Oral History of Tom Malloy, part 2 of 2

Interviewed by:
David C. Brock

Recorded December 14, 2018
Mountain View, CA

CHM Reference number: X8837.2019

Brock: Well, thanks again, Tom, for joining us.

Malloy: No, you're very welcome.

Brock: Again. I appreciate it very much. At the end of our last time together, we were talking a bit about your efforts working with Charles Simonyi and Butler Lampson, at the beginning of the Bravo text editor word processor project, and we also discussed that this was in part framed within the context of Charles's PhD thesis, dissertation work for Stanford.

Malloy: Right.

Brock: And that the initial work on the editor took place, at least according to Charles Simonyi's thesis, between July and September of 1974, about 13 weeks, I think.

Malloy: Okay.

Brock: There was some-- and I was just wondering about-- well, two things I wanted to ask. One was about the sort of collaboration between the three of you or anyone else involved in kind of building Bravo, and also to talk also about the piece table as a kind of fundamental data structure for Bravo.

Malloy: Okay.

Brock: And it's your thoughts about the importance or the implications of that data structure.

Malloy: Okay. Well, in terms of the collaboration, my collaboration with Charles was on a daily basis, and the whole philosophy that he brought to programming was that there was a team, in this case, a pair of people. He was the meta-programmer and he was writing specifications for me in essentially somewhere between English and pseudocode, and handing that off to me to, as the technician, to implement in BCPL, which was the language that we were working in. Now, I think he and Butler also probably had a fairly intimate relationship, professionally intimate relationship, on I don't know whether it was a daily basis or close to it. My interactions with Butler were more casual, I would say, in those early days, because Charles was the interface between the two of us, but he was always open to me walking into his office and asking questions or getting guidance. But I don't remember it being, you know, kind of a daily kind of thing.

Brock: Okay. And the piece table data structure that you were busily implementing for Bravo, could you talk about it and--

Malloy: Well, I think the piece table-- I know Charles, I've heard Charles speak, and I know he considers it a, you know, one of his great accomplishments, and I think it is, his and Butler's.

But I see it more as an example of the kind of design that really permeated the Alto, and I think it, the philosophy, largely came from Butler and maybe Chuck Thacker as well, but-- Alan Kay as well, I think, this notion that they were going to build something that could just barely do <laughs> what it was they wanted to do, because they wanted it to be as inexpensive as it could. They wanted it to be as much in software as opposed to hardware as it could. So software, or sorry, the hardware was very simple for the Alto. Didn't have, I mean, it had a lot of memory for 1974, but it didn't have a lot of memory for the kinds of tasks that they wanted to accomplish, and so there are a number of design decisions throughout the design of the Alto and the software that ran on the Alto that reflect this sort of minimalist philosophy, and I think piece tables are an example of that, because the idea was that we wanted to have documents that were larger than would fit in memory, and if that was going to be the case, then we had to have some kind of data structure which would let us keep most of the document on the disk and only bring into memory the portion that was necessary for whatever was going on at that particular moment, and the piece tables were the vehicle for doing that.

Remember, because it was so simple, there was no virtual memory. If I remember correctly, we barely had parity on the memory in the early days. We'd have to look that up, but, you know, everything was bare bones on the Alto. Another example of this, which I think we talked about last time, which I was very impressed with as I was reviewing what we were doing for the demo last year, was this feature that Chuck and Butler built in-- I assume was Chuck and Butler-- built into the hardware where you didn't actually have to allocate a full bitmap for the bitmap display, if you didn't want to.

Brock: Oh, right.

Malloy: Remember that Smalltalk did, and that was half of the main memory, essentially, was the bitmap. But we did not, in Bravo. You could allocate blocks of memory that just represented, like, each line. It represented horizontal swaths of the display, and the microcode would pick up those blocks and turn them into the output for the scanned CRT, and that allowed us to have much less than half of the memory devoted to the bitmap for the display, and there's really numerous other examples of things like that, of how we made design decisions, or they made design decisions, that were direct consequences of this minimalist philosophy, and the piece tables, I think, is the one that Charles is most proud of, and rightfully so. I implemented all of my text editors and word processors with piece tables right up through Apple and Adobe and elsewhere. That just--

Brock: Well, you're prefiguring a whole set of <laughs> questions about the piece tables and your subsequent work.

Malloy: Yeah, sure.

Brock: Which I'm so glad we can-- glad to hear that and glad that we can get to it. Could you-- I think I have-- well, I would love if you could take a stab at explaining for a viewer of this interview, and for me, <laughs> who doesn't have a tremendous depth of detailed technical knowledge in this area, you know, what the piece table was and how it allowed, afforded this kind of minimalistic approach?

Malloy: Sure. Sure. Well, first, let's just imagine or take it as a given that all of the characters and the formatting, everything that makes up the document, is residing on the disk somewhere.

Brock: Right.

Malloy: We'll take that as a given and then we talk about how we make sure that happens a little bit later. Well, to begin with, that means that you can describe the document in main memory with a single piece or a single record that says, "Go to the disk here and read for however many characters there are in the document the information that's stored at that place on the disk." That's the description of the document. That's a piece table with one piece in it.

Brock: Right.

Malloy: But now let's imagine that you remove a paragraph from the middle of the document. Then you need two pieces to describe the document. You say, "The first piece starts at the beginning of the document on the disk and it goes up to wherever that missing paragraph started," and then there's a gap, which is described by adding a second piece that says, "The second piece begins after-- on the disk-- after where that paragraph resides, and then goes to the end of the document."

And then with each edit you build up these pieces, and eventually there could be many pieces, one or two for each edit, depending on what kind of edit it is, and so the amount of memory that was required, main memory that was required to describe the document, was a function of the amount of editing you've done, not of the length of the document. That was the essential idea.

Brock: And was-- and then it would only be at certain defined intervals that the kind of edits represented in the piece table would be sort of written to the disk in creating a--

Malloy: Resequenced.

Brock: Resequenced in creating--

Malloy: Yeah.

Brock: --kind of a new saved--

Malloy: Right, yeah.

Brock: --version of the document.

Malloy: Yeah, exactly.

Brock: One thing that I haven't been clear about, forgive me, is how-- whether or not the piece tables were implicated in Bravo sort of going to the disk and getting information for putting on the display, or was that a separate function? You know, so let's say I'm using Bravo and I want to, you know, go down a couple pages. I'm going to edit Page 11 and insert something there.

Malloy: Right.

Brock: Was that separate functionality that was handling--

Malloy: No, no. You'll-- think of it in object-oriented terms. The object that was the document was encoded in this piece table, so regardless of whether or not you were displaying a portion of the document on the screen or searching through the document or performing an edit, everything went through that piece table data structure.

Brock: That was how you went through the piece table to get to the information that was stored on the disk.

Malloy: Yeah, always. Always.

Brock: Interesting. For...?

Malloy: Everything.

Brock: Everything that you were doing.

Malloy: There was no way to even refer to pieces of the document or resolve references to the document except by going through the piece table.

Brock: Okay. Okay. So, for something like, you know, I want to look at Page 11 and do something, it would, the editor would be sort of calling or referring to the piece that contained or pointed to Page 11 and then--

Malloy: Well, all the references--

Brock: --put it in the screen?

Malloy: --started out as a document and a essentially character index.

Brock: Right.

Malloy: And then that character index in the document was passed against the piece table or resolved through the piece table to come up with a location on the disk and a character offset from that location on the disk.

Brock: I see.

Malloy: Where that document character index could be found.

Brock: Okay. So it's-- yeah. I think I follow. <laughs> Thank you, so-- and yeah. So that's fascinating. So it really is, suffuses, the entirety of the functionality of that.

Malloy: Yeah, absolutely.

Brock: Okay.

Malloy: Yeah.

Brock: One-- thank you very much for that.

Malloy: Sure.

Brock: Could you talk about the relationship between Bravo and this, and I guess it's a language, Press, for printing? Was Press something that Bravo could kind of call upon externally, or was that somehow--

Malloy: We generated Press from Bravo when we printed to the first of the printers that Ron Rider designed. Well, no, let's see. First of all, you should check my facts on this, but I think the first printer that was designed was EARS, if I'm not mistaken.

Brock: I think that's correct.

Malloy: And I think that was before Press. But Press was a printing language, and in general, once these printers came online in the research center, we developed code in our print command that would generate whatever print language was necessary, first EARS and then Press, I guess. We'd have to check on that.

Brock: But it would be--

Malloy: But that's totally different representation of the document than what we worked with on an interactive basis.

Brock: But it would've been the print functionality within Bravo--

Malloy: That’s right, yeah.

Brock: --to create the Press file.

Malloy: That’s right. It was all-- I think we discussed this last time, but remember, Bravo, it didn’t quite run on bare hardware, but almost. So, there was no print libraries or anything like that. We started with our representation and we wrote the code that generated the print representation. We took some code from Bob Metcalfe to actually send it out over the Ethernet, but that wasn’t a library, that was, I mean, it was-- then it was built into and part of Bravo. It was not part of any operating system. So yeah. When we brought up the first printer, we were also debugging the first Ethernet code. It was--

Brock: <laughs>

Malloy: Well, I mean, they had used it for a few other things, but we were still shaking down that code too.

Brock: We also talked a little bit last time, but I’d love to hear you expand a little bit more, about the fact that initially, or the most avid use of Bravo once it really kind of got going, was as an editor to write programs, to write other software.

Malloy: Actually, I don’t think that’s quite accurate in terms of the number of users. There were probably more users using it as it was intended as a piece of office software, because, remember, the number of developers was-- well, there was our team, and then I guess there were quite a few devel-- and we’d have to count noses.

The Smalltalk team wouldn’t be using Bravo to write their code, I don’t think. They’d be doing it in Smalltalk. I guess the rest of CSL [Computer Science Laboratory] probably used Bravo. I don’t-- I would think. They could still use a line editor if they wanted, but they probably used Bravo. Certainly we used Bravo. But everybody that had an Alto, from staff people, management people, administrative people, and researchers and technicians, everybody used Bravo for writing memos and other office uses, so it’d be-- I would, as I just said, we’d have to count noses, but--

Brock: Okay. Yeah. So it wasn’t--

Malloy: --it was being used extensively in both--

Brock: I see.

Malloy: Both ways, I think, and if somebody came back to me and said, “No, no, there weren’t nearly as many people writing code with Bravo as we’re hypothesizing here,” I wouldn’t necessarily disagree.

Brock: Okay. And I suppose by this point in time in the sort of story of text editors, it was known that the features that would make an editor good for writing code would also be helpful for writing other kind of literary documents: memos, reports and things and--

Malloy: Well, I think it's the other way around. I think Bravo was really designed to be a office document creation tool, and there was this period when it was also better than anything else or as good as anything else for developing code. If we think forward not very many years, probably '76 or '77, when Mesa comes on the scene, Mesa will have its own editor somewhere in that time frame, '77, '78, I don't know when.

Brock: Right, right.

Malloy: And so, clearly, program editors develop a life and a history of their own fairly quickly, but remember, in 1974, the first Alto came off the assembly line in '73, I think. When I got there in '74 there were only about six or seven of them.

Brock: Right.

Malloy: And so we were building from scratch, and so in terms of programming, other than the Bravo team itself, my guess would be that they heyday for using Bravo as a program editor was probably, I don't know, '75 to '77. I'm just guessing. I don't know.

Brock: Okay. That makes sense, and also your comment about the Mesa environment and its editor resonates with a question that I had written down here about the Interlisp environment.

Malloy: Ah, yeah.

Brock: Which had--

Malloy: Right, exactly. They had their own editor.

Brock: And I had a chance to talk to J Strother Moore, who evidently kind of independently and almost simultaneously with Butler Lampson, also conceived of basically the piece table structure as well.

Malloy: Oh, really?

Brock: And he was at the University of Edinburgh, making an editor for a time-sharing system, the same space-power constraints and came up with a similar notion, and he was in fact at PARC for about a year and...

