the word lists all the time, I needed some kind of relatively fast algorithm to quickly collect information about the patterns in the word lists with hyphenation points and then to test out various theories. Because these were just simple strings, it was natural to look at variations of standard data structures like tries for that and to read related papers like Don’s on pattern matching.

The problem with the tries — tries are very fast because it is just based on indexing, but they tend to be very sparse so you then have to use various tricks to speed that up. And the idea, actually, for doing this weird packing, if that is what you are referring to, was I read another paper by a Stanford professor at the time, Andrew Yao, which was talking about storing a sparse table. It was a somewhat different application, but he had this idea of when you have these sparse things you sort of interleave them all in one array and thereby save space while maintaining speed. So that’s where I got that idea.

Moderator. A question for another panelist?

William Adams. My question is for Tom Rokicki. Ages ago, when the WorldWideWeb.app was written on the NeXTstep, there was also available on that same platform TEXview.app, and we’ve seen an awful lot of effort in trying to get mathematics and nice fonts and nice settings onto the web. Why didn’t we just start out with an extended version of hyperTEXview.app and cut to the chase?

Tom Rokicki. [to the moderator] I’m sorry. Can you repeat the question?

Moderator. I don’t understand the question. Maybe William can say it again, and I can try to repeat it.

William. We started out with TEXview.app on NeXTstep and on that same platform, WorldWideWeb.app was developed by Sir Tim Berners-Lee. Why wasn’t TEXview.app used as the basis for the WorldWideWeb.app so that mathematics and so forth would have always just worked instead of us constantly working to try to make them work on the web?

Tom Rokicki. Basically, you’re asking why TEX was not chosen as the basis for mathematics on the web, based on the NeXTstep. Boy, I’m not really the right person to ask. Gosh, you know, I really don’t have a good answer to that. The HTML stuff was really crude in the beginning, and it’s still pretty crude, but it’s getting there. So I don’t really think anybody spent a lot of focus at that time. Back then, it was just “let’s get the links working, let’s get the text working, let’s draw around images, and stuff like that, and call it a day.” Which was pretty amazing in itself. As far as what went after, I really can’t say.

I’m not sure that it really fits though. Because the web, HTML, is all XML-based. And TEX is rather different. And I think there was a very strong reason to keep it as a markup language like SGML or XML, that could be easily automated, and restructured, and all this type of stuff. So I think that there were good reasons why TEX was never used at that point. But I was never really part of that, so I can’t say for sure.

Moderator. Another question? Nelson.

Nelson Beebe. Question for David Fuchs. It’s important to remember that in this audience there are a lot of younger people here who have grown up with laptop computers, that TEX was designed on a machine that cost roughly half a million dollars at its entry level price. And there were two people, one of whom is sitting here, who really changed that and made life different for an awful lot of people, and Dave Fuchs is one of those; the other one is Lance Carnes, who unfortunately isn’t here today. I’d just like Dave to comment on his work with MicroTEX, which brought to you the first live preview, while TEX was running, of what was happening, and how difficult it was, and how much hair he lost trying to get TEX to work on the little machines of the day.

David Fuchs. Well, that was fun. It was a lot of kind of hackery and trickery down at kind of the bit level, and the intricacies of the 8086. At the time, there were no good Pascal compilers, if there ever were, ..., well, that’s not true — there was a good DEC one. So that work involved writing kind of a limited Pascal-to-C translator that had special hacks — it didn’t bother with the parsable language that TEX didn't use, it had special hacks in it so that I could insert using the change file scheme from WEB. I could insert magical keywords that I had laid out that said, oh, here's one of the big arrays, and then I could write some assembly code. . . . Gee, there was even a version, there were different instantiations of it. One early version, I just pretended to the C compiler that the arrays weren't really big. It always produced one of three different sets of machine code instructions to address those arrays; I had a post-processor that would look through the object code that the compiler created, look for those three or four possible patterns, insert some special other code that called some of *my* code to look at the real array thing, ... That only lived for a version or two. Then there were other cute tricks: it kind of used a sort of VME system where I chopped up the arrays into pieces, and those got swapped in and out of, I think off of the disk. I even stuffed some of them, as an experiment, into the . . . it turns out the video cards that you had, that drove your displays, they had some extra RAM in them, so you could swap stuff out, and it was kind of fun, 'cause if you put the video card into a different mode you could actually see that stuff on your screen going by.

So, the early PCs had a maximum of 640 K, and a lot of people for a while had 512 K, so that was marginally not quite enough to do everything, so you had to replace all the run-time libraries that came with the compiler. I was down to . . . I wasn't even using . . . what's it called that's usually linked in with the C program, even the startup code, so you got . . . there was no standard files, there was no anything. So that was all to cram it down into 512–640 K.

Lance did a job too. Because his stuff was rather commercial — that's his livelihood — he was always somewhat circumspect about it. But, obviously, it wasn't complete black magic.

Tom also had some work in this direction, if I remember correctly?

Tom. Absolutely not, absolutely not. [laughter] This accomplishment of David Fuchs' was one of the most amazing things I've ever seen. All my platforms had *plenty* of memory. I never had those issues. I cannot *believe* what Dave accomplished — it was absolutely amazing! Okay, truly a hero.

Moderator. Anybody else have experiences with these tiny machines?

Another question, please. Hans.

Hans Hagen. Aren't you somewhat disappointed after 32 years that not more people made fonts using METAFONT?

