CHM Computer History Museum

Interview of Butler Lampson

Interviewed by:
David C. Brock

Recorded October 29, 2018
Cambridge, MA

CHM Reference number: X8828.2019

Brock: To really kind of try and dig into that, first with a discussion with Charles Simonyi, to talk to him about-- to talk to him about Bravo and then the transition to Microsoft Word. And, you know, trying to get some more detail about things that, I think, in the literature, as it stands today, you know, had been a little hand waving. And I had a nice interview session with him, where, probably very unsurprisingly to you, but surprisingly to me, I learned about the whole story about piece tables in that, as the kind of fundamental data structure. And Charles was very eloquent about its importance, and how adopting the piece table approach, for Microsoft Word, was very kind of central to the story of its development and, I suppose, its success.

In the context of that conversation, he mentioned J Strother Moore, and, you know, his work on an editor, and him being at PARC, and that J's editor embodied kind of a very similar data structure, or the same data structure. And so I was able to contact him, J, and I did an interview with him, and talked to him about where his editor came from, and his involvement in artificial intelligence, and how his development of what became called the piece table related to his work in automatic theorem proving-- you know, sort of how he came to the idea. And then, in-- and very interesting story. And in kind of checking in with Bob Sproull about this whole matter, he mentioned to me that, well, of course Charles Simonyi talked about, of course, your place in the Bravo story. But, then, Bob Sproull was also telling me that this was a case of independent, near-simultaneous invention, my words.

Lampson: Right.

Brock: And I thought that was just absolutely fascinating. So I read some more, and I read some of the things that you wrote. Then I found out about the QED editor, and so I've been just kind of washing around, reading what I could about these editors. So what I would love to do, if it's okay with you, is to make a recording of this discussion that we can-- that then I could have the museum transcribe, and you could look at it, and hopefully we could eventually, you know, put it into our collection, as a resource for other people, beyond me. But I would love to talk with you about-- oh, just this kind of larger story about text editors and word processors, and, you know, the relation of them to timesharing computing, and your experience with QED, and then how that came into Bravo, and the whole piece tables development, because I think it's very fascinating.

Lampson: Okay, sure.

Brock: Great. Well, I wrote down some questions for myself and maybe if it's okay to you, I'll just start proposing them to you.

Lampson: Sure.

Brock: I wondered if your experience of-- well, how much of an experience of computing did you have as an undergraduate at Harvard? Did that come into your picture very strongly at that time?

Lampson: Well, I first got involved with computers when I was in high school, because one of my classmates found a IBM 650 at Princeton that was not being used very much, and he sweet-talked the woman who was in charge of it into letting us play with it.

Brock: And so, at that time, the 650 would have been very much kind of composing your code on paper and then putting it into card decks, and that sort of a-

Lampson: That’s right.

Brock: Yeah.

Lampson: But of course, when you ran it, you got to sit at the console of the 650. So it was very interactive, in a weird sort of way.

Brock: Well, when did-- so, at that time, while you were-- so, and your association with computers then continued as an undergraduate?

Lampson: It did.

Brock: Yes.

Lampson: There was not any-- very much in the form of formal computer science education at Harvard at the time, but Gerry Salton [Gerard Salton] was there, as one of the pioneers of information retrieval. And I guess, in my junior year, Ken Iverson [Kenneth Iverson], who invented APL, came on sabbatical from IBM, and gave a course on APL, which was pretty interesting, because at that time there was no implementation. Right? It was purely a notation. And, in fact, Ken was strongly opposed to the idea of an implementation, because he was sure that the implementors would have to make so many compromises, that his beautiful language would be wrecked. And he was extremely fortunate, three or four years later, to run across Adin Falkoff and his colleagues, who figured out how to implement APL without making any compromises. So I took his course. And I also fell in with a professor in the physics department who had been taking spark chamber photographs, and wanted to use them. That PDP-1 that the physics department had, to analyze these photographs-- and I did a lot of programming for him. That was my first real experience with interactive computing.

Brock: So this would have been the time that some of the very first text editors, as I understand it, were being developed at MIT for the PDP-1, just around this time. Did you-

Lampson: Yep, it was Expensive Typewriter. That was what the MIT guys called it.

Brock: And did you use that for this spark chamber effort?

Lampson: I don't remember exactly, but I definitely remember playing around quite a bit with an interactive editor that used the PDP-1's somewhat feeble graphics display. This was a display in which you had to turn the dots on, one by one, by executing a pair of IO instructions for each dot. This was before the days when you could tell the hardware to automatically display a whole character.

Brock: Right. And that would have been-- that sort of an editor, at that time, that would have been primarily as a tool for creating software for writing code?

Lampson: Right.

Brock: Okay. Well, I was very interested to see that, in the development of these early text editors, that I could-- that I have been able to kind of read about to date, you know, they really seemed to emerge in this MIT, Lincoln Labs, MITRE, BBN, System Development Corporation sort of scene. And I wondered if there, you know, that the same sort of scene in which the movement toward timesharing computing was really starting to gather steam, and I wondered if there was a particular relationship between text editors and timesharing.

Lampson: Of course.

Brock: Could you tell me a little bit about how you see that connection?

Lampson: Well, if you're going to do interactive computing, you need a text editor. There's just no way around it. If you're doing batch computing, you don't need a text editor, because you can use the key punch. Although, as soon as batch computing got a little bit more advanced, and they started compressing the card decks, then you needed the equivalent of a text editor, in the form of a bunch of cards that you appended to your card deck, to edit it. To produce the next version. So there was just-- there was one called Crunch, and there was another one whose name I can't remember. I think it began with a B. I used both of those at MIT, when I was an undergraduate at Harvard. So those-- you didn't-- people didn't really think of those as text editors, because there was nothing interactive about it. You submitted the last version of your source deck together with a bunch of cards that said, "Insert after line seven, the following three cards," and things like that. But the commands were recognize-- not too surprisingly, recognizably the same as the ones that we used in systems like QED.

Brock: So that would-- well, that's very interesting, because one thing that really-- just to jump ahead, one thing that I thought was very relevant, or at least seemed to me like possibly relevant to the piece table development was-- there's a section in the paper that you did with Peter Deutsch on QED that talks about re-editing, which was the idea of-- you would have kind of a single copy of the original text, and several other text files that represented QED commands, or edits, to that original. So you could have various versions. And it's the idea of kind of treating the edits and manipulations, you know, separate from-- while kind of leaving the original in place, and just treating the edits as a file. It almost sounds, from what you were bringing up about the punch card decks, and these programs that you worked with at MIT as an undergraduate, that in some ways, you know, those final cards in the deck were, you know, cards of the edits. There seems to be a similarity there.

Lampson: I guess so. I don't remember thinking about it that way at all.

Brock: Okay. Well, if the-- well, perhaps, then, we could talk about-- well, it also seems that there was kind of an extension of text editors, to create what one might-

Lampson: But no, just to wrap up-

Brock: Oh, I'm sorry-

Lampson: What we were talking about before, you didn't need a text editor for batch processing, for the reasons we just discussed. But for timesharing, there were no card decks. The stuff was-- the text lived in files on the timesharing system. And so you had to have an editor to edit it. So it was not accidental at all.

