Oral History of Steve Russell

Interviewed by:
Al Kossow

Recorded: August 9, 2008
Mountain View, California

CHM Reference number: X4970.2009

Al Kossow: We're rolling. It's August 9th, 2008. This is Al Kossow we're in Mountain View, with Steve Russell. Thanks for coming by.

Steve Russell: You're welcome.

Kossow: So I wanted to start with early history. So where were you originally from and what was your family like?

Russell: I was born in Hartford, Connecticut. My father was an engineer; my mother was a teacher turned housekeeper.

Kossow: What sort of engineer?

Russell: Mechanical. He told me a bunch of stories, on his knee.

Kossow: Oh, okay. So where did he work?

Russell: He worked for the Hartford-Empire Company which became part of Emhart, which is still around in a much different incarnation. They built glass machinery, and he had worked on building and installing annealing lehrs, which take the hot glass fresh out of the mold and heat it up again to the point where it oozes but doesn't fall down, you hope, and then cool it down very slowly, which relieves the stresses and makes it possible for the glass to survive. If you don't do that, if you just take it out of a mold after it's been molded and let it cool down quickly, the glass becomes very fragile, like the demonstration, Prince Rupert's Drops and Bologna bottles, where you just do something very small to it and it explodes.

Kossow: So did he end up going around the country when they were installing these?

Russell: He did a fair amount of field work. He also had several patents on construction details for lehrs. He worked on automatic control systems, which at the time were vacuum tube chart recorder systems where a chart recorder would record something and provide control signals to control it. And he also worked on installing the actual molding machinery and giving advice on the operation of the glass furnace, which at the time— and I think it's still true— but at the time the construction of the glass furnace was there would be a large central furnace, maybe 50 or 100 feet in diameter, and the raw— the batch that was going to be melted, would be put in there, in one side, and then there were long tunnels, called feeders, which took the glass out of the central furnace and heated it up to— or cooled it down, as the case may be— to exactly the right temperature for molding, and ended up with a device that would drip glass and cut off pieces of red-hot glass and send them off to molding machines. It took something of the order of five or ten days for the batch that went into the furnace to actually get out there to the far end of the feeders where it turned into bottles. But it was— what went in affected what went out, and so there was a considerable element of back magic involved in figuring out when things weren't coming out right, at the far end, what was wrong somewhere along the way.

Kossow: Did you have any siblings?

Russell: What? No.

Kossow: There was no siblings.

Russell: My father grew up in Worchester. He was born in 1898, so he grew up in Worchester around the turn of the 1900. And he told me stories about that, and one in particular that I remember is he was— one of his things was when he was a kid he went down to a foundry, which was relatively close to where he lived, and learned about the engines and what was in the foundry. It occupied two city blocks, one for the foundry and one for the machine shop, and it had its own power; it had a 300 horsepower Corliss engine, and that ran the whole two blocks of the factory. The biggest load was the blower on the cupola.

Kossow: It was all electric?

Russell: No.

Kossow: Well the mechanics—

Russell: Belts and line shafts.

Kossow: Oh, okay.

Russell: With a line shaft that went over the street that separated the two blocks of factory.

Kossow: Okay, so it was a single motor driving the entire—

Russell: Yes.

Kossow: So this would've been in—

Russell: 1905 maybe, somewhere around there. And one of the— in the, well I think, sometime between 1945 and 1948 his company acquired the Pitkin Governor Company who had been making steam engine governors for— and governors for anything, for that matter.

Kossow: Like fly ball governors?

Russell: Yes. Which had a factory somewhere near Hartford, I don't remember exactly where, and they were shutting down the factory because it was woefully obsolete; namely, it had a central steam boiler and steam engine and everything in the factory ran off line shafts. And they had a beautifully maintained Corliss engine, and the last day they were operating it my father got me in and I got to see it. The reason they were operating it on a Saturday was because the manager was cleaning out his office, which was on the second floor, and he was kind of lame, so they had to keep the elevator running.

Kossow: Oh, and it was all driven off a common—

Russell: And the elevator was run off clutches off the line shafts.

Kossow: So was that one of the last Corliss engines running in Connecticut?

Russell: I don't think it was the last.

Kossow: No.

Russell: But I don't know. It was certainly very late for a Corliss engine. They did have electric lights, but there was no sign of electric motors. And it was very interesting because it was very quiet. Even though there was sort of some belt noise, which was just occasional flapping and a tick as the—

Kossow: The leather belts.

Russell: Yes, a tick as the splices went over the pulleys, but the Corliss engine itself made no noise at all when it wasn't putting out any power.

Kossow: So when did you get interested in trains?

Russell: I don't remember, and I'll tell you why.

