Oral History of J Strother Moore, part 1 of 2

Interviewed by:

David C. Brock

Recorded October 17, 2018

Georgetown, TX

By telephone

CHM Reference number: X8816.2019

Moore: Hello, J Moore.

Brock: Hi, this is David Brock calling from the Computer History Museum. How are you?

Moore: Right, hi, David. Yes, I'm fine. We're putting our house back together after a lot of repairs. Not repairs, we had it painted, and then we had the carpets and floors replaced. And everything has been-- we stayed in the house while this was being done. And everything was packed into bathrooms or places that weren't being touched. And so, today is the first day that we can start to unload all this stuff and put it back together. Anyway--

Brock: Wow, thanks for taking the time to talk to me given--

Moore: Well, it's actually nice to sit down. I've been carrying things and lifting things and squatting. And it's just-- it's nice to sit down.

Brock: Oh, good, good. Well, I'll try and ask a lot of questions, so you can rest.

Moore: Okay.

Brock: At least physically. I am using a call recorder. So, I just wanted to let you know that I think it's-- yeah, it is running automatically. So, I just wanted to let you know that.

Moore: Okay.

Brock: And I am going to send you-- I was-- I forgot to do it ahead of time, but we do have a release form that we ask people to sign that just clarifies that we can transcribe the interview and after you review it, make it available for other people to look at. So, I can just send that along to you by email after we talk.

Moore: Right, I'll-- I probably won't look at my email until tomorrow. But yes, I'll get it, and I'll sign it provided it's what I expect.

Brock: Yes, have a look at it first. It's pretty standard, but whenever you get a chance to. Again, sorry I forgot to send it beforehand.

Moore: No problem.

Brock: Well, I have-- I've created a whole set of questions for myself. And if you're comfortable and ready, I thought I might just begin.

Moore: Yes, before we start, just to clarify the context here, you have spoken with Charles Simonyi?

Brock: I did.

Moore: And he was the one who mentioned my name, is that correct?

Brock: Exactly right.

Moore: Okay, okay.

Brock: And I went to-- yeah, I went to talk to him to interview him and to try and elucidate a little bit more for me, and maybe for eventually the readers of something that I write, about exactly what this connection was that I had of course heard about between Bravo X and Microsoft Word.

Moore: Okay.

Brock: I've also done an oral history interview with Larry Tesler. So, I knew about the kind of connections between Gypsy and Bravo and wanted to-- so, then once I learned from-- once I learned more from Charles about the connection between this data structure that you brought to PARC that became known as the piece table, how he had used that in Bravo, and then how that had-- and then subsequently, how that had gone through Bravo into Gypsy, which I confirmed with Larry Tesler, and then how he and I guess Richard Brody had used the piece table to create Microsoft Word. And when I was talking to Charles about it, he was talking about how you had been instrumental with the concept at PARC. And that's how he had come to it.

Moore: Okay.

Brock: And-- yeah.

Moore: One thing that we should-- I should mention before we get started is the name Butler Lampson.

Brock: Yes.

Moore: Have you talked to Butler?

Brock: I have not talked to Butler Lampson yet about this. But I had a qu-- his name appears on my list. And I had also asked-- what did-- I also I think emailed Bob Sproull, who I've gotten to know a little bit and asked him just because I find him so knowledgeable. And he's actually so patient in explaining things to me. I emailed--

Moore: By the way, when did you contact Sproull?

Brock: He said he had just talked to you about this stuff. So, it was a weird--

Moore: Okay, good.

Brock: Really weird synchronicity.

Moore: Yeah, he-- I wondered whether your contact with him prompted him to contact me or whether it was synchronicity. But okay-

Brock: Yeah, totally random. And Bob Sproull suggested that I also talk to Butler Lampson.

Moore: Right.

Brock: Which I have not done.

Moore: Okay well, I've never quite understood Butler's role in the adoption of the piece table or even perhaps the invention of the piece table. I have understood from certain people, I don't

remember who, that Butler might have come up with or suggested the basic idea of representing the text by this sharing like data structure independently of what Bob Boyer and I did. But I am fairly convinced, but Charles is the expert, that it was my presentation at PARC to Charles that convinced-- well, to all of PARC, I think it was in a dealer.

Brock: Okay.

Moore: That convinced Charles to give it a try. And this is just speculation. But I would not be surprised if Butler independently came up with a similar idea.

Brock: Right.

Moore: But that the difference between Butler's idea and our idea, "our" being me and Bob Boyer--

Brock: Right.

Moore: Was that we had actually implemented it, and it was in use rather heavily back in Edinburgh.

Brock: Right.

Moore: And so, we had proved that it worked out. So, in any case, I just wanted to get Butler's name on the table because I truly don't know how much influence he had on Charles, although Butler had a lot of influence on a lot of people. But I know that I hadn't ever met Butler before I went to PARC. And we invented our data structure before that.

Brock: Right, right. Yeah, it will be fascinating to-- I can't wait to talk to Butler. Well, I hope he agrees to talk to me about it. But I'd love to talk with him about it more because the way that Charles Simonyi told me the story was that when he was at PARC, Butler Lampson had some notes about creating a new editor. And one-- and that it used, in Charles' recounting, it was really to incorporate two new algorithms or kind of approaches, two new developments. One was an algorithm for fast display on the screen, I guess.

Moore: Yes.

Brock: The details about which we did not go into, Charles Simonyi and I. The second was the piece table.

Moore: I see.

Brock: So, I don't know if it was-- so, I'll talk to Butler about that. And hopefully, we can both get some clarity about that.

Moore: By the way, it's too bad. Do you know Danny Bobrow? Did you know Danny Bobrow?

Brock: No, but I did hear of his recent passing.

Moore: Right. An interesting fact is that Danny was at Edinburgh on a sabbatical--

Brock: Oh.

Moore: While Bob and I were using-- while people were using our editor. And we would have naturally talked to Danny about it. And Danny was the one who convinced me to come to PARC. I was a PhD student and he said, "Well look, when you graduate--" and that would have been next year. So, this conversation would have been in '72. Danny said, "You know, you should definitely apply to PARC." And Danny was basically my advocate when PARC invited me to come and give an interview.

Brock: Oh, okay. Well, I do-- I had a question about that too, about how you got to PARC.

Moore: And so, I don't know what Danny-- Danny then-- Danny was at BB&N and then went to PARC prior to my joining. But he was already committed to going to PARC. And he was kind of recruiting for PARC while he was at Edinburgh. So, in any case, a good question to ask Butler was whether he had talked to Danny.

Brock: I will.

Moore: Okay, now having kind of gotten all the names that float around in my head, I'm now happy to hear your questions.

Brock: Well, great. I wanted to-- I really wanted to maybe just talk a little bit about your life before you got to Edinburgh, before digging into some questions about Edinburgh, and then into PARC. So, if you--

Moore: Okay.

Brock: I would be kind of interested in capturing the whole-- that whole kind of picture because I think it's-- well, it's absolutely to me so fascinating that this data structure, if you will, that becomes widely known as a piece table comes out of this context of research in artificial intelligence or people working on artificial intelligence. I'm just fascinated by that, so I wanted to ask you a little bit about getting to Edinburgh as part of kind of you getting into the AI scene, if that's okay.

Moore: Okay.

Brock: So, it was just-- I did see that you attended MIT as an undergraduate at the end of the 1960s. But I wanted to talk a little bit even how you got to MIT. And I was just curious about when and where you were born.

Moore: Okay, I was born on September the 11th 1947 in a little town in Oklahoma named Seminole, as in the Indian tribe.

Brock: Right.

Moore: But at the time I was born, my parents resided in Galveston, Texas. And it's just that my grandparents lived in Seminole. And for some reason, my mother went to Seminole. And that was actually my father's parents. My mother went to Seminole to give birth. I think the situation was my dad was on some kind of trip. And rather than be alone in Galveston, she went to-- she went up to her in-laws.

Moore: It makes sense.

Brock: In any case, I was born there. But as-- and my father, as soon as she went into labor, came back from his trip. And as soon as I was able to travel, they took me back to their home in Galveston. And so-- and my mother, by the way, was a registered nurse. So, I kind of think I probably got back to Texas at the age of one week or something like that.

Brock: What-- oh, please.

Moore: A nurse. And she already had one child. So, she knew the ropes.

Brock: Right. What did your-- what was your father doing for work at that time?

Moore: At that time, he was in the Boy Scouts. He was a-- you may not understand this, but he was a professional Boy Scout leader. And that's not like a person who has a troop with a bunch of boys. It's a person who organizes the troops in a region like Texas and New Mexico.

Brock: Got it.

Moore: And he-- but I don't remember exactly-- I don't know when exactly my father decided to quit the Boy Scouts, and he went to work in an oil refinery because he wasn't making enough money in the Boy Scouts to support his growing family. So, for all I know, he was quitting the Boy Scouts by the time I was born. The only person who would know that now is my older sister.

Brock: Right.

Moore: But in any case, I think of him as a professional Boy Scout shortly to become a management person in an oil refinery in Texas City.

Brock: And how would you describe kind of like the major themes of your household. You know for some people, religion is a very prominent theme, or for some people, it's reading, or politics, or music. Was there a-- business. Was there a theme?