Brock: Right. That it was essential to-

Lampson: It was absolutely essential, right.

Brock: It was essential to the scheme. Yeah. And it does seem that some of these text editors, like Colossal Typewriter, Expensive Typewriter, and then-- I'm not quite sure how it's property pronounced, TECO, T-E-C-O, this editor for the PDP-1, these were kind of a mixture of the two. Let's say if-- an interactive timesharing editor and this kind of batch mode of editing, which was, you know, the idiom would be the actual manipulation of paper tape. So, as I understand, TECO, originally, was-- you would load the original tape, load the-- or I guess you would load TECO, load the original text, and then load all of the TECO edit instructions, and then output your new tape. So that sounds kind of like-- more like-- but you could sometimes run it kind of online in the machine. It seems like kind of a transitional, or hybrid, mode.

Lampson: I think that's right. I mean, again, this was driven by the pragmatics of storage on the system. These PDP-1-based systems didn't have file systems. They had no disk. So the only way that-- and they typically didn't have card readers, either. So you had to be able to edit the paper tape. And that completely drove the design.

Brock: Right, right, because the physical manipulation of the tape, with splicing and all this sort of thing, was just-

Lampson: That's not practical.

Brock: Completely heinous, it seems.

Lampson: That's not-- that's never been practical. That's right. So, again, because it was paper tape and not cards, you had to have something that resembled an editor. And-- because the systems were actually interactive. But they were never run in batch mode. It made sense that the way the editor worked is that you sat there and typed in commands, and things happened.

Brock: That makes perfect sense. It-- then it-- well, I was interested, around this same time, we begin to see, you know, by '64-ish, we begin to see the appearance of programs that are sometimes called-- I've seen them called formatters, formatting programs that would essentially take these-- take the output of these text editors, which were, you know, primarily designed to create program listings, to-- the formatting programs could kind of make what one might call literature, you know? Instead of programs.

Lampson: Yeah, I think the first of these that I'm aware of was Jerry Saltzer's RUNOFF system.

Brock: Yes, that's-

Lampson: Which I believe he wrote so that he could write his thesis.

Brock: But I thought it was-- and do you think it's a correct characterization of that-- as kind of a augmentation, or a translator from, you know, a tool that was for writing programs, to make a tool for writing documents?

Lampson: Well, no, I think that people thought of the editor and the RUNOFF program as being very different things. They thought of the RUNOFF program as being like a compiler.

Brock: Oh.

Lampson: And, in fact, some of those systems were actually called "text compilers," or "document compilers." I'm pretty sure. I don't have a crisp memory which ones those were, but I think-- I don't think there's any doubt that people thought of all those RUNOFF programs as being like compilers, and not as being closely related to editors.

Brock: Okay, that's really interesting.

Lampson: The first system that I know of-- I think maybe Bravo was the first system to combine those two pieces of functionality.

Brock: Right, right.

Lampson: I'm not certain about that, but I think that's true.

Brock: No, it very well may be, because-- yeah. I mean, I definitely-- in my reading, to date, I have seen that these formatting programs like RUNOFF were very much thought of as separate from the editor. You know, I was just thinking of them as a-- what would I say-- kind of like an extension of it. A program that could, you know, take a tool from-- use the results of a tool for making software, and take those results and translate them into a document, a literary object, if you would. But thinking of them-- you know, them being framed as compilers is very interesting because-

Lampson: Yeah, here it is. PUB. Larry Tesler's PUB system, that I'm sure you know about-

Brock: Yes.

Lampson: It says right here, "PUB, the document compiler."

Brock: Right. Oh, I hadn't realized that they were-- that RUNOFF was also thought of in that way.

Lampson: Well, I'm sure-- it wasn't called-- I don't think RUNOFF was called a compiler, but I'm sure it was thought of that way.

Brock: Yeah, which is interesting, because it also makes perfect sense, because of the programming language aspect that many of these early text editors had, like TECO.

Lampson: Yes. Well, that-- I think that that is really from a different space.

Brock: Okay. How should we understand that?

Lampson: I'm hesitating to answer, because that whole TECO, Emacs, programmable editor line of thought was one that I never participated in, personally. I was certainly aware of it. And it became very clear in the '70s-- people who liked Bravo couldn't stand Emacs, and vice versa. And we did have people at PARC who came from the MIT Emacs culture, and never really liked Bravo that much, because it's a very different way of approaching life. But each one has its strengths and its weaknesses. The TECO Emacs path was a little bit older, because you couldn't make the Bravo thing work unless you had a decent display. And I believe the Alto was the first time that anyone had decent displays in noticeable quantity. Before that, you really were relying on a teletype-like terminal. Which means that the whole idea of thinking of the document formatting program as an editor just isn't viable, because it takes too long for the terminal to generate the finished version.

Brock: Right. Well-

Lampson: I think the first system that-- probably that was anything at all like that was Doug Engelbart's NLS system, where they did have a significant number of displays. And people--

you know, everyone in Doug's group had one. But, of course, Doug was not terribly interested in document formatting.

Brock: Right.

Lampson: And you probably already know this, but Peter Deutsch and I both worked for Doug, off and on.

Brock: Oh, I did not know that.

Lampson: In our spare time, as students.

Brock: Oh, I had-- I actually had no idea. So you had-- that would be at the same time that you were-- that would be on the same time that you were working on the QED editor, or the QED editor was under development? You-- that would have been that same time period?

Lampson: Might have been-- it probably would have been a little bit later than that.

Brock: Later.

Lampson: Probably a year or two later, because it was-- well, I guess when I-- the first time I worked for Doug, he had yet gotten his 940 [SDS 940]. He had a CDC 3100, and I tried to write a SNOBOL system for him, which I believe never got finished, because it was a little bit too big a project for my spare time during the summer.

Brock: Sounds like it. Well-

Lampson: We had done several SNOBOL systems at Berkeley, so I was just transcribing.

Brock: I see.

Lampson: In fact, one of those systems was done by Charles [Charles Simonyi].

Brock: Oh, yes. At Berkeley.

Lampson: Yes.

Brock: Yes. Well, maybe I could-- well, maybe just to follow that line about this kind of division, or the two paths between the kind of TECO Emacs, programmable editor line and the Bravo world, one thing that I realized in getting ready for this call was, I did not know-- I of course know about people using Bravo to create all sorts of documents, obviously. Now-- but what I have to confess that I'm ashamed that I'm ignorant about is if people were also using Bravo to compose programs.

Lampson: Oh, yes, that-- Bravo was the editor.

Brock: Okay, it was. All right. I just-- because so much prominence is-

Lampson: There was another-

Brock: Yeah?

Lampson: There was another editor floating around that Bill Duvall built, that was-- Bill had been working for Doug at SRI, and when he came to PARC, he actually built a sort of extremely stripped-down version of Doug's system. But not very many people used it. It was really hard to compete with Bravo.