Kossow: Did you do a lot of train traveling?

Russell: Something which I've been told about, and which there are pictures of me, is when I was about three, we took a train across the country to visit my mother's folks— my mother's father and her sister and brother-in-law in the State of Washington. So that was a long train trip, which I don't remember. But the picture shows me lying down on his living room floor looking admiringly at his O Gauge model, which was running on track on the floor. So I was definitely interested then, but I don't remember it, and the next Christmas he gave me a Lionel train set, and I haven't recovered. And that was— I got interested— well I liked running the trains. When I got to be around ten somewhere, I'm not sure exactly where, I started getting interested in signals and doing more elaborate layouts and that sort of stuff, and that's what got me started getting interested in electronics. And then my father got laid off in, I guess the end of 1948 or the beginning of 1949, and my folks decided to move out to the State of Washington and go into farming with my grandfather. And we made a long auto trip out of that. We went down south and around and covered almost all the extant narrow gauge railroads at the time: the East Broad Top, the East Tennessee and Western North Carolina, the Denver and Rio Grande and— that was about it; we didn't cover the California and Sierra Railroads.

Kossow: So you father was a big train fan then too?

Russell: Well he was tolerant at least.

Kossow: Oh. Oh, so he was doing this for you.

Russell: Yes.

Kossow: So you'd just try to visit the track or actually try to get rides?

Russell: Yes. Well in some cases they were still carrying passengers and in other cases they were— the crews were susceptible to begging.

Kossow: So you'd hop on and then your parents would meet you.

Russell: Yes. So anyway my parents, my father went into partnership with my grandfather, which didn't work out very well and lasted less than a year, and then we moved up to a small farm in Burlington, Washington. And so I went to Mount Vernon High School, which was near Burlington, and when we moved— I started my freshman year in high school when we were in Mount Vernon, and when we moved to Burlington I kept going to Mount Vernon High School. And I had some friends. My friends were mostly interested in electronics and radio and that sort of thing.

Kossow: Did you get a ham license then?

Russell: No, I got a ham license much later, much, much later, but I let it lapse. But I learned in high school a great deal about radio and World War Two radar because there was plenty of surplus World War Two radio and radar around. One of the guys who was a little older than I was, not much, was an avid ham and his equipment tended to be made by butchering, really crudely, surplus radio gear. One of the things— his favorite metal working tool was an ice-pick.

Kossow: Ugh…

Russell: Which tended to leave volcanic holes in the aluminum chasses but it was effective, when wielded with enthusiasm.

Kossow: Okay, so then did you end up getting a scholarship to go to Dartmouth, or how did you end up back on the East Coast?

Russell: Well my aunt married- my father's sister married G.W. Pierce, who was a Harvard professor, and they felt that everyone should get a proper education, and so she paid for my tuition. Uncle George was a— started out as an acoustics expert, and in the 1930s he studied the acoustics of sound waves in quartz and how you could use the sound waves in quartz as a resonator for electronic oscillators; the Pierce oscillator circuit was one he designed. And he then went off— one of the things that he showed me one time when I was visiting, in the late forties, was— in fact there were several things on that visit. One was a closet in the Applied Science Laboratory where he had attempted to build a pendulum clock that would compete in accuracy with the crystal clock. But it wasn't successful, the crystal was still better, but it was an electrically driven pendulum in a vacuum chamber and quite elaborate. The same visit he also took me to see Harvard Mark I, which was busily printing tables for something or other, and Professor Aiken showed us around, and he checked to see what it was doing and decided he could demonstrate the error detection feature. And so he put his finger in one of the typewriters so that it kept a number from striking, and there were contacts on the typewriter that noticed whether the numeric keys had actually completely hit, and so that rang a bell and caused operators to come out of their offices and so on. And they were also— down in the basement they were working on an electronic machine, but that wasn't nearly as interesting and apparently it wasn't working yet.

Kossow: So that would've been the Mark II.

Russell: Two or three, I'm not sure.

Kossow: The first drum calculator.

Russell: So I thought that was interesting.

Kossow: And so was that your first exposure to computing?

Russell: Other than just doing- learning math, or learning grade school math.

Kossow: Automatic computation.

Russell: Yes, yes. Well let's see, had I— I had seen adding machines before.

Kossow: But there was no machines up at Dartmouth yet?

Russell: Oh no. My senior year at Dartmouth, 1958, '57, '58, I worked on the most advanced data processing operation that Dartmouth had, which was a card shop with a 402 or 403, the slow tabulator. And that had just been installed in the last year or so.

Kossow: Okay, so here it is, it's 1958, and somehow or other you get hooked up with John McCarthy.