Don Knuth. Well, I can't say I'm disappointed in the fonts we have now. And I'm happy with the ones that my students have made, and I made. So I'm not . . . . When I wrote it, I just had the idea . . . everybody's entitled to have some mistakes in their life, and so I didn't have to worry about the fact that not everybody would use every program I wrote, and this one happens to be a very personal thing, and so the way I look at it is, how wonderful that John extended it to METAPOST, which I use a hundred times more than I use METAFONT. I've already done most of what I ever need to do with METAFONT.

Hans. But didn't you overestimate the font designers then?

Don. Well, I thought it would be easier to teach the font designers about the notion of parameters and metadesign than it was. Computer scientists, we're used to writing something that's going to work under many different conditions as the parameters change. But to most of the rest of the world, to my big surprise, they never heard the word "parameter" — they thought it meant "perimeter".

Moderator. John?

John Hobby. Yes, I certainly agree that the METAPOST application was more popular, but Don has a very unusual set of skills, and indeed, there aren't many other people who are good at that kind of thing. I think the real thing is, it's just too hard to create a really meta-font, as far as the art community is concerned.

Moderator. Any other comments on that from the panel?

Okay, another question — for Oren, or Michael, or Luis, or Howard?

Karl Berry. I have a question for Howard, which is, I've spent a lot of my life looking at TEX.CH, and at the top in all those change files, pretty much all of them say "Howard Trickey and Pavel Curtis". I see Howard Trickey, who I never quite understood was at Stanford before this conference. I wonder if you could tell us if you worked directly with Pavel, or if it was two independent things, or how that came about.

Howard Trickey. This is pretty interesting. The fact that that change file has my name in it meant for many years it was really easy to find me on the World Wide Web. There was, like, 300,000 references. So thank you all for putting that page up on the web.

Don had done his thing on the DEC computer, and I had worked on VAXes — I didn't like this computer. And I wanted this thing to work on VAX.

Moderator. The VAX was from DEC. [laughter from the audience]

Howard. You're right! I'm sorry!

The DEC 20 as opposed to the VAX. And so I did the work that was necessary, which turned out to be evil, although not nearly as evil as the things Dave had to do because he had to change the Pascal compiler default clauses to fool around with the [inaudible]. And I had to do the change file that did the system-y stuff that Unix needed, so that's where that change file came from. And then I found out, hey!, this guy named Pavel Curtis had done the same thing, unknown to me. It happened almost simultaneously. So we were hooked up together and cooperated together.

Moderator. Anything else on that from the panel?

Don. Can I ask a question of Michael Plass, to describe his experiences in 1978 when I went to China and asked him to implement the prototype of TEX.

Michael. Don had this trip to China, and he left Frank and me with a few pages of what his ideas were for implementing — what he would like us to implement over the summer. (I actually wonder whether I still have those pages somewhere. I tend to keep stuff so it's possible — that would be interesting.) Frank was tasked with the hyphenation, and I was tasked with building up starting with storage allocation, I guess (the very bottom), the macro processor, and up through ... I think by the time he got back it was about ready to start implementing some of the line breaking stuff. I guess Frank was also working on the output — the printer driver end of this so we were able to make some prints by the end of that summer. One thing I remember is with the macro language the way Don had spec'd it out, I did some experiments, and he decided it was too powerful — that you could get too tricky with it and do too many things. So he redesigned it to be much more token oriented than it was in the original.

Moderator. Frank, do you have anything to add to that?

Frank. Well, what I remember in addition to the hyphenation which actually I did while Don was here was that, after looking at his notes, Mike decided — he was sort of in charge — we decided to split it up and I would do the output and he would do the rest. And I said this didn't sound like much. At the time he thought output sounded pretty difficult because maybe he didn't know how to do it right off the bat, and of course I had already been playing around with the XGP so I knew how to do it. So for me actually it wasn't that much work and he ended up with much more than he thought. What he gave me was a list of boxes, graphics boxes, and I said, okay, I'll just put them on the printer so that wasn't much work for me. Obviously we way underestimated how much work it was going to be and it was two more, or several more years of Don's work later.

Moderator. I have a followup question. In Michael's interview, on the TUG web site, he says that after Don got back, then he rewrote it all. And so my question for Don is, you didn't like Frank and Michael's work?

Don. Oh, no, actually I liked it, although there were basic changes made. I think control sequences were sort of considered as one character at a time instead — this tokenization idea was coming along at the end — so really the main thing is it gave me the idea for an architecture for the program. But I never expected that I was actually going to use that exact code. I wanted to see it in place; I wanted to see how big it was, what kind of subroutines you needed, and things like that, so that was a key step in getting going. But I knew my sabbatical year was coming up, and that during that time I would ... I always intended to look at what they had and then work over again, and say, okay, now back to square one. Now we know what it's going to be like. Now let's design the right data structures that go with this kind of architecture.

Moderator. Question for Oren or Luis or Joe?

Boris Veytsman. Question for Oren. I always wanted to know, what was the inspiration for the style of BIBTEX language?

Oren Patashnik. For the style of the BIBTEX language?

Boris. No, for the BIBTEX language itself — for the `.bst` files.

Oren. Leslie Lamport needed somebody to do a bibliography program to go along with LATEX, and his idea was to use, sort of as a model, Scribe — it had a bibliography program. So he and I sat down, and he had some ideas about things that he'd want in this `.bst` language. So he and I sat down and kind of discussed it, and I was the one who implemented that. So BIBTEX itself is really, to a first approximation, an interpreter for this `.bst` language. That's really what BIBTEX is.

Leslie Lamport — I think the main ideas for what (he had thought about it before) what was going to go into that language, came from Leslie. And then I implemented it.

