Dick Koch

Dick Koch is the creator and lead developer of TeXShop, a renowned front end for TEX on Mac OS X. [Interview completed 12 July 2007.]

Dave Walden, interviewer: Please tell me a bit about your personal history independent of TEX.

Dick Koch, interviewee: I grew up in Haven, Kansas, population 1000. The land there is so flat that you can see all the way to the grain elevator in the next town. I got a scholarship to Harvard because the admission folks thought that Kansas is esoteric, sort of like Kazakhstan. One or two of us might liven up the place. When I left for college, my dad arranged for the Sante Fe train to stop next to a wheat field; we waited in the darkness at midnight with only lights from distant farms visible miles away, until the train appeared and I left Kansas behind.

My first Harvard adviser was in the building housing the original Mark 1 computer, still there in 1957. Harvard said it was the first real computer. After getting a PhD in mathematics from Princeton, I worked at the University of Pennsylvania across the street from the Moore School of Engineering, where the Eniac was made. Penn said it was the first real computer.

I've been teaching mathematics at the University of Oregon since 1966 in the oldest building on campus, Deady Hall. The citizens of Eugene guaranteed that it would last 1000 years when they gave it to the state. It will.

I play harpsichord on an instrument built as a kit, and your readers are very lucky that they will never have to listen.

DW: When and how did you first get involved with TEX?

DK: At Harvard in 1957, I went to a couple of free lectures on programming before deciding to attend a lecture by Robert Frost instead. But I saved the programming handout packet and filed it away. Years later I rediscovered the packet, which explained how to read a paper tape on a Univac by writing assembly code.

No more computing for me until 1978 when I got a Radio Shack TRS-80 and was hooked. In 1984 I got one of the first Macs. But it couldn't be programmed, and two months of MacPaint is enough. Still, I pulled it out of the closet when visitors arrived and explained that it represented the future of computing. In the years after 1986 I learned to program the Mac, even working during a nine month sabbatical at Tektronix as one of three Mac folks in their printer division.

A colleague, Ken Ross, was secretary of the MAA and got a personal demonstration of TEX in Donald Knuth's office shortly after the program was introduced. Knuth told him that a high quality laser printer would cost $100,000, but an adequate printer for drafts might be had for $50,000. Ross reported back to us that we could ignore TEX.

In 1989 I got a NeXT machine. By that time I was old enough to deserve a toy, and

it was a NeXT rather than a Ferrari. The idea was to demonstrate it to my friends whose envy would be unbounded. But in practice I never had a successful demo. Mac owners said "it's just like a Mac and why did they put the scroll bars on the left?" Unix people opened a gigantic Terminal window on the screen, typed a few lines, and complained that it didn't have the latest version of curses.

So I gave up on demos. But gradually I discovered that Unix and a GUI make a powerful combination, and that NeXT provided an understated but wonderful working environment.

In the years when it was marketed for academics, the NeXT came with a free TEX including a previewer named TeXview by Tomas Rokicki himself. After a few weeks, I was a TEX fanatic. The NeXT didn't have much software, but for a mathematician it was wonderful. Mathematica and TEX — who could ask for more?

Although I still worked on Macs, my mathematical work was done on the NeXT from 1990 on, and I celebrated wildly on December 20, 1996 when Apple bought NeXT. This meant a free TEX on the Mac. The command line programs would port since "it's just Unix", and Rokicki's source code was available. But then Apple dropped display PostScript and it looked like rough sledding ahead for TEX on OS X.

DW: I'm not sure what you mean when you say "dropped display PostScript." Please say another word about that.

DK: NeXT machines used Display PostScript to draw to the screen. This brought many wonderful consequences: rather than a discrete drawing palette $Z \times Z$, the screen looked like $R \times R$ to a programmer, and its symmetry group was the full affine group rather than just rotations and reflections by multiples of 90 degrees. Adobe Type 1 fonts were used for text on the screen, and any drawing or text could be rotated arbitrarily on the screen.