Russell: John McCarthy was— in '56, '57, and maybe a little earlier— was a professor at Dartmouth. And he had arranged to get the SNARC. Marvin Minsky built the stochastic Neural Analog Research Computer, which was an electronic rack about this big, a tabletop rack, which was attempting- demonstrating an electronic mechanism for doing learning the way it was thought that human nervous systems did, and it was a bunch of vacuum tube, relatively straightforward vacuum tube electronics, and you connected them randomly when you got the— and you would put a pulse in and then you got something out maybe. And when you got a result you liked you said, "Yeah that's good," and that caused all the neurons that had fired in the last operation to crank their little pots, probability pots, up higher. And when you said, "No that's bad," it caused the ones that had fired in the last operation to crank their probability pots down lower. And this was done with— they remembered whether they had fired, and what that did was cause a magnetic clutch to pull in, and then the reward or punishment was done by turning a- by a motor running for a bit to crank the pots over a little.

Kossow: And they all changed by the same amount.

Russell: Yes. Much later Minsky fought the Perceptron wars and apparently proved that that couldn't do it. And so I had worked on that, and McCarthy, John McCarthy had arranged that, and he offered me a job at MIT working on computers, which he thought I could handle. The summer— I didn't fulfill all of my graduation requirements. I was supposed to produce a senior thesis but I never got around to it. That summer I had a job in summer stock at the Berkshire Playhouse in Stockbridge, Massachusetts, which at

last observation was still running, under a different name. I had been quite active in theater work in college, and I was student technical director of the Dartmouth Players my senior year.

Kossow: So this was stages and props and lighting.

Russell: So I spent the summer in summer stock getting paid $25.00 a week plus board, or plus room rather. The room was in the basement under the scene shop, not a particularly wonderful place but it didn't rain— well it didn't rain there often. Well when the— there was one incident where the scene designer had gotten enthusiastic with washes where you'd put all the flats down and then throw a big bucket of colored water on them. And it turned out that he did that one Saturday, and that got all the beds kind of wet down in the basement. We were not happy.

Kossow: So there was more than just you down there.

Russell: Oh yes. Yes, well it was some of the apprentices and the rest of the technical crew. Anyway—

Kossow: So you got through that.

Russell: Yeah, we got through that. Well actually I learned a lot and— no I actually had fun at it.

Kossow: Did you do much theater work after that?

Russell: I did a little when I was working at MIT but not too much. Computers were more absorbing, at the time anyway.

Kossow: So you get to MIT then, it'd be—

Russell: The fall of '58.

Kossow: So at that point they had the 704.

Russell: At that point they had the 704 installed and running. I had worked the previous summer— actually I had worked at MIT for John, and he sent me to work using FORTRAN, which was just out of the box, and I did a little, some matrix inversions and some other stuff in FORTRAN. So I sort of knew my way around there. Anyway, when I arrived— I don't remember the detailed sequence of who did what but my impression is that John had figured out the basic machine language code for CAR, CDR and CONS, on the 704, and had—

Kossow: So he had in mind what the basic atoms of Lisp were going to be at that point.

Russell: Yes. I'm pretty sure— he may have had garbage collection in mind but I don't remember that it was mentioned first off, but it came fairly- it came sometime that fall. And it was abundantly clear, from later events, that no one had thought through garbage collection completely at that point. So anyway we

started writing some sample code. We did list input and output routines and maplist and a few other of the standard functions; I don't remember the other ones, but we had written all those in assembler.

Kossow: So this is just you and John working on that.

Russell: No, there was— Klim Maling was another full-time employee, and there were several students and grad students associated with it one way or another. I don't remember their names but there were a couple who were seriously thinking about compiling Lisp, at least by December they were. And we were certainly very conscious that it would be a good thing to compile Lisp, although it was much easier to write an interpreter. And given the size of the 704 it was probable that compiled code would have given even less space for free storage. So compiling wasn't on the high priority list.

Kossow: So just to give a background of what was going on at the time, there was John working on developing Lisp, and that was in which department? So what was going on around him at the time?

Russell: I think that was in electrical engineering. John and Marvin had started the artificial intelligence project I guess that summer; so the summer of 1958, something like that.

Kossow: What building was it in?

Russell: Building 26, and I think Building 26 was mostly electrical engineering. We started out in a dungeon in the basement next to the coin machines, which was echoy and noisy. We later on moved up to the second floor. So anyway we had been, Klim and I some of the students, had been hand assembling, or hand compiling, Lisp M-expressions for map list and print and read and that sort of thing. And my remembrance is that John sort of came in one day, in late September or October or something like, that with the universal M-expression, that is the Lisp interpreter written out as an M-expression, and we sort of looked at it and said, "Oh yeah, that'll work," and I looked at it and said, "Oh, that's just a matter of doing more hand compiling, like I've been doing. I can do that."