Brock: I would imagine so, because I had seen, in a couple different sources that I was looking at, you know, it was this idea that the very tools that we associate today as being absolute hallmarks of word processors, as many people are familiar with them, you know, were orig-- in their original context of development and integration into text editors, these were seen as features and capabilities that would make the programmer's job so much easier, and enhance their capabilities, you know? Some of the features that, you know, that were implemented in QED, for example-- very familiar things for a user of a word processor, but all of the example-- what I thought was so great about reading the article was, all of these examples were how these features, like search, or, you know, kind of a find and replace sort of function, were, you know, wonderful affordances for the software maker and the code writer.

Lampson: Yeah, well that was probably the most important application for QED, although it was certainly also used for preparing documents.

Brock: Oh, it was?

Lampson: Oh, yeah. I mean, we tried to do all our work on the machine, and QED was the only editor.

Brock: Okay. Well, that makes sense. But to me, do you think that is a true kind of statement, that these features that make these programs good for creating documents, you know, were initially developed because they were primarily, you know, helpful features and capabilities for making the programmer's life easier?

Lampson: I'm not sure I would say that. I think we were aware, from the beginning, that you could use the text files that the system supported for programs, and you could also use them for documents.

Brock: Right.

Lampson: And I don't think we made any sharp distinctions.

Brock: Okay, perhaps-- okay, right. Because obviously, you were using it for both, so both of those uses were in mind. And it just-

Lampson: Right, I don't think we made any sharp distinctions.

Brock: Yeah, because I guess there really aren't any, you know? It's-- in some ways, there's an interesting dimension to the whole story, where writing is writing, you know? Whether it's, you know-- whether that string of symbols, you know, is a memorandum or is a sub-routine, it's writing.

Lampson: Yeah, well, of course, later on, the distinctions became much-- the two paths definitely diverged, if you look at the way program editing is done in modern, interactive development environments, it's extremely specialized to the programs. And there's all kinds of stuff that wouldn't be at all useful for documents.

Brock: Right, like a debugging window for your essay, or something-

Lampson: But, more than that, there's highlighting of syntax, and there's some auto-commands for automatically renaming things, and ways to jump around in different parts of the program, and the so-called intelligent features that will pop up the parameters for a function when you just type the function name, and just hundreds of things, which were things that wouldn't really have been feasible on the machines that we had in the late '60s.

Brock: Right. And so, in a way, you know, perhaps it's a story of eventual divergence between-

Lampson: Yes.

Brock: You know, one's-- tools aimed toward creating literature, and tools aimed toward creating programs.

Lampson: Yes, and I think, while that's the trajectory that was successful, it is also true that, at PARC, a lot of programmers used the formatting features of Bravo very heavily to make their programs look prettier. And, I think, eventually, after a number of years, people decided that that was-- that game was not really worth the candle. That it was extremely tempting, and a lot of time and energy was spent on using the formatting features of Bravo to make the programs look good when they were printed, or displayed on the screen.

Brock: And, literally, you know, this aesthetic quality?

Lampson: Oh, yes.

Brock: That it looked like a lovely, laid-out page.

Lampson: Yep, yep, yep. And it has some utility for programming as well, because you want-- nowadays, I think people generally take the view that the way to get that is with a prettyprinter, not by manual editing.

Brock: Yes.

Lampson: But we-- nobody ever did that at PARC, as far as I know. We never wrote any prettyprinters that-- whose output was pretty, in the typesetting sense. But people definitely did that manually.

Brock: And forgive my ignorance about a prettyprinter, because I've only just encountered the term last week, but, to me-- so is this the case, that it would be a program that would sort of understand features of the text as features of computer programs, and then would kind of render it pleasingly and aesthetically, based on its sort of understanding of these-- its interpretation as a program?

Lampson: Yes, that's exactly right.

Brock: Okay, whew.

Lampson: And, by the way, at MIT, the popular versions of this were called grind, for reasons that I believe had to do with the fact that they were fairly slow, although I'm not certain about that. You would have to ask someone like Gerry Sussman or Tom Knight, what the origin of that is.

Brock: Thank you.

Lampson: They didn't call them pretty grinders, they called them grinders.

Brock: I'm just taking some notes.

Lampson: And this whole prettyprinter thing actually has been very controversial in programming for many, many decades because there are many programmers that don't trust the prettyprinter to get it right.

Brock: Huh. To not be able to correctly interpret these kind of functional or these functional aspects of the code to be--

Lampson: Right. To not render the program in the way that the programmer thinks it's the clearest and most attractive or some value of attractive.

Brock: <laughs> Are there prettyprinting programs that are then manually editable by--

Lampson: Well, that's the source of enormous conflict. Because yes, of course, typically the way the prettyprinter work is they produce another text file.

Brock: Right.

Lampson: But if you edit it, then you-- if you prettyprint it again, your edits will be lost. I'm saying if you edit it for format and then you prettyprint the text again, which you might well want to because you also made substantive changes, then your edits will be lost. So that doesn't really pay.

Brock: Hmm.

Lampson: But if you're signed up for prettyprinting, then you don't go in for manual editing.

Brock: Right.

Lampson: But it's also possible to operate in another mode and-- I don't know a lot about this. I'm sure there's a bunch of prettyprinters where you can insert commands to the prettyprinter into your program text.

Brock: That's what I was just thinking. Tag it.

Lampson: Tell it how you want things laid out.

Brock: Right. Tag it and say this part is actually this sort of entity or--

Lampson: Or I really want this carriage return right here, no matter what you think.

Brock: <laughs> Right.

Lampson: Well, we never did any of that stuff at PARC as far as I can remember.

Brock: Okay. Well, maybe I could ask you-- thank you. That is extremely helpful. I wondered if I could talk to you about-- well, if we could talk a little bit about QED and just the whole idea of developing your own editor rather than adapting an existing or-- editor and just your experience of that and what, if any, connections there are to your QED experience to the piece table development?

Lampson: I'm not aware of that. What I remember about the piece table development is that I was sitting around at PARC sometime, it must have been early 1973, mid-1973, reflecting on the fact that we really didn't have an editor for the Alto and that we better do something about it.

Brock: Yeah.

Lampson: I was thinking new thoughts because, as I was saying before, most of the existing editors didn't have a display as part of their design space, and it makes a huge difference whether you have a display or not.

Brock: Right.

Lampson: So I was starting more or less-- and certainly, all of our experience with QED was heavily affected by the fact that the IO device was a Model 33 teletype.

Brock: Right.

Lampson: So I was more or less starting from scratch.

Brock: Okay.

Lampson: So I was sitting around in my office thinking about how to go about doing this and I thought of piece tables and the other thing I thought of was the algorithm that Bravo uses for updating the screen. And I had written both of these things down on a couple of pieces of yellow paper. And then, Charles Simonyi wandered by and told me about his software factory, which I'm sure he told you about.

Brock: Yes.

Lampson: And he had just done the first small software factory project, which was called Alpha, and he was looking around for a second project. And I said, "Gee, Charles, why don't you do an editor?"