Moore: Yeah, of course every-- you ask my sister, she would have a different theme. But my father was-- just loved the outdoors. So, my theme growing up was outdoor adventures. We--

shortly after Galveston, we moved to a town called Dickinson, which is-- well, it was in the news last year because it was heavily flooded by Harvey. But in any case, Dickinson has a lot of bayous. You know what a bayou is?

Brock: No.

Moore: Some people call it a bayou. It's a--

Brock: Oh, yeah.

Moore: It looks like a river except it flows both ways because the tide drives the water in and out. In any case, Dickinson is just-- has bayous all over it. And we lived on one. And I had a canoe. And I would spend my free time on the water or building rafts or fishing or hunting or just doing the things that a boy would do in a small Texas town. And my father would take us on camping vacations. Even after he quit the Boy Scouts, he became a troop leader. So, he was the troop leader of my troop. And he had a rule that the entire troop had to go camping once a month. And so, once a month, all of us went out. But my brother and I would often just go out on our own in a canoe. One of my hobbies from probably age ten was learning how to survive on just living off the land. And it wasn't so hard there because there are lots of fish and snakes and rabbits and whatnot.

Brock: Right.

Moore: So, a not uncommon thing for me to do on a weekend was to put my pocketknife in my pocket and wear the belt that I had previously already pretty well cut up to get pieces of leather--

<laughter>

Moore: And go out and build a little lean-to and make a fire and find something to eat and come back Sunday. But in fact, when I was, I don't remember the year, but it was about two years before I went to MIT, I was on the staff of Philmont, which is a Boy Scout ranch in New Mexico, teaching survival. So, anyway-- so, definitely my theme was that combined with almost constant experimentation. I was always building things. And probably the most interesting things

that I built were bombs. And we graduated from building bombs out of gunpowder we got from firecrackers to making our own gunpowder with potassium nitrate and other easily obtained chemicals at that time and building pipe bombs and setting them off in the bayou to stun fish. And that went on for a while. I had some co-conspirators, my brother and a couple of friends. That went on for a while until one of our bombs blew up while we were assembling it. And one of the boys was put in the hospital. And we decided that was pretty much the end of our bomb building days. But in any case--

Brock: Did that translate into an interest in chemistry in school?

Moore: I took chemistry. But I took every math and science course I could get. I-- another curious fact is that because I was born on the 11th of September, I was eleven days too young to start 1st grade. And so, but my parents didn't realize that. And so, I went to Kindergarten. And when it was time for me to go to 1st grade, they took me to 1st grade. And the school wouldn't take me because I was eleven days too young. The cutoff date was September 1, and mine was September 11th. And so, rather than just sit around for a year, my parents took me back to Kindergarten. So, I spent two years in Kindergarten. Now, this is a small Texas town. And I graduated from high school with the same kids I went to that second year of Kindergarten.

Brock: Okay.

Moore: And boy, in Kindergarten, I became the leader of everybody because I knew everything about Kindergarten.

<laughter>

Moore: And so, by the time I was in high school, I was president of the science club and valedictorian and quarterback of the football team. And it's all because I spent two years in Kindergarten.[1]

<laughter>

[1] Response corrected for accuracy by J Strother Moore.

Brock: Maybe it's a good technique.

Moore: Yeah, I really recommend holding people back and letting your child be older than everybody else. In any case, so I always had an interest in science. And I was-- but I did play football. And I had a lot of outdoor adventures. But I think of myself primarily as just an explorer. I mean explorer in the broad sense. I love looking-- trying to understand things. And so, that was--

Brock: Was science fiction in the picture at all? Were you reading?

Moore: Yeah, I read some science fiction at that time. I don't remember any particular author besides Asimov.

Brock: When did kind of computing emerge in your consciousness?

Moore: Well, that's also-- in, let's see, between my sophomore and junior years in high school, I went to a National Science Foundation summer program to teach-- to learn computer programming. So, this would have been-- let's see. I graduated in '66. So, this might have been in '63. I can't quite do the math. But anyway, it was the year before I started my junior year in high school. And I graduated from high school in 1966. So, one of us could do that math. And so, I was at the University of Oklahoma where Daddy had-- Daddy had gone to school there. And so, he kind of knew what was happening on OU campus. And when he learned about this, he said, "Hey, do you want learn this," because it was about computers. And none of us knew anything about computers. So, I said yes. And I went to that summer program. And that was my first exposure to programming. And it was-- I mean it just fascinated me. And programming still does. One of the-- a famous quote <laughs> of mine that-- I was once in my office, and a colleague called me up and said, "Hey, J. I'm writing a book about someone. And I would like to use a quote from you." And I said, "Well, if I said it in public, you're welcome to use it. But I'm curious. What was it that I said that you found so useful?" And so, the quote is, "Programming is like sex. Why would you pay anybody to do it for you?"

Brock: <laughs>

Moore: That's pretty much how I still feel about programming. I just love it. And unlike building with blocks or tinker toys or erector sets, you basically just never run out of parts.

Brock: Yeah.

Moore: And that's what's so fun about it. But in any case, so that summer, let's call it '64. That summer, I learned how to program in Fortran.

Brock: Okay.

Moore: And when I got back to school, I actually convinced the math teacher to let me give a one-month course on programming to the advanced math-- which was probably an algebra or trig class. It was a trig class, yeah. And so, I did that. We didn't have access to a computer, but I still taught them the basics. And then by the time I went to MIT, I was a competent enough programmer that I could-- well, I-- it's a long story. My mother got ill a couple of years before I graduated from high school. And she was in the hospital for a full year. And it broke the family in the sense of financially.

Brock: Right.

Moore: And so, I managed to go to MIT on a full scholarship, but with some loans. But in any case, the family didn't have any money. But Daddy wanted me to go to Rice, which was free to Texas residents at that time. But I got into MIT with enough financial aid to go, and that's what I wanted to do. I had wanted to go to MIT all my life. I can remember in the 7th grade looking at Time magazine and just spotting the name MIT over and over again whenever I would read an article I was interested in. So, MIT had always been the place I wanted to go. And when I got to MIT, I didn't have any extra money. And I thought it would be nice to like go home at Christmas and stuff like that. So, after I kind of got myself-- once I understood I could handle it-- you know, the first semester was, as usual, kind of scary and anxiety provoking because you don't know what it's going to be like. But after I established to myself that I could handle it, I looked for a job. And I found a job in a physics lab, the laser research lab. And my job was to be a gopher, you know, a person who goes to the physics store on campus and brings back glass tubing or electrodes or whatever the grad students who are assembling things need. So, I'd just

show up at the lab and hang around. And they'd say, "Hey, would you go get me a meter of ten-centimeter glass tubing." And I'd go to the store and bring something and then hang around. And while I was hanging around, I was looking over their shoulders. And they were writing Fortran programs to solve differential equations. And I knew more Fortran than they did. And often, I would bring back the wrong thing because I didn't know anything about electrodes. And after a while, the grad students started saying, "Look, can you debug this program while I go to the store and get some glass?"

<laughter>

Moore: And by my first semester of my sophomore year, I was put in charge of the computing budget for the lab. So, everybody brought their computing problems to me. And I would write Fortran programs to solve them and try and get the money to stretch out for the year.

Brock: And were these jobs-- these were-- were these scientific calculations, or were they like engineering calculations for building--?

Moore: It was scientific calculations. Yeah. I don't really know what the physics was. They would just say, "I need to solve this differential equation. I need a numerical solution to this differential equation." And why they needed it, I never asked.

Brock: Okay.

Moore: But by the way, that experience-- that first semester that I was in charge of the computing budget, the very first job I submitted ran all night because of an infinite loop.

<laughter>

Moore: And I-- when I picked up my output the next morning, I had not only had useless output, but I had incurred a bill of something like fifteen thousand dollars. And the total budget for the lab's computing for a year was ten thousand.

Brock: Oh gosh.

Moore: And so, I had to go to my boss and fall on my sword. And he taught me a very, very useful thing, which he said, "Well, everybody makes mistakes. I'll see what I can do." And he got the computing lab to-- he got the comp. center to cancel the bill on the grounds that the operator shouldn't have let this stupid job run.

Brock: Yeah.

Moore: And in any case, that was a good lesson in management. I kept my job. And anyway, but another fact I should mention is that so back in high school, by the time I got to my senior year, the advanced math class had dwindled down to one person. And that was me. And so, they canceled it. And that was the year I was supposed to take calculus. And so, rather than not take a math course in my senior year, I enrolled in a junior college nearby, Alvin Junior College. And I took calculus at night. And interestingly, the professor of that calculus course was a professional mathematician who worked for the Manned Spacecraft Center, which is the town next door to Dickinson where I was growing up.

Brock: Okay.

Moore: And so, that man's name was Nagle, Norman Nagle, N-A-G-L-E. And Dr. Nagle, as I always called him, was skeptical of having a high school kid in his class, but he soon realized that I understood it. And he-- and I just fell in love with calculus. I mean it was just the most beautiful mathematics I'd ever seen. It was the first mathematics that felt to me like real mathematics and not just rote memorization.

Brock: Yeah.