Kossow: So that's how the first one was bootstrapped.

Russell: Yes.

Kossow: Just by hand.

Russell: And actually most of the code is probably in still in the Lisp 1.5 listing. But I got something working before Christmas, which was a useable interpreter; no garbage collector, but there weren't any big programs yet. And my remembrance is that in the beginning of December or so Jim Slagle, a blind grad student, had a seminar where he explained how he was planning to do formal integration in Lisp, and he started writing his Lisp expressions on the board, or getting them on the board somehow, I don't remember exactly how that was done. But he had no trouble understanding Lisp and writing Lisp, but he used a Braille typewriter. There was no connection between that and the machine so somebody had to read the output to him. And I, and I guess Dan Edwards— I think Dan Edwards was an undergraduate at the time, I'm not sure— and a couple of the grad students who were worrying about how to compile Lisp were sort of sitting in the back, and he wrote something and then it had a clever double recursion in it, and we sort of started muttering, the interpreter can't handle that. What happened was, the initial version

there was a problem with something like maplist where there was a functional argument. And the problem was if the functional argument had a variable by the same name as one of the internal variable's in maplist, that actually— if it was all interpreted— that was actually visible on the association list, the A list, then inside the functional argument you would see the map list version of the variable name before you'd see the intended version, which was out in the outer call to maplist. And so anyway we had been using a dodge like defining a function- or using internal variable names like *terribly unique program variable*, and it was clear that with this double recursion that wouldn't work. And so there was a great deal of muttering—and, I think I invented it, I'm not sure; I'm sure it got critiqued by plenty of other people— what we fixed on what was generally called the function funarg hack where when the functional argument got passed a map list, it got extended with- it got transformed from the function to funarg, which had the function and the current A list at the time of binding. So what happened in the machinery of map list or whatever else was churning around with the functional argument wouldn't be visible to the internal function when it actually got called. And at the time we thought it was a rather despicable thing, and we didn't like it all because it was going to be hard to compile. Nowadays there's a lot more patience for things that are hard to compile and a lot more memory to use for such things. But it worked, and so it stayed and— oh, and also no one ever suggested a better idea. I don't remember— I believe the garbage collector was proposed in the fall of '58 and I don't think we started— I don't remember when we started working on it seriously. I wasn't— I guess it started in parallel with the second version of the interpreter which had the function funarg hack. That got working sometime in the spring of 1959. And the other project was to get- was to make the garbage collector work. And one of the things which became evident much, much later, as garbage collector bugs kept showing up for years and years, was that it would've been a really good idea if we had segregated the storage for list structure, and the stack for list structure, from other kinds of storage that weren't involved in garbage collection. But we hadn't and so what we did was make up a table of all the locations where list structure could be stored and use that to interpret the stack— let's see, I think we also must have had a template for the function. So anyway so there was a list of all the places where list structure could be stored and a way of determining what things on the stack were list structure and what were not. And it turned out that those lists were not fully correct, because they were made by hand by reading the code, and so for at least a year, and I think longer than that, we would periodically get bugs where somebody had done some perfectly innocent function and had gotten the free storage list tacked in the middle of its list structure, because something that shouldn't have been garbage collected was. So I'm sure that we hadn't thought that out when we had fixed the programming conventions.

Kossow: So you had something sort of working then in the spring, '59?

Russell: Yeah, I don't think we had the garbage collector fully functioning in the spring. I think that was- took another while.

Kossow: So were there very many people trying to use the language at that point? Because you said somebody was trying to do integration with it.

Russell: I think there were of the order of five or ten serious grad students, and John inflicted it on his Introduction to AI class, and so there were a bunch- there were maybe another 20 more or less student users. And I'm not sure exactly how big— I don't remember exactly when this happened and how big the user base was at the time. But sometime later John figured out that CAR and CDR weren't really good names. And so he tried to launch a program to call them *FIRST* and *REST*, which seemed like much better names. And I remember supporting this and trying to help in it. But it didn't work. There were too many entrenched users who were used to CAR— had internalized CAR and CDR and didn't want any

change. And I don't know how many there were at the time but it couldn't have been more than 50. But the damage had already been done. I'm reminded of the original, the landline Morse code story. When Morse first developed the telegraph he had— there were, some words had long spaces in them and there was a long dash that was assembled, and it very early became evident that this led to transcription errors. The original Morse receiving instruments were siphon recorders.

Kossow: What's that?