Malloy: We implemented his search algorithm in Bravo.

Brock: The Boyer-Moore--

Malloy: Boyer-Moore algorithm, right, yeah.

Brock: Right. Well--

Malloy: I mean, he invented it and then we implemented it.

<laughter>

Malloy: Pretty much in real-time, I think, if I remember correctly.

Brock: Oh, yeah. I think it must've been right after he and Boyer conceived it.

Malloy: Yeah.

Brock: It was when he came to PARC, but he was telling me that he was really working with the editor functions for Interlisp and bringing his kind of-- the same piece table approach into that world, so it's an interesting case of simultaneity and--

Malloy: Yeah, yeah.

Brock: --these contexts mixing. I wondered if you--

Malloy: Well, and the people were mixing too. I mean, everybody was on the same hallway, so...

<laughter>

Brock: Right. I wondered if you had any recollections about just that, about Moore being at PARC and--

Malloy: Well, I do because of the search algorithm. I didn't have a lot of interactions with him in other contexts, but I have a vivid recollection of, you know, somebody coming to me, whether it was J or Charles or somebody, and saying, "Hey, we've got this new search algorithm, and it's better than the--" I think we actually implemented Jim Morris's algorithm first. That's got somebody else, a few other people's names on it too. I don't remember what-- anyway, I don't remember what the name of that search algorithm is or the names that go along with it, but I think we implemented that one first, but then very quickly after that J arrived and we learned about this algorithm and somebody said, "This is the best thing since sliced bread. We've got to go implement it." We did, and so that was the standard search algorithm in Bravo from then on.

Brock: It's my impression that it's still widely used.

Malloy: I would imagine. I don't know of any better to this day. It's a sublinear algorithm. It's--

Brock: <laughs>

Malloy: You don't get to find those very often.

Brock: Well, I was wondering about, I guess, the picture of the overview of your roles at PARC, because I wasn't quite sure when you departed PARC for Apple and just with the sequence of things you were doing at PARC was.

Malloy: Sure.

Brock: Yeah.

Malloy: So, remember, I started as a summer intern, essentially. What we would today call a summer intern. But I had already graduated, so it wasn't like I was going back to school.

Brock: Right.

Malloy: And I was a temporary employee the entire two years that I was at PARC proper, and, you know, they just kept paying me my hourly rate, whatever-- some ridiculous rate. I think it was like, started out, at $10 an hour or something like that, and then in the grand scheme of how PARC was going to be successful they started spinning off development organizations. After-- that was in '76. So, the first two years Charles and I and everybody else was in PARC proper. That's basically the organization that was there, inside the Computer Science Lab, and we developed Bravo there.

In, I'm pretty sure it was '76, the Systems Development Division was founded. Dave Liddell was the head of that, and a lot of people left PARC at that point to join the Systems Development Division. Bob Metcalfe was a executive in that group, Charles went there and took me, as well as the Bravo team was probably five or six people by then. I don't know, and the entire Bravo team went there, and actually, we didn't go there to work on Bravo. This, the Systems Development Division, was the organization that would ultimately build and ship the Star. They were going to do all their development in Mesa. They did all their development in Mesa, or at least all the software that shipped on the Star was-- I think was written in Mesa, and our assignment, Charles's group's assignment, was to start developing substantive software solutions applications in Mesa in anticipation of the D hardware coming online and all that sort of stuff, and I don't quite remember whether we were supposed to be reimplementing Bravo in Mesa at that point or whether there was some other assignment we had, but the tools, the Mesa tools were so primitive or they weren't up to the task in 19-- in whenever it was, mid-1976, and so we basically brought the development environment to its knees in a fairly short period of time, and you'd have to check with, like, David or Bob or Charles to get the real story about how this pivot happened, but we basically had time on our hands and we went off and started just working on other projects, and it seemed like almost everybody in Charles's group had a project of their own for a while, and my project, which I came up with, was called-- I named it DeSoto, after the car, and it was an early incremental compilation environment maybe or it had the following features that you would now find in integrated development environments, and this was all in Mesa.

So, it's not like we went back to developing Bravo in BCPL. Instead of developing-- this is my recollection. Instead of developing, you know, office applications in Mesa, we went off and started developing tools for the Mesa environment, because the tools were so lackluster at that point, and so what my tool did was the compiler was really, really slow and there wasn't really much of a linker. That's my recollection. In any case, the process of recompiling an application was excruciatingly slow, and so I came up with this notion, which now people would call incremental compilation, I think, which is sort of like a step beyond MAKE, so MAKE keeps track of the static dependencies between different files or modules in a larger program and schedules recompilations of things that if you change one source file, it'll automatically recompile things that depend on that source file. So we went one step further, or I went one step further, and I would actually go inside the compiled object file and find out what changed from one compilation to the next, and if there were no externally visible changes, you know, if you just changed the inside, the stuff that's only visible inside the source file, I wouldn't schedule any recompilations, and then I would go even further and at a symbol, you know, basically at a symbol-by-symbol level, like a procedure or method call level or a instance variable or global variable level, I would look to see whether any changes had been made to that particular procedure or method, I don't remember what they're called in Mesa, and I would only schedule a recompilation for a dependent file, Mesa file, if it actually used that, one of those methods, that changed, and so it made rebuilding of the system go exponentially faster, and then after that I also added-- so there were three levels, I think. There was sort of the basic MAKE level and then this fine-grain recompilation level, and then I also did a very primitive source code control system where you could check changes into an IFS server and thereby share them, and so that's-- I probably spent a year on that, and Greg Shaw, I think, developed a debugger, and... But we weren't part of the tools team, we were part of Charles's team. So even though the tools, the folks that were really responsible for Mesa tools, I think learned-- were interested in what we were doing and learned from it. I'm not sure anything ever became of any of those tools that we built.

Brock: Was Chuck Geschke in charge still of Mesa? Because I thought he had been initially--

Malloy: Not in SDD. I don't think he ever left PARC.

Brock: Okay. Got it.

Malloy: Because when he left to found Adobe, he was still at PARC.

Brock: Yes.

Malloy: So...

Brock: Okay. So, no. This would be getting, again, to, like, the production version of Mesa.

Malloy: That's right, yeah. Yeah.

Brock: Yeah, okay. So different group.

Malloy: Right. Right. And Chuck, I think of Chuck as being primarily responsible for the language, but...

Brock: I think that's right.

Malloy: So this is sort of-- his baby had grown up and had gone off into the world.

Brock: <laughs> Moved out. Was there-- and forgive me for not knowing this. There was a physical move too, out of the building.

Malloy: That's right.

Brock: Where you had been. Where was the Systems Development group located?

Malloy: Where were we? We were across the street, I think.

Brock: Okay. <laughs>

Malloy: I have a definite picture of what the office looked like, but I can't remember where it was. I think it was just across the street.

Brock: Very nearby.

Malloy: Yeah.

Brock: Yeah.

Malloy: So that was probably... So, if first two years were at PARC, then we probably spent another year in SDD, maybe more, I don't know. Was there almost five years, and then there was one more move. There was yet another spin-off from PARC called the Advanced Systems Division, ASD.

Brock: Okay. I've heard of that.

Malloy: And that was founded-- or the founding head of that was Jerry Elkind, who used to be the head of CSL, and he had a different mission. So, let's say that-- let's just say for the sake of argument we spent about a year-- SDD had been in place for about a year or however long it took. Became obvious going to take a whole lot longer to build the Star and ship it than people thought, and-- thought initially, and so there was a desire, on some people's part, I assume it was driven by Jerry, to get Altos out into the world in sort of like this expanded beta test almost, and that was ASD's charter was to go out and do prototype deliveries of Altos with Bravo and the mail program.

Brock: Laurel.

Malloy: Laurel and a few other things, and put them in real offices and get some real feedback, and so we moved to that-- Charles's group moved to ASD at that point and then we went back to working on Bravo again. And BravoX was developed in ASD and we delivered to places like the White House, and I don't remember where else but I distinctly remember a White House installation, and that's where I finished my career at Xerox is I left in late '78, early '79.

Brock: Okay. And--

Malloy: I think that's right.

Brock: Huh. And while you were doing this work, you were also doing a master's at Stanford?

Malloy: Yeah. That was '77 and '78, I think, yeah.

Brock: Okay.

Malloy: Yeah.

Brock: And the BravoX was, if I understand correctly, developing Bravo so that it could incorporate some features and functions that had been pioneered or implemented in Gypsy; is that correct? Or were there--

Malloy: Well, we definitely wanted to get a better user interface. So, remember, Bravo was not a modeless user interface. It was a very modeful user interface, and I think everybody knew early on that that was not the right choice, but it was-- it wasn't what we were focused on in 19-- I mean, user interfaces were not what we were focused on in '74, even '75.

So, it was probably '78 by the time we got around to having the resources and the desire and the-- and we were going to put this in, you know, customers' hands. So my high priority was developing a modeless interface for sure, yeah, and also just, you know, since we were now going to have real-life customers, they had enhancement requests, and so we were putting more features in, and I don't really remember too much about what the exact features were, but the user interface was a big part of it.

Brock: And I suppose this would be when the first external users come into the picture in a way.

Malloy: That's right. Yeah.

Brock: So, it makes you look at the-- I would imagine it makes you look at the program with fresh eyes and about how these people would approach it.

Malloy: Well, yeah. I mean, when your customers are your colleagues and fellow developers and researchers, you have-- they have one little-- one set of expectations. When they're somebody in the White House there's a different set of expectations about what your software does and doesn't do.

Brock: I was curious, what engagement, if any, you had with the microcomputing scene at this, in this later portion of your time at PARC?

Malloy: I wasn't really plugged into it at all.

Brock: Okay.

Malloy: You mean, like, the computer clubs and things like that?

Brock: Yeah. Were people making home microcomputers, anything like that?

Malloy: No. No, wasn't--

Brock: Okay.

Malloy: Wasn't anything I was involved in.

Brock: I think that would've put you in the mainstream of people <laughs> at PARC.

Malloy: Yeah. Yeah, right.

Brock: Not having-- well, the reason for my question is just about your departure to Apple.

Malloy: Well, first of all, I didn't go directly to Apple.

Brock: That's what I was curious about.

Malloy: So, I left, as I said, in late ’78 or early ’79, and when I decided to leave, I sent my resume to a few companies in the Valley. I mean, it wasn’t like, anything like it is today, in terms of being obvious how you go about, you know, find another job in high tech in Silicon Valley. Now you just go out, stand on a corner, and somebody picks you up. But I did send my resume to a few different companies. Apple was one of them in that when I was leaving Xerox, but another one was Zylog.

Brock: Yeah, sure.

Malloy: And I sent my resume to Zylog. I don’t know where else I might have sent my resume, and to show you how naïve I was at the time, I wanted to leave Xerox or at least my recollection of why I wanted to leave Xerox, was because, you know, I wanted to get my work product into products and into people’s hands, and also, I wanted to work for a smaller company. You know, Xerox was a big company, a big corporation, and I didn’t like the-- there were things about the bureaucracy I didn’t like, and so I wanted to work for a smaller company.

So show you how naïve I was, I didn’t realize until I got to Zylog that it was owned by Mobile, Mobile Corporation. It was a fully owned subsidiary of Mobile. So I thought I was going to, like, this little kind of startup. I’m not sure we even called them startups then, but it was established company for sure, but I thought it was a small company, and it had aspects of a small company, but it was in fact owned by the largest corporation in the world at the time, I think, Mobile.

So I went there and I joined a group that was, whose mission was to develop software and hardware based on the new Z8000 processor that was just coming off the chip lines, and they had aspirations to build office equipment kind of hardware and software. But it didn’t take me long to figure out that I was in the wrong place. First of all, this group was run by Charlie Bass and Ralph Ungermann. By the time I had gotten there, Ralph Ungermann had already left, and Charlie Bass left before I did, and I was only there for 10 months, and they founded Ungermann-Bass.