Brock: <laughs>

Lampson: "And here are my ideas." And that's why it's called Bravo, because it's the next step aftr Alpha.

Brock: <laughs> Well, may I ask you about-- now, I-- Charles did tell me about the-- that you had these two kind of-- well, the data structure-- the piece tables and the kind of fast display algorithm.

Lampson: Mm-hmm.

Brock: And I tried to dig into and find some details about that fast display algorithm. And what I have been able to find to date was describing it in some ways also as relating to the use of a table. That is to say that the-- if I was understanding what I was reading, that treating the screen as these kind of horizontal sections that were maybe edited and controlled through a structure like a table, is that correct?

Lampson: Well, yes and no.

Brock: <laughs>

Lampson: I mean, almost everything in the computer is a table in one sense or another.

Brock: Okay. <laughs>

Lampson: And it's-- on the one hand. On the other hand, it's sort of an accident that a piece table is called a table.

Brock: Okay.

Lampson: It could have been called a piece array or it could have even been a tree.

Brock: Right.

Lampson: So no. I don't think there's much in that.

Brock: Okay. I was just wondering if there was-- if my maybe just simple-mindedly seeing the same term used, I wondered if perhaps there was a-- you-- there was some similarity between the algorithm for handling the fast display on the Alto and--

Lampson: No. They're totally different.

Brock: Okay. Totally different. Just the same word appears. <chuckles>

Lampson: Yeah.

Brock: Okay. And so-- and you had the-- could you describe what occurred to you in the-- well, did you kind of conceive of this whole schema of the piece table on those pieces of paper, that you would have this series of records that would point to the original and you would have this kind of proliferation of records pointing to the original as the editing session went on? Was that all-- how much of it did you conceive originally?

Lampson: I think it was all of a piece.

Brock: Okay.

Lampson: To use a pun.

Brock: <laughs>

Lampson: Didn't actually intend until after it had come out of my mouth.

Brock: <laughs>

Lampson: The basic notion was, okay, we're going to have-- at any given time, the document's going to be a single giant string.

Brock: Yeah.

Lampson: And edits are kind of going to replace one substring of that giant string with another and this-- the obvious way to implement that is to make the string into a big array. And then, just move the characters of the array enough to make room for the replacement. And-- but there's two things that are wrong with that from the point of view of an editor for the Alto. One of them is that, as the document gets bigger, it's going to be too slow.

Brock: Right.

Lampson: And the other is that there's not going to be room for the whole document in memory.

Brock: Right.

Lampson: So both of those considerations mean that that naïve approach is not at all attractive.

Brock: Which is, in a way, I mean, more or less how QED worked. Right? It would be to--

Lampson: Yes.

Brock: --divide the document into pages and you would have just maybe one page in the memory or something? I mean, I was reading about that, but in essence, it was hold the document in memory as to this big string and move it-- move the text around and add text and remove it. So this is-- okay. So I think I-- I'm following.

Lampson: I mean, we did this kluge thing of breaking up the text into pages and leaving some extra space on each page.

Brock: Right.

Lampson: That didn't-- the Alto was not at all-- unlike the [SDS] 940, the Alto is not at all a page-oriented machine.

Brock: Right.

Lampson: And so, we're not-- that didn't seem very attractive.

Brock: And--

Lampson: And then, of course, the other thing that happened later that was quite big that was not part of my original vision at all was this notion of being able to attach formatting operators to the pieces.

Brock: Yes.

Lampson: Which I believe was Larry Tesler's idea.

Brock: Oh. I did not realize that.

Lampson: I'm not certain about that. It was not mine.

Brock: Huh.

Lampson: It was either Charles' or Larry's, and I think it was Larry's.

Brock: I'll ask him about that.

Lampson: And that was very appealing because the idea that you could make the whole document bold by just changing a few bits of the memory is-- was very sexy.

Brock: <laughs> I can imagine. Well-- so it must have been around this time that J Moore came to PARC, however briefly. Did you-- do you remember when you became aware that he had developed a similar scheme?

Lampson: That was not until much later.

Brock: That was much later.

Lampson: That I became aware of it.

Brock: Okay.

Lampson: But I think he transcribed his-- his tables came into Interlisp.

Brock: He did.

Lampson: But I wasn't paying-- I did do some programming in Interlisp, but I wasn't really paying that much attention to it and I was not aware of that until several years later.

Brock: Okay.

Lampson: At least, as far as I can remember.

Brock: Yeah. But to your memory, it was not even something that you were aware of?

Lampson: No.

Brock: Okay. Interesting. Because it's-- in talking to J Moore and as he was kind of developing this editor-- these kind of editor functions in Interlisp, he described kind of having discussions with Charles Simonyi, who had mentioned things like, oh, yeah, this idea of adding the sort of formatting operators to the pieces. And then, Moore would kind of put those-- put that development into his Interlisp editor and play around with that. So it seemed like that they may have been having more of a--

Lampson: But that was later because the formatting operators didn't come along until a year or two later.

Brock: Right. Okay. All right. And-- let me see. Let me just look quickly at my notes here. So how close were you-- once Charles Simonyi took on this-- took on Bravo as a project, how closely did you follow what was happening with his development of it and his work with Tom Malloy?

Lampson: Oh, very closely. Charles and I would talk probably at least a couple of times a week about how things were going and what the latest ideas were and so on.

Brock: Okay. And--

Lampson: And we continued to work together very closely until Charles went off to the development group. And even after that, we would get together pretty often.

Brock: Right. Well-- and so, should we really think of this as a really-- years of ongoing development as the Bravo code base goes into Gypsy and then some of the kind of modeless operation and the copy and paste sorts of operations come from Gypsy back into Bravo. Is that the right way to think of it, as just this kind of years of evolution with different people contributing, some programs emerging as separate entities, but just an ongoing research and development program on editors?

Lampson: Well, yes and no.

Brock: Okay.

Lampson: Yes in the sense of the things that you described. No in the sense that I think we always thought of what was going on as, really, two parallel paths. There was the editing engine path and there was the user interface path. And they were pretty distinct.

Brock: And-- yeah.

Lampson: So for example, when we started Bravo, we made a very deliberate decision that we wouldn't work on the user interface.

Brock: Hmm.

Lampson: Not because we didn't think it was important, but we didn't think we had the resources. So we just kind of hacked together these interfaces in a way that seemed sort of reasonable.

Brock: Okay.

Lampson: And we were extremely fortunate that Tesler and [Tim] Mott came along.

Brock: Right.

Lampson: With their Ginn and Company project.

Brock: Right.

Lampson: Gypsy, which-- their primary interest is in the user interface. So they took-- essentially, they took the Bravo editing engine and completely ripped out its UI and put in the new one.

Brock: And the heart of that editing engine is the piece table data structure?

Lampson: The piece table and the display update.

Brock: And the display update. Okay.

Lampson: And a few other things, like there's quite a lot of hidden machinery for paging both the text and the code. And then, later on, of course, a whole bunch of stuff having to do with fonts and formatting and so on.