I just wanted actually to follow up, in addition to the discussion of literate programming from earlier [Bart Childs's presentation, "Thirty years of literate programming and more?", pp. 183–188], I sort of thought about this a little bit over the break, and I went up to my room and got this — it's my copy of `bibtex.web`. I think `tex.web` is probably the largest piece of software in `WEB`, and `bibtex.web` is maybe the second or third, I'm not sure. But it's a third or maybe two-fifths the size of TEX. Actually, I hadn't — all the stuff I've been doing with BIBTEX since, has been looking at the stuff with the `.bst` language, and not with BIBTEX itself. The only bugs have been really, really minor, except for one; there was one kind of majorish bug, which is that it didn't handle URLs.

Well, back when BIBTEX was written, there were no URLs, so I didn't have a chance to test it out on that. I think I misunderstood something that either Don said, or Dave said, I don't remember, about how control sequences are handled, and so basically, BIBTEX doesn't handle the line-breaking right for very long URLs; that's the issue. And so finally, I was convinced that I should . . . , and rather than release a version of BIBTEX with all these kind of minorish things, that's probably not worth wasting people's time to install a new version for that. Rather than just doing that, I thought I would finally release a new version of BIBTEX that had as its only change that change to how BIBTEX handled the long URLs. And so I recently did that a couple of months ago — I was working with Karl. And this is getting back to the literate programming in `WEB`. Every time I look at BIBTEX itself — sometimes I look at the bits on my computer, sometimes I look at the hard copy — every time I look at it I think, kind of, what's going on here? But pretty quickly I then sort of get into it. But I hadn't really had to make a change before, until this time. And I realized . . . same experience, I looked at it, I knew I'd taken a peek at this, and I thought, well it's not completely trivial, so I'll do this eventually. So now is the time to do it; we finally decided now's the time to fix that bug.

I looked at the code, and after not very much time — it always takes me a long time to do context switching — I looked at it and said, oh, gee!, the structure of what's going on in the program became completely clear. It was like I was in a zone; you hear athletes talk about "being in the zone", a basketball player, all of a sudden the basket looks huge, and it's easy for them to make a basket, or a baseball player, the pitch coming in from the pitcher looks like a grapefruit, and it's easy for them to hit it, or a soccer player knows they're going to have a 28-yard direct free kick, bend it around the wall and bury it into the upright corner of the net. I mean, talk about athletes getting into the zone, and I sort of felt like that was the experience I had here: After a little bit of looking at this program, all of a sudden the structure and what was going on became completely clear, and it was really easy for me to make the change.

I had just switched computers, so I didn't actually have a TEX implementation on my computer, and I had to use Karl as my debugger — KBDB or something like that — so in the change I made there was one minor mistake. So it took two passes. But I was amazed at how, initially you look at this code and think, what's going on? But very quickly I completely realized the structure.

I think what happens is, when you do literate programming, it imposes in your mind a map of what's going on in the program. It had been 22 years since I'd looked at this code, I think; after 22 years, it didn't take very much, and all of a sudden it was crystal clear. I've never felt that experience before. When looking at `.bst` code, that doesn't happen to me. [laughter] But with a literate program, I think it's because literate programming imposes on you a structure that, even when you haven't looked at it for 22 years, it comes back; it's still there.

Moderator. Don, you have something to say?

Don. Well, speaking of literate programming reminds me of a question for Joe Weening, because it was Joe who suggested the idea of mini-indexes that I used in the `TWILL` program for literate programs, and Joe made some mockups, so maybe he can remember something more than I can.

Joe Weening. I remember them, but I don't think there's much to say beyond what you just said. It's a pretty simple idea: looking at a page of a literate program, there'd be a lot of names you hadn't seen before — names of variables, names of other sections, and so on — and so rather than go from each page to an index and then to another page, the idea was, let's go there directly. Of course, this was before hyperlinking. You must have heard about hyperlinking! Nowadays, what you'd want to do, you would be looking at this on line, and you'd just click on something. That's probably what [. . .]

Don. Certainly we do have the hyperlinks and the clicking now, but a lot of times there's still a value for this in hard copy, when you have a book and you're sitting in your chair, or you want to keep coding without clicking to the other part. But the thing was, you not only suggested the idea, but you also showed a really nice way to present it. I guess it seems simple to you, but it was a real revelation to me.

David. `TWILL`?

Don. Yeah, TWILL. It's on my web site. It's not that easy to use, so I don't advertise it much. It requires running in several passes. When you have a literate program and you have a variable called '*x*', there might be thirty variables named '*x*', so you have to disambiguate which one you're talking about, at least when you write code the way I do, which is maybe not the best. So you have to go through several passes, and then give hints, and say "No, I didn't mean *that* '*x*', I meant *this* '*x*'." And you have to tell it to say "This is something in such-and-such a C library." So there's a bit of hand-tuning that goes on, and it's not a trivial thing. I just went through a book coming out later this year called *Selected Papers on Fun and Games*. In there I have a hundred pages — I took the original program of "Adventure", the cave game, of Don Woods, and I rewrote it as a literate program. And it appears with these mini-indexes, but I had to go through carefully and do it. But the original program used to do Volume B and Volume D, you know, *TEX: The Program*, *METAFONT: The Program*, it was called TWILL. And now I have CTWILL.

David. Wasn't there an early version of WEB? Did Luis work on it? Wasn't there a version of WEB that was before WEB?

Moderator. Perhaps one of you could speak a bit about that? Luis, perhaps?