Brock: Right.

Malloy: And, you know, Zylog just didn’t have the commitment or the resources to do what I thought we were going to do, even though the people that I was working with, you know, thought that we were going to do that, and I met some of the-- some people who went off to really do wonderful things in the industry like Judy Estrin and Dave Folger and people like that. So was a great experience, but, really, literally, I was there from, like, January to October of 1979. Because it just was obvious that what I-- my job was to build a word processor, but I was building a word processor on hardware development systems, so it just didn’t make any sense, and there was no sign that there was actually going to be some kind of computer that made sense for my software.

So, in parallel with me saying “Oh, geez, I wonder what the heck I’m doing here,” I’m sitting at home one night, and I don’t know whether I got a phone call or a letter, but somebody from Apple-- this is six months after or eight months after I had sent in my resume-- called me up and said, or wrote me a letter and said “Hey, we just saw your résumé, and we think you’d be great for Apple, and we’d like to have you in as soon as possible.” I think it was a phone call-- I’m not sure-- because I said “Well.. that’s kind of old news. I left Xerox back at the end of last year, and now I’m at Zilog.” But they persisted, and also the reason I sent my résumé to Apple to begin with is because I had a friend who I was an undergraduate at Stanford with, who graduated, went to business school and then went to work for Apple in about ‘77 or ‘78 probably. Probably ‘78 would be my guess; I don’t know. And so he was also trying to find me a place at Apple, and so finally kind of the stars aligned, and somebody introduced me to the guy who was the engineering manager for the Lisa group. Oh, I’m blanking on his name right now. That’s horrible. Anyway, it’ll come to me.

And I interviewed with the Lisa group, which was, mm, smallish, but not small. There was probably half a dozen people at least in that group at the time. And they wanted a word processor, too, and they claimed they were going to build a real computer for people to use as opposed to a development system for people to design hardware for the Z8000. And so it all sounded much better to me, and I had these friends there that kept saying- or a friend who kept saying “This is a great place, and you should come, too.” And it was a real-life start-up. <laughs>

Brock: Not owned by an oil giant.

Malloy: It was not owned by Mobil. So I quickly pivoted. I don’t really talk about my Zilog years other than the people I met. In fact, that’s where I met Bernard Peuto, who’s so important to the [Computer] History Museum.

Brock: Oh, yeah, sure. Absolutely.

Malloy: And we remain friends to this day.

Brock: I did an oral history interview with Bill Carrico, who told me all about that Zilog context, because he was over there in that mix.

Malloy: Ah, yeah, Bill was there as well. Right, yeah.

Brock: And also in that interview--

Malloy: Manny Fernandez. Do you remember Manny Fernandez? He went off and formed Gavilan.

Brock: Sure, Gavilan. Sure, yeah. I think that was how Bill Carrico got to Zilog.

Malloy: Is it? I don’t--

Brock: He had worked with--

Malloy: Manny?

Brock: For Manny previously at Fairchild, if I'm not scrambling my story.

Malloy: It could be.

Brock: But one thing that Bill Carrico did talk to us about in our interview with him was that there were close social ties between people in the Zilog scene and people from the PARC scene, and I wondered if that had at all been a factor for you knowing about Zilog or thinking of going over there.

Malloy: I don't think so.

Brock: Okay.

Malloy: And I may have been part of that catalyst for all I know.

Brock: Oh, yeah, true.

Malloy: I was certainly part of the catalyst between Apple and Xerox, so I may have been part of the catalyst. I don't know.

Brock: And may I ask about your friend who had been at Apple?

Malloy: Mm-hmm.

Brock: Who?

Malloy: His name is George Johnson. He was a marketer.

Brock: Okay.

Malloy: And he was at Apple in the early days, and he moved up to the San Juan Islands many, many years ago and still lives there and very happy. He's done a variety of things since then.

Brock: Okay. And so this would be sort of in the second half of 1979-ish that you joined Apple.

Malloy: October of '79 is when I joined Apple.

Brock: Okay.

Malloy: I know just because your hire dates at small companies make a lot of difference <laughs> financially, so I remember my hire date at Apple and at Adobe, or at least I remember the months.

Brock: Yes. Well, great. So had Larry Tesler gone to Apple at that point?

Malloy: No, Larry came afterward.

Brock: After. That's what I thought.

Malloy: I was the first Xerox person to show up at Apple to the best of my knowledge.

Brock: Okay. Yeah, that's what it looked like to me, so I'm glad to hear that confirmed.

Malloy: And it was before Steve got his demo, so it was not like--

Brock: It was two months earlier, right?

Malloy: Yeah, right.

Brock: Yeah.

Malloy: So I think Larry came shortly after the demo.

Brock: That's right.

Malloy: Probably he gave the demo, so I assume that Steve recruited him because of that. But my arrival was kind of serendipitous.

Brock: It wasn't part of a movement toward that visit. Right.

Malloy: It wasn't like they knew about Xerox and they were trying to recruit people out of Xerox. I had left Xerox months before, so it was really kind of coincidental.

Brock: That's fascinating. How did Charles Simonyi and the other people you were working with at Xerox, how did they react to your departure? Could they understand? It sounds like you had maybe a shared concern with some of them that it was going to take longer than people would've wished for things to really get out.

Malloy: Well, lots of people left Xerox for those reasons, and I think that was part of the reason I left Xerox, but I think part of it-- it's hard to know when it's so far in the past, but I think part of it was just it was time for me as an individual to go. Remember, I started out as a snot-nosed kid. I was working as one of Charles's technicians. I do remember that I kind of thought I was underpaid, and I felt like I was in this big bureaucracy. Oh, by the way, and as I thought about what I wanted to do, I believed what now, in historical hindsight, is the motivation for many people having left PARC, but that was just one factor for me.

Brock: Right.

Malloy: So I was just this-- let's see. 1978, I was this 26-year-old guy. I'd been in my first job for five years, which now is an eternity, and my career wasn't advancing quite as fast, and I was in this company that I thought was kind of a bureaucracy. Charles was telling me "No, just stay here. I'm going to take care of you." Jerry Elkind was telling me "No, I know you think we're a big bureaucracy, but I can fix this." In fact, as I was out the door, they were actively trying to fix it so I would stay, but my motivation for leaving was personal rather than this grand professional perspective that-- by the time Chuck and John left to form Adobe, or by the time Bob Metcalfe left to found 3Com, or by the time Charles left to go to early Microsoft, I think their motivations might've been more clear and more pure in this historical storyline, but mine was really just more mundane.

Brock: I imagine it's a mix for everyone.

Malloy: It is a mix. It is a mix for sure.

Brock: It has to be a mix for everyone. Well, could you describe the Lisa project as you found it then? What was it? You kind of walk in the door. What did it look like? Where did you sit? What was it like?

Malloy: Well, the most important thing for you to realize is that the day I arrived, I thought I had made another terrible mistake. So, the week before I was scheduled to arrive on Monday, probably on Friday or something, I called up Ken, Ken Rothmuller. That's the guy's name who ran the project. I called up Ken, and I said "Look, I'm coming on Monday. Do you have anything I can read in advance to sort of be prepared and hit the ground running?" And he said "Sure, here, I'll give you the project notebook. You can take it home for the weekend and read it." And it literally was a notebook, and it had mostly hardware stuff in it, and I was reading through this notebook, and it was describing the original Lisa hardware. I don't know whether you have this in any of your interviews or historical archives anywhere, but the original Lisa hardware was something that was designed by Wozniak. It was a eight-bit bit-slice processor that was going to run what you would think of as byte codes today. It probably was bite codes then. And I could tell just by looking at the specification it was dog slow, and it was going to be a horrible, horrible problem that we weren't going to have the oomph to do what we wanted. But it was a classic Woz design, right? Talk about minimalist. It was <laughs> six chips or eight chips or something like that.

Brock: A half a chip, yeah <laughs>.

Malloy: Except he had passed that minimalist design off to some other hardware designers who now had it expanded out to a board or two. Anyway, so I was kind of depressed going in on Monday, and I thought I was going to have to have a conversation about how maybe I made a mistake coming to Apple. But before I could say anything, Ken grabbed me and everybody else in the office. Office: This was a suite behind the Good Earth Restaurant in Cupertino. It wasn't even on Bandley Drive. It was a tiny, little suite with a couple of cubicles. And he pulled us all into his office slash conference room, and he said "We're throwing out the old hardware design.

I, Ken and somebody else--" I don't remember who, maybe Rich Page-- "just flew down to Austin, Texas, on Friday. We went to Motorola, and we saw this new chip called the 68,000. Not even off the assembly line yet, but it's going to be here whenever it's going to be here, and it's going to be so much better than this hardware design that we have here. We're going to throw out this hardware design; we're going to start over, design something based on the 68,000." And I went "Whew. Oh, god, I'm so happy <laughs>." I didn't even know the details of the 68000, but at least it was a 16-bit processor, not a 8-bit processor, and it was an integrated CPU, not this-- I shouldn't complain about bit-slice processor. That's what the Alto was, too.

So, the project actually was about a year old then I think at least, but it restarted on the day I arrived, again, coincidentally. It didn't have anything to do with my arriving; it had to do with the fact that they'd gone to Austin on Friday. And so we started over. And what else can I tell you about the early days? I told you physically we were in this tiny, little suite. Apple itself was tiny at that point. There was only a few hundred people there, but there were bigger buildings on Bandley Drive, and we were in literally just a tiny, little suite.

The overall design of the Lisa was, I have to say, also pretty uninspiring in that first iteration. Ken and a guy by the name of John Selden and probably a bunch of other people whose names I don't remember were from HP, along with-- the engineering VP for all of Apple at that time was guy who had come from HP, Tom somebody. I can't remember his last name. Anyway, I don't know whether you have any HP 100s in the Computer Museum or not, but they were inspired by I think it was called the HP 100. Anyway, it was a computer with a keyboard and soft keys along the top of the keyboard, and then it had a CRT, but not a bitmap CRT I don't think, and then the keys appeared at the bottom row of the CRT, whatever the label was for the button. No mouse, no bitmap display. I think, if I remember correctly, the display was going to be yellow, because the best ergonomicists of the day were Swedish-- at least this is my recollection-- and they had determined that the most ergonomically pleasing or satisfactory color for a display was yellow, because it was easy on the eyes. So I think we were going to have a yellow display, no mouse, no bitmap, no graphical user interface, soft keys. And so that was sort of the starting point when I arrived. Now, I don't think I honestly marched right into Steve Jobs's office and said "We can do better than that," but the marketing manager or whatever his title was, director for Lisa at that time was Trip Hawkins, who went off to found Electronic Arts and many other things. And so I was interacting with him almost immediately, and another guy by the name of Glen Edens. I don't know if you know Glen.

Brock: I've heard the name.

Malloy: He went many places, but he was at Sun for many years. And so, in addition to talking to my engineering colleagues, I was also talking to them, and they were, I'm sure, picking my brain about Xerox and PARC and things like that, and so I started describing the mouse and the bitmap display. And it wasn't very long before Steve [Jobs] got involved in those conversations, and it wasn't very long after that that they somehow wormed their way into PARC for their famous demo. And it wasn't long after that that the whole project got reorganized again around building what we now know of as the Lisa.

Brock: Okay. And during this time when you're having these discussions in the earliest months of your time at Apple, what was your kind of--? What was it that you were supposed to be doing?

Malloy: I was supposed to be building a word processor--

Brock: Okay.

Malloy: Just like I did at every job for many, many, many years, many iterations. I didn't start out building word processors at Adobe when we get to that. In fact, I never built a word processor per se, but I built the guts of word processors into Illustrator and other things.

Brock: Hm. Well, okay, so it was very clear.

Malloy: Yeah. It was very clear.

Brock: And were there a set of other people in the Lisa group who were going to make other application programs?