Moore: Instead of adding up numbers, you were sweeping over curves and things like that. It was just full of motion and forces and all kinds-- it was great. And so, I had this-- I developed this long association with Dr. Nagle. I mean every time I'd come home in the summer I'd visit him. And my-- the second, I guess-- in 1968-- I went to MIT in '66. So, the summer of '68 would have been the summer after my sophomore year. So, I'd been programming for the laser lab for basically a year and a half. And that summer, I went home to Dickinson. And I had a job.

Thanks to Dr. Nagle, I had a job working for TRW Systems Group, which was a subcontractor to the Manned Spacecraft Center. And my job was to help debug the Apollo onboard computer.

Brock: Oh, wow.

Moore: And again, that was because Dr. Nagle had written a letter of recommendation for me along with my boss at the laser lab. But that was an interesting summer because I walked into the TRW group on the day that the group was formed. And our boss said, "Look, there's software in the onboard for doing many different phases of the flight like lunar orbit insertion and trans-- navigation between the Earth and the moon and all that." Each phase of the mission had some associated software. And so, they wrote down each mission phase and put it in a hat. And we each drew a piece of paper. The piece of paper I drew was lunar orbit insertion, LOI.

Brock: Oh my gosh.

Moore: And when I drew it, the boss said, "Now, that's a really critical piece of the flight because it's the only phase of the flight that happens when the spacecraft can't communicate to Houston. It has to happen on the backside of the moon. They-- the spacecraft had to fire its main engine to slow down so that it would go into orbit around the moon. And you have to do that on the backside of the moon. And I said, "Well, you really want an intern to have that?" And he said, "Well, you know as much about this as the rest of us because none of us have worked on the onboard before." And so, I kept that task. And it was advantageous because that was the highest priority thing to debug, every time I would put in a run on the computer - I'll tell you about that in a second - my job would get the highest priority. So, I could do as many runs in a week as I could program up, whereas the others had to kind of wait for me to be not on the computer. But the-- it's a long story. But we-- what we were really doing, the Draper Labs at MIT had written-- had not only written the onboard computer software, but they had a written a simulator for it. And so, it was possible to simulate-- you could submit a deck of cards that said the spacecraft has this mass and is at this position with this velocity, and we're going to simulate thirty minutes of flight. And the moon is here. And Saturn is here. And Jupiter's there. And Earth is there. And then you could program up events like the astronaut pushes this button at time fifteen, and then the astronaut pushes this button when such a signal happens, and so on. And our

job was to fly the mission according to the flight plan, but to deviate from it in minor ways allowed by the flight plan. So, for example, if it was assumed that we knew the mass of the vehicle within five percent-- and I'm making that number up because I don't remember the numbers, then we could pick any mass we wanted above or-- within that window. And so, we were just simulating lots of Apollo missions. And I was simulating a lot of orbit insertions. And we would get a second-- we'd get a line printer listing second by second of all the parameters at the end the-- at the end of the run, I'd get pages and pages of line printer stuff that I would then give to another guy. And he would go through it looking for anomalies. And I luckily found a bug that would have caused the spacecraft to crash on the backside of the moon.

Brock: Oh my god.

Moore: And the way that that bug was fixed was to put a-- this was in the summer of 1968. And there was no way they were going to change the software.

Brock: Right because it was already wound into those rope core things.

Moore: Yeah.

Brock: Yeah.

Moore: Yeah. So, what they did was they put a what you would today call a post-it note on the console saying don't do this. And actually, I was told by people there at NASA that the-- when Apollo actually flew, the console was covered with post-it notes like that, just little reminders. Don't do this. Be sure to do that. And so, when you go to the Smithsonian, and you see the Apollo capsule, the console is all clean and nice looking, if somewhat outdated. But at the time it flew, it was just covered with sticky notes. Anyway, so that was my summer of '68.

Brock: My gosh.

Moore: And that was an instrumental summer because-- oh by the way, given the computing-- given that you are talking about the Computer Museum, it may be sometime interesting to you to interview me about that bug.

<laughter>

Brock: Yeah, I would. I--

Moore: I don't know if you have an Apollo flight control computer there.

Brock: We do.

Moore: But yeah, anyway, I have the programming manual for it. Anyway, I could show you some stuff.

Brock: That would be great.

Moore: But anyway, meanwhile, that was an interesting summer because prior to that, my idea of what a computer was for was to crunch numbers to solve equations. It was just purely a mathematical engine. But that summer, I flew Apollo over and over and over again. And if I didn't put Jupiter in the simulation, that rocket wouldn't get to the moon. You actually had to go out there far to get the gravitational model accurate enough. It was so real to me that when they actually did go to the moon-- I think in December of '68, they flew a mission that didn't land on the moon but just went into an orbit and came back. When they flew that mission, everything they were saying I knew. You could hear the astronauts talking to Houston. And they were reading out numbers. And all of those numbers were just utterly familiar to me and everybody else who had been part of that program because we had seen those numbers all summer. And anyway, so when I went back to MIT for my junior and senior years, I changed-- I was a math major, Course 18. But when I went back, I started focusing on AI and computing because that summer had shown me that, inside a computer, I could make a world, any world I wanted. And I could have it obey whatever laws I wanted. And it could be as real as real, you know. I mean it was really eye opening that one summer. And I'll tell you another thing that was interesting about that. A few years after that, the Apollo program was abandoned. I mean budget cuts and all kinds of stuff. And the guys I worked with, including Dr. Nagle, who I didn't work with, but he was there at the Manned Spacecraft Center, those people all lost their jobs. I mean by 1975 or so, those people who had gotten us to the moon didn't have jobs anymore. And it wasn't because we hadn't done our job. In fact, we-- and here, I include myself because I spent three months at it,

and so I get to do that, we had gone to the moon. And they fired us all. And why did they do that? It was just politics. And it didn't have anything to do with the value of what we had done or the quality of what we had done. And that taught me a lesson that I have held closely ever since, which is you can't evaluate what you're doing based on what other people think. You have to feel that you're doing the right thing. You have to feel that you're doing something worthwhile for you. And if you're doing it worthwhile for you, then it doesn't matter what they think. And yes, they might fire you. Yes, you might not get a job. But stick it out and do what you think needs to be done. And that's been my credo all along. I mean from graduate school onward I pursued my dream. And I pursued it whether I had funding for it or not. And yeah, sometimes, it meant I had to do other things in order to pay the bills. But at night, Boyer and I would sit down and do what we need to do. And in any case, that was because of what happened to the space program. But anyway, so yeah, and then after that, I started programming-- I quit working for the laser lab and started as a contract programmer. And my last year or so at MIT, I was a full-time employee of State Street Bank and Trust programming up the software that prints out the statements every night on all the transactions. So, if you bought stock-- State Street managed thirty-three mutual funds at that time. And if you bought some of those mutual funds, then a program I wrote would print up the certificate that you'd get in the mail. And I had done that as the summer job for State Street. And when the summer ended, they asked me to just stay on as a State Street employee. They gave me a terminal at home so that I could work. And so, my last year at MIT, I was a full-time employee of State Street--

Brock: Wow.

Moore: And an MIT student. And so, when I look back on my MIT career, it was really programming.

Brock: Yeah.

Moore: The other stuff I had was programming a page fault simulator for IBM for the System/360.

Brock: And that was a contract job for IBM?

Moore: That was a contract job, yeah. Once I got back to MIT and started associating more with the computing side of things than the math side-- I graduated with a math degree. But that last two years, I was at Project MAC a lot.

Brock: Okay.

Moore: And I met people. And they said, "Instead of working for the physics lab, why don't you work for this little software house that contracts out programmers." So, I did. That's how I got to IBM one summer and how I got to State Street Bank the next summer. And anyway-- and also, by the way, my senior year at MIT, I did an undergraduate research project under a grad student named Gene [Eugene] Charniak. And he was doing a PhD on understanding children's stories. And I did a little project on kind of a rule-based AI system. So, you'd put in a hundred rules that say if this happens, do this, if that happens, do this, and so on and kind of simulate it and see how it attacks a problem. And it wasn't a very deep thing. But it did-- it kept me busy. And that's when I started thinking about logic and AI. And in fact, when I went to Edinburgh, I intended to major in understanding natural language. But I didn't. In any case--

Brock: Well, I--

Moore: So, that's my MIT story.

Brock: Was that-- were you getting into the use of Lisp for that sort of work?

Moore: Yeah.

Brock: Yeah.

Moore: Yes, I learned Lisp from Marvin Minsky. And he taught the Lisp course. And yeah, I was-- so, I was programming in Lisp there. And unbeknownst to me, my last year at MIT, also on campus was Bob Boyer, who--

Brock: Okay.

Moore: Bob's name will come up when we start talking about the piece table.

Brock: Right.

Moore: But Bob was a graduate student at the University of Texas. And his advisor was Woody Bledsoe. And Woody took a sabbatical and spent a year at MIT. And Bob went with him.

Brock: I see.

Moore: And so, we happened to be together at MIT, Bob as a graduate student and me as a senior undergraduate. And-- but we didn't meet.

Brock: Okay. Well, before we get to your decision to go to the University of Edinburgh, could you talk-- in the essay that you put online about the piece table recently, you talk about-- or revised recently, which I-- was a great help for me to read. So, thank you. Could you talk a little bit about your experience with this TECO editor? I don't know if I'm saying that right, the T-E-C-O editor.