Russell: Ink siphon, recording on paper tape, driven by clockwork. But after the first year or so they realized that the operators weren't bothering to keep the ink supplied, because they were listening to the tink-tink of the recorder and recording it directly, writing it down, having learned what the tink-tink sounded like. So then they said, "Okay, we don't need that," and they built the “modern”, that is, the telegraph sounders that are around, which were simply designed to sound like the siphon recorder. After about ten years or so, when Morse telegraphs were being expanded to Europe, they had to revise the character code to handle some of the European characters, and at that point they eliminated the long dash and the inter-character long spaces, and they tried to persuade the U.S. operators to adopt that. [Shakes his head ‘no’] And so landline Morse kept its intra-character space and long dash until the last landline Morse circuits were shut down. Same problem.

Kossow: Yes. Now just so that it's on here, CAR and CDR come from—

Russell: Contents of Address Register and Contents of Decrement Register.

Kossow: Which are two registers in the 704.

Russell: They are two fields in a 704 instruction, and because they were used for indexing— they're used in the indexing instructions— and because they were used for indexing there were special load instructions which would load either address or decrement into the accumulator. So this made it easy to do instruction in CAR and CDR. And that was the reason— that was clearly an effective way to do list structure, and that was the reason they were called that. There was also another—

Kossow: John had come up with that?

Russell: Yes.

Kossow: Those two functions?

Russell: And we had no problem thinking about it that way for the first year or two. It was only after trying to teach it. Another feature of the indexing instructions on the 704 was that they used two's-complement arithmetic and so the number that you wanted to store in the list structure was actually the two's complement of the location in memory that you were pointing to, and it turned out that made—

<video breaks up >

<Crew talk>

END OF TAPE 1

Kossow: All right, we're rolling. So this is tape number 2 of the Oral History of Steve Russell. So we're at mid-1959, MIT. So at that point, you're concentrating mostly on working with John on Lisp. At that point, had Jack Dennis gotten the TX-0? So what's going on? Computing at MIT at that point, was this 704 by 1959?

Russell: 704 but not much else; Whirlwind, I guess, in the fall of 1958 or 1959—

Kossow: That's when they were shutting it down.

Russell: They were shutting down Whirlwind, and since a bunch of people in the Computation Center had worked on Whirlwind, they arranged some tours of Whirlwind. And I took one of them, and one of the things I remember was the “bouncing ball” program. Now, much, much later I have gotten the impression from reading some of the Whirlwind literature that somehow that was felt to be very important. I think it may have been a benchmark for Whirlwind's ability to do real-time stuff, but I haven't seen any direct evidence of that. But there's some indirect reference that—

Kossow: So was this running on the large scope?

Russell: This was running on a large scope; this wasn't one of the Cape Cod scopes. That was still classified. But this was running on a scope that was in a rack by the operator position. It was more or less the size of the PDP-1 scope, it may have been a little bit smaller. The PDP-1 scope is 10 inch, I believe. It's 9.25 inches diagonal, x and y size for the display area. At any rate, the bouncing ball display— there was a wall and a floor that was drawn. The floor had a little hole in it. The ball started off on the top of the wall. I don't remember whether it was just a single spot or whether it was round or something more. But anyway, the ball got pushed off by an unseen hand, and it fell down and bounced on the floor, in smaller and smaller bounces, showing that the physics were correct, and it either bounced off the edge of the screen, or it bounced into the hole and went down through the bottom of the screen. And when it went through the hole in the bottom of the screen, it then stopped changing the amount of force it got pushed with. So you got push… bounce, bounce, bounce, bounce… push a little harder… bounce, bounce, bounce, bounce… push a little harder yet… bounce… bounce… bounce… boip! And they had a speaker on Whirlwind, and so it made relatively distinctive sounds for the bouncing. I don't know how they contrived that, whether they contrived that, or whether that was just the way it worked; but it did come with sound effects.

Kossow: So did you just get a tour of the control room and the Flexowriter array area? Or did you actually go into the computer itself?

Russell: We went into the computer itself, but they kept us on the main pathways. It was live; it was 300 volts exposed on those racks. It wasn't a place where you wanted a large mob.

Kossow: So, were there fans above it, cooling? I'm just curious as to how loud it was inside.

Russell: I don't remember an impression that it was loud. Of course, I was used to 704 computer rooms, which weren't exactly quiet, and tab shops, which weren't quiet. But by the data processing standards of

the day, I don't think it was loud. I think they had air conditioning and ducting, but I don't remember for sure. Anyway, there was more stuff to do on Lisp, and we worked on Lisp for a while. I don't remember anything about TX-0 appearing. At some time around, I think it was 1961, late 1961, early 1962, I had gotten bored with working on Lisp and got a job at Harvard. And part of that involved my not having a deferment anymore, so I went into the Army Reserve, and I think—

Kossow: So you had had a deferment while you were working for John?