Malloy: Yeah. Ultimately we had five applications, right? I don't remember whether or not we had team leads for all five of those at the time I arrived or not, but probably not. I don't remember which ones were there and which ones weren't. Bill Atkinson was there, but Bill, rather than working on an individual application, more worked on what we would think of as the application layer.

Brock: Right.

Malloy: Who else was there at the time? Well, Jef Raskin was not part of the Lisa team, but he was there, and he was having an influence, because he was imagining his vision for the Macintosh at the same time we were building the Lisa. I'd have to scrub my memory for who else was there.

Brock: Okay.

Malloy: Rich Page was definitely there. He was the hardware guy, or a hardware guy. And then there were some people who were there to develop an operating system, because this, unlike the Alto, the expectation from the start, it was going to have a real operating system running on it.

Brock: Well, let's see. So I suppose it was by early 1980 that the second reorientation of the Lisa towards the kind of PARC idiom, if we could say, that gets going. Could you describe then the unfolding of LisaWrite and building that?

Malloy: Well, I'm not sure I have any particularly memorable stories off the top of my head, but it went on for a couple years after that, right? We didn't ship until 1983, so there was lots of iterations and team growth and drama. Steve [Jobs] was very involved in the early days, but then he became involved with the Macintosh; then he wasn't involved at all.

Wayne Rosing came in after Ken Rothmuller to lead the team, hardware and software. We had the five applications; we had the application layer that Larry [Tesler] and Bill [Atkinson] were so influential and important to. The Lisa operating system was actually pretty heavyweight for what we were trying to do, and in those days it would be more like a macOS 10 or a OS/2 kind of operating system as opposed to the Mac operating system, which was more like a DOS, C or a single-user, lightweight, very little protection kind of operating system, so that was an extensive effort.

It was a controversial effort, because a lot of people, and maybe even me included-- I don't remember-- thought it was kind of overbuilt to the task. And even though we had a megabyte at that point-- I think originally Lisa was a megabyte of memory-- we were using it up at a furious rate. So those of us that came from a more minimalist background would have preferred a more lightweight operating system I think but be that as it may-- and so we just plugged away.

We had lots of problems to solve that are documented by history. There was the whole Twiggy thing. It needed a floppy disk, and for years it seems like, it was going to be this floppy disk that was going to be designed at Apple called Twiggy, which nobody could ever get to work, and eventually we switched over to a Sony smaller disk. So there was lots of drama around that, delays. I'm sure we weren't supposed to ship in 1983. We were probably supposed to ship in <laughs> 1981 or '82. I don't know. And at some point, the Macintosh team spun off, and there was more drama there. I don't know. You'll have to ask me something more specific I guess. <laughs>

Brock: Okay. Yeah, well, maybe we could just have you speak to-- you said that you used piece tables in LisaWrite.

Malloy: Mm-hmm.

Brock: Could you talk about just your comparison between Bravo, BravoX and LisaWrite, how your earlier experience shaped--

Malloy: Well, my earlier experience shaped not just LisaWrite but other things as well. Probably the first thing I did-- Charles [Simonyi] doesn't mention this, I think, as one of our more memorable accomplishments, but I actually think right up there with piece tables was the memory manager that we built for Bravo, which also was, I think, designed by Butler [Lampson]; I'm not sure, and it was a relocating memory manager. So the standard memory manager even to this day, malloc, is a fixed memory manager in the sense that, if you call malloc and ask for a block of memory of 100 bytes, you get a pointer to that, and that object or whatever you want to call it, that record is always at that address in memory.

Brock: Okay.

Malloy: Again, we had so little memory, that causes fragmentation over time as you allocate and then free things. And back at Xerox, we were so concerned, or Charles and Butler were so concerned about not wasting any memory, they designed this relocating memory manager. We call it the heap, but there's so many things called the heap--

Brock: That's what I was just going to wonder, because--

Malloy: Because so many things--

Brock: I've seen heap throughout the Bravo source code.

Malloy: Right, there's so many heaps today which are not relocating memory managers that the term has kind of been co-opted, but our heap was this relocating memory manager. And so instead of pointers, we had pointers to pointers as the reference to these records or objects or whatever you wanted to call them, and so there was only ever one pointer to the actual object, and that was inside the implementation of the memory manager. Everybody else used these pointers to pointers, and so that meant that you could actually compact the heap, move all of the objects so that they were contiguous, and then just update this one pointer, and nobody would be the wiser, basically, on the other side of the interface.

Brock: This is a form of indirect addressing, correct?

Malloy: Exactly, absolutely, yeah.

Brock: Okay, so I get it. You point into the memory manager, and it's figuring out all these smart things to do to keep things together and avoid fragmentation--

Malloy: Exactly, mm-hmm. Right.

Brock: And really be smart about where you're putting things.

Malloy: Right.

Brock: And the program doesn't have to worry about it.

Malloy: Well, the program has a pointer to that pointer.

Brock: Right.

Malloy: It's not as seamless as modern-day virtual memory, because it's all in software. Clearly, when that application software wants to reference the stuff in the object, it has to go through that pointer, and maybe it ends up putting that pointer into a temporary location, so you have to be careful about not having those dangling pointers. You can get yourself into trouble with this, but in the absence of hardware support like modern-day virtual memory, it was a very elegant solution.

So the reason I bring this up is because the very first thing I did pretty much at every job I've ever been at, I think, well, for some number of years, was I implemented a heap. For whatever application I was building, I developed a heap, an implementation of this heap memory manager. So I did that for LisaWrite, but actually, all the applications used that code, so that became part of the application environment that we associate with Bill and probably Andy Hertzfeld on the Mac side and other people. In fact, I saw in one of the many books written about the Macintosh that Andy Hertzfeld gave me credit for that, because he said that was the only- I think he said that was the only piece of software that ever actually came from the Lisa and was adopted by the Macintosh team. It was a very simple piece of software, but nevertheless. So I also developed a number of things that went into the sort of core application environment. But getting back to LisaWrite, LisaWrite I think was a fairly natural evolution from Bravo to BravoX to LisaWrite, so it had a modeless user interface like BravoX, had piece tables, and it had the heap storage inside of it. It had most of the features. It probably had a Boyer-Moore search algorithm in it.

Brock: That's what I was just going to ask, yeah.

Malloy: It had a user interface that was consistent with and inspired by the work that Larry [Tesler] was doing for the overall look and feel of the Lisa. It had one user interface element that I think I was the first one to actually design and develop, which is the "find and change" dialog for LisaWrite. I wanted to take the modeless notion as far as I could, and so this is not the same user interface that's used in every find dialog that you see, but it's used in many where you have a "find" and a "change" and a "change all," and when you hit "change," it goes and it searches to the-- you have find, change, change and find, and change all, and that was completely modeless as opposed to what you did before where you might say "find," and then you would say "replace" or something like that. And so that completely modeless interface was one that I developed, and I think LisaWrite was the first place that that appeared.

Brock: And that happened in a dialog window that opened up?

Malloy: Right, but it was a modeless dialog.

Brock: Right.

Malloy: When you create a dialog even to this day, you get to choose whether it's modeful or modeless. Most find-and-change dialogs were modeful in that you would put everything in, and then maybe you would say "change all," and it'd go off and do everything. This one, because we added those extra buttons, you could change one and then just be back in your document and continue, or you could change and find, but in all circumstances the dialogue box was modeless, and so you could go back and forth and edit in one or the other.

Brock: I see.

Malloy: What were the other interesting features of LisaWrite? It was kind of just like Word was kind of just like Bravo. <laughs>

Brock: Yeah. <overlapping conversation>

Malloy: And then the thing about word processors is there's a thousand features, but which are the ones that stand out? I don't know.

Brock: And when I talked to Charles Simonyi about the very first Microsoft Word versions of it, his description was kind of like by employing the fundamental structure of the piece table, kind of latent in there was the ability to-- once the kind of host system for Word became more capable, you could add more features to this, but it was a similar sort of task at the beginning of trying to shoehorn kind of a lot of functionality into a underpowered system. Was it the same for you with LisaWrite? Was it the same sort of shoehorning problem?

Malloy: Well, I think the more accurate way for me to put it was that, because of my history-- my kneejerk reaction was just to build word processors that way. In fact, to this day, my kneejerk reaction would be to build word processors that way, because even if you have lots of memory, it's an elegant solution. And we didn't think of the Lisa as being that small. A megabyte was pretty hefty. It was bigger than the amount of-- it was more than the amount of storage available, for instance, to an early Windows application on a 640K PC. So on the one hand, it was kind of a kneejerk reaction. The other data point is that it was abandoned by the Macintosh team, which had an even smaller computer, but they were not steeped in this philosophy, and they were content in the early days of the Macintosh to have very small documents, so they were also content-- who was it? Was it Andy who wrote MacWrite?

Brock: I don't know the answer to that, and I should.

Malloy: I should know the answer to that, too. I did. Whoever did MacWrite simply chose to have their document in main memory, and so even 128K Mac, which was I think how big the first Mac was, you could do a couple-page document and keep it all in memory, and they were happy to do that.

Brock: That's interesting.

Malloy: And in fact, I think they probably stuck with it for a long time, because Moore's Law let them get bigger and bigger documents without using this piece table kind of abstraction.

Brock: And just to clarify for somebody who may be listening or reading to this or watching this, the idea there is one way to handle a text editor or word processor is just to load all of the characters into the main memory of the computer--

Malloy: Yeah, exactly. Take everything in the file and read it into main memory, yeah.

Brock: And actively edit that, and then at the end, write it back to the storage.

Malloy: Mm-hmm.

Brock: Okay.

Malloy: But hopefully our viewers will see some of the inelegance of that even if you're not short on memory, because let's say that you want to insert a character at the beginning of your document. Even though it's in main memory, you got to take every character in the document and make space for that new character, whereas in the piece table case, you just have to make space in the piece table. So instead of the time to insert one character being a function of the length of the document, which it is if you keep everything in main memory, it's a function of the number of edits.

Brock: Right. Right, otherwise, if you follow this approach as with the MacWrite of having the entire document in the memory, each edit you're essentially having to do a transform on all the characters.

Malloy: Well, you're having to move the characters around, and if your document is only two pages long, who cares? And if you only have 128K of memory, your document's not going to be very big. So it was a good design decision for them. I'm not saying they did the wrong thing, but it's not the design decision I made, and I think the reason I made the decision I made was more because of the professional or intellectual culture that I was steeped in than a detailed comparison of the pluses and minuses of those two approaches. It's just like if you learn how to write with your right hand, then you don't get up every morning and say "Well, am I going to write with my right hand or my left hand today <laughs>?"

Brock: Right, it becomes sort of a design style.

Malloy: Yeah, exactly. Mm-hmm.

Brock: Yeah. Two questions. One is: For many people, working with and collaborating with Steve Jobs was very memorable. Was that the case for you?

Malloy: Of course. How could it not be?

Brock: Could you care to elaborate on that. What was it like in this period, '80 through '83?

Malloy: Well, I don't think I have anything to add to the mythology of Steve Jobs. He was incredibly charismatic and passionate, and he was the driving force that transformed the Lisa and then the Macintosh-- well, first the Lisa from this HP 100 style of computer to what we know of as the Macintosh today. We were both young. We were both in our 20s. We had a lot of fun together. But he was also the person that we read about. He was difficult. There's a word that we didn't use back then, but I think when you characterize Steve's weaknesses, it might be pretty accurate; he was kind of transactional. He could bring all of his charm and charisma to bear on you when he wanted something from you, but if he didn't want something from you, you kind of quickly faded into irrelevance.

Brock: Would you characterize sometimes some of his perhaps you saying whatever the inverse of charm is but sort of a negative charismatic interaction also as transactional? Sometimes people have described extremely critical--

Malloy: He was always extremely critical, because he had very strong feelings, and he was brilliant. So, the advice I always gave to people at Apple about Steve was that "90 percent of what Steve tells you is going to be bullshit, and 10 percent is going to be sheer brilliance, and your job is to figure out the 10 percent is." And I think that's true. He was equally passionate about the things that he was wrong about as the things that he was brilliant about. He only had one setting. <laughs>

Brock: Fully on or something? Yeah. Huh.