Moore: TECO.

Brock: TECO.

Moore: It's pronounced TECO.

Brock: Thank you.

Moore: TECO was the standard editor at MIT, at least at Project MAC, as far as I know. And it was-- all editors of the time were command line editors. We didn't have-- to me, the big advance of Bravo was that it was what we called a WYSIWYG editor.

Brock: Yeah.

Moore: What you see is what you get. Then displayed on a bitmap display are the actual characters including even fonts. But back in the '60s, the basic way you interacted with a computer was either via punch cards and a line printer, or a Teletype machine. And at MIT, it was Teletype machines. And so, if you wanted to edit a document you would say edit document

or whatever, and then it would basically give you a prompt. And then you could say some single letter command like P for-- I'm making this up. I don't remember the particulars. But if you typed a P, it would type the current line, print the current line. And if you typed M 5, it would move forward five characters. And then if you typed fast, it would insert f-a-s-t right there where you were standing. And you wouldn't-- and so, that's what a command line editor is.

Brock: Yeah.

Moore: Is you-- and that's what TECO was. And it-- to me, the thing that made TECO most interesting was that it was programmable. That is, you would actually write programs in the command language. So, you had an IF, and you had loops. And so, you could do something like write a program that would find the word "the" and replace it by "that," and do it globally through the whole document. And that would be just an incomprehensible sequence of characters in this weird programming language. But the point is it was universally powerful. And it was a programming language that presented itself as a text editor. So, that was what TECO was. And as far as I know, the way TECO managed the document was to read the document from the file into memory. Now, I don't know because I never looked at the insides of TECO. But that would have been a standard way back then to manage editing. So, you just read the entire file into memory. And then you address bytes in memory. And if you have to insert something, you actually have to shift all the bytes in the document down to make a gap so that you can insert the word that you're inserting. Do you understand?

Brock: I do. I do.

Moore: Now soon-- it could be-- because sometime around that time, editors started using-- implementing what was called the gap. And the gap would be a big blank spot, or maybe several gaps, in a document, in the internal representation. So, maybe you'd have the first thousand bytes and then a gap of a hundred bytes and then another thousand bytes of the document and then another gap. And that way you could shift less. And because-- and then when the gaps were filled up, you'd have to shift the whole thing. But this is basically just a way to handle inserting into a large document without paying a big price.

Brock: Right.

Moore: Of shifting megabytes of text. You also have to remember there weren't megabytes of text <laughs>.

Brock: Right, <laughs> yeah.

Moore: MIT had way more memory and way more disk storage than we had at Edinburgh, but nevertheless, my laptop has got just gobs more memory than anything computers of that day had. I mean gosh, our mobile phones are more powerful than the thing that went to the moon.

Brock: Yeah. Well, could I-- now, is it true that for TECO and these other editors, they were originally developed and used in the context of editing programs. These were to edit the text, if you will, that comprised computer programs. Is that correct?

Moore: Yes, that's-- that's right. They're basically-- to create a program prior to that, you had to type a punch card.

Brock: Right.

Moore: Well, or paper tape.

Brock: Right.

Moore: And then they started-- there was enough software and enough memory and whatnot so that you could eliminate that physical medium and have the-- create the program in a file and use that text editor to create and correct that file. And so, basically-- so, for example, I wrote the software for my dissertation. I wrote the-- my dissertation described some programs.

Brock: Yeah.

Moore: And I wrote those programs with a text editor. But I wrote the-- I wrote my dissertation, which is an English document, I wrote that on a typewriter, okay?

Brock: Yeah.

Moore: Now, I believe that Bob Boyer at MIT wrote his dissertation with a text processing system, TECO. I mean with maybe some kind of formatting. But in any case, it was-- even in the early ’70s, it was possible to create documents with text editors. But virtually, I suspect that most applications of text editors were to create programs and manage programs.

Brock: Okay. Okay well, could you talk about your thinking then of deciding to continue in artificial intelligence and deciding to do that at Edinburgh?

Moore: Yeah, okay well, so it came time to apply to graduate school. And I talked with both Marvin Minsky and Seymour Papert. Papert was my undergraduate advisor.

Brock: Okay.

Moore: That sounds fancy, but it’s not.

Brock: <laughs>

Moore: Papert was a busy man and probably didn’t know me or any of the other people he was advising. But MIT assigns a faculty member to every student. And that was mine.

Brock: Okay.

Moore: And in any case, they recommended-- I had decided I wanted to get a PhD simply for the reasons I said earlier, namely, there was still so much to learn. And I wasn’t anywhere near satisfied with what I knew. So, I had to just keep going. It wasn’t like I had a job in mind or even an area. I just needed to keep learning.

Brock: Got it.

Moore: So, I had to get a PhD. And there was never any question. Just like there was never any question would I go to college. I mean it’s just what I had to do. So-- and I told them I wanted to learn more AI. And they both recommended three places: MIT, Stanford, and Edinburgh. And I applied to all three places. And I didn’t get into Stanford, which is really interesting because on my application, I wanted to work with John McCarthy. As it turns out, the work that my entire

life thereafter has been devoted to is one that was very close to John's heart, which was Lisp and Lisp used as a mathematical logic to reason about computer programs. And that's what I've spent my life doing. And John was impressed by the work that Boyer and I did.[2] So, it's really ironic that I could have done that work under John, but I didn't.

Brock: Right.

Moore: And didn't get into Stanford. But I did get into MIT, and I got into Edinburgh. And I loved MIT. But I had spent four years there. And I had picked up on the fact that MIT was very focused on itself. And it basically-- it sort of hired its own, and it explored ideas invented there. And if I wanted to get a different perspective on AI, I would have to go somewhere else.

Brock: Right.

Moore: And that left Edinburgh. And the other interesting fact is that this-- so, this was 1970. I graduated from MIT. The United States was in turmoil because of the Vietnam War. And it was just a chaotic and almost hopeless kind of feeling. But in any case, the appeal of Edinburgh, aside from being not MIT, was that it would let me look at America from afar.[3]

Brock: Interesting.

Moore: Instead of being in the middle of it. And Boston was in the middle of it. I mean every weekend there were marches and whatnot.

Brock: Yeah.

Moore: You know the story. Martin Luther King and--

Brock: Oop.

<phone ringing>

[2] Reply corrected for accuracy by J Strother Moore.
[3] Reply edited by J Strother Moore.

Moore: Hello.

Brock: Sorry, my-- I lost the connection there. I'm sorry about that.

Moore: Yeah, anyway, so I was opposed to the Vietnam War, but that wasn't the reason I left. I left because the country was just so chaotic. And it was hard to understand what was going on. And I wanted a broader perspective. So, I went to Edinburgh sight unseen. Yeah, but when I got to Edinburgh, it was the Machine Intelligence Department. I was just blown away by-- it was like stepping off a boat and-- in the sense that life was so peaceful and different and quiet. And I had gone from MIT to this 16th century building the edge of a park. And anyway, it was-- when I met my advisor, Rod [Rodney] Burstall, he said-- he took me to a room, and he said, "Here's your desk. And we keep pencils and paper in that cupboard. And this little sheet tells you how to log on to the timesharing system. And we have tea every day at ten and three. And it's really nice to have you here." And he left. And there's no classes. It was just research from the moment you stepped in. And that was contrasted to MIT, where I was constantly going to class and doing problem sets and then doing my part-time job. It was just amazing. And three years later-- well actually, about two years later, Rod came and said, "Well, I think you need to write this up," meaning my dissertation. And I found myself finished before I was ready. But that's the way it is. It's just all research. And now meanwhile, because Edinburgh is such an interesting place, it had a constant stream of interesting visitors like Minsky, like McCarthy, like Dana Scott, like Danny Bobrow, Robin Milner. All kinds of people came through. And there were constant offerings of lectures and seminars and stuff by our colleagues plus these distinguished visitors, who might spend weeks or months there.

Brock: Right.

Moore: So, it was a very exciting place. But it was completely unstructured. And that really, really appealed to me. And so, I started working on understanding children's stories, and my dissertation supervisor was going to be Donald Michie. And Michie was the Minsky of Great Britain.

Brock: Right.

Moore: And Michie also had the interesting background of having worked at Bletchley Park with Turing as a graduate student. So, Michie could tell all kinds of stories about Bletchley Park. And I worked-- I spent a year working as-- working on this understanding natural language in the guise of understanding children's stories.

Brock: Yeah.

Moore: So, you know, the game was--

Brock: <coughs>

Moore: A computer program to read "Little Red Riding Hood" and then answer questions about it. But as happened at MIT, I needed money. I needed-- you know, we all need to eat.

Brock: Yeah.

Moore: And so, I got a job programming in the building next door, which was the Metamathematics Unit. And their interest was building mechanical theorem provers. And so, I was working on understanding children's stories and working as a programmer for Bob [Robert] Kowalski programming up theorem provers. And after that first year, I realized that I was a lot better at building theorem provers than I was at understanding children's stories. So, I switched advisors from Donald Michie to Rod Burstall. Rod was partly associated with that group and changed my dissertation topic from understanding natural language to automatic theorem proving.

Brock: Was that still part of the Machine Intelligence Department?