Don. Yes, I was going to ask Luis about the original ... I mean Luis was involved with this project so early on that you don't know any more all the key things that happened. There are, for example, questions of how did we port TEX to a hundred different computers and get the tapes out and everything like this. People were asking about Maria Code the other day, and somebody said they didn't know if she was a real person. And also, you might be able to also speak as to what Ignaki [Ignacio Zabala] did — he's the main person of the original team who isn't here today.

Luis Trabb Pardo. The question about the literate programming, it was originally a much more mundane thing. We were distributing tapes, and people wanted to know where was the "Main", where does the program start? I said, "At the bottom." And my primary function at the TEX Project was to answer the phone. And I essentially answered that question, "It's at the bottom." In some discussions with Don .... Also, there was the issue of documentation. So, the idea of blending that came very naturally as the necessity to not answer questions but have the questions essentially be answered by what you said. That was the suggestion; essentially Don did the whole thing.

The issue about what Ignaki worked on originally was to start thinking in terms of graphic objects that were going to be put on a workstation. We were in an environment at Stanford that was a timeshared system, and we didn't have much ... well, we did have interesting things going on there, but the concept of a resource available to you like we have today, on our desktop, on our personal computers, was not there. So he started working from that component. He actually put together a system that had all kinds of graphic things. He called them graphic objects. And you could do what you could do today in a display environment on an existing system. I think that answers more or less the level of what Ignaki did.

Unknown. Was Maria Code a real person?

David. I believe that was Ron Code's wife. I remember dealing with Ron more than Maria. The ARPAnet was there, but that was only academic and government institutions, so it used to be, you'd send in a tape to, I think, Ron and Maria Code, who were entrepreneurs, I suppose; I don't know how they hooked up with us. And they would spin off a copy for you and send it.

Unknown. So they weren't members of the Stanford CS department?

David. No, no, I saw them in person.

Moderator. The statement from a member of the audience [Gio Wiederhold] is that Ron [Code] worked for him in medical information systems.

Gio Wiederhold. Maria did all the hard work, and Ron managed her.

David. I believe I gave him the tapes after a while, whenever there was a new release. Following up on the thing that Luis said, when he stopped answering the phone, I was the one who started answering the phone.

John. Actually, I answered the phone for you.

David. I apologize. [laughter] It turns out, there's really only five questions that anybody ever asks, so after awhile, if you called, and you happened to get me, you could start asking your question, and I'd be able to answer it before you were done. And, not only that, after awhile it got so I could say, "And, by the way, the next question you're going to ask ...." So people thought I was brilliant from this, but it turns out it's all fake.

Moderator. There's a question out there.

Didier Verna. Something totally different. The first part of this question is probably for Don, and the second part for everyone. I would like to know why TeX was designed as a macro expansion system, and the second part of the question is for all of you: How do you regard that design decision thirty years later?

Don. The way TeX was designed was the following. I thought I would have a language for myself and my secretary, and I sat down one night and I wrote out — I chose about seven pages of *The Art of Computer Programming* that had different kinds of features on them, and I said, how, if I were entering this into a computer, how would I like to do it? And I wrote that down; it's all there in the book *Digital Typography*, the memo that I stayed up late one night writing it out. And then I figured out, I changed it a little bit to something I thought I could implement, and gave it to Michael and Frank while I went to China. But the design, it was natural to have macros rather than procedure calls, the way I looked at things, because of the way I could conceive of writing this program.

So, the second question is, is it worthwhile? Well, you'll have to wait for my next talk. But I don't know what the other people on the panel [think].

Moderator. Does anybody else have a comment on that?

David. Yeah. One of the things to keep in mind, people going, oh, my gosh, how can you possibly fit it in 640 K back in the PC days? Well, it's important to realize that the big DEC-10 that it was developed on was a 36-bit-word machine — let's call that 4 bytes, more or less — and you only got 2^{18} of those, so that's only a megabyte. And the whole thing, even in the big version, fit in a megabyte, and, boy, when you look at a lot of the decisions, at least from my perspective, it was driven by that. It's amazing that you can do that much in that little memory.

Don. But when I got to METAFONT, it was macros gone berserk, because I had object-oriented macros in there.

Moderator. Another question from the audience?

Hartmut Henkel. This is for Don Knuth. Was it backslash from the beginning? Why, actually, is it backslash introducing any TeX command?

Moderator. The question is, was it always the backslash that introduced TeX commands?

Don. You can see exactly what it always was just by reading that chapter — I guess it's *two* chapters — in the book *Digital Typography*. It answers every question about what was there in the beginning.

Actually, I think, in my very first draft I did *not* have reserved words, because I was thinking of nroff. They didn't have any backslashes, so then I wouldn't be able to use certain words. But that didn't last very long.

Moderator. Question over here.

Unknown. This is partly for Don, but partly for anybody else. When did you first become aware of the PostScript language or its predecessor JAM and to what extent was there any influence of that design to TeX, or perhaps backwards?

Moderator. Michael has an answer, maybe?

Michael. After I graduated from Stanford I started at Xerox PARC in the lab where Chuck Geschke and John Warnock were, and at the time JAM was in use. By that time, they had already translated TeX78 into Mesa, so it could run on Altos and the other machines in use there. So I really don't know how much they influenced each other, but I think the TeX stuff probably predated a lot of the ...

John. I did learn JAM one summer very early in my graduate career, but it was just part of my education.