Malloy: Fully on, right. And I think that was good advice for new people coming to Apple. Well, I felt like it was good advice.

Brock: Right. I suppose it would be difficult advice to follow. You must have--

Malloy: I know. I was trying to give people difficult advice as to how to really take advantage of Steve without, on the one hand, ignoring him-- a lot of people ignored him at their peril**-- a**nd just being a sycophant, which I was never interested in. So the sycophants maybe ended up being more successful in life. <laughs> I don't know, but-- <laughs> so this is the way I sort of threaded that needle.

Brock: Well, I think I saw somewhere that you began consulting for Adobe in 1984. Is that correct?

Malloy: I think so.

Brock: Okay.

Malloy: So I left Apple in '83 after the Lisa shipped. Maybe mid '83? I don't really remember. And my model of my career, or a career in what we now think of as high tech, at that point in my career was you go off and you spend four or five years developing a hardware or software solution, because that's what I'd done at Xerox working on the Alto and Bravo, and that's what I did at Apple working on the Lisa and LisaWrite and other pieces.

And I didn't want to make that commitment of another four or five years, so I said "Oh, I'm going to work for myself for a while, just be a consultant." Coincidentally, we weren't married at the time, but my current wife, who we were together-- we wanted to take this long vacation, and I didn't want to get involved in a project where they were going to give me a hard time if I wanted to take a month or two off, and so that was also part of the motivation. So I did consult for a while, and I don't think Adobe was my first client, but I always refer to them as my sugar daddy client, because I knew everybody there. I knew Chuck [Geschke]. I didn't know John [Warnock], because he came after I left PARC proper. I knew Dan Putman really well. I knew

Doug Brotz and lots of other people. I knew Chuck really well, because he was the guy who managed me and my college roommate going way back to when I was still in college and I came in and did the printing at night. But actually, the person I knew best was Dan Putman, and I could pretty much walk into Adobe in those days and just sort of wander around and say "I've got some time. Is there anything you need doing?" and somebody would think of something.

Brock: Okay.

Malloy: One of the things we didn't talk about in my career at Apple was that I was in charge of networking software for a while. The Lisa was supposed to have a hardware network called, if I remember correctly, AppleNet, and I advocated to adopt the Xerox network protocols, XNS protocols, as our network layer, and, I don't know, maybe it was one of those situations where, if you say something needs to be done, you end up being told to go do it.

But in any case, I ended up in charge of a small group of networking software people for a while, and so we were in the process of implementing the XNS protocols on top of this AppleNet hardware. Never really came to anything I don't think, because AppleBus came along with the Macintosh, and I think Lisa adopted AppleBus, and the XNS was not adopted, which in hindsight was probably the right decision, because TCP/IP came along and sort of swept it away.

But the reason I bring that up in the context of my consulting is that, since I had gotten involved with networking at Apple, I managed to find some consulting work in the networking space, and so I developed some TCP/IP protocols for various people. Ungermann-Bass was one of my clients at some point. This was a period that lasted from '83 to '86, so somewhere in there I worked for Ungermann-Bass, and I worked for Stanford for a while in their networking group and various other people. As all consulting is, it's just piecemeal, whatever you can find.

But whenever I couldn't find something easily, I would go over to Adobe. So the first thing I remember doing at Adobe was I ported PostScript to the Lisa, believe it or not. Now, why would you port PostScript to the Lisa? It's a piece of software that's designed to run on printers and typesetters and things like that, image setters. Well, turns out that there weren't any devices whenever this was, '84 I guess, probably '84. There weren't any LaserWriters to be had. They didn't have the first image setter hardware up and running, which was from a company that I don't remember. And they, "they" being Chuck and John and the other people at Adobe, thought that PostScript was going to be this language that lots of developers used. Now, in fact, in hindsight, the only people that ever programmed in PostScript were printer device driver writers, but at that point, they thought that they wanted to get the language out so people could start programming.

And so the Lisa was actually a pretty good surrogate for the hardware that was going to be the LaserWriter. The LaserWriter had a 68000; the Lisa had a 68000. LaserWriter had a megabyte of memory or something; Lisa had a megabyte of memory. So they had this idea. It wasn't a profound part of their strategy. It was just like “Wow, wouldn’t it be nice if we could have PostScript running on the Lisa, and then we could hand it to people so that they could start writing PostScript before the LaserWriters showed up?” And so I did that, and we did get it running, but for reasons I can’t remember, it didn’t run very well. I think there were some hardware issues or something. I don’t remember. In any case, it went on long enough that, by the time we got it running, the first LaserWriters had started to show up, and then people just used that, so I don’t think anybody ever really used that for much of anything. Then subsequently-- I think this was still while I was a consultant; I’m pretty sure it was-- I ported it to the IBM, PostScript again, to the IBM PC, and this was pretty much in support of the sales team, who was trying to get IBM to adopt PostScript. And again, it’s wasn’t that they thought that the IBM PC was ever going to run the PostScript language in that form, but IBM was going to develop printers, and inside the printers were essentially going to be PC hardware of some sort, and so IBM wanted to know if you could run PostScript-- as part of the negotiation for this OEM contract, they wanted to know if they could run PostScript on the PC, so I got it up and running on the PC.

Brock: That’s really interesting, because in both cases of putting it onto the Lisa and putting it onto the IBM PC, it really does highlight the fact that, in these laser printers, they were powerful computers that happened to print, and it really does--

Malloy: Well, the LaserWriter was more powerful than the Macintosh at the time it came out I’m pretty sure.

Brock: Yeah--

Malloy: They both had 68000s, but it had more memory. The truly extraordinary thing about the LaserWriter, or the radical thing about the LaserWriter was that it had this huge program in read-only memory. It wasn’t programmable read-only memory. It was read-only memory that, once you ship that printer, if there was a bug in the PostScript language, <laughs> it was going to be bad news.

Brock: <overlapping conversation> yeah.

Malloy: So that was pretty radical at the time. I don’t remember how much read-only memory and how big the code was, but it was at least hundreds of kilobytes. Lot of code; many thousands of lines of code; much more complex than people were used to putting in a nonvolatile, non-changeable memory. So I did projects like that. I don’t remember what else I did as a consultant, but in spring/early summer of 1986-- my hire date was in June, so it happened somewhere before that-- two things happened. First of all, the way I describe it is that the rabbit died. You know what that expression means?

Brock: No. Oh, oh, yes, for a pregnancy test? Yes, yeah.

Malloy: Yes, right. Not very many people know that expression anymore, but any case, my wife became pregnant. We were married in '85; she became pregnant in '86. And I had already taken a part-time permanent position at Stanford in the networking group so that we could have health care, but it was like "Okay, I'm going to have a kid. I'd better get a real job with health care and blah, blah, blah, blah, blah, blah, blah." So that was the first thing that happened: the rabbit died. The second thing that happened is that Chuck came to me, and he said "Tom, we're going to be going public here pretty soon, and if you think you might ever want to join Adobe as a regular employee, this would be a good time to do it." <laughs>

Brock: Very nice of him.

Malloy: Yeah, yeah, right. So the stars aligned; I became a full-time employee. My daughter's 32 now? 31? Let's see, 1987? 31 I guess. That went all went fine.

Brock: Good.

Malloy: And then I spent the next 26 years at Adobe.

Brock: Well, what were your initial roles within Adobe, the first things that you were working on? I saw some indication that it may have been typefaces. Is that wrong?

Malloy: No, it's true.

Brock: Okay.

Malloy: But the thing I used to say about my early days at Adobe was that I was the only person at Adobe that could spell PC, because in my consulting-- I'd worked at Xerox on the Alto, I worked at Apple on the Lisa, and none of that had anything to do with the IBM PC, but the IBM PC is happening over this time, right? Windows 1 probably came out in about '83 I would think, wasn't it? '81?

Brock: It shipped in '85 I think?

Malloy: Windows?

Brock: I could be wrong. I could be wrong.

Malloy: Maybe, maybe, because I wasn't working on Windows in my consulting. I was working on DOS, but that networking work that I did was on DOS, and other contracts I had, I was-- I had a PC in my home office, so I knew how to navigate around a PC, and nobody else did. PostScript was developed first on a VAX I think and then on Sun Workstations, and anybody that wasn't working on PostScript was probably working on the Macintosh because of the LaserWriter and the Apple connection. The typefaces, which were the second product after PostScript, were first released on the Macintosh. When I arrived in '86, Mike Schuster was working on the first version of Illustrator for the Macintosh. So since I knew how to turn on a

PC, my first jobs were in taking things that were running on the Macintosh and getting them running on the PC, so the first thing was the typefaces. There wasn't much software associated with the typefaces, but we had to have software to download to the LaserWriter and to the typesetter, image setter, and probably some installation software. Who knows? It was very simple stuff. So my first product release at Adobe was the typefaces for the IBM PC. And then I went on from there, and my next project, which was a big project, was to try to get Illustrator to run on Windows, so I ported Illustrator to Windows, and I built a team to do that. And the first attempt at doing that was, from a computer science point of view, it was extremely elegant and ambitious, and from a practical point of view it was totally useless.

Male: Why?

Malloy: Because this was-- I don't remember what version of Windows, but it was an early version of Windows still running on 640K PCs. They had this hack where you could have more physical memory than the 640K, but you had to map it from the application into this one or two 16K blocks up above 640K. Maybe there were four of them. I don't remember. In any case, not much memory. And so we have Illustrator; it's running on a one megabyte Macintosh, and I'm not sure the bitmap for the Macintosh was actually in that one megabyte. It might've been in some other part of the address space. I'm not sure about that. But in any case, now we've got Illustrator, and we want it to run on this IBM PC with 640K, and it's got to have Windows running. It's got to have DOS in there; it's got to have Windows in there; it's got to have a bitmap for the display in there I think. I'm not sure. Anyway, it was maybe half the memory space of the Macintosh when all was said and done. No, of the Lisa, excuse me, of the LaserWriter. I'm getting confused here. How did that work? I guess it must've been the-- so Illustrator for Macintosh was released in '86, so by then, it had to be at least a 512K Mac.

Male: I think that's how they were.

Malloy: Yeah. So we'd have to go back and resurrect the numbers. Anyway, there wasn't as much space for Illustrator on a PC running DOS and Windows 2 or 3 or whatever it was as there was on the Macintosh. So I huddled up with John Warnock, and he clued me in to this data structure that he'd used on flight simulators. In a flight simulator, you had this huge world, right?

Brock: Oh, yeah, he had a table, too, didn't he? Is that--

Malloy: Yeah, right, yeah, but it's a different data structure entirely. It's where you carve up either the 2D space into quadrants and recursively divide the quadrants. He was doing it twice-- so he had a 3D space into I guess you'd call them octants. Anyway, I said "Well, John, what do you think about what-- maybe we'll take the Illustrator image," which could get fairly complex, "and we'll carve it up into these quadrants recursively, and then we'll put it on the disk sort of like we did with word processor documents, and then we'll swap in the pieces that we need to display, just the piece that's on the screen." And we got that working. It was a very elegant data structure I think. The problem is that, unlike flight simulator worlds, a very typical use case was to want to see the whole Illustrator picture, <laughs> at which point it was way slower, <laughs> because you had to read everything off the disk. Basically it just thrashed.

Brock: I see.

Malloy: And so I'm not sure that version of Illustrator ever saw the light of day. Maybe it did, maybe it didn't. I think maybe we shipped it, but it was a miserable failure. So time passes; Moore's Law acts on the world, and the amount of memory on PCs gets bigger; Windows develops into sort of a real live operating system layer, and it can map memory, and I'm sure virtual memory came in there at some point. And I don't remember what the dates were, but it was probably more like '89 before we had a viable version of Illustrator on the PC that really worked well. Illustrator 3 is what I remember on the PC. Illustrator on the Macintosh was being extremely successful all along, but the one for the PC really didn't take off until about Illustrator 3.