Another advantage was that the same PostScript language drew to the screen and the printer, so anomalies when printing vanished. NeXT's printer was quite inexpensive because it didn't need PostScript — the computer rasterized images for it.

At the first developer conference after Apple bought NeXT, the company revealed plans to base Mac OS X on NEXTSTEP. In particular, the Mac would use Display PostScript as an imaging model, and adopt NeXT's object oriented programming system that was based on Objective C and a class library later named Cocoa,

But Apple had a difficult time convincing programmers to adopt this model. All of the major developers like Microsoft and Adobe were noncommittal. So a year later at the next developer conference, Apple's plans changed considerably. Instead of Cocoa, the company announced and pushed the Carbon programming model, which was essentially the old Mac OS 9 programming system modified in small ways to support Unix and its multitasking environment. Although they didn't make a big deal of it, Apple dropped Display PostScript in favor of a new graphic environment to be developed based on PDF. These major changes postponed the introduction of Mac OS X by a year or more.

The switch from Cocoa to Carbon was particularly significant. I wrote several NeXT programs, so I knew that Cocoa was a jewel. In contrast, Apple's system 9 had become more and more baroque and Carbon didn't look much better. About this time I got a chance to go to a developer conference. Great masses of programmers attended Carbon sections, but the Cocoa sessions, sparsely attended, were run by former NeXT engineers who seemed to be hanging on by the skin of their teeth.

I believe the success of TeXShop is partly due to my one programming advantage: I knew from the beginning that Cocoa would be the future of the Mac. It took many years for the majority of Mac programmers to come to this conclusion.

Let me come back to that new graphic system, now called Quartz. At the time, the announcement of this system was a great disappointment because it meant that Apple would not use Display PostScript. But it gradually became clear to me that Quartz was as powerful as Display PostScript, and its introduction meant that PDF would be fundamental for Mac OS X, easy to create, and trivial to display.

DW: You said "it looked like rough sledding ahead for TEX on OS X," but you are well known for TeXShop (`http://www.uoregon.edu/~koch/texshop/texshop.html`), so obviously you either decided to take things into your own hands or somehow got roped into taking things into your own hands. Please tell us about how that came about.

DK: I have spent a lot of time on NeXT and Cocoa because it is intimately connected to the history of TeXShop.

By the early 1990's I had come to depend on four NeXT programs: TEX, Mathematica, Mail, and a Web browser. Although I didn't mention it earlier, I was writing a certain amount of Macintosh software and used a Mac regularly for graphics and other tasks.

The sale to Apple meant that these two worlds could come together. I was so excited by the prospect that I switched to Mac OS X while it was still in deep beta mode, two or three years before the actual release. I only needed Mail, which was part of OS X, a Web browser, but Omniweb soon appeared on OS X, Mathematica, but Wolfram was one of the few developers committed to OS X, Cocoa, but that existed even if it wasn't promoted, and TEX. In short, I'd be in heaven as soon as TEX made it to the beta OS X.

After only a couple of months on OS X beta, I discovered a TEX distribution for it. This was pre-Gerben Wierda, and I have forgotten the details. Perhaps it went something like this: someone ported TEX to the PowerPC version of NeXTStep, producing a NeXT install package; I tried to use this package on OS X but it wouldn't open in Apple's installer; then I discovered that the package could be opened as a directory containing a pax file. I had never heard of pax files, but man pages told me that they were sort of like tar files, so I used a Unix command and surprisingly got a working TEX distribution. Something like that!

The trouble with this working TEX is that it produced DVI files and I had no DVI previewer. So no cigar. But then, quite by accident, I discovered that one of the command line programs installed was pdfTEX, and pdfTEX output pdf files rather than DVI files. This was the single most important discovery of my entire TEX adventure because PDF is the native graphic format on OS X. So instantly I could preview TEX files produced by the TEX distribution. The inventor of pdfTEX, Hàn Thế Thành, became my hero. To be sure, it was a shock to realize that the current version of his software was 0.14! But gradually I realized that version 0.14 worked fine.