Russell: Yes. I'm not clear on the exact sequence of events— anyway, I started working for Harvard; I had to go off for six months of active duty; I came back, management had changed; new management was not nice, and—

Kossow: So you didn't stay at Harvard very long?

Russell: No. And when I came back from— about the time I was convinced that I didn't want to continue working for Harvard, John was moving to California, to Stanford, and offered me a job at Stanford, which I took. So I ended up at Stanford in the fall of 1962. But, popping back a little earlier, when I was working at MIT, I got active in the model railroad club and continued to be active when I was working at Harvard, and a number of the people in the model railroad club were associated with Jack Dennis's lab, and TX-0 and the PDP-1 when it arrived in 1961. I don't know exactly when TX-0 got going, but it was going noticably at least a year or more before the PDP-1. That had a display. It had roughly the same complement of I/O gear that PDP-1 had, and you could write similar type programs with it; it wasn't quite as fast, but it was transistorized.

Kossow: So was Jack's lab in Building 26, too?

Russell: Yes, it was also on the second floor of building 26, just down the hall. One of the things— a missed opportunity in the museum setting up its PDP-1 restoration display I thought, was that it would have been very easy to recreate the building 26 décor, which was painted cement block and tile floor. Probably not as interesting as the mural of the production line.

Kossow: Trying to pretend it's The Mill.

Russell: Anyway, the PDP-1 arrived, and Marvin Minsky wrote the tripos demonstration, generally called the Minskytron, and there was the famous weekend where the mob of undergraduates transcribed the macro assembler from TX-0 to the PDP-1, because they didn't like FRAP. And then fairly quickly thereafter, they wrote DDT and connected up the macro symbols to DDT. So that was all sort of in place by the middle of the fall of 1961. And the combination of the Minskytron and having DDT with interactive debugging with symbols was very tempting. I don't remember the exact order of things, but I'm pretty sure I started talking up a better demonstration program than the Minskytron, and eventually, Alan Kotok went up to Maynard and collected the sine and cosine routines from DECUS, presented them to me, and said, "Okay, here are the sine and cosine routines; now what's your excuse?" And I discovered I had run out of excuses; I had to actually think. And so I started work and figured out the basic trick of Spacewar! display which is that you only need to calculate a unit vector pointing in the direction of the spaceship. And you can express everything else the spaceship does, and the outline of the spaceship in terms of that unit vector, suitably scaled. So it's basically a lot of addition in the usual program upkeep.

Kossow: Do you want to just give an overview, then, of how Spacewar! actually works?

Russell: It's one big loop, and the loop is on the displayable objects. And I called them displayable objects, although I didn't know about object orientation or object-oriented programming at the time.

Kossow: So you have the sun—

Russell: Colliding objects, not displayable objects. The colliding object is a space ship, there are two of those. And that has a lot of extra data with it. It shares the position and velocity tables with all of the torpedoes and explosions that are running around. So there's just one big loop through the colliding objects, and it looks at all the higher-numbered colliding objects to see if there's a collision, using an octagon because you don't need to calculate the square root of anything, you can do that by work on X difference, Y difference, and X+Y difference—

Kossow: So the bounding box for the collision detection is an octagon?

Russell: Yes. So it goes through, it sees if this object is colliding with any higher-numbered object. If it is, it replaces the calculation routines. That's another thing that every colliding object has, is a calculation routine. It replaces the calculation routine with the explosion calculation routine. And then things take care of themselves. Then, after it's decided whether it's an explosion or not, it goes off to the calculation routine. And the calculation routine updates the position, since all colliding objects have velocity; and if it's a spaceship, it worries about reading the controls and updating the other things about the spaceship in deciding whether to launch a torpedo or not. And if a torpedo needs to be launched, it searches up the colliding object table for an empty slot, indicated by having no calculation routine. It searches up the table for an empty slot, puts a torpedo calculation routine there, and the spaceship position plus the suitable increments, so it won't run into its torpedo, and of the velocity of the spaceship plus an increment for the torpedo, and it goes on. When that the main loop gets done, you go off and do some star display and display the sun, and calculate — and part of the spaceship calculation is to calculate the effect of gravity on the spaceship. Originally, there wasn't any gravity, and I had an interpreter, which interpreted the outline description, and Dan Edwards, sometime in late 1961 or early 1962, looked at that code and decided if he could write a special purpose compiler which would compile precisely the right code, and proceeded to. And there's one compiled outline for each spaceship; each spaceship actually does half of the spaceship outline and then you twiddle the vectors and do the other half. That keeps the display running just as fast as it can. That gives time to calculate the effect of gravity on the two spaceships, but not on the torpedoes. So we decided that they were photon torpedoes not affected by gravity.