Brock: And is that what you were focused to in that period? Illustrator for the PC?

Malloy: Probably at that point, yeah, because the first big pivot for me at Adobe was Adobe Type Manager. So there's plenty of drama around this story. Oh, in '88 or '89, Apple announces that-- so we had this typeface format that's called Type 1, and it was only for printers and image setters. The screen was still using bitmap fonts on the Macintosh and on Windows. Obvious desire is to have scalable fonts on the screen, but at the time it's a very hard problem, because it was hard to get quality-looking type on 300 dot-per-inch laser printers, and now we're talking about getting quality print on 72 or so dot-per-inch screens. So we were feverishly working to adapt the software that renders Type 1 fonts to work on the screen, expecting that, well, our PostScript is a emerging standard, Type 1 is going to be a standard as well, and that's going to be wonderful for Adobe, but Apple didn't want that. They didn't want our format running on the Macintosh. They didn't want our software running in their operating system. Again, you'd have to ask somebody like Chuck and John to reproduce the history of the interactions. They were actively trying to convince Steve and the powers that be at Apple to adopt Type 1 and take our typefaces, and we had dollar signs in our eyes. It was like "Well, they're paying us a royalty for the LaserWriter now. What if they pay us-- even if it's a small royalty, what if they pay us a royalty for every Macintosh?" Well, you can see why maybe they didn't want to do that. So this goes on-- I don't know, let's say it's happening in '88. I don't know when it's happening, these discussions. And there's a famous Siebold conference where Apple gets up and announces that they're going to create their own scalable type format, TrueType, and I think even at that early meeting, they had already lined up Microsoft as well--

Brock: As well, yeah.

Malloy: To embrace the standard with them. So now all of a sudden, the two major personal computing platforms were going to have a new scalable type format, but it was not our scalable type format. So John had a very emotional reaction at that Siebold conference, and he came back, and basically he and Chuck said "Well, we have to get our software running and our typefaces running on the Macintosh ASAP. We have to beat them to market," and they asked me to lead that effort. And it's one of those cases for an engineering manager, which is pretty idyllic, where actually you get the golden finger. You can walk around the company, and you can point at anybody and say <laughs> "Guess what? You're on my project now." It's more complicated than that. Obviously people were involved, and they didn't like being pointed at with a golden finger; then you had to deal with that, but basically, from the point of view of Chuck and John, I had a golden finger. Just go find anybody in the company to make this happen ASAP, so I pulled people from my team. Mike Shuster joined us. Bill Paxton was part of-- I don't think he was ever formally part of the team, but he's the one that came up with the initial enhancements, I think, to the rendering algorithms to make them work at 72 dots per inch more or less as well as they did at 300 dots per inch. I subsequently made some additions to those algorithms that we patented and got factored into the core algorithms. So that was one big piece of the puzzle, but there was this other big piece of the puzzle, which is probably pretty foreign to people today because it was virtually impossible to do, but we were a third-party software developer, and we wanted to basically add something to the Macintosh operating system. Think of it that way, right? Because we didn't just want our applications to get the benefit of scalable fonts. We wanted any application that used fonts to have those typefaces rendered at any point size using our software and our scalable fonts, so that meant we had to basically splice ourselves into the Macintosh OS without the cooperation or even over the objections of Apple, because <laughs> they have a different strategy. So we had to reverse engineer the Macintosh operating system and figure out how to find the places where applications were calling to have things rendered onto the screen, patch those places, and then have our software take over at the appropriate times or defer to the bitmaps if that made sense.

Brock: So if I were a Macintosh user and I installed this--

Malloy: Adobe Type Manager, yeah.

Brock: Adobe Type Manager, would I be altering my system?

Malloy: Essentially, yeah.

Brock: Okay.

Malloy: Yeah. You could think of it as sort of like installing a device driver--

Brock: Okay.

Malloy: Except we weren't using--

Brock: An agreed-upon--

Malloy: An agreed-upon device driver interface. <laughs> We were just hacking into the system, so it's more like a virus <laughs> in some ways, right?

Brock: Yeah, sure.

Malloy: So figuring out how to do that-- nobody knew how to do that. Certainly Apple wasn't going to help us figure out how to do that, so there were two big pieces to the Adobe Type Manager project initially. One was to revise the algorithms to get them to work at lower resolutions acceptably, and the other was to figure out how to hack into the operating system and install ourselves as a virus. And we did both of those things, and we shipped that product in something like six months, incredibly fast. I think you'd have to go back-- yeah, we'd have to go back and look at the history books, but it was a-- and we beat them to market, and Type 1 fonts were the more important typefaces on the Macintosh until TrueType really matured.

And if you fast-forward a long time-- I don't know how many years, a decade maybe-- as a result of that, Apple and Microsoft as well had to support Type 1 fonts even when all these hacks got removed and even when we weren't supplying them with the software to do it anymore. They had to interpret those fonts, because the industry was dependent upon them and expected them. So we won in that respect. We succeeded in salvaging our format as a standard. It was a secondary standard over time as TrueType matured, and as a business, the existence of TrueType and the business motivations of Microsoft and Apple, i.e., they didn't really care about making much money off of typefaces-- in fact, they didn't; they bundled their typefaces-- it kind of eviscerated the typeface business, which is what we were interested in. We weren't doing this because we wanted to have our software running in the Macintosh. We were doing it because we wanted to sell typefaces-- that's what we made money on-- and then have them used on the screen. So at its peak, the typeface business for Adobe was probably tens of millions of dollars a year, but it's one of those businesses that came and went, clearly--

Brock: But it still continues to this day, does it not?

Malloy: Yeah, but--

Brock: Not with the--

Malloy: But it's not an important product line for a-- what is Adobe? Six billion-dollar company today?

Brock: Yeah, who knows? Yeah, yeah.

Malloy: So it's just one of those sort of legacy things that will never die I don't think but became immaterial to the business fairly quickly. 20 million dollars a year in 1988, '89, '90, that was probably a percent or two of the business. I don't know.

Brock: Right. Well, my impression is that this Adobe Type Manager project was a huge priority for Adobe at the time.

Malloy: It was, yeah.

Brock: Maybe this is wrong, but it didn't turn out that long-term financial benefits of the success of Adobe Type Manager didn't-- it maybe wasn't that the upside was so huge but that, at the time, circa 19--

Malloy: It was an incredible threat.

Brock: That's right. The downside would've been so dramatic.

Malloy: And it could've threatened the PostScript business, which was much larger and is probably still larger today. I don't know.

Brock: Why do you think that they asked you to shoulder the burden to get this thing done?

Malloy: <laughs> I don't know. You'd have to ask Chuck and John. I was good at my job. <laughs>

Brock: Were you surprised? Well, yeah, right. I just wondered if you had-- it's a big job to get, and I wondered did it surprise you that they asked you, or did you feel like "Yeah, that's right. I'm the one to do it"?

Malloy: No, I don't think it surprised me. I was still young and cocky and ambitious. The one that surprised me comes later when they asked me to take over the Advanced Technology Group. We'll get to that eventually, but that one surprised me.

Brock: Well, I wanted to ask-- so you had I suppose this kind crash project or this intense project and playing it out during the Font Wars so to speak. I was interested in where PDF starts to emerge in this milieu, or what was your involvement with--

Malloy: I was not very involved with PDF in the early days.

Brock: Okay.

Malloy: So that would've been the early '90s I think, or--

Brock: Yeah, '93?

Malloy: Yeah. So now my grasp of dates might get a little looser, but whenever they happened, I went from Adobe Type Manager to being in charge I think maybe of all of the application software. I'm not sure from an engineering perspective. That wasn't much then. It was Illustrator and typefaces and Adobe Type Manager I think. But somewhere in the early '90 timeframe, I took over all of type, not just engineering but production as well, and that was a fun job. So during that period, in addition to managing all the Adobe Type Manager stuff from a software perspective and managing the typeface library, we were building type tools to develop Asian fonts, Japanese fonts first and then Chinese fonts, and so that was exciting and challenging, and I got to work with designers and production people. That was a fun variation on my normal work. And it was during that period I think, when I was in charge of the type group, that John and Bob Wulff and a bunch of other people were doing the first version of Acrobat. Bill McCoy.

Brock: Yeah, that would make sense.

Malloy: Yeah. So I really didn't have much to do with it. Then my involvement with Acrobat started in the mid '90s or even late '90s, probably mid '90s. So, I just said wonderful things about running the type group, and they're all true, but for a technologist, it's kind of a dead end. We developed these tools; we developed these rasterizers. The way I describe type technology or the type business is that's it's a 400-year-old business that's had a few technological discontinuities, three or four, Gutenberg being one and scalable type and raster-based image setters being another, and there weren't that many in between, and the periods of innovation were not that long. So, we sort of solved the problems of scalable type for all practical purposes between when Adobe was founded in '83 and maybe '93 or '94 or '95. There just weren't that many interesting technical problems to solve anymore, and so I found myself in this career juncture of, well, I'm getting more and more management responsibility, but I'm not really challenged very much technically, so I made what I think in hindsight was a fairly radical choice.

I turned over the typeface management to somebody else, and I went back into the Advanced Technology Group that was being run by Jim King then, essentially as a free electron, unsigned, individual person, and it was just a place where they let me park for a while while I figured out what to do next. And what I chose to do as an individual was develop what we today would think of as the security features of Acrobat. First of all, I went off with a marketing guy and licensed all the crypto libraries from RSA, and then I built two suites of features inside of Acrobat. One was for doing encrypted secure documents but with distribution lists. So I was using public key cryptography, which was pretty cutting edge at the time, to attach a distribution list to a document in such a way that only the people on the distribution list could open the document, and that was pretty radical and pretty innovative for the time. And the other feature was signed documents, which are used to this day, and again, I used public key cryptography, which is why we went off and licensed the RSA libraries, because if you were just going to do a password encryption on a document, you could implement your own encryption and decryption algorithms, but to do public key, you'd want somebody who had really pored over that code. So we licensed that code. I implemented those features as a individual contributor inside of the Advanced Technology Group, and then the Acrobat team picked those up and put them into the product. Probably re-engineered the whole thing before they did it. I don't know.

Brock: Could you speak a little bit about how you became focused to these issues of security at this moment?

Malloy: Again, it's all just serendipity. When I was a master's student at Stanford, I took a graduate study group, kind of reading group seminar from Bob Metcalfe, and we read papers about networking stuff, and one of the papers I read was Diffie and Hellman's paper <laughs> about public key cryptography. So in the back of my mind I knew a little bit about this public key cryptography, and I thought “Well, this is a good match for me.” It’s a meaty, technical problem. I had a vision for how it would fit into the Acrobat application with these two features. I had enough management expertise or pluck or whatever you want to call it to not be intimidated by going off and licensing the library and wrangling this marketing guy to go with me to Redwood City and tell them we wanted to license this library. And so I did it.

Brock: And I suppose it’s also kind of inherent in the very notion of PDF. What is that for? Distributing documents electronically, so it fits perfectly.

Malloy: Oh, yeah, it fits into the vision for PDF for sure, but nobody else was making that connection in ‘95 or whatever it was. ‘96? I don’t know. So it was a project to keep me busy, and it turned out to be really successful.

Brock: That time when you were working on that as an individual contributor, did that also give you the opportunity to kind of see everything else that was going on in the Advanced Technology Group?

Malloy: Yeah, sure.

Brock: Yeah.

Malloy: It wasn’t that big a group.

Brock: Okay.

Malloy: And I think I was aware of it before, but it was a pretty small group at that time.