Moore: It would become so. Right at the moment-- at that moment, the Machine Intelligence Department and the Metamathematics Unit were two separate entities within the university. But before I graduated, they merged into the Department of AI.

Brock: I see.

Moore: So, actually, where they're-- what name I used-- I'm probably inaccurate about some of the names because they kept changing their names as they merged. I don't know when it became the AI department.

Brock: Yeah.

Moore: But the first year, it was the Metamathematics Unit and Machine Intelligence Unit. In any case, in that second year, the beginning of my second year, Boyer showed up. And because we were both Americans, and Boyer showed up in the Metamathematics Unit because his interest was in-- his dissertation had been on mechanical theorem proving. And I was writing mechanical theorem provers. And we were both Americans. And we were both in the Metamathematics Unit. And so, they put us in the same office. And so, I kind of had the natural job of teaching Bob how things worked in Edinburgh. And one of the first things I taught him was how we edited-- how we interacted with the system in order to create programs because that's what we did. And now, I have to digress and tell you that. How did you edit documents back then in 1971 in Edinburgh?

Brock: Before you do that, could I ask you one quick question before you do that, which I would love to hear, but I just-- it's been kind of nagging at me that-- to understand this. It sounds like to me that this topic of automatic theorem proving in this Metamathematics Unit, that that stands-- it sounds, on the one hand, kind of like theoretical computer science. And on the other hand, it sounds like artificial intelligence research.

Moore: Yeah, at that time, it was mainstream AI.

Brock: Okay.

Moore: Just like the chess problem was mainstream AI. Back in the early '70s, tackling the question of "How do you prove a theorem?" was considered a real challenge for an artificial intelligence. And as time has gone by, automatic theorem proving has become more a field unto itself. This is a common fact about AI-- is basically every time a problem has gotten understood enough so that you're actually making progress at solving it, it's not AI anymore.

Brock: <laughs> Right.

Moore: AI, by definition, is always working on poorly understood problems. And then it becomes vision, or machine learning, or other things that aren't exactly AI anymore. But back then, it was AI.

Brock: Got it. Yeah, so, I'm sorry. Yeah, thank you.

Moore: How do you edit programs back then? And the answer is they had, as I wrote in that little page I sent you[4], they had a simulated paper tape editor. Now, so think about paper tape with holes punched in it that denote letters of the alphabet. And suppose you wanted to create-- to edit a paper tape. Well, of course, you can't because the holes are punched in the tape <laughs>.

Brock: Yeah <laughs>.

Moore: However, there are paper tape editors. And those were machines. And you would put a paper tape that was punched and that you wanted to edit into one reel. And you would put a blank paper tape into another reel. And then you would copy the tape punching holes into the blank tape until you got to the point where you wanted to change something. And then you would manually punch the holes you wanted at that spot. And then you could stream the rest of the thing through copying the rest of the holes in the original tape onto the blank tape. And that's--

Brock: Okay.

Moore: Okay? And so, you would pass through the tape linearly. And you could make edits, but you had to think really far ahead because you'd scan the tape until you got to the first place you wanted to change. And then you'd type in the stuff you wanted. And then you would scan more until the next possible change, and so on.

[4] http://www.cs.utexas.edu/users/moore/best-ideas/structure-sharing/text-editing.html

And so, now you can simulate that with a program so that it just reads from a file and copies to an output file. And it stores that character or byte that you're on in a buffer. And you can either send that buffer-- that character to the output file, or you can type a different character. And so, that was the editor. And compared to TECO, it was just amazingly frustrating because, for example, if all you can store is one byte, how can you find the word "the." Say you wanted-- you had a document, and it had the word "the" in it, and you wanted to find that word "the" and put an n after it, turning that word into "then". Well, you see a T, and either you write that T to the output file, or you don't. But as soon as you let go of that T, you get the next character. And it may be an H, and it may not be. But the point is you couldn't do contextual searches because it'd only look at one character at a time. And you couldn't remember what you had already seen. I mean the machine didn't. It just had that one-character buffer.

And so, it was a very frustrating editor to use to those of us who were used to TECO where you could think of yourself as seeing a whole line at a time and move backwards and forwards in that line before you'd go to another line. In any case, so meanwhile, Boyer and I were writing programs to do theorem proving. And we had come up with a way to share the structure of one formula as you derived another formula from it so that you wouldn't have to use so much memory to represent the two formulas. So, in any case, the-- we decided that after a while using this text editor to write some pretty nice theorem proving programs, that we just couldn't stand it anymore. And we'd write our own editor. And then we basically borrowed the spirit of our theorem proving research.

And the spirit of that was don't create new structure when you can get what you want by sharing structure from something you already have. And so, that's-- and, as I wrote in that page I sent, the challenge of text editing at that time was memory. I mean memory was so limited that you just couldn't store a large program or a large document in memory. And so, we had to have a way to represent the text without using a lot of memory. And that's what-- we did what I showed in the page.

Brock: Yeah, and I was-- if I could ask you a question about that. It seemed-- you know, I tried reading through your explanation of structure sharing in automatic theorem proving. And I-- being uninitiated, it was a little difficult for me to follow it, not because of your writing, just

because of my background. And but what I was getting from it was just-- loosely, was that with structure sharing in a theorem prover, there was a sense in which, as kind of the process is going along, instead of doing a lot of copying or repeating of let's say things like earlier propositions or something like that, there's a lot of instead kind of pointing back to other propositions or theorems or statements that you're drawing from. Instead of kind of reproducing them, you would point to them. Is that true?

Moore: You would point to them and attach a note that sort of modified it. So, for example, you would point to a formula and say I am that formula over there, except his x is now three.

Brock: Okay.

Moore: That would be instantiating that formula. Or I'm pointing to-- I'm that formula over there, except the third hypothesis is now gone.

Brock: Got it.

Moore: So, that would be a deletion from that. Or I'm that formula over there, except at the end, tack on this formula over here. And that way, instead of copying and instantiating variables with new terms, you would just point to old things. And you would see everything through the sort of rose-colored glasses of these pointers.

Brock: Okay.

Moore:

And that, by the way, is how computer programs actually execute. I mean this may not make any sense to you if you're not a programmer. Suppose I have a program named "Foo"," and it has a variable x as its input, and it adds 5 to x to produce its output. Suppose I want to use Foo to add 5 to 3. So I call Foo with input 3. The computer doesn't actually take the text of Foo, namely "x+5", and replace x by 3, and execute "3+5". Instead, typically, it executes the pre-existing text of Foo but with a "binding" – a note – that says "When you see x use 3 instead." That is, every time the computer comes to a variable it looks up the value of the variable in a table or stack or

some data structure. You don't copy the program. You just execute the original text in different "environments". That's what we were doing with our theorem proving code: seeing pre-existing formulas in different environments.[5]

Brock: I get it.

Moore: And that way, you don't create much structure. You don't use much memory. That was a good way to do that kind of theorem proving and the text editing idea is spiritually similar.[6]

I mean, for example, if somebody sends you the page proofs of a document, you don't copy it all out and write corrections in the new copy. Instead, you take the existing page proof, and you write pencil marks on it that say here, insert this, or x out that and replace it by this. But the point is that's a perfectly sensible representation of the new document that you have mind. And that-- so, that's what our representation does is it takes the old document, and it annotates it with sort of editor's marks.

Brock: Oh, interesting. I hadn't thought of it that way. I mean I had gotten the idea that-- I mean just crudely in my mind, the way that I was thinking about it was that in the automatic theorem proving work, that had structure sharing, which was kind of "don't copy, just point" as an approach. And then when you got over to what became the 77-Editor, it kind of did the same thing. Don't copy again, don't make another copy of the original, just point at it.

Moore: Right.

Brock: And but I hadn't thought of it as just like-- as either the proofreader's marks or, as I have done many times working when you're doing copy edits and manuscript revision, where you'll end up just sending a person a list of all the changes.

Moore: Right. That's right.

[5] Reply edited for clarity and accuracy by J Strother Moore.
[6] Reply edited for clarity by J Strother Moore.

Brock: Paragraph four page six, strike x.

Moore: Yes, exactly. And that's basically the representation that we used. And one beautiful fact about it is if you remember the order in which those changes were made, then you can undo a change by just erasing it from that little list.

Brock: Yeah, right.

Moore: Okay, and so that gives you this undo facility, which is really hard to do if you have the text concretely represented in the computer because then you have to know where in the text to erase. And then you've got to slide all the text behind it back up to join them again. And but by kind of just tracking the changes, you can make that be the representation. Another neat fact is that the original version of the document is preserved on disk as you edit it. If your system crashed in the middle of your session you still have the original, not some copy that has been mangled by partially completed edits. When you've reached a good stopping place, e.g., the end of a section or chapter or whatever, you save your work. Save just makes one big sweep through the representation carrying out each change. That creates a new clean version.[7]

Brock: And the editors that you had used before at MIT, TECO, that had a different structure. That was kind of like-- did that also-- well, did that editor basically create a whole copy?

Moore: In a sense-- yes, you would-- to edit a document, you would read it all into memory. That's a copy.

Brock: Okay, yeah. Yeah.