Don. I didn't look at PostScript very much myself, but I *did* visit PARC rather often, and I saw, well I remember one of the first times I went there, going by a room, somebody sitting by a terminal, and there was a big letter "B" he was measuring. So when I took my sabbatical year, that first year in 1978, I asked Xerox PARC if I could work there, to do my font work. And I was going to measure all the letters in my book, and fit splines and everything, and they said, well, that would be fine, but then all of your fonts belong to Xerox. So I went back to the Stanford AI lab and decided to do it myself. So there was a lot of work going on at PARC. And Warnock actually brought his stuff from Utah before, which I only learned later. But then the other main influences — afterwards, in the '80s I'm meeting the leaders of the font industry. Mike Parker comes from Mergenthaler, and says, boy, there's some guys over ... that have this PostScript language that renders fonts in an incredibly fast way from outlines, and things like this. And he was all excited about it. And they had new ways of tuning the fonts to the raster dynamically; hints, they call it now. And so that's when I first learned, myself, about that kind of work, the PostScript.

Moderator. I have a question. When in his interview, John Hobby says he worked with Don on METAFONT, that he primarily worked on the algorithms, and Don did the coding himself, I'm wondering with Frank and Michael, with hyphenation and paragraphing, was it the same situation there, or did any of the rest of you actually touch the TeX code?

Don. So, let me say that they were watching me, over my shoulder. [laughter] Especially David. But if you look at the error log you'll see, for example, like I think there will be several entries that say "HWT", and that's Howard Trickey, and so on. And the error log is also printed in *Digital Typography*. So the idea was, really, I was the filter and the final end. And we had this group that would meet once a week, and we would discuss over lunch, maybe two or three hours, and we would toss around whatever the topic of the week was — everybody was participating and looking at these decisions during that time, and then I would have to go back. Usually I would have a new chapter of the manual with me that they would look at, and say "Well, why did you do that?" Or they would shoot ideas, "Let's put this feature in." But then I didn't want the project to diverge, so I always decided basically, okay, we're going to make sure this is a little bit unified by being the only person who wrote the code. And I guess it's also because I guess I'm a little afraid of using something that I don't really understand all the way through.

Moderator. Anybody else want to comment on the collaboration with Don?

David. Let me just follow up quick with that. You know for awhile when I was answering the phone, I was also kind of the gatekeeper — people thought they found a bug, which frequently they hadn't, but when they reported it, I'd go, okay, I'd check it out, and see that it seemed like it didn't work, and I'd go into the code — it didn't happen that often — but I'd try and come up and find the exact line that was the problem and come up with a suggested solution, you know, in real code and test it out, and then I'd send it along to Professor Knuth, and not once did a corrected version come back that matched what I had suggested. [laughter] Never happened. It's all his code.

Moderator. Luis?

Luis. I think I have a comment in the opposite direction, which is the influence of what was happening, what Don's area of work and expertise meant to all of us who were working in there. I want to counter that with my experience in industry later on.

One of the things that you see in the development of TeX and METAFONT is the concurrent solution of a large body of problems that were not solved in a particularly efficient way, or were solved in different places. And many algorithms were redone, or adopted or whatever, but there was always this ... this group of people is the group of people who had learned about algorithms and about how to do them efficiently, and to solve *complex* algorithms, not trivial [ones], and solve them in an efficient way, not just take the trivial answer and be just happy with it. One of the things I've seen repeatedly in my experience in industry is that many, many engineers just decide, "Oh, it works. Bye." If you try TeX or METAFONT that way, it will not work, because maybe some things will be solved, but the entirety will not work.

The other thing is the issue about efficiency and optimality of things. My own little experience on this was, at the beginning we needed to interface things. And I remember a conversation with the engineers at Canon, who had brought to Stanford a printer, an OEM printer with no controller, and they said, "We did the printer, but the controller is very difficult." So we said, "We'll do the controller. Can you give us the printer?" We had a discussion about it. And what we did in there was essentially to think about how to use things that we had, like a little microprocessor, and built a few things, and we got a controller in there. The industry was not willing to go that way. They would say, "Oh, it's a complicated thing, we need a lot of memory, a lot of power, ..." Well, what we had at that point were the people around who just figured out how to solve it with the things that they had there. And I guess that that's what made it possible.

Moderator. Karl?

Karl Berry. I guess this is a question for everybody except Don, per his request. Given all the comments about literate programming, both by Don and now by Oren, I wonder if *any* of you have seen literate programming after your TeX life? [silence] I was afraid of that. [laughter]

Moderator. Has anybody seen literate programming in their life after Stanford, after TeX?

Karl. Used by, done by somebody else?

Tom. I don't claim to be capable of writing very well, but on a number of our projects I *have* used literate programming techniques for a delivered product. In my modern life, I don't. For fun, I do, but not for anything I do commercially, or anything like that. But for a number of projects I delivered both during and after grad school, I based on literate programming ideas. I didn't necessarily use CWEB. For instance, one project I did — it was a SCHEME-language program — I did use CWEB for that. But certainly the concept of the linear exposition of the ideas, small sections, and focusing the effort on allowing readers to understand the program was the goal I was shooting for, as opposed to just letting the compiler accept it and letting it pass all the unit tests and stuff like that.

Moderator. Luis?

Luis. I think if you just take a sample of what you have out there, you would say that probably documentation is never done within two or three years of release of a project. So essentially, there is no priority anywhere in the workplace to anything close to this idea of presenting the totality, of the concept, the implementation details, and the actual implementation in one place. You'll be fired if you try to do it.

Moderator. There's a question back there?

Oh, wait a second, Don wanted to say something. Hold that question.

Don. About 20 years ago when Bill Gates visited Stanford I gave him a copy of the book *Literate Programming*. I just wonder, Frank, if anybody in Redmond ever saw it.