Kossow: So when the explosion routine starts, it continues calculating motion, so the explosion moves?

Russell: Yes. If you see two spaceships collide, if you watch closely, you will see that there are two explosions that continue off in the direction that the two spaceships were going. There is another number in the table for all colliding objects, which is the size. And this is, roughly speaking, proportional to the amount of computing it takes to compute that object. And at the end of the loop, as you go through the main loop, you accumulate the sizes also; and so at the end of the loop, there's fritter away time loop that attempts to keep the frame rate approximately constant. It doesn't do a wonderful job; it's visually adequate, but God help you when you try to take a movie of it.

Kossow: One of the complaints with all modern, little kids trying to play it now is that it's TOO SLOW.

Russell: Kids who are used to something like Asteroids seem to think that. But when we do the demos, we get a number of people who seem to be still quite addicted to it with the old, slow version. Now, that was always a complaint; the reason that all the parameters got accumulated in the first page of the listing, which says you can put that first page of the listing of the console, and anyone who wanted to try a different set of parameters could. But the ones that were compiled in or assembled in were the ones that I thought were good. Now it turns out I'm not a representative arcade game player, and so my version of the parameters is slower and gives more opportunity for marksmanship than the arcade version.

Kossow: Right, that's the whole thing with gravity and doing the, what's that called, where you whip around the sun? Does it have a name, where you whip around the sun and you shoot?

Russell: The closest name is the "CBS maneuver" is what happens when two lazy experts fight each other, which is they both turn at right angles to the sun, and fire for 3.5 or 4 seconds, so they're now in stable orbits, and they know it. And then they turn at each other and start trying to place torpedoes where they think the other one is going to be. And so the trails turn into an eye around the sun, and CBS used that as a logo, so it got called the "CBS maneuver". One of the spaceships can go the other way around the sun, so that both ships meet on the same side of the sun. But that seems usually to have less chance of winning. Not much less, but somewhat less.

Kossow: It's better that you stay on opposite sides, then?

Russell: Yes.

Kossow: And then at some point, you added hyperspace?

Russell: Yes, and we realized that that was going— I don't remember whether we actually had it. I may have had it for a little while with no limit, but it became very clear that someone who didn't understand could use hyperspace to escape their proper justice forever, and so we added the unreliability of hyperfield generators very quickly. One thing that Asteroids and the arcade versions, the later arcade versions of Spacewar! added, which actually was a big help, especially with their high acceleration rates, was “training mode” where space was actually viscous. So if you got your ship accelerated so that it was going across the screen so fast you couldn't understand what was going on, if you took your hands off, the situation would gradually become understandable. I don't think I would have been persuaded to do that in the original Spacewar! because it was “unrealistic.” But it definitely made it easier to learn.

Kossow: So are there any other favorite anecdotes about Spacewar!, or just the spread of Spacewar!?

Russell: Well, the "imitation is the sincerest form of flattery"— many people saw Spacewar! as, some people ask for copies of the source, and of course we gave them out because we very briefly considered trying to sell Spacewar!. We realized the only possible customer was Digital Equipment, and we also, on a little reflection, that they were too cheap to do it. So we gave it out to anyone who wanted it, and some people got the listing and thought about it, and by reading it— a lot of people simply saw the game and had a computer that wasn't the PDP-1 but did have a display, and implemented Spacewar! their own way. I suspect most of them figured out the “basic trick”, but I'm not sure.

Kossow: What “basic trick” were you thinking of?

Russell: That you could do everything based on the spaceship unit vector. How are you fixed for the source code for different implementations of Spacewar!?

Kossow: We have a few. I don't know if we have the PDP-10 version. We have the 12 version, and I think we have a PDP-7 version.

Russell: I think Bob Saunders wrote the PDP-7 version. I think one 6 version simply ran on the PDP-1 simulator. I think there must have been others, but I don't know. Something for some history grad student to pursue. When we were running a demo for the Yelp event, there was one woman who had done Spacewar! in turtle graphics as a high school programming project, which she wasn't too happy with. She seemed to like the demo of the original Spacewar!.

Kossow: So turtle graphics running on a micro or something like that?

Russell: I didn't quiz her. I didn't have the opportunity to quiz her further.

Kossow: So she thought this version was better?

Russell: No, she just thought it was nice to see the original.

Kossow: So she was in her mid-20s?