Moore: Okay, so now you need twice as much. You need the bytes you have out on the disk. And you have the same bytes in memory. And then if you want to insert something, you have to shift part of the text down. That's more copying in a certain sense. You are taking a megabyte of text and moving it down five bytes so that you can insert a five-byte word in there. And as long as you have enough memory to do this, then nothing-- then you still have your good copy out on the disk. But some editors didn't have some-- some systems didn't have enough memory to even

[7] Reply edited for clarity by J Strother Moore.

suck in the whole file. And so, they would have to edit in place. So, that would actually write to the disk. It would-- in order to insert something, you would give I/O commands, input/output commands to copy the last megabyte of text down five bytes on the disk so that you could have room to put in that five-byte insert.

Brock: Oh, my gosh, which must have been very slow.

Moore: It's very slow because it's all disk based. And it also corrupts your original document so that if you-- something bad happens like you lose power, then you've just got garbage.

Brock: Yeah, okay.

Moore: So, this had a nice way of basically preserving your documents and, at the same time, giving you freedom to make many edits on it and then save the final version. And then the other thing that's neat about it, it's not so evident in this metaphor about the editor's marks, but you can-- because every change is associated with a piece, what became called a piece in the piece table, those pieces, you could associate-- you could add metadata to. You could say this piece was added by Charles. Or this piece was added on this date. Or all this piece does is change the font from normal to bold, and then this piece changes the font back to the other. And the idea then is that you can attach all this metadata to the edited document and not affect the content so that you still have all the same bytes in the document. But when it's displayed, there's this extra stuff that it either may or may not be displayed like the change tracking or the fonts. Now, in Edinburgh, we did not have the ability to display fonts. We didn't do change tracking. But-- so, the 77-Editor didn't have any of that, but the representation would support it. I mean that is to say when I got to PARC and started talking about the way we represented documents, it was just pointed out that each piece could give you an opportunity to store additional information. And of course, Word uses that. But anyway.

Brock: So, you could essentially-- with your 77-Editor, it was-- so, there's the original copy. And then there's initially one record that just points to that whole original. And then you can add additional records in a sequence, I guess. And those records basically contained kind of a pointer to a section of the original and then sort of like a command, right? Like insert this, or--

Moore: Well, what it actually did was-- if we called each record a piece, then the document is the concatenation of all the pieces.

Brock: I see.

Moore: And in memory, you just have a list of pieces. And originally, there's just one piece. And it's a pointer to the original document with delimiters that say from beginning to end. And then if you want to insert something in the middle, you first split that piece into two pieces so now you have two pieces that are the first half of the document and the second half of the document. And then you insert between those two pieces a new piece that has the new text. And so, now you have three pieces in memory. But all of the text is either out there on the disk from the original document, or it's stuff that was typed as part of this editing session.

Brock: Right, and it's-- okay.

Moore: Now, I think, given that Charles uses the name piece table for this, it suggests that Charles just arranges these separate pieces in a table, what would be called an array in programming language. Boyer and I didn't have it in a table. Boyer and I had it as a linked list so that each piece pointed to the next piece. Okay, so each piece had the-- some stuff talking about the text it's supposed to represent, characters one through seven on this file, and then a pointer to the next piece in the sequence. And that way, you could insert with no overhead. You just change the pointers between pieces to link in another piece.

Brock: I get it.

Moore: Now, if in fact, the pieces are stored in a table, then you have to, if you want to insert a new piece you have to shift down all the pieces below it. Now, that's not shifting down megabytes of text. It's just shifting down big chunks of it in this piece table, but you would have to ask Charles how he does it because I haven't looked.

Brock: I will.

Moore: And what we did was the pieces were linked. It was just a linked list.

Brock: And that-- and may I ask, a linked list is something that's very familiar from this kind of AI Lisp sort of world, right?

Moore: Yes. Oh, absolutely. Absolutely, yeah. It-- every record or-- every record has an address in the machine, and to link to it you just write down that address so if-- and yeah, and that makes-- that's a really neat representation because insertion is constant time. It doesn't-- you don't-- no matter where you insert it's the same amount of work because all you do is change a pointer (an address).[8]

Brock: Well, so once you-- so your frustration with the existing editor and trying to work on these automated theorem prover programs prompts you and Boyer to develop this new editor based on these kind of new ideas. How long did it take you, and were there unexpected challenges or changes that came during the implementation?

Moore: Well, it-- I don't recall how long it took, but Boyer and I had a motto back then saying that anything-- any program worth writing could be done by two people in a month. So probably it took us a month to get something that was usable and then, as usual, a lot of tuning and stuff but one of the problems is you've got text spread over these successive pieces and you want to be able to-- you want to be able to search it. You want to be able to find the word "elephant" in the text and the E-L-E might be on one piece and the P-H-A-N-T might be on the next piece and so you have to write a search program that understands that the document is in pieces. And anyway, so that was an issue and another thing that we did was we found it useful for-- you often want to display the document and as displays got better you'd display more and more of it, not just a single line and so what Boyer and I did was we-- I don't know how to describe this, but if the user was working at a particular place in the document that would be the piece he's focused on is literally some particular piece. Okay? Now, then we could-- when the user's focus would change from one place to another in the document, we would reconstruct the text that the-- that was currently represented by visiting 100 characters in-- before that point and 100 characters after that point. I don't remember if it was 100 or 10 but some number, and we would just make a string of that text, okay, so that we could rapidly answer questions about the stuff around you

[8] Reply edited for accuracy by J Strother Moore.

right now without having to look at each piece and jump over piece boundaries. And so this-- I don't know how to put it, but it's kind of like virtual memory in that you would-- you have a representation, one-- we'd actually end up with two representations of the part of the document. The entire document would be this linked list of pieces, and there would be a more conventional representation of the part that you're focused on right now, the page or two, and that turned out to be useful when I got to PARC where they wanted to display the whole page because I could just take that process of, we called it reification, to reify the document was to turn the-- I didn't want to use that word but it just came out naturally. But we would-- to reify the document around a point, you would just basically read every character from the beginning to the end of that window you were interested in and copy it into a string or a page somewhere and if that page happened to be the screen then you have the document printed out in front of you, that part of it, and so we would-- when I got the part and I implemented-- by the way, when I got to PARC one of the first things I did was I implemented these editing commands, not the editor itself but the representation. I implemented it in Lisp and it was a package of Lisp programs called TXDT, the text editor, and there's a memo about that package and that was a Lisp, a collection of Lisp functions for manipulating pieces and it would-- it included this so-called page mapping that would take part of the document and map it on to a page. And anyway, that package was what Charles actually experimented with. I mean I don't know what Bravo was written in, but I wrote that package in Lisp, Interlisp, and I think Charles probably played with it there before he reimplemented it knowing what he-- no, he didn't. Ultimately, he didn't use my particular programs. He just used the ideas, but he understood those ideas by playing with my program.

Brock: Wow. Okay. Well, let me just-- so I was-- so you wrote, you and Boyer wrote the 77-Editor in POP-2. Is that correct?

Moore: Yeah, yeah.

Brock: And is that a Lisp-like programming language?

Moore: It doesn't look like Lisp. It looks like ALGOL or, in fact, you can even think of it as kind of looking like C or Java. I mean it's a pretty conventional-looking programming language, whereas Lisp is full of parentheses and is-- and looks quite odd compared to other languages. But

sematically, POP-2 was like Lisp in that its main data structure was linked lists and it had automatic storage allocation and garbage collection.[9]

Brock: And-- oh, go ahead.

Moore: Yeah, we wrote the editor in POP-2 and POP-2 had a very-- it had a very succinct syntax so if you could, if you named a function F and another function C, then if you typed F-space-C, you'd execute F and then you'd execute C and so making an editing language out of it, like TECO had its own editing language, it was really easy. We just gave each of our functions one-letter names and then you could just type POP-2 but calling those functions to insert and delete and move around the text. So-- but it was fully programmable because it was just POP-2 so if you just wrote a program that called F and C, then it would do it under program control and so anyway, it was a pretty elegant command line editor, the 77-Editor, but nowadays people would look at it and think it was incredibly clunky because you couldn't see the document, that you had to-- if you wanted to see what was on the current line you had to print it and-- but I don't know. At the time, it seemed nice.

Brock: And was the-- were those Edinburgh computers on a network? Did the editor spread to other locations through network use or...?

Moore: Well, no. There was only one computer. Everybody used it and it was on Track 77 of the disk system there so if you ran the program on Track 77 you got the 77-Editor and it was just a big-- it was a time-sharing system and there were-- when I was there, 8 people at a time could use it but they were all sharing 64 kilobytes of memory. I mean that's tiny and the disk farm, as it were, had 3 disks and each disk held 4 megabytes so there was a total of 12 megabytes of disk and 64 kilobytes of memory and that's what you had to do AI in and edit and anything else you wanted.

Brock: Wow. Amazing.

[9] Reply edited for clarity and accuracy by J Strother Moore.

Moore: Yeah, and over time, of course, the system expanded. I understand-- I left Edinburgh in December of 1973 and joined PARC in January of 1974, but I understand that the 77-Editor continued to be used about-- for about 10 years until the early 1980s when that British machine, the ICL 4130, was replaced by a PDP-10 or something more conventional.

Brock: Right. Well, could you tell me again a little bit about your transition to PARC and your discussions with Bobrow and others about the shape of the AI activity there?