Frank. Twenty years ago I think I had already left Microsoft. My office was right next to Bill Gates's for about a year. But I hardly ever saw him except he would walk by in the morning and he would walk out in the evening. So I'm sorry. We used at Microsoft, when I was there, we had all our own development system which was kind of inspired by another Stanford graduate, Charles Simonyi. He had something called Hungarian. I don't now how many of you are familiar with that. We used those conventions at Microsoft in the early days. Now the organization is so huge I don't really know.

Moderator. Let's go back to the question there. . . .

Idris Hamid. This is a question for Donald Knuth, and it relates to, I guess, what we might loosely call the mysticism or spirituality of TEX, what I would loosely call the mysticism or the spirituality of the topic. One of the classes that I teach in the philosophy department is religions of the West. And in the Christianity segment, I actually used *Bible Texts Illuminated*. We don't have time to read the whole Bible, and of course that covers a number of religions, and even if it only covered Christianity we wouldn't have enough time to cover the entire Bible. But taking your own approach, this is one of the texts I used *in* that class. And, before I ask the question, I want to make one more brief, contextual comment about this. In my own work, which involves the study of Arabic manuscripts, a lot of which relate to spiritual literature, one of the things that I've encountered is the need to begin to deal philosophically with the æsthetics, for example, of the Arabic script. And then I start coming across certain sayings in my own tradition, as a Muslim, for example, a beautiful script makes the reality that it represents become more obvious. Or, that God is beautiful, and he loves beauty, loves to see his creation create beauty. So my question for you is, when you reflect on your years of work developing TEX, and when you look at your interaction with the written word, what spiritual or æsthetic reflections or wisdoms would you like to share at this juncture, given these years of digression that turned into such a beautiful product; what kind of spiritual or æsthetic reflections or philosophical reflections would you like to share with us?

Don. So, in the first place, it sounds like we're on the same page with respect to our feelings about the primacy of having beautiful things. The second thought that came to mind quickly was, when TEX itself became a reality was the moment that it had a name, and that also goes into the religious concepts, of naming something. When Duane Bibby came along, it actually got more of some kind of a soul; I don't know. But anyway, the project from the beginning was definitely driven by the idea that I wanted to have the things that I myself was writing would be something that people would enjoy looking at — not just reading, but also somehow the idea that I liked it enough to also present it well. I would dot the 'i's and cross the 't's and care about ligatures and things like that, instead of just getting [inaudible]. So that's not shared by everybody, and all of these questions are very personal; so also, I think that the Muses would agree with this kind of opinion.

Moderator. Bart?

Bart Childs. This is for everybody *except* Don. When Don was about to release, started in to do 3.0, several of us got an e-mail, "are there any features that you think should be added to 3.0?" I sent a list off, and later I thought, well, Don, mine didn't make it, but did any of you say, "here are features that you ought to be putting in TEX" that didn't get there, and, if so, what were they?

John?. I remember one that made it there, which was the one-character font name argument. I remember the day we all yelled about that back and forth, and it was nice to get multiple character font names.

Don. No, that was before 1.0. The original TEX78, there was `\font a`, `\font b`, `\font c`; there wasn't room for storing more than 32 fonts, so why should we allow multiple [character font names]?

Joe. I was trying to remember, what year was TEX 3.0? 1990. Yes, and the big thing was 256-character fonts and features like that. I think most of us were gone from Stanford by then. Tom — maybe you were still around.

Don. What ideas did you guys propose at our weekly meetings that I ignored? [laughter]

Joe. I think I wanted to make control sequence names *not* be case sensitive, and you insisted that they be case sensitive. [audience: why?] Just a matter of personal preference. [audience: I agree with Don! laughter] I guess I was just a Fortran programmer for too long.

Moderator. I have kind of a follow-on question here. The question that was asked was kind of a legacy question. I have a slightly more down-to-earth legacy question — I'll direct it to Tom, but I think it's probably true for all the rest of you in one way or another. Tom, at one time you developed `WEB2C`, and you developed `DVIPS`, and then somehow you got somebody else to take it over, or somebody else took it away from you. How do you feel about the separation, and the continuing life [of your project]?

Tom. Oh, boy. Appreciative, I guess, is the word. And truly so! I mean, not actively using that much, actively supporting a bunch of people — it's a different world when you get out of it. And it was hard for me to deal with some of the problems people were having, because I wasn't even running anything like that. And there were a number of issues with `DVIPS`. Part of it was that there was a certain effort to make it be GNU, and at the same time I really wanted to keep it free of the copyleft. And it turned out to be what I ended up doing. But I really tried to keep a separation there for awhile, and it turned out to be a bad idea .... I'm just really grateful that people took up the leadership role and made it all happen and kept it all working 'cause I wasn't doing it.

John. I could add a comment to that, in that I clearly also gave up a leadership role. In my case, clearly I was eager to give it up simply because I didn't have time to adequately maintain the program. But anyway, I think we're all glad to give up the leadership role.

David. Oren needs to comment on that. [laughter]

Oren. Yeah. My comment was that I never let that stop me. [laughter]

Moderator. You have a new release of BibTeX coming out?

Oren. BibTeX 1.0 is going to come out any decade now. [laughter]

Moderator. Another question from the audience, please? There's one way back there. Frank — is it Frank?

Frank Quinn. Leslie Lamport has been a big influence on all of these developments. Could you perhaps comment on how you saw his motivation, and what sort of interaction he had with the group?