Brock: Oh. I hadn’t realized that, that it was--

Malloy: Well, the original vision behind the Advanced Technology Group was the following: Company was founded by these world-class researchers from Xerox PARC, like Bill Paxton and Doug Brotz and other people of their ilk whose names I should remember. Ed McCreight was there by then I think. And in the startup phase of the company, it’s all hands on deck. You do what has to be done, so these world-class researchers were basically talking on the phone to OEM customers and fixing bugs in PostScript. And so as soon as they could afford it, I think Chuck and John wanted to have a little walled garden where these people could go off and maybe think about something <laughs> a little more substantive than how to fix the latest bug that was preventing the last OEM to call from shipping their product. And so that was the vision, and there weren’t that many people at that time like that, so that’s where those people went, and they worked on the things that were important to them. That’s where Bill Paxton was when we did Adobe Type Manager, and since he had done the original rasterizer for PostScript, type rasterizer, it was there that he developed his improvement for the screen rasterizer. I don’t remember what Doug Brotz was doing or Ed McCreight, but it was just their little walled garden. I’m sorry, Jim King came at some point maybe in the early ‘90s? I don’t know. And he hired some more people, but that was the sort of founding principle. So it was probably bigger than just that handful in ‘95, but not that much bigger.

Brock: And was it right after you kind of handed off the security work with PDF to the Acrobat team? Was it at that moment that you were asked to lead the Advanced Technology Group?

Malloy: It was more or less at that--

Brock: Okay.

Malloy: And I don’t remember anything significant happening between those two events. Yeah, actually I was lobbying for a different job.

Brock: Which was?

Malloy: I wanted to be the VP of engineering, and I think it was for the whole company at the time, because I think they were formalizing a functional organization for the whole company before there was any divisions. And I was lobbying for that job. Smokey Wallace ultimately got that job. And John came to me, and it seemed like kind of a consolation prize at the time. He said “No, I don’t want you to do that. We’re giving this job to Smokey. I think he’s better qualified for it, but I want you to go off and lead the Advanced Technology Group,” so I said “Okay.” <laughs> I never thought of myself as a researcher, and it was called Advanced Technology at the time, because we didn’t really call it a research group, although it is very much a research group now and evolved that way during my tenure. But that’s how I ended up with the job.

Brock: And you were in that role for--

Malloy: Until I retired.

Brock: Until you retired.

Malloy: Yeah.

Brock: Which was in 2000--

Malloy: ‘13.

Brock: ‘13. Well, on the one hand, listening to you talk about it as a walled garden for these highly skilled people, that sounds like a leadership challenge to me. Well, I guess the whole thing about sort of research management is a continuing conundrum that groups grapple with. How did you--

Malloy: Right, and we had for the time I think a fairly unique perspective on it.

Brock: What was it?

Malloy: So when I would talk about managing research groups inside and outside of Adobe when I was in that position, I would say that all research groups have the same top two priorities. One priority is to advance the state of knowledge in your discipline, be at the cutting edge, probably have good working relationships with academic research, things like that, so it's what I call the outward-looking priority. The other priority is an inward-facing priority of being relevant to the company that's supporting you in a corporate research context, and that means having an impact on their products and services.

So I say everybody has the same top two priorities. How do you distinguish between any research philosophy or practices based on that? And I said "Well, the way you do it is you turn to the research manager, and you say 'Okay, I understand you have those top two pr-- but what's your top one priority?'" And research groups always fall I think pretty much into one of two camps: Either the outward focus, research focus, advancing state of knowledge, working with academia is the top priority and being relevant to the company is a secondary priority or vice versa. And if you looked at the prominent research groups of the time, most of them were of this kind; the outward-facing priority was the most important: IBM Research, HP Labs, Microsoft Research probably even in those days, certainly during most of its history if not to this day. That was their priority. Sun Labs was a little different. I don't know. Intel Labs if it existed I'm sure was of this kind of priority.

We very deliberately reversed those priorities, and our top priority always was to be relevant to the company. And I think Chuck and John established that philosophy based on their experience of being in a research organization and then feeling that they needed to go leave that research organization in order to do what they wanted to do to have an impact on business and society and get their research results into the hands of people. And I think I took that vision to an extreme in the sense that it was my mantra throughout my entire dozen or so years of tenure that, both internally and externally, this is our priority. And so you talked about managing these very talented, perhaps high-strung technical people, and I was straight with them every day in and out. It was like "I love what you're doing, but it's got to be relevant." Doesn't have to be relevant tomorrow. It has to have an impact on the company. My timeframe is three to five years. If you've been working on something- if you've been working in the advanced technology group for years and you've never transferred any technology or you never had an impact on a product or people on the product teams don't come to you for advice, then you're not succeeding.

And so we just inculcated that in the culture. People knew it when they got interviewed. People knew it when they got reviewed, when they got raises, when they got stock options. And so that was our culture, and that's why my challenges as a vice president of research, though we didn't call it that then, of Advanced Technology, my challenges were the opposite of most of my peers' in the industry. I think they would view their biggest challenge as justifying their budget to corporate and trying to get their people to do something that corporate thought was valuable as opposed to just hobnobbing with their friends at universities, collaborating with their friends and colleagues. Certainly I had to keep beating that drum, but it was so ingrained in that almost everybody in my group was always looking for those opportunities, and my challenge was to make sure that we didn't fall into just being the 13th developer on a project. And I don't think we ever did, but that's what I would think about.

So that was our philosophy. We were inspired by Bert Sutherland at Sun Labs. I think he might not have been quite as militant as me about it, but I think he was very attuned to being considered useful to Sun. I had a very early meeting when I first took over the Advanced Technology group with him, and he sort of gave me some pointers, and we incorporated that into our practices and philosophy. And then, over the course of a decade or so, we actually were able to evolve all of the outward-facing strengths that perhaps we had de-emphasized or not emphasized in the early days. So David Salesin was a key hire that I made, say, early 2000s I guess, I don't know, and he came in and really codified for us how to work with universities in an effective way. And I think we had again kind of a different approach to working with universities than--

Male: Could you describe that?

Malloy: Well, what were the differences? The first and most important difference was that we insisted there be an honest and bona fide collaboration, so money changing hands was of zero interest to us unless there was a collaboration. So our collaborations tended to be s- our university collaboration projects tended to be smaller scale, but we had more skin the game, and we weren't like an Intel that was going and dropping a hundred million bucks on Stanford to build a fab or something like that. We were identifying like-minded professors and students to our own advanced technologists and researchers, and they were working hand in hand, elbow to elbow on some interesting project. Not necessarily a project that was going directly into a product, but nevertheless they were working together on it. So collaboration was the first key.

The second thing that I think that David figured out that was critical was the intellectual property model. If you are a research manager and you want to collaborate with a university, your worst nightmare is getting their lawyers or your lawyers involved in setting up the collaboration, because it's guaranteed to come to a screeching halt and maybe never happen. This was David's inspiration. What we did is we convinced our lawyers that we didn't need an intellectual property agreement with any of our university collaborators and that we were going to be just fine, because every collaboration we did, every project we did with a university was a bona fide collaboration with somebody that was employed by Adobe, and therefore the fruit of that research project, the intellectual property fruit, which is almost certainly going to be a patent, first a paper, then a patent, was going to have an Adobe employee's name on it. So we didn't have exclusive rights to the work that happened in university collaborations, but neither did anybody else, and we had equal rights to anybody in the university, and we convinced our lawyers that this was a valuable trade-off to make. Give up the exclusive rights in exchange for them not having to negotiate with the university's lawyers' impossible terms for sharing intellectual property that didn't exist at the time that the lawyers were involved, and therefore they were going to bring all of their expertise and power to bear to make sure that every unanticipated consequence was accounted for. So that was the second thing.

Then a third thing that I think we got right is we really focused on an internship program that was unparalleled at least at Adobe. In the Advanced Technology Group, when we were, say, 75 people in, say, 2010 when this was sort of at its height, we might have 40 interns. There might be 30 interns, because Adobe didn't have a great internship program at the time company-wide, but we would have maybe 40 interns, and that had some obvious benefits and maybe not-so-obvious benefits. The first obvious benefit is it was great recruiting tool. The not-so-obvious benefit is it played into this intellectual property strategy that we had, because now the graduate student, who in fact is the one that does all the work anyway in the university, is spending a significant part of their time on the Adobe payroll, and so the energy they invested, the intellectual property they developed while they were an intern was Adobe's exclusively.

Male: Interesting.

Malloy: And so now, the intellectual property conundrum was solved. In an even more powerful way, the lawyers were even happier. And we really believed that supporting the students was more important than supporting the university or the faculty members. Faculty members didn't usually agree with that. And they were-- so as such a commodity financially in the university community, we would pay them 100- in 2010 we'd pay them 90, 100 bucks an hour for 10 weeks, and they'd make more money in their internship than they did in their whole year as a graduate student. So we thought we were doing a real service to the community that far outweighed what the professor really wanted, which was for us to give him tens of thousands--

Brock: Right, a grant.

Malloy: Or hundreds of thousands of dollars to run his lab. We did some of that, but not nearly as much as hiring interns. So that was the third or maybe the last big innovation we had in terms of university collaborations, and credit for that all goes to David.

Brock: That goodwill multiplier for those graduate students from that treatment must've been unbelievable.

Malloy: Yeah. Yeah, it was fabulous, yeah. But we could only do that as our staff in the Advanced Technology Group matured enough to want to go back to having those university collaborations, because we kind of beat it out of them in the '90s, right? It was like, well, if you come to Adobe, you're leaving research and becoming an advanced technologist. But then we sort of had to reintroduce that and let them know that it was okay, university collaborations were valued, and we wanted them to mentor graduate students and collaborate with faculty members. It took us a while to sort of get that back, but we did, and in all modesty, I think it was the most powerful research model that I've ever come across.

Brock: <overlapping conversation>

Malloy: Much more powerful than sort of the standard one where a poor research manager's constantly being beat up by executives about how much money he's spending and by product teams saying "We don't get anything from you yahoos over in research."

Brock: Well, listening to you, it's an interesting way to-- because it's not you telling other people "Oh, I think you should go in this direction or that direction to be directly relevant to the others in the company"--

Malloy: No <overlapping conversation>

Brock: But rather saying it's up to you, dear member of the group--

Malloy: That's right, because--

Brock: To go out and find the thing that comports with your curiosity and inclination and talent.

Malloy: Exactly. Exactly, because they're senior people. You can't go and assign them to create the next innovation for Photoshop. Either they're into it or not, and if they are, then they'll find that innovation as long as you have the sort of infrastructure to help them meet the people on the product teams. I had lab managers who were really charged with trying to do this matchmaking process. But, yeah, the individuals got to choose themselves, and they got what I thought was a fairly liberal timeframe to demonstrate success. And it was extremely, extremely successful from my point of view, because I don't think anybody ever had an easier time during budget planning than I did, because the product teams by and large would go-- when I was sitting on the executive staff, the business unit managers would look at my budget and say "Is that enough?"

Brock: Yeah, enviable.

Malloy: Now, I got to be honest. We weren't as successful in all disciplines as I would've liked, but the general model was those folks in ATG, they produce-- in the context of then a four billion-dollar company, it was a pretty insignificant amount of money, I don't know, 10, 20 million dollars a year. I forget. And we just had all these advocates in their engineering teams. It was like "Don't mess with those people over in ATG. They're the ones that produced this feature for Photoshop and this feature for Acrobat and this feature for Premiere."

Brock: Also, being very diligent, I would think about protecting, if you will, the poor people in the product groups from that kind of 13th developer or whatever you call that, this super developer that has a lot of maybe weight and status in the overall company to make it-- yeah.