Moore: Danny Bobrow asked me to consider PARC. I didn't even know about PARC. It was only just being created when I was a student in Edinburgh and it sounded intriguing and so when I kind of unexpectedly got my Ph.D. because it was time to write it all up, I went on a-- I went, I applied to jobs, four jobs, and one of the jobs I applied at, one of the places was PARC and I went and gave an interview. And I met-- the person I remember most meeting, aside from Danny who was there and I already knew, was Bob Taylor. (Taylor and I became really good friends and in the years between 1974 and his death in 2017 – long after we'd both left PARC – I visited him and we stayed in touch by email and phone.)[10] But the-- yeah, so I got the job and it seemed interesting. Now, Boyer was also-- he was a postdoc at Edinburgh and he left the same time I did but he went to SRI in Menlo Park and I went to PARC in Palo Alto. So Boyer and I continued to do theorem proving at night together but in the daytime I worked for PARC on two things. One was Interlisp. I wrote-- Interlisp was a version of Lisp that was being developed at PARC. It stood for Interactive Lisp and I worked with Warren Teitelman and Peter Deutsch and my job was to specify Interlisp so that other people could implement it. So basically, I had to figure out what parts of Interlisp an implementer needed to code up in machine code in order to get the rest of Interlisp to work on a new platform. So I wrote the Interlisp virtual machine specification for one of my jobs and the rest of my time was working on text editing, not with-- by coding up my stuff in Interlisp. But PARC had a tradition of "Dealer," which was a meeting held once a week in which somebody would just give a presentation to everybody at PARC and it was a great tradition, "Dealer", and so I gave a "Dealer" once on this text editing and I actually don't remember much about Charles and his uptake of this idea. I mean I knew that he was working on another kind of editor that would display the text on the bitmap displays we had and that was all

[10] Reply edited for clarification by J Strother Moore.

very interesting, but I was really working on Lisp and so the fact that the piece table made it into Bravo, much less that it made it into Word, wasn't known to me for years. I mean I only learned about it-- I only learned about it being part of Word probably 20 years after I left PARC.

Brock: Hmm. In the early 2000s?

Moore: Yeah, probably. I got...

Brock: And how did you-- how did you find out that?

Moore: I got into the-- I got elected to the National Academy of Engineering in 2007 and my host at the induction was Charles, which is the first time I had seen Charles since then and I had a long chat with Charles then about that but-- and I had heard about it prior to that. In fact, it's mentioned in a book. I don't remember which book and all my books are piled in a bathtub right now but it's one of the books about PARC, "Dealers of Lightning" or "Dropping the-- well, I forget...

Brock: Oh, yeah. "Fumbling the Future," or whatever it is. Yeah.

Moore: Yeah, "Fumbling the Future." Yeah, one of those books about PARC, "The Dream Machine," maybe.[11] Anyway, one of those books just mentions that I passed on this idea to Charles, and it could've been that book where I learned that it was used. In any case, but I wasn't part of that. I just gave my talk, implemented the code so that people could understand truly how it worked, and then I would answer questions like, can you do this? Can you do that? And I would make it happen in my utility package. I mean for example, the standard way to-- here's a neat problem and it doesn't make much sense, but I'll explain.

So suppose you have a piece of text and you're pointing at a word in the text. Now, how do you remember that place, that point in the document? Now, the normal way to remember it is that it's

[11] Added by J Strother Moore: On page 199 of "Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age" by Michael Hiltzik, Harper Business publishing, 191, in a discussion attributing the piece table to Butler Lampson, it says "These [ideas] had been first developed by a programmer named Jay [sic] Moore." Technically my first name is "J" not "Jay," I was a Research Mathematician (but proud to be called a programmer!), and the ideas had been developed by Boyer and me.

so many bytes from the beginning, okay, but now if you go to an earlier place in the document and you insert something, then that number that represents that place doesn't represent the same place anymore. Okay, so the question is, can I get a pointer to a place in the document that stays pointing to that same place even as I edit the document around it? Okay, and the answer with a piece table is yes because you can just remember the piece containing it and the location within that piece.[12] I mean the-- even if you insert other pieces before it, that piece is still there and anyway, so there was a way to get a pointer to a place that was immutable even if you edited around it and that was a very cool idea because then an editor could mark something and then go edit around somewhere else and then come back to where he was and really be back where he was, not to a place offset by some arbitrary amount.

Anyway, so Charles or other people would come to me and say, "Can you do this with your representation? Can you attach metadata? Can you make-- can you give me a way to say this is in boldface?" and I would do that but all I would-- I wouldn't make it print in boldface. I would just add a field to my pieces that said this is a font change. Anyway, so but that was all almost-- I didn't think of myself as doing that. That was just doing favors for people in the lab. I was focused on other things, on Interlisp, and I should say that my relationship with Bob Taylor kind of was a similar one because Bob loved video games and he was constantly playing. In those days, he was constantly playing a game called "Star Trek," which was a line-based video game where you would say, "Raise the shields," and let's say it would say, "Okay," but then the Romulans shot at you. You were safe for a little while and then you could ask what the status of your shields was and it would say, "Fifty percent," and you could say, "Go to Klingon star" or whatever. Anyway, he would type these commands and interact with this thing and play and pretend you were Captain Kirk and fly around on the Enterprise and Taylor was playing it constantly and griping about it because he couldn't do things he wanted to do and I said, "Well, do you have the code for it?" and he said, "Well, here's the program I'm running," and it was written in Fortran. So I would modify it and I fixed-- I made a version of "Star Trek" for Taylor and every time he wanted-- he wanted to undo. He wanted to be able to go back and get into a state he had been in earlier and play it differently. Okay, and so I would do that for him and so, again, that was just one of these favors for somebody at PARC. It wasn't my job, just like the

[12] Reply edited for clarity by J Strother Moore.

way I thought about the work on the editor features. It wasn't-- it was just Charles wanted things. I'd add them. Bob Taylor wanted things. I'd add them, but...

Brock: Now, would you be adding like, for example, I think when I was talking to Charles Simonyi he recalled like for instance, the stuff around formatting information to be able to put that into a piece, he described that as like attaching a message to the piece.

Moore: Yeah, right...

Brock: And...

Moore: Put metadata in the piece. Yeah.

Brock: Okay, and now would that be something like that you would then also-- like if would you implementing that yourself in TXDT or...?

Moore: Yes, yes. Yeah, but it wouldn't have been my idea. It would just be Charles would ask me to do something like make a place for him to store some metadata and I'd make that and how he would interpret the metadata would be up to him. I mean he would take the pieces and print them out, and he would interpret his metadata however he meant to interpret it. So my job was just to store the data in this representation somewhere so he could get at it and read it and change it.

Brock: But you would be doing this to your-- you would be making these changes to your editor and utilities in Interlisp?

Moore: Yes, that's right. That's right, and it would just be-- again, I don't know. You'd have to talk to Charles, but I understood it as just he was just experimenting with my package to see if he could get everything he might want and at this point he stopped talking to me and just started doing it himself but...

Brock: Right. Then, they were doing their own kind of implementation of these ideas that they had been experimenting with or becoming familiar with through your TXDT and these other Interlisp utilities.

Moore: Yes, that's right. That's...

Brock: They used those to develop Bravo or Bravo X or whatever it was.

Moore: Right. That's what I understood was happening. It's referenced in that memo, in that, the PARC document. Right.

Brock: Right, from 1981.

Moore: Yeah, and that was-- it's so late because I had left PARC. I left PARC in '76 and went to SRI to work with Boyer more closely because I could work with Boyer all the time and-- but the-- eventually PARC asked me if I would document that package and probably it was associated with Charles leaving. I don't know when Charles left PARC. Do you?

Brock: He left in either 1980 or 1981.

Moore: Yeah, so I suspect what happened is Charles knew everything there was to know about that package and then left and so they wanted to write down what that package could do.

Brock: Oh, wow. Okay.

Moore: And that's my theory and so I was asked to write a tech report long after I was no longer associated with PARC. Most of TXDT was written between 1974 and 1975, but I continued to add features and fix bugs for Charles even after I left PARC in 1976.[13]

Brock: Oh, I hadn't realized you left in '76. Can I just ask you, so when...?

Moore: It might been '75. I am not good with dates, but it's somewhere-- it's not later than '76.[14]

Brock: Okay. Well, when somebody-- so let's say we're in 1976 and I'm using Interlisp and I was just reading something that was describing kind of the graphical environment that was

[13] Reply edited for clarity and accuracy by J Strother Moore.
[14] Added by J Strother Moore: I left PARC in 1976.

eventually developed for Interlisp. I would-- well, when people were using TXDT, you're at an Alto, or where you running it off MAXC?

Moore: I think I was running it off an Alto or a Dorado.

Brock: Okay, and was it-- when you were using TXDT, would it be-- would you see a whole page of text, or what would you...?

Moore: No, no.

Brock: It would just-- yeah.

Moore: It would just be typing the commands to insert into this piece or delete that piece and so on and behind the scenes changes would occur in the data structure and if you wanted, you could print out, you could reify the data structure and see a string of characters but the taking that and putting it on a display with fonts and all that is something that the package didn't support. I mean the package was just managing the representation of the document, not displaying it.