For a while, I edited files with Apple's Edit, typeset from the command line, and previewed using Apple's Preview. It was awkward, but it sort of worked, and showed that an adequate TEX must be just around the corner (modulo a serious glitch I'll explain in a moment).

So by a miracle I had TEX and only needed a front end. In the next years as I worked on TeXShop, Gerben Wierda appeared and produced an honest TEX distribution designed specifically for OS X. At the time I didn't know how important Gerben would become, but it was certainly a relief to have the real thing.

At first I believed that a TEX front end would appear soon after his work. I kept asking my Apple representative, who assured me that Apple understood the importance of TEX. But nothing seemed to materialize.

Then I had a chance in 2000 to go to WWDC, the Apple developer conference. I was

still teaching at the University, but my assignment was Discrete Mathematics, a required course for computer science majors, so that seemed to justify the trip (!!). Sitting in the sparsely attended sessions on Cocoa, I began to realize that it might not be difficult to do a TEX front end myself.

After I was back from the conference and school ended, I began experimenting. For a front end, I'd need an editing window, a preview window, and a typesetting command which connected the two. A third party editor could be used, but I couldn't figure out how to get that third party editor to call TEX and typeset. On the other hand, making an edit window is trivial with Cocoa, and adding a typesetting button to my own program was easy. So TeXShop got an edit window because I was lazy.

To call TEX from my Cocoa program, I needed an appropriate Cocoa command. About a half hour's work of thumbing through the Cocoa documentation led to the `NSTask` class, which makes it trivial to call Unix programs from within a Cocoa program. An hour later I was typesetting.

Finally I needed to display the result. Everything depended on Cocoa's facilities to display PDF files. Could Cocoa open a PDF and tell me the number of pages? Yes. Could it be instructed to display page 15? Yes. Indeed, a little reading showed that all of the necessary commands were present in the three or four pages describing classes to display PDF. After that, finishing the first version of TeXShop came quickly.

A short time later, I released the program under the GNU free software license. I don't know the date of the first release, but my guess is that it was in the summer of 2000, when the essentially free "public beta" of OS X was available, but before the first official release version.

However, there was one "minor" problem: Apple's PDF display routines could not interpret fonts embedded in a pdf file. They only worked if the font was one of the standard OS X fonts installed by default. Thus a TEX document typeset fine with TeXShop, with only the following mild restrictions:

1. main text must be Times Roman
2. no mathematical symbols allowed

As a stopgap, I added an alternative preview mode to TeXShop. Rather than using Cocoa to display the output PDF, it could call Ghostscript to rasterize this PDF file to a bitmap and then display the bitmap. This worked, but the result was much slower and much fuzzier than direct Cocoa display of the PDF. Slowly, a few users on the network began typesetting with TeXShop in that "fuzzy mode".

In March of 2001, Apple released the first official version of OS X. I now know that this release was called "Cheetah", although I began learning cat names quite a lot later. To the annoyance of developers, there was no prerelease version to test until a couple of weeks before the official release. I recall my excitement when I realized that Cheetah could display embedded fonts — suddenly TeXShop worked. There was just time before the official Cheetah release to rip out that alternative preview mode.

DW: I see from your university web site (`http://cc.uoregon.edu/cnews/summer2002/koch.html`) that Apple gave you an award for TeXShop. Do you think TeXShop (and perhaps XETEX) are increasing the popularity of the Mac for TEX use, particularly among mathematicians?

DK: No.