Brock: Right.

Lampson: Were all part of the editing engine.

Brock: Okay.

Lampson: But copy and paste were a pure UI construct because Bravo, from the very beginning, of course, had the ability to chop a piece of text out of one place in the document and put it down somewhere else. It's just that the interface was funkier.

Brock: Right.

Lampson: So copy and paste has been enormously influential as a user interface mechanism, but it had no impact at all on the editing engine.

Brock: That's interesting. I had never thought of it or heard it described that way, of these kind of--

Lampson: I mean, it's sort of-- and that goes back to QED, and QED did that by going through these text buffers.

Brock: Right.

Lampson: Which is also the way it was done in TECO and other typewriter-terminal-based editors.

Brock: Yes. And that's where you could either put a string of characters into a buffer that could represent a string of program operations.

Lampson: Of course, we didn't do that in Bravo.

Brock: Right.

Lampson: Yeah. They thought that was very important.

Brock: But in a way, for-- you had some of that in the ability to-- I guess-- well, you still had the kind of-- the-- with the piece tables came the ability to rewind or unwind the string of edits. You could kind of--

Lampson: No. No, no, no. Absolutely no.

Brock: No? That didn't come from the piece table?

Lampson: Bravo only had one level undo.

Brock: But wasn't there-- I mean, there's-- didn't Bravo have this feature where you could do some sort of interrupt and then put it into a replay, where it would step through--

Lampson: Yeah. But that was done a completely different-- that was done by feeding the same commands through the same initial state. That was not done-- had nothing to do with piece tables. That was done at the-- basically at the interface between the user-- between the UI and the editing engine. You would just log all of the commands that were coming from the UI. I think there were actually logged in the form of the characters that you typed, plus some encoding of the mouse position.

Brock: Oh.

Lampson: And the only thing you could do with that was replay it strictly from the beginning.

Brock: Okay. So I had a completely wrong impression of that. I thought that--

Lampson: That would have-- it’s actually quite difficult to make that work. And if you look at how undo works in Word, for example, it doesn't work that way. It works by making-- they make a complete copy of the piece table after every command. Nowadays, there's probably a bunch of hacks to optimize the copy. But conceptually, they copy the piece table after every command, and what that means is that undo is very reliable because you're just putting the state back to where it was before. And that is taking advantage of the fact that the-- in some sense, the piece table is recording the edits. But it's very crude. There's no notion of being able to apply an undo action to the piece table. It's just done by putting back the old piece table.

Brock: So if I were-- so--

Lampson: And we didn't do that in Bravo because we didn't have enough memory.

Brock: So it would be-- if you, for example, wanted to have a two-layer undo, then I would have kind of three copies of the piece table. Right? I would have--

Lampson: Well, Word keeps lots of copies.

Brock: Huh.

Lampson: Bravo's undo was done basically as a hack.

Brock: Okay.

Lampson: And I'm pretty sure that there were commands for which it didn't work because that hack wasn't good enough.

Brock: Huh.

Lampson: And if we wanted to do multi-level undo, the hack would have gotten much hairier.

Brock: And was that--

Lampson: And you probably would have had to rethink everything from scratch.

Brock: Wow.

Lampson: So you could get the effect of multi-level undo, because when you did the replay, you could stop it.

Brock: Right.

Lampson: But that was a different kind of hack.

Brock: And the--

Lampson: And the original motivation for the replay was not undo.

Brock: Well--

Lampson: As you can tell from its name.

Brock: Right. <laughs>

Lampson: Right? Its name is Bravobug and the original motivation for it is so that people could report bugs.

Brock: Oh, so they would just send you that--

Lampson: They would send us the record and then we would replay it ourselves.

Brock: And you would see where the bug occurred.

Lampson: And we would see where it went off the rails.

Brock: Oh my God. So that is kind of a debugger. I mean, that is--

Lampson: Yes. Absolutely.

Brock: Okay.

Lampson: Well, it's not really-- I mean that's-- No one would have called that debugger at the time. A debugger was a program that you typed commands into just to see what the contents of memory of was.

Brock: Right.

Lampson: But yes. It definitely was a debugging tool and as I say, it was called Bravobug.

Brock: I did not know that. That's fascinating. And did-- and the single level undo hack is unrelated to this entirely?

Lampson: Totally unrelated.

Brock: Fascinating. Well, I wondered in your-- so when I was talking to Charles Simonyi, he was very eloquent about how essentially in the creation of Microsoft Word, they drew exactly on what he called the same set of tricks <laughs> that had been at the heart of Bravo, most especially the piece table, to create Microsoft Word.

Lampson: Absolutely. And I was quite startled to find maybe eight or nine years ago now, I was talking to Antoine Leblanc who was running it at the time. Actually, I was talking to him about how they could implement multi-version undo better than they had. And I was a little bit startled to learn that they're still using those data structures.

Brock: That's what I was wondering. You know, I saw that--

Lampson: I don't know whether that's still true today, but it definitely was true 10 years ago.

Brock: Oh my gosh.

Lampson: They're good data structures.

Brock: Yeah.

<laughter>

Brock: Yeah. I had-- that was one thing I wondered if you knew because I had seen at least through Microsoft Word for Windows that it was still based on the piece table, but I didn't know about how long it has persisted. So could very well still be there?

Lampson: Yes, absolutely.

Brock: Part of my quest in this is to find someone from the current leadership of the Word effort who might be willing to answer that for us.

Lampson: <laughs>

Brock: Because I don't what the sensitivities are around that, but I sure would like--

Lampson: No, neither do I.

Brock: I sure would like to find out. <laughs>

Lampson: I'm sure that Charles can find you somebody.

Brock: Well, I was thinking of asking. I was thinking of asking him. I-- in fact, I recently wrote him an e-mail with the edits to the edits of our transcript of our interview together and I did ask him if he would be willing to put me in touch with somebody there to try and see if we could get that answer. So I'll continue to try different avenues to see if that can happen. And I wondered if you had an opinion about-- Charles had a very interesting way of saying that by using the piece table in Microsoft Word, that they had all of the kind of latent potential that had been expressed in Bravo by using a piece table in Word that they kind of built in that latent potential into Microsoft Word. So that when the kind of-- what would you call it-- enveloping system environment was appropriate, they could add into Word things that you-- had been in Bravo, essentially. And--

Lampson: Oh, absolutely. And that happened very, very early in Word for the Mac.

Brock: Right. And in some ways, I was wondering if that was very important to Word's success, because it had this structure that could support a very kind of fine display-oriented editor that was Bravo with that-- maybe the essence of this editing engine, as you were describing it before, was in Word. So that gave it kind of an advantage over other rivals because it could-- it had this great potential for evolution.

Lampson: It's a very interesting story, which came to fruition in the days of Windows 3.

Brock: Hmm.

Lampson: Because in the eighties, Word and Excel were not the leading word processor and spreadsheet.

Brock: Right.

Lampson: The leading word processor was Word Perfect and the leading spreadsheet was Lotus.