Malloy: Well, that was part of the culture, too, right? Because my people were senior, because I gave them all this latitude in exchange, what I demanded as part of that is it's your responsibility, the researcher, to figure out how to get your technology into the products. Some of you are going to have a very easy time. Metaphorically speaking, you're going to set your work product outside your office door when you leave, and when you come back in the morning, some earnest product engineer is going to have integrated into the product. God bless you; I'm glad it was so easy, but some of you are going to have to work hard, and you're going to have to sit side by side with the product team through the first release and maybe the first couple releases and make it work, and if that's what it takes to be successful, that's what I expect you to do. In many things, I was very undemanding. What you work on, how long it takes to prove viability, I think I was very flexible and patient, but in terms of having people committed to the end result, I was very demanding and very unforgiving.

Brock: That's fascinating. I wanted to ask you in the time we have remaining about maybe-- well, three questions. One is about the culture of Adobe, which John Warnock and Chuck Geschke talk a lot about, and I just wondered your take on the story of Adobe culture, its importance, its--

Malloy: Well, I don't think I would've stayed at Adobe for 26 years if it had a culture like-- should I name names? <laughs>

Brock: It's up to you. Other companies.

Malloy: Other companies. Well, from our generation, Oracle was probably a very contrasting culture, very kind of hard-nosed and unforgiving and demanding. I think Chuck and John are two very humane individuals, and they brought that humanity to the culture of Adobe, and it meant it was a nice place, or even when there were fights and feuds and disagreements, there was an understanding that we were all on the same team. Chuck used to have this expression, "Everybody sweeps the floor at Adobe." I don't think everybody did sweep the floor at Adobe, but it sets a tone. And so it kind of took the air out of a lot of prima donna behavior or privileged behavior that just made it a nice place to work.

Brock: And I also wanted to ask you if you could provide some characterizations or reflections on the two of them, John Warnock and Chuck Geschke, just impressions, observa--

Malloy: Well, they're just two of the finest individuals that I think I've ever known. As I said, I just described them as two very humane individuals. I could try to find more adjectives, but they're just genuinely good people, and even when they were being executives and making tough executive decisions and competing aggressively, their humanity was always right there close to the surface. They just inspire loyalty and trust, and as I said, I can't imagine having worked there for 26 years if it wasn't for them and the culture that they established.

Brock: Well, then perhaps I'd like to ask you a little bit about your life outside of Adobe, just your other interests and activities outside of the context of work--

Malloy: Well, I don't have a Adobe life anymore. It's been five years since I retired.

Brock: Right. Well, but just over the course of your life, have there been, and today, any particular passions or interests?

Malloy: Well, there's work, family. It's kind of the usual list of things. Adobe had a very nice policy that came about in the mid '90s when we acquired Aldus. Aldus had a six-week sabbatical program every five years; Adobe had a no-week sabbatical program, and so the compromise was that we ended up with a three-week sabbatical program at Adobe, which is true to this day I think. Every five years you get three weeks, and you have to take all at once, and that every five years became a really important part of my family's life, because we would always go on some big trip. I actually went on the first sabbatical ever. I know, because the sabbatical program didn't start until January 1, 1995; the acquisition was in '94 I think. And I left on my sabbatical before the holidays in '94, because I was taking six weeks, not three weeks, and my family and I went to Ecuador for those six weeks. Had a great time, and then in 2000 we went to Africa, and 2006 my wife and I went to New Zealand, and 2010 my wife and I went to Italy. So that was an important part of how we lived our life. It was not a very frequent thing every five years, but it was something that we all looked forward to. I didn't have that many outside interests. I have more now that I'm retired.

Male: Do you care to share any of that--

Malloy: Oh, sure. I have two main structured outside interests in my retirement. The first is that I wanted to spend more time outdoors, and so I'm a docent at Stanford's Jasper Ridge Biological Preserve, and I lead tours, and I help out on various projects over there. But actually, for the sake of this conversation, probably the more interesting one is that I volunteer at Woodside High helping to teach AP Computer Science. This is my fifth year doing it. It's really a big part of my retirement.

Brock: And so does that run throughout the school year every year?

Malloy: Mm-hmm, yep.

Brock: Okay. And I know in some states, it's harder than others for somebody without some sort of educational credentials and certifications to teach in a public high school. Was that something you faced?

Malloy: Well, I don't teach. I co-teach or assist.

Brock: Oh.

Malloy: So there is a teacher in the classroom.

Brock: That's a great workaround.

Malloy: But also there is an organization which deserves a plug here called TEALS. I don't know whether you've ever heard of it.

Brock: I haven't.

Malloy: It's a nonprofit founded by a guy out of Microsoft, and the whole vision behind TEALS is to take professionals in industry and match them up with schools and teachers to bridge the expertise gap that prevents schools from having a computer science curriculum. So it's not just AP Computer Science; there's Intro to Programming, and there's now AP Computer Science Principles, and at Woodside High there's even a fourth class that's kind of unique to Woodside. But TEALS is the organization that does that matchmaking, so the way I ended up at Woodside High is I found TEALS. I said "I'm interested in volunteering," and they said "Great. Here's the guy to talk to at Woodside," and it's a huge, huge lubricant in the process, because if you just call up a random high school and say you want to volunteer, I guarantee you'll get nowhere. You'll get a very polite "Thank you very much. Don't call us; we'll call you." So they know who the computer science teachers are. If there's a department head, they know who that is. If there's a instructional whatever they call them in high schools, vice principal or something like that, they know who that is. They know what classes they aspire to teach. They know what the skills of the accredited teachers are and what their skill gaps are. So they know how to put together a package of volunteers and curriculum and other things that really greases the skids to make this happen, and they operate in hundreds if not thousands of schools around the country. They started in the Bay Area and in the Northwest, but they've expanded beyond that for sure now.

Brock: And how would you characterize what led you to connect with that organization?

Malloy: Well, it wasn't a straight line, and it didn't happen overnight. So, as I said, one of my priorities when I retired was to spend more time outside, and that was easy, because I happen to live next door to Jasper Ridge Biological Preserve, and my wife has been a docent for many years. And so the obstacle to me becoming a docent was that you actually have to train to be a docent, and I didn't have time. I didn't feel like I had time to take the training when I was working, so I started that immediately upon retirement. That was easy. The other area that I told myself I was interested in, objective I set for myself is I wanted to do something in education, and my first idea was that I was going to be a math tutor or something like that, and in fact I did. I tutored at one of the local high schools, a high school that was a very spoiling experience, because that high school, unlike other high schools, was very well organized and set up to take advantage of volunteers. And they got a commitment out of the kids to show up when they said they were going to show up, and we met with an individual student over a period of months rather than-- there's also some models where you just sort of show up at a bullpen and wait for kids to show up. It doesn't work very well. So I did that for one semester, the spring semester, and then they actually were so well organized, they didn't need anybody the next year.

So the next year I tried doing something at another nonprofit that was not a school, but it was an after-school program, and that was more like the bullpen situation. I didn't do that for very long, because I would have to drive all the way over to East Palo Alto from Portola Valley, and I would sit there, and maybe somebody would show up and maybe somebody wouldn't, and then I'd have to drive back through traffic. So I didn't do that for very long, and then I was just trolling the web looking for volunteer opportunities when I came across TEALS. And this was also mid academic year, and about this time or maybe in January I filled out an online application. And the material on their website seemed very clear that you had to take some training and start at the beginning of the year, and they wanted to make sure you were really committed, blah, blah, blah, blah, blah. So I filled out this application. I figured I'd start the next September at some school. I hadn't even identified Woodside as the school. And I think maybe my résumé was a little different than the average volunteer, so I got a call or an email back within about 24 or 48 hours and--

Brock: I'm not surprised.

Malloy: <laughs> "We think we could fit you in right away." And so I started whenever that was, five years ago, so it must've been 2014; might've been even 2015. I don't know. And I've been doing it ever since, but I'm never in the classroom by myself, and that's why I don't have to be accredited.

Brock: Right, that makes a lot of sense.

Malloy: And it's good for the volunteer, too. It means that I don't have to deal with the immense trivia, and I don't have to deal with any awkwardness about interacting with young adults, that sort of stuff, because I'm never alone with them. <laughs>

Brock: Well, to have done it for five years, you must be finding it rewarding.

Malloy: I do, I do.

Brock: So I wonder if you could just say a few words about-- if you can, characterize why it is rewarding.

Malloy: Well, there's a couple reasons. I enjoy working with young people. I enjoy tutoring, and a big part of being a co-teacher or assistant teacher is tutoring, because it's a hands-on class. So most of most classes, the kids are at the computers working on programming projects, and so my job is just to circulate around and ask them how they're doing and, when they're stuck, to help get them unstuck. So I did that with my kids in math, which is why I thought a math tutor would be nice, but now I do it in an area that I know even better than math, and so hopefully I'm even better at it. So there's that. The other thing is that I probably took a slightly unusual approach to my mission in that I've actually been developing curriculum as well in the form of projects, and so I've developed three or four projects that the kids do, and so then I get to program, and I get to test these things out; I get to build libraries for the projects. And I find that even after, let's see, it's going to be 50 years next year since I learned how to program, that I still enjoy programming, and so that's a real draw for me. I don't know that I'll do it forever, but I'm still enjoying it now, and I've really found it rewarding over the last five years.

Brock: Well, maybe this is an appropriate time to break.

Malloy: Great.

Brock: Well, actually, before we do that, I'll ask kind of a catch-all, a safety, a cover myself question, which is: Is there anything--

Malloy: No.

Brock: That I haven't asked that I really should've asked you about? Do you feel that we've--

Malloy: Well, there's only one story I can think of--

Brock: Please.

Malloy: That I didn't get a chance to tell, and I think it's an educational story, and it's got to do with Steve Jobs and Larry Tesler and the Lisa and the Macintosh, so we were looking for things to say about that period of my career. So there was, as I said, lots of drama in the early days of Apple, lots of pivots and false starts, but one of the bigger dramas was around-- we had a mouse, but we had to decide how many buttons to put on the mouse. You're probably familiar with this decision. Now, for context, the Alto mouse had three buttons, and nobody wanted three buttons on the Lisa mouse or the Apple mouse, but there was a strong camp for one button, and there was a strong camp for two buttons. Larry and Steve wanted one button, so you can be pretty sure the way it turned out. I was actually an advocate of two buttons. I don't even remember why. I think because I just wanted that extra functionality, and I wasn't quite as convinced that the press-and-drag metaphor was going to be easy enough to use, which is basically the way you get around not having more buttons. But I think the interesting thing to take away from that drama is not that the Apple mouse ended up with one button and one button was right, but to use another quote from Bill Atkinson, when we were designing the Lisa, having these fevered conversations about things like this, he would raise his hand periodically, and he would say "We're arguing about diamonds and rubies here, and everybody else in the industry is selling coal."

And I think this is a perfect example of that in the sense that there's very few design decisions where, if you wait long enough, you can see both options explored at market scale, but this is one of them. So to this day, an Apple mouse has one button, but everybody else's mouse has two buttons. All the mice that run on Windows machines and other machines pretty much have at least two buttons. And you know what? It's all fine. People that have the one-button mouse, they love their one-button mouse. People that have the two-button mice, they love their two-button mice. So we were arguing over whether we were going to have a one-button diamond or a two-button ruby, and in the end, it didn't matter. The Lisa and the Macintosh would've been equally successful with either of those design choices, and so I think it's important to remember that from time to time, that sometimes in your passion for whatever perspective you're advocating, you may be passionate about something that's just not as important as you think it is.

Brock: Right, yeah, yeah, and I guess the trick is figuring out when you're in that sort of situation versus when you're in a situation where--

Malloy: Yeah, exactly.

Brock: It's diamond versus coal or something.

Malloy: Well, mouse versus soft keys I think is not a diamond-versus-ruby choice or bitmap display versus character generator display. That was not a diamond-and-ruby kind of trade-off. Anyway--

Brock: Well, thank you.

Malloy: I don't know whether you can slip that in somewhere, but--

Brock: We will put it in just where it landed, yeah, yeah. Well, thanks again.

Malloy: You bet. My pleasure.

Brock: Thank you so much.

END OF THE INTERVIEW