Oren. Well, I'll comment a little bit, since my real first .... Other people here really know more of the TeX internals, everybody else does than I do, and so my first contact with a lot of the TeX stuff was through Leslie, and as I said before, he needed somebody to do a bibliography processor for LaTeX. And I know his thinking was, he kind of liked the path that Brian Reid took with Scribe; it was kind of a very simple interface, you could describe things fairly succinctly. Somebody who's an English major could easily use it and get nice output. And so, I think that was the first contact I had was with him. I don't know how much he was involved with the TeX project before that. I think people at — where was he then? I think he was at SRI — he had written some macros that people liked there, and I think they encouraged him to do something with it. I'm not sure where his other influences were, but certainly, once he got going, I think people, obviously they're fairly happy with it.

Moderator. Anybody else?

Don. He's a very independent spirit, like I am. He does a lot of work on his own. Every once in a while he would run into something he couldn't do, and so then I would have to put in another feature, while kicking and screaming. But it was totally independent work from Stanford.

Moderator. Right here — front row.

Robert Cristel. We know how the problem of ... *The Art of Computer Programming* [printing] caused TeX to be more [æsthetic]. So my question really is, apart from making *The Art of Computer Programming* more beautiful, how, in other ways, did TeX affect *The Art of Computer Programming*?

Don. Well, it set it back about fifteen years. [laughter] On the other hand, I'm writing a little bit faster now, so maybe it'll save twenty years if we amortize the whole thing.

I thought you were going to ask about other things *besides The Art of Computer Programming*.

Robert. I mean the stuff that's inside, not necessarily the other.

Don. To my great surprise, right from the get-go for example, Barbara Beeton came with a few other people during the summer of 1978,[1] and she showed me all the kind of things that she wanted to do with *Math. Reviews*, and then also the AMS

[1] Actually 1980. — bb

was having trouble typesetting their journals and things like this, so I started looking at applications of other users. And so then this meant that TeX had to grow in lots of ways. But it was sort of going from one user to ten users, and then from ten users to a hundred users, from a hundred users to a thousand users, ... each time the language had to change in some way. And I *think* the fact that all these things had to be filtered, had come down to me, helped to keep the thing from diverging, although of course for complicated systems ... it would have been a lot worse if we hadn't done it that way.

Moderator. Frank?

Frank Mittelbach. A question for Michael, I guess. I consider the paragraph algorithm as one of the very central algorithms within what actually has been achieved with TeX. It grew in time. I know your thesis, and I've seen other papers around it. My question is, as far as anybody can remember, have there been things you were sort of experimenting with that you *would* like to have in addition to that kind of algorithm, like in parameterization or something that you either never got around to doing, or found too difficult, or got shut down for other reasons? Is this the ultimate thing you wanted to have, as a group, in terms of being able to do this kind of thing, or is there some stuff that back then was not possible for some reason, but conceptually was on the horizon?

Michael. A lot of the features that were built into the line-breaking algorithm itself that's in TeX, I think Knuth ended up putting in there based on experience. As far as the actual coding of the algorithm, he did that. The origin of the problem was actually in the first graduate seminar programming class, where there was a problem for doing this; I guess he was thinking ahead a little bit to a sabbatical year at that time. But that was for breaking up — I think the problem was musical composition, if I remember right, but it's a very similar thing.

I guess as far as something that it would be nice if more of it were used in practice is the stuff that's in my thesis about arranging figures, moving figures from page to page, which at the time, it was certainly way too expensive to actually consider using in a real typesetting program. Maybe today it's not.

Don. So homework problem, Frank, go look at Michael's paper. He wrote a short version of our joint paper, which was published in another book about typography at the time. He generalized what we had and had an idea of a kerf (spelled 'k e r f'), and I don't remember what it was, except that it was good.

Moderator. A kerf is something to do with a saw, in real life.

Don. Okay, but anyway, it was in his paper, and I can't remember it today either, but anyway, I think it's worth resurrecting.

I wanted to mention something before I forget it. Although I wrote the main code that people saw, for TeX and METAFONT, there were also drivers and many other programs that had to be written, like TFTOPL and all kind of other what we call utility things. And Tom Rokicki did the things associated with PK fonts, for example. But David remarked briefly about having to take all the TeX code and convert it to C; well, he wrote a long WEB program that did this, and then I modified it slightly so that it would make profiles of the TeX system, so that it could instrument the whole program and find out how many times every instruction was done. And then David worked out a very clever thing that would work on our computer, and it — I think Joe Weening worked on this too — there would be daemons that would keep track of these statistics, and so over a whole year's time, every time anybody ran TeX at Stanford, these statistics were kept, and the counts were accumulated, and carefully saved, with machines crashing every day, but still pretty good stuff altogether. And then, using David's profiling program, I could make a pretty-printed version which would associate with every line of TeX exactly how many times people had used that line. And it was really important, for example, how many times did each error message get issued during the year. And we could figure out what the bottlenecks were. So anyway, to make a long story short, there's lots of other programs that were written at that time that were necessary for the development, that didn't go out to the world.

Moderator. Is there another comment down there?

Joe. I had completely forgotten all about that; it sounds familiar. I don't remember the details.

Don. The DEC-20 had memory that was divided into two parts, and there was one part that was sort of always there for the system libraries and things like this, and that's where all these statistics were living. There might be ten people using TeX, but only one copy of TeX is running somehow on the machine. And that took a lot of system wizardry, and I have no idea how they did it.

David. You wanna know? [laughter] So this was actually on the DEC-10, which had the WAITS operating system, which was custom-built at Stanford. That 2^{18} address space, it was half code, half data.