Brock: Right.

Lampson: As I'm sure you know. And that changed very rapidly in the early nineties. And the-- it changed-- you know, there's an iron rule of technology, which is that the only way you can displace the leader is to change the rules. And the reason that Microsoft Office took over was that it was successful in changing the rules. And it changed the rules in two ways. One was the graphical user interface and the other was suites. And Microsoft had a huge advantage with the GUI because, basically, Word and Excel for the Mac had been around for 10 years and we understood extremely well how to make them work in the context of the GUI. And neither Lotus nor Word Perfect had that experience and it took them a long time to catch up, because it's actually quite hard. And of course-- it was a somewhat different story for Excel, but it-- the GUI is a less traumatic addition to a spreadsheet. From the beginning, even if it's on the character screen, has some of the aspects of a graphics program.

Brock: Right.

Lampson: But a typical word processor doesn't have any of those aspects. And so, yes, the fact that Word came out of Bravo is a huge advantage.

Brock: Hmm.

Lampson: And as I say, the whole thing was completely debugged on the Mac.

Brock: <laughs> Right. For-- in Word for Mac.

Lampson: In Word for Mac, which shared most of its code with Word for Windows.

Brock: Right. Well--

Lampson: In fact, there's an interesting story about that too. I think it was Word 6, they actually merged the code bases because it seemed crazy to have these two separate code bases now that Word for Windows was getting to be so graphics-centric. And that turned out to be a disaster.

Brock: Oh, no. <laughs>

Lampson: And it was a disaster for an interesting reason that had nothing to do-- much to do with technology, which was at that time, somewhere between 90 and 95 percent of the market for Word was on the PC and 5 or 10 percent was on the Mac. So the way that the merged system was released was it was released to the PC first and then, a few months later, the Mac was-- would be shaken down, would be released.

Brock: Right.

Lampson: But in those days, major software products that were produced by Microsoft, and I think everyone else too, ran on two or three-year release cycles. And the problem that they had with Word was that during the time that the Mac version was being shaken down, it was essentially impossible to start on the next release of the PC version because those two-- there were too many distractions. And so, the results of-- was that by catering to this 5 or 10 percent of the market, you delayed the next release for 90 percent of the market, by 20 or 30 percent.

Brock: Wow.

Lampson: Which was just a disaster. And I think this was completely unexpected.

Brock: So how was that resolved?

Lampson: They split the code bases again.

Brock: Oh, really? Oh my gosh.

Lampson: Yeah. And they make some effort to code-- they can easily be made common between the two systems, it's done that way, but they have separate development trees.

Brock: Oh, I did not realize that.

Lampson: Which is what they had had before Word 6 as well.

Brock: Hmm. And-- maybe you can-- I'm sorry. This is kind of looping back a bit to features and features that may or may not be related to the piece table structure. Is it that-- well, it seemed to me that track changes would be-- is that related to the piece table?

Lampson: I do not know how track changes works. I'm sorry.

Brock: Okay. <laughs> Because it seems like some sort of annotation that you could do in codes to a piece. You know?

Lampson: It's certainly possible that it's implemented that way. I just don't know.

Brock: Okay. I'll try and figure that out. And do you know if there are other word processors or text editors that they're based on the piece table data structure outside--

Lampson: I'm sure there are, but I'm ignorant.

Brock: Okay. And do you think that-- I was wondering about this kind of-- I understand that you came up with the piece table because of these sort of very specific constraints that you had at the time.

Lampson: Mm-hmm.

Brock: Dealing with this new display and contending with--

Lampson: Well, the piece table actually has nothing to do with the display.

Brock: Well, wasn't it to keep things from running slowly in the context of the display-oriented Alto?

Lampson: Well, you wouldn't have wanted things to be slow even if we hadn't had a display.

Brock: Yeah, I suppose so.

Lampson: I mean we used a much less-sophisticated hack in QED to keep it from being too slow, but definitely the shortage of memory was critical. I don't remember this, but it may be that the way that I, the path that I followed to come up with the piece table was to say, "Gosh, what we did in QED was really a kludge," and I shouldn't say "we" because Peter Deutsch wrote QED. I just-- I wrote the paper because he didn't want to write a paper, but as I said in the paper, I didn't write any of the code.

Brock: Well, I saw-- yeah, I saw that at the end but then I thought, well, maybe this is a collaboration on the design. I didn't know how to...

Lampson: We talked to each other a bit about the design, but basically it was Peter's design, which he did because we had another graduate student who was supposed to be writing the editor with him. He turned out to be incompetent and we'd gotten most of the rest of the system to work really well and we didn't have an editor and that was extremely painful. So Peter wrote the original guts of QED over a weekend.

Brock: Oh, jeez. Well, I was just-- the question that I had was, so maybe better put that the piece table is an approach in the data structure that is appropriate to this kind of constrained memory condition, let's say. It seems that, and it embodies kind of this idea of...

Lampson: Well now, remember there's two good things about the piece table. One is it's good in a constrained memory condition and the other that it-- is that it allows you to make the cost of editing even a large thing proportional to the amount of edit, not to the size of the whole document. And that's a consideration that continues to be important even if you're not short of memory.

Brock: That the...? I'm not sure I'm following.

Lampson: They make the changes...

Brock: I'm sorry. Could you walk me through that again?

Lampson: The piece table allows you to do an amount of work on the computer that is proportional to the size of the edit, not to the size of the document.

Brock: I get it. Right, right. So if you're doing very few changes to a gigantic thing, the cost is only a very few number of operations.

Lampson: Right, and that's important because people deal with the multi-megabyte things nowadays routinely.

Brock: Right. Well, that is a very, very interesting observation because I was thinking of these, I don't know what, how to properly characterize the genre, but these kind of browser-accessed editors like something like a Google Docs or even sort of the online version of Word in Office 365 that you can access kind of in the Cloud or what have you. And it seems like in those instances, well, with these kind of contemporary editors that you're not in that same memory constraint environment. You're dealing with these questions of bandwidth now but I wondered if would piece tables still be an appropriate data structure for this kind of Cloud environment rather than-- which I guess is kind of a time-sharing environment so maybe it would be. I don't know. I just couldn't figure it out. I wonder what your thoughts are.

Lampson: Well, I haven't thought about this deeply for quite a long time. I have thought deeply about a related question which is, okay, how can you efficiently represent multiple versions of a document, and the piece table ideas are extremely relevant there. And if you really want to pursue that into the early 21st century, you should talk to a Microsoft employee named Steve Lucco.

Brock: Hmm. Could you spell his last name for me?

Lampson: L-U-C-C-O.

Brock: Thank you.

Lampson: My intuition says that the piece table ideas are probably still extremely viable for a number of reasons. One of them is that it's pretty easy to use the browser's local HTML data structure like a piece table, and I think if I were setting out to build one of these collaborative editing systems from scratch I would probably try very hard to take advantage of that. Now, what they actually do, I have no idea.