Brock: So when people were-- was this the main editor then that people were using when they were developing programs in Interlisp and developing the Interlisp system itself?

Moore: No, no. I don't know what-- I don't even remember what text editor we used but Interlisp provided its own editor so you could create Interlisp programs by interacting with Interlisp and it was more or less a command-line type editor even then. But there was also some kind of text editor because we could write documents and memos to each other and I don't remember what that editor was but of course it was eventually Bravo but I don't know what it was before that. It could well have been TECO. Lots of people at PARC had come from MIT or BB&N which was also MIT.

Brock: Right. I guess I'm just trying to understand how the TXDT package would have been used by people in Interlisp at this time.

Moore: Yeah. I don't think it was. I think that that package was-- the way I always thought of that package is it was-- it could be used by someone who wanted to create a text editor. There

were two people who wanted to do that. One was Warren Teitelman, who wanted to make an editor for Interlisp and he might have eventually used that. I don't know. And the other was Simonyi, who wanted to make a document system and all that-- all that package was for was to clearly communicate and experiment with that representation so that people could prototype editors and-- but users, just like users today, don't have a clue that there's a piece table behind Word. Users of all these experimental editors didn't know that TXDT was involved and I mean anyway, I thought of it-- I really thought of that package as just a service to Charles and to Teitelman.

Brock: I get it. I heard somewhere-- I think I read somewhere that Teitelman made an editor called Poet, but I haven't been able to find any info about it. I'm still looking around.

Moore: Yeah, I don't-- I do recall that name, but I don't remember anything about it. I'm peculiarly focused on my own stuff and this, it's funny to say because I recognize that the piece table idea is kind of, in a certain sense, fundamental to some of the capability of Word but I think of it almost in the same genre as fixing things for Taylor's "Star Trek." It was just I needed to communicate ideas to people or build a prototype and I just did that and what they did with it didn't concern me. I never watched-- I never played "Star Trek."

Brock: Did you have any interaction with some of the other people who were working on or interested in these kind of document programs...

Moore: Yeah, yeah.

Brock: ...like Tom Malloy or Larry Tesler or Tim Mott?

Moore: Well, the main person was Chuck Thacker actually, and not-- he wasn't working on documents but there's another part of my life that I think should be interesting to the Computer Museum and that's Boyer's and my invention of what's called the fast string-searching algorithm.

Brock: Yes, I was reading about that. I mean when Charles Simonyi was talking about you and the piece tables he also very much mentioned that fast string searching, and I went through the information about it on your website.

Moore: Right, so that's an algorithm that's widely used and I mean, for example, DNA searches often use that and the virus detection mechanisms on-- if you have a virus detector on your home machine it's probably using the fast string-searching algorithm because basically it takes a known virus and searches your gigabytes of memory for it. And so that algorithm is used a lot and I'd be a rich man if I got a penny every time that algorithm ran, but again, that was just something-- Boyer and I needed to do string searching and we thought about how to do it and that's what we came up with but when I first coded it in assembly language for MAXC the-- it wasn't as fast as TECO. TECO's string search was incredibly fast and I couldn't figure out why our super algorithm wasn't faster than TECO and so I went to Chuck Thacker and I asked him to look at my code and answer that question and he explained that on a machine like the PDP-10 or MAXC, the fastest-- memory, it takes time to pull a program-- to pull an instruction out of memory and that's a limiting factor in lots of instructions when you have to go to memory. And so to make a program really fast you would load the program itself into the registers and the registers of the machine you could get at more or less instantly and so if you had a tight loop, a small loop of eight instructions and you needed to execute them over and over and over again then you'd load them into the registers of the machine and execute them from there. And then he also showed me a way to increment a byte pointer. You have this address in the machine that points to a particular byte and a particular word and there was a very clever way to increment it and Thacker showed me that. So using Thacker's coding tips, Boyer and I recoded our algorithm and then we-- I gave a "Dealer" on the algorithm and I showed-- I actually, right there on the screen I showed TECO doing a search and then I showed us doing the search and we beat TECO and when TECO-- when we beat TECO the "Dealer" room erupted in applause and it was so cool because TECO was just-- it was the finest, best-written, best-tuned piece of software anybody knew. And we just beat it on its bread-and-butter issue which was searching text. Anyway, but yeah, but I don't remember working with Tesler or Mott or anybody else. Sorry.

Brock: Yeah, so let's see. I was just closing the door because my dog was barking. Sorry. Let's see. Well, it seems that-- well, maybe what we could do is there's some other questions I wanted

to ask you, but I've had you on the phone now for over two hours. But I wanted to-- maybe what we could do is I did want to learn more about that fast string-searching algorithm and maybe we could just follow this line of the piece table forward for a few more minutes and we could talk again sometime about some other aspects, about what you did after you left PARC and about-- more about the story of your involvement with artificial intelligence after that. Maybe that would-- does that sound okay?

Moore: Yeah. I'm happy to wrap this up however you want, and then at your discretion we can have another conversation sometime. I should tell you that in two weeks my wife and I are going to Scotland, and we'll be there until Christmas.

Brock: Well, maybe what I could do is I can work on getting this transcribed and edited, and then we can have another session after-- when you get back from Scotland.

Moore: That sounds great.

Brock: Okay. Well, let's just, if we could for a few moments talk about, just follow out the piece table and word processors and text editors. Did you-- after you left PARC to go to SRI and then to the University of Texas, did you have any more involvements with this sort of world?

Moore: No. I made small changes to TXDT for Charles after leaving and I wrote that TXDT memo for PARC in the early '80s. If that came out in '81 I was either in Texas or about to go to Texas because I joined UT in 1981.[15] Yeah, I mean basically the main reason I left PARC was that I wanted to work-- I wanted to work on theorem proving and PARC was more focused on document preparation and office information systems and I had made a real push to try to get Jerry Elkind, the leader of the Computer Science Lab, to go-- to make an investment in automatic theorem proving and program verification and Jerry just didn't think it was the right thing for them. He was probably right about that, but it wasn't what I wanted. I didn't want to be working on document preparation and office information systems anymore and when I first joined PARC the way it worked was that you could spend about a quarter of your time doing anything you wanted and the other three-quarters on tasks that PARC was interested in and-- but over time, I

[15] Reply edited for clarity by J Strother Moore.

felt the screws tightening in the sense that I had less freedom to do what I wanted to do, less time to do it. And so eventually the attraction of joining SRI and working directly with Boyer all the time on theorem proving was just overwhelming, and that's what-- and once I left, I didn't look at-- I mean I used Interlisp. That's how our theorem prover that we worked at SRI was written, in Interlisp, but I didn't participate in its development anymore and I didn't work in text editing except when somebody would ask me to do something to the-- to that package. Yeah, so-- and I should note that the string-searching algorithm was developed while I was at PARC but Boyer was at SRI and when-- because it was really developed by Boyer and me for use in our theorem prover. I mean we needed to do some string searching and that was the algorithm that occurred to us but when I first presented it to the PARC authorities with asking permission to publish a paper about it, it was-- PARC denied it because it was so close. It was kind of like interesting proprietary information but because Boyer was at SRI they couldn't have stopped Boyer from publishing it and I pointed that out and said I would be happy to just take my name off of it and let Boyer publish it because it's a cool algorithm. And so they relented and agreed to let us publish it, but in any case, that was pretty much my last interaction, aside from minor changes to TXDT and writing that memo.[16]

Brock: Okay, and I think like from speaking with Charles Simonyi, they definitely-- I mean he definitely said that when he got to Microsoft they implemented their-- the word processor in the C language but using the piece table with these ability to have metadata attached to the different pieces, and I also saw that at least through the first Word for Windows, at least Word for Windows was also based on the piece table. I haven't yet been able to track down if it continues to be. That's something I want to take up with the people at Microsoft and see if I can find that out.

Moore: Right. I don't know the answer to that.

Brock: Okay, and I'm...

Moore: Yeah, I'm a very-- I don't know how to put it. I'm just focused on my stuff.

[16] Reply edited for accuracy by J Strother Moore.

Brock: Oh sure, sure. No, no, no. I get it.

Moore: I'm happy to do things for people and then I-- and I hope they make use of it, but I don't often even follow what they're using it for because...

Brock: It's probably why you're successful doing what you're doing.

Moore: Yeah. Oh, it is. Focus is 99 percent of the thing.

Brock: Well, maybe this is a good time to break. Thank you so much for this interview today. It has been just so helpful for me and so clarifying about-- I mean I think it's this-- to me, an absolutely fascinating story about how these tools, this kind of approach to text editing developed squarely within the context of AI research ends up kind of in this program that is what a colleague of mine, Matthew Kirschenbaum, has called "the No. 2 pencil of the digital age." So it's such a great story and I hope to be able to kind of work on it further and I'll look forward to continuing the kind of overall story about your work in AI after-- when you're back from Scotland.

Moore: Yeah. Okay. That'd be great but I'll leave it to you and I really appreciate your interest in this and I'm really happy that it will be documented so that it won't be lost to history when we all pass away.

Brock: Me, too.

Moore: Okay. Well, thanks for talking with me, David, and I'll talk to you again sometime when you send me the edited transcript.

Brock: Will do. Thanks again so much.

Moore: Okay, bye.

Brock: Take care.

END OF THE INTERVIEW