Sometime between 1990 and 1995, TEX captured the mathematical world completely. It was a revolution. Before that revolution, mathematicians wrote in longhand and departments had a platoon of mathematical typists; after the revolution the typists were

gone and virtually all mathematicians used TEX. I watched that revolution at the University of Oregon. At first our new hires all used TEX, sometimes on the Mac and sometimes on Linux machines. Soon the older faculty switched. At about the same time, graduate students switched to TEX for everything from quizzes to conference talks.

The last people to switch were older faculty. Often they made the change when they began a book project. But even here the revolution was complete. I helped some of these faculty buy a new machine and answered questions about "cutting and pasting" for weeks. I thought to myself "if even the fundamentals of computing cause this much trouble, it will be hopeless to get this guy to learn TEX." But after a few days explaining TeXShop or another program, I found that mathematicians take to TEX like ducks to water. It speaks their language. I got lots of computer support questions, but essentially never about TEX.

My first TUG meetings were a great surprise — listening to talks about "does TEX have a future?" I gradually learned that switching to TEX in other fields is controversial, and dealing with academic journals can be difficult. All of these problems seem foreign to a mathematician.

I am proud that TeXShop (and TEX and many other related programs) are free; this makes it easier to introduce mathematical undergraduates to TEX. At Oregon, we have a small center for undergraduates; students writing honors theses tend to pick up TEX from older undergraduate peers. The standard question "how do I get TEX on my own machine?" has easy answers on Windows, Linux, and the Mac.

Over the years I've had interesting interactions with users outside mathematics. Many developments, certainly including XETEX and ConTEXt, make TEX attractive to these users. From my vantage point, the TEX world is very dynamic and active.

DW: You have had many collaborators; they are listed in TeXShop's "About Panel" and elsewhere. Please tell me about these collaborations.

DK: TeXShop is provided with source code, so users proficient in Cocoa can easily add features. Over the years, many of these features were sent back to me and integrated into the program. This includes many of the "power user" features because I am not such a user myself. By power user features I mean things like auto completion, command completion, macros, and AppleScript support.

This collaboration was done very informally; I'm sure it has been frustrating for some contributors. The main attribute contributors need is persistence, because I often ignore email when I'm working on something else and "no answer" doesn't mean "I don't want it." Let me tell a story to illustrate that point.

Geoffroy Lenglin was a master's degree student in Aeronautical Engineering at MIT. He sent a "LATEX Panel" which lists mathematical symbols; clicking on a symbol inserted the corresponding TEX code in the source file. I thought the panel was important as soon as I saw it and sent Lenglin some encouraging words. But I was teaching at the time, so nothing happened. Common courtesy would require sending an explanation of the delay, but no such message was sent. Maybe six months later, spring break arrived and I got around to integrating Lenglin's code into the program. When I wrote Lenglin asking if I could list his name in the contributor list, he wrote back "thank goodness you wrote; I finished my degree and am leaving for a job in France, and this email address will be active for four more days."

I will not list numerous other such cases out of embarrassment. If some of your readers contributed something and got no response, they should write now. I will specifically mention another two or three participants in this informal collaboration.

Nicolas Ojeda Bar was a high school student in Argentina who sent code to preserve source code tab indents. When TeXShop won an Apple Design Award, I converted the computer equipment into money and distributed funds to the contributors. Bar got a small amount, and asked for an Amazon gift account due to Argentina's financial crises at the time. I was delighted to learn that he bought theoretical mathematics books!

Mitsuhiro Shishikura is an important theoretical mathematician in Japan. His collaboration is a model for the type of interaction I prefer. Shishikura revised the PDF display code to permit continuous scrolling through pages, side-by-side display, and many other features. He added a magnifying glass, which I wanted desperately. When the Tiger operating system from Apple appeared, much of Shishikura's code could be replaced by code from Apple's PDFKit framework, but that framework was virtually isomorphic to Shishikura's approach, a great compliment to him. The vital magnifying glass code still comes from Shishikura.