Brock: Well, I believe that you're absolutely correct in that. I was in fact talking to someone who had been involved with Google Docs about just from wearing a different hat thinking of the museum and what could there be to preserve of Google Docs. And he was saying that, just as you had, that there is, and you'll have to forgive my kind of technical ignorance, but there is some significant fraction of it that is running in the browser, using the sort of browser's native capabilities. But there is issues of code obfuscation with that but in principle, one could kind of preserve that browser side part of the editor but then, of course, there are these other functions or services that it's accessing over the network. But so I think...

Lampson: Right, and of course, the most important thing it's getting from the network is the collaboration stuff.

Brock: Exactly.

Lampson: And I think Google Docs uses this, the operational transform scheme. The problem with collaboration is when two people make updates more or less simultaneously. How do you keep the multiple versions from getting confused? And loosely speaking, this is somewhat oversimplified but loosely speaking, the answer is you make it so that the updates that the two people make commute with each other. So it doesn't matter what order you apply them.

Brock: That's interesting.

Lampson: And then you just have to make sure that all the updates eventually make it everywhere.

Brock: Well, that's fascinating, so yeah. So perhaps even if there's this idea that for the collaborative browser-accessed editors that if they are still really trying to leverage the kind of capabilities that are built into the browser, then you're still really dealing with these same, the same sorts of constraints and considerations and advantages that the piece table afforded in other contexts because it's-- that's interesting.

Lampson: When I first-- to some extent, that's been built into the browsers' data structures from the beginning because if you're doing the piece table in the modern world, of course, it's not going to be an array. It's going to be a tree, and that's exactly what the HTML document is.

Brock: And let me just-- well, I was wondering if, to hear your opinion about the importance of an editor or the editor to the creation of software. We talked about these integrated software development environments. I just became interested in the concept of like the persistent centrality of an editor to making software and I was just interested to hear your thoughts about that. Do you feel that's true, that is, this kind of essential tool for software creation whose importance is by no means diminishing, or do you have a different opinion? I'd love to hear your thoughts about that.

Lampson: Well, it depends on your point of view about what's the right way to build software. There's a radical point of view that says that you should think everything out very carefully and then you should write your code very carefully and you should type it in and it should work. and there are people that can actually program that way. But that's definitely not the way 99 percent of software development is done. The way 99 percent of software development is done is very incrementally and certainly when you're doing things incrementally, the editor is a crucial tool. So yeah, I don't think there's any doubt about it, and as I say, modern-- the editors in modern IDEs, a large part of their functionality is devoted to supporting this extremely incremental and typically collaborative kind of development, which is a good description of the way almost all software is built. Anyway, that's been true almost from the very beginning, in spite of this sort of purist view that it shouldn't be that way. And to some extent, that has to do with what people are good at conceptualizing, but also there's a more fundamental thing going on which is that most software-- another aspect of the idealized view of software development is that you write down the requirements and then you write the code to satisfy them. But in real life, it's not like that because you don't actually know what the requirements are and they're changing. And so the fact that the requirements are changing means that it's unavoidable that the code is going to have to change. Even if there are no bugs, the code is still going to have to change.

Brock: Right, because the...

Lampson: So that is really quite intrinsic.

Brock: And there's no-- as I was kind of writing down that question, I was kind of puzzling to myself because I couldn't really even imagine what an alternative tool would be to the development of software to an editor. It just seems-- I certainly couldn't think of one, so it really seems so integral.

Lampson: Well, but it's very much a question of where you're spending your time. If you only have to write the code once and then just get rid of a few bugs, then of course you're still going to need an editor but it's not going to be that important because basically you're going to be typing the program in and that's going to be it. But if you're in a mode where the code is constantly changing, either because you're constantly removing bugs from it or because you're constantly adding features, then you're in a very different world where the ability to support all that activity of change is crucial, and that's the editor's job. So you're also going worry about multiple versions and the fact that in many cases the same piece of code is being worked on by more than one person and a whole bunch of other things for which, again, the editor is going to be quite central.

Brock: Right, to managing those challenges. Well, what I thought was also-- so we have this, you have the editor as very central to software creation and therefore to computing. And it also seemed very interesting to me that I think there's an argument to be made that text editors and word processors, these tools of writing with computers is also kind of a major development in human, literate, practices of culture. I was thinking about how many people are using a computer to write, either to send that writing to another person or send that writing to the computer to get, to achieve some desired outcome. It just seems like the quantity, the overall quantity of kind of human writing activity that's going on through the means of editors, either standalone or integrated into other software. It's just tremendous. I just wondered if you had any thoughts about that, about the story of these text editors and word processors just in the kind of idea of people writing.

Lampson: Well, I think that there are many different angles on that. Certainly, it's true that not much writing gets done nowadays that isn't mediated by the computer. It's also true that—and I think there's a fair amount of evidence—that there's a lot more writing being done now than there used to be because of the computer. Maybe the more interesting aspect of this is to what extent are the things that the computer has made easier having an influence on the way writing works, and the most obvious things are the tools that people have for making the resulting product pretty, the tools that people have for working with things collaboratively, and the tools that the computer can give you for structuring the thing that you're writing. I mean you can see that a lot of work has been done in all those domains, and doubtless in the not-too-distant future a fourth thing that will join those three is the ability that the machine will have to do some of your writing for you.

Brock: Yeah, to have like a built-in developmental editor or something in there.

Lampson: Well, Google now has these smart replies for email. That's a very primitive form of that, but I'm sure it's going to grow.

Brock: Yeah, a different take on...

Lampson: The machine will get better and better at doing more or less routine things or maybe pointing out more and more places where you have things that you might want to reconsider. We started out with the wavy red underline for spelling errors, and then we had the wavy green underline for grammar errors, and now we have the double blue underline for stylistic issues, and that's not going to stop.

Brock: Right, right. And it is-- and people certainly have looked at some of how the, some of the more informal language practices in things like short messaging have come to inflect people's writing in other contexts and things like that. But yeah, and I, but I just was, to me, I guess I was still just struck by even just the quantity argument and that, and just the computer as a writing medium and its reach. I hadn't really tried to think about it in the aggregate before, but it's certainly striking in a number of different dimensions.

Lampson: And you can see, for example, a lot more books are published nowadays than were published 30 years ago, and that's not because the population has grown. It's the various aspects of computing have made it a little bit, a lot easier to publish a book and cheaper.

Brock: Right. Absolutely. So I really do think it's kind of a, it's a very important development in this kind of history of human literacy. It's quite astonishing.

Lampson: Yeah. Absolutely. And of course, people have, on the computer side have been very interested in all these things, not quite maybe from the very earliest of days but from pretty early, not just editors but programs like RUNOFF and programs like Engelbart's NLS which addressed the beauty aspect and the structuring aspect of writing. And of course, both of those have been pushed a very long way from those early days, but you can see people very interested in that, more or less from the first instant at which the hardware was capable of supporting it at all.

Brock: Yeah. No, I suppose with the collaboration, too, at the moment that that sort of became available obviously through networks, but I would imagine that the collaboration as a kind of a major theme that...