And the code segments were all shared, so if many people on this time-sharing system were running TEX at the same time, or any program — you know, the editor, the system editor — there'd only be in physical memory one copy of the code, but everybody had their own separate data, so I was editing my file, and you're editing your file. Well, the trick was, we wanted to count every execution of every basic block of TEX — every line of code, more or less. So the compiler used on that machine was this terrible thing from Hamburg — half the comments were in German, which I suppose was okay, but the compiler was not very good — but I managed to modify it so that — what! a lot of modified compilers here [laughter] — so that it would spit out little increment, atomic increment instructions every time it entered a basic block. The trick was that the place, the memory locations it would increment were in what should have been the read-only code segment that was shared among all the users. Yeah, right. People who know hardware are raising their eyebrows. [laughter] So that was fun, and every day or so, it would save itself out to disk, and there was special code that could retrieve this stuff out of the code segment, and it would actually increment; if the program counter was dot, it would increment dot plus two and then jump over it. So there was enough room for all the data. So that was another great piece of fun; this was back in the day ... the point is that the resources were really tight, and it was hard to get stuff in, and you had to do all sorts of hackery.

Moderator. I think we may be getting a little overly sentimental now. [laughter]

Well, over the last couple of days, trying to help this panel go more smoothly, we invited questions to be submitted in advance. And I have two, so I'd better ask them, or else I'll be very rude. And I'm going to combine them, in a sense.

The one question is, in your legacy, where does your work with TEX stack up? Now, that's kind of a TEX-centric question, but that's the question. And the other question is, because people are curious, what are you doing now? I think those two questions go well together — you've done a lot of things since then. Where does TEX fit into your life adventure? If anybody would like to answer that.

John. You say you'd like us all to answer?

Moderator. No, no, anybody who wants to answer.... We don't have to have everybody answer.

John. Well, okay, I'll give you the answer first of all. Certainly, METAPOST is one of the most visible things I've done. I'd sort of not like it to be the highlight of my scientific career, but all I can say is, sure, it was a great experience and I'm happy to have worked on it, and I'm not surprised that it can somehow overshadow a little bit of the other stuff.

Tom. Well, let's see. Coming and working on the TEX project for me was absolutely changing, because TeXas A&M ... I was a bit of a cowboy, I wasn't really — I wasn't CS, I was EE. I was always pushing electronics, not bits. But I enjoyed programming a lot, and I learned a lot myself, and all that, but coming to Stanford and being with some of these people really taught me a lot about how to program correctly, and the importance of literate programming, and that sort of thing. So, absolutely critical. As far as what I'm doing now — I'm a web programmer nowadays. I've got a little startup down in Santa Clara. We write huge enterprise applications for Fortune 500 companies.

As far as legacy, you know, I could not ask for anything better than to be associated with this group of people and this project. So I have absolutely no problem with this being "the big thing".

Moderator. Anybody else have a comment? Not required. Luis?

Luis. It essentially sent me in a direction in life that I had never expected to be. I was a student of Don's, and I was going to do something academically "tainted". [laughter] And I ended up in industry. And essentially, the one thing I think I have the most ... what I recall my time at Stanford as part of the project is the quality of what was being done. The point I was never able to achieve in industry, because I want into the printing industry, who were creating laser printers and doing some architecture for that. You can never do it; you can never go back to that level of excellence and I have to thank all the people here and of course Don, for that.

Howard. I was going to say something. I'm think that I'm very proud to be associated with this because there's the comparison between things like Scribe and things like TEX when the kind of utilitarian thing that was easy to use and then TEX appealed to me because of the beauty part of it, which I mentioned earlier. And I'm proud to have been part of something that brought a lot of beauty to texts for many, many years, that everybody has been producing. I think the world would have been a much uglier place if we hadn't done what we had done, but especially, of course, Don, to do this. I like the fact that many, many people have run a piece of code that I have written because of this project; so that's the legacy part there.

Now I work on Google maps, and wrote business ranking code, which maybe is starting to exceed the number of instructions executed.

Joe. Yeah, I guess just thinking about that, what I enjoyed the most — it's always been sort of part of what I like to do — is to see something that I like and make it better by adding features or doing things that I would like to see in my project. And I think that all of us at this table have done that. So we were each able, in our own way, to contribute to the TEX project, and just thinking of it in the larger sense, this is what Don has done for this whole community, is to take what he needed and what he wanted to see for his books and then make a really big software project out of that, that had a lasting [effect]. So I think we're all proud to be part of it in that sense.

Moderator. I think we're almost at time out, and I'd like to make an observation. I joined the TEX community, I don't know, ten years ago, roughly, something like that. And at the time, it seemed to me that there was some depression I sensed. You know, things weren't changing; things were getting . . . . Of course, a lot of development was going on all the time, but today, at this meeting, I have a sense that it's a *very* active development community. People are excited about things, and I'm just so impressed that something that started thirty-two years ago was done in a way that enabled that group of people to pass it on to another set of people, that group of people to pass it on to yet *another* set of people — I'm not sure what generation we're on now. Surely there's some people who have been involved more or less the whole time. And today it remains a vibrant community trying to achieve the beauty that this group of people set out to achieve.

With that, I think we should call an end to this session. The panelists — I'd like to thank you all for both your participation today and for everything you've done for us, that led us to be here today.

[applause]

From left: Luis Trabb-Pardo, Michael Plass, Tom Rokicki, John Hobby, David Fuchs, Don Knuth, Howard Trickey, Oren Patashnik, Joe Weening, Frank Liang.