When Shishikura made his modifications, he sent a revised code base so I could immediately experiment with his modifications. The modifications were carefully set off with comments, and I could pick and choose what to use and what to avoid. Shishikura sent a detailed document listing exactly what he had changed and why. This made it very easy to add his modifications with certainty that I understood what he had done. Thanks, Mitsuhiro.

It is easy to work with collaborators when they make one addition, possibly in several different files, and clearly explain the change. A few programmers work globally, making many changes at once and even changing the program interface in dramatic ways... In these cases I have to either accept or reject the entire change because it isn't possible to pick and choose. I've found it impossible to deal with that approach. Some of these proposed changes were probably excellent, but I couldn't digest them. If I adopted the change, I'd no longer understand the code.

It isn't my goal to be fair here by listing colleagues in order of importance. But I must list one recent worker, Max Horn, who has had to suffer through long email silences just like everyone else. Horn rewrote large sections of TeXShop, cleaning up the code dramatically, and improving the speed of the program in several important ways. But miraculously, I still understand his code; and making changes is as easy as it ever was. Horn managed to preserve all of the important method calls and procedures. It is sort of mysterious that he could revise so much and still make me happy. The current version of TeXShop uses Horn's revisions.

DW: I looked you up in the Mathematical Genealogy Project (`http://genealogy.math.ndsu.nodak.edu`) database and see that you are descended only a few generations from David Hilbert! This brought a question to mind. Do you think the availability of high quality mathematical typesetting has significantly changed mathematics or how it is done? Would someone like Bernoulli have been able to write more papers if he had TEX available to him, or was he better off without it?

DK: My previous comments apply here. I don't think that Bernoulli would have written more, and to use the extreme case, I don't think Euler would have written more. But I think they would have have seen TEX as a completely natural tool.

DW: There is a lot of talk these days about the new version of Word having better math typesetting capabilities. Do you think TEX will still continue to hold its own (minor) position in the mathematical-document preparation market.

DK: My department has one Word user, and the rest of us use TEX. So I'm afraid the question doesn't interest me.

Incidentally, nobody in my department promotes TEX. This would be considered unseemly. They just use it, on various operating systems, and get on with their mathematical lives.

DW: To what extent, if any, is there coordination of the TeXShop effort with the XETEX effort?

DK: I think Jonathan's work is very, very exciting. But so far there has been just a little coordination. I added the "engine" architecture and the ability to set a file's encoding using a line at the top of the source code when I saw the harder instructions Jonathan provided. Later I added a template for XELATEX to encourage its use, and Jonathan criticized my initial attempt and provided a reasonable template.

I should add that Jonathan went to WWDC this year and I saw him in several sessions. He is always bubbling with ideas, and some of these ideas will make it into TeXShop. So in the future . . . The trouble is that I can absorb about one idea a day and Jonathan can provide two or three an hour.

DW: Does your TEX work get support, or at least encouragement, from your university? You obviously have done lots of development related to TEX; how do you fit that in with your academic responsibilities and ambitions?

DK: The department has been great. For a number of years I was head undergraduate adviser and spent a lot of time working with undergraduates, so my TEX work may have seemed an extension of this sort of "academic service work". I understand the thrust of your question, which can be a real problem for younger faculty, but it hasn't been a problem for me.

DW: I have one last question. You have mentioned Mathematica several times, but I'm not sure what it does. How is Mathematica used in combination with TEX by a mathematician?

DK: Mathematica is a symbolic mathematics program, able to perform algebraic manipulations, integrate symbolically, solve differential equations, and graph in two and three dimensions. It is one of a small number of such programs; others include Maple and Macsyma.

I often use Mathematica to create TEX illustrations. But it really is a complementary tool: TEX for writing mathematics, Mathematica for symbolic calculations, and the blackboard for serious work!

DW: Thank you very much for participating in this interview. Even though I am not an OS X user myself, I can tell by the buzz in the air that there is a lot of excitement among TEX users about TEX and OS X and that your contributions are very much part of that excitement.