Lampson: No, you didn't have to have a network. It was collaboration.

Brock: ...certainly must be there in time-sharing, right?

Lampson: Collaboration was a major theme for Engelbart, and he didn't have a network. He just had one time-sharing system.

Brock: Right, but I was thinking that collaboration would also-- the affordances of time-sharing would lead to these, could lead to these same concerns for like collaborative editing of some document or other forms of computer-mediated collaboration.

Lampson: Yeah. I don't know, other than Doug's system. I don't know that a whole lot of work was done in that area. I mean the first networked collaboration system that I'm aware of was a whiteboard system that was done at PARC in 1975 maybe by Niklaus Wirth who was on sabbatical visiting us and it was funny because, of course, Niklaus is Swiss and so he likes for things to be very organized and when he came to grips with the question of control in a collaboration system, he wanted to have some sort of protocol by which, at any given time, one of the collaborators would be in control. And I tried fairly hard to persuade him that this was not a good idea, but he's pretty stubborn and so he went off and he tried two or three things, none of which worked very well at all and finally gave up and adopted my suggestion, just-- it should just be anarchy. If you and I are collaborating and I move the mouse to the left and you move it up then the mouse will go to the left and will also go up. We'll just add the Delta Xs and Delta Ys, and if you move it to the left and I move it to the right then it won't go anywhere and that'll be a little bit frustrating, but we'll work it out. But it took him a couple months to become reconciled to this very undisciplined way of doing things.

Brock: Was that a projection from an Alto, or what kind of a...?

Lampson: No, it was screensharing.

Brock: Screensharing, I see.

Lampson: No, there were no projectors in those days.

Brock: Yeah, I was trying to...

Lampson: The only projectors were Eidophors.

Brock: Which were these...

Lampson: They're hideously expensive and require a sizable staff to keep them working.

Brock: Right, and they sound horribly dangerous, too, somehow. Don't they involve something with oil and...?

Lampson: Yes, they do. I don't know dangerous it is.

Brock: It just doesn't sound very, well, maybe not dangerous but certainly just...

Lampson: Messy. Yes, definitely messy.

Brock: Messy, perhaps, is a better word.

Lampson: Yeah, that's what Engelbart used for his famous demo.

Brock: Yeah, that's where I had heard the detailed description of these things, and I guess also I heard about them from-- did I hear about them from Chuck Geschke, who was saying that's what were used at Futures Day, they needed to use these?

Lampson: I wasn't aware of that.

Brock: I think they may have.

Lampson: I wasn't aware that you ever hooked up an Alto to an Eidophor. Maybe so.

Brock: It may have been just to some of these films that they had made for Futures Day. I'm not quite sure, but I-- yeah, but I did hear about them certainly in the context of Engelbart's demo.

Lampson: Of course, they were items of commerce at that time. He didn't have anything to do with inventing them.

Brock: Right, right, right. It was just my first introduction to the...

Lampson: And mine, too. I had no idea that such a thing existed.

Brock: Well, I think...

Lampson: Anything else?

Brock: No, I think I've run through my question list. I will-- thank you so much. I really appreciate your insights into this. I think it's-- I mean there's just kind of a-- well, to be perfectly honest, what I got today was, in addition to understanding the story a lot more, was a confirmation of what I thought the basic kind of structure in a history article that I might write is confirmed by our discussion, which was I have in kind of, from talking to you I can and better articulate that there's going to be this kind of, from time-sharing to the word processor, that kind of line. And there's also an interesting line that is about time-sharing and interactive computing that leads to the word processor and also from J Moore, his kind of development of the piece table ideas are coming out of artificial intelligence research in the same, and also in a kind of a time-sharing environment.

But I think just for people who are using tools like Word today, to see them through this lens of embodying some big, big developments in the history of computing, I just think it could be a nice little essay because it's sort of like these big themes and big stories in the history of computing crystallized in a tool that people use today without giving it a moment's notice to, and sort of denaturalize it for people and bring up some of the, even to raise up for people that there are people who made these things. I think it's just a, for me I think, a wonderful case study of where we could take these programs with which people are so familiar and kind of open people's eyes to the fact that they-- yeah, they kind of crystallize these big important stories in the history of computing. So I'm encouraged.

Lampson: And I guess you can do a similar thing for desktop publishing and for spreadsheets. To my knowledge, neither one of those has really been done.

Brock: No, and for spreadsheets is another one that I'm very interested in, in terms of-- yeah, I mean there's a whole story about the recalc algorithms in spreadsheets that's very interesting, but I think for me in the spreadsheet story what is, what hasn't been-- the thing that I would like to read or maybe even try and write some version of sometime is its influence on capitalism in terms of how the spreadsheet has become just a ubiquitous and indispensable tool for many of these practices of finance and a lot of these creations in the financial world that would be inconceivable without the spreadsheet and which-- in which the spreadsheet plays a vital part. So...

Lampson: An interesting riff on that is that I've been told by my colleagues in the Excel group that they have telemetry that tells them that only 10 percent of Excel users know about the formula bar.

Brock: Oh my gosh.

Lampson: The other 90 percent use it for making lists, for which it's extremely good.

Brock: I will confess that I use it for lists frequently, but I have clicked on the formula bar. I will confess to that, as well, but yeah, it would be actually interesting to see if you could mine the telemetry data from the use of spreadsheets to predict financial events.

Lampson: I'm pretty sure that you wouldn't be able to get your hands on it for that purpose.

Brock: Well, not I, but one could perhaps.

Lampson: Well, they-- I'm sure Microsoft is pretty compulsive about that. The other interesting angle on the spreadsheet from my personal point of view is that we failed to invent them at PARC and I think it's—Peter Deutsch sort of had an abstract idea of spreadsheets but it was never at all concrete. And I think the reason for that is that we didn't have any use for them.

Brock: Well, also I mean I think there it's the sort of, the context of, if you look at the context from which it really arose, Dan Bricklin sitting in that Harvard Business School classroom, it's a different-- it's a...

Lampson: Yeah, that's what I meant by saying that we didn't have any use for them.

Brock: Yeah, yeah, yeah, but even-- yeah, the, so the-- yeah, right, the need was not there.

Lampson: We did have accountants at PARC.

Brock: The need was not there.

Lampson: Yeah, we did have accountants at PARC, but we didn't get along with them so we didn't think of what their needs were.

Brock: Maybe it required a nonadversarial relationship with accounting for the emergent...

Lampson: Precisely.

Brock: That would be a funny essay. Well, thank you.

Lampson: Okay. Well, I'll look forward to seeing the transcript.

Brock: I will get it transcribed and I will clean it up a bit and send it to you and I will also keep you updated about my quest to see what's going on with the piece table in Word today. I'm also going to try and talk to, try and get contact to some of the people doing, who have done and are doing Google Docs to see what kind of data structure they use there. So I'll let you know what I can find out.

Lampson: Okay. Great.

Brock: Thank you so much. Take care.

Lampson: Yeah, bye.

END OF THE INTERVIEW