Russell: 20s or early 30s; probably 20s. The DEC field service story where the DEC production people got into the practice of loading Spacewar! the last thing before PDP-1 shipped, and field service would then unpack it, make sure that nothing horrible had happened, tried turning on power, and starting Spacewar!. And if it worked, they would call the customer over and say, "See, it works." If it didn't work, then they'd worry about it. In the restoration project, we had a little reflection and we decided that probably if Spacewar! works, just about everything works, as far as machine instructions go. It doesn't guarantee all the I/O gear works, but it does multiply and divides, and just about every instruction. So a lot of people implemented Spacewar! just from knowing that it existed and having seen it maybe once.

Kossow: Were there any other interesting hacks you remember from back then, at MIT? There's all the TMRC train stuff.

Russell: Yes, under the layout provided, the TMRC had developed a telephone equipment scrounge method, which consisted of the Western Electric Educational Gift Program. The things that Western Electric was giving away to educational institutions came out in a catalog toward the end of the school year. So for years and years, TMRC members would go over to the electrical engineering department supply room and get a copy of the catalog and look to see what they thought the layout could use, ask for it, and then Freddie Broderick, who ran some part of that, got a station named for him on TMRC, would then duly request it from Western Electric, and most of the time it would come through. So there was a carefully sorted store of relays and switches and general electrical stuff, telephone quality, under the

layout, which got used for making parts and layout, and also got used for the first two versions of Spacewar! controls.

Kossow: Those were the three-way lever switches?

Russell: The version that I remember quite clearly, and nobody else does, so I think they must have died very early— there was a 1930s design, 4-pushbutton block that was used for buzzing buzzers in an office; so you'd have a receptionist who had maybe one or two or three lines on a phone and when one of them rang, she'd pick it up, find out who it was and then buzz the buzzer to get that person to pick up the phone. Those were nice because you could hold them in one hand and play them like a flute. But they died very quickly; they were designed for being pushed a few times a day, and it turned out that there was a fatigue problem when they were pushed hundreds of times a day. And so they died fairly early. Then somebody, it may have been Bob Saunders, made up the second version, which is the one that's Shag sketched and everyone remembers, which is a wood box with a slanted masonite top, and a left-right-off switch, and up-off-down switch, and a pushbutton. And that was used for rotate left, rotate right for the horizontal one, accelerate or fire torpedo for the up down one and hyperspace for the pushbutton.

Kossow: And were quiet enough that you couldn't tell what the other guy was pushing?

Russell: Yes. They were switchboard type switches, so they didn't click; also the first ones didn't click. And one of the problems with clicking switches, which is what people tend to start with, is you could tell when your opponent is out of torpedoes or rockets because you hear the switch clicking and nothing is happening on the screen.

Kossow: So most of the play, then, was done with the lever switches?

Russell: Yes.

Kossow: So the exhibit that we have isn't really accurate with the pushbuttons?

Russell: No, but those are genuine— those are clicking arcade type pushbuttons. But usually when we do the demo, the demonstrators don't get to the point where that matters. Occasionally, it does. But the improvised quality of the control boxes is quite authentic, and they always were pretty improvised.

Kossow: But they've held up pretty well. It's been running for two years now, and haven't had any problems with the boxes?

Russell: No, in fact, there have been occasional problems with the cables, but given the craftsmanship of the original cable construction, that's not surprising.

Kossow: So go back to 1962, then, when you came out to the west coast?

Russell: Yes. At that point, they were building what was then the computer center complex; I forget the names of the buildings, but anyway. Eichler of Eichler Homes was building them. We were scrounging

for— we had some time on a 7090 down at IBM San Jose somewhere. I don't remember exactly where, but we had weekend or nighttime time. So we did a little work on Lisp, and—

Kossow: So this was all 1.5 at that point?

Russell: Yes, and John also had the idea that we could build, we should build a display-based time-sharing system using PDP-1 and what we would we could get with custom displays. And he and Pat Suppes worked up a joint proposal to do this. I don't remember the exact dates on that, but at any rate, it arrived, and we also got custom displays from Philco. That took a while, to do the request for proposal and evaluate that sort of stuff.

Kossow: This is all coming out of ARPA funding?

Russell: I think so, I don't remember. Stanford should have it all in their archives somewhere.

Kossow: So how many people were working with John, then at that point? You had come out, so you were working for him full time?

Russell: I don't remember. There was the usual herd of grad students. It wasn't a big herd; I think it was the order of four or five. I don't remember if there was anyone else full-time. I think there were, but I don't offhand remember who all they were. The PDP-1— one of the things that DEC figured out— I'm not clear on who did the pushing on them; I suspect some combination of John and Ed Fredkin provided a lot of the motivation. But DEC figured out that they could build a swapping drum which would replace the contents of 4K of PDP-1 memory in one drum revolution, which meant that the drum had to be 4000 words around, and the data rate had to be a little slower than the 5 microsecond memory of a PDP-1. It was about 8 microseconds, I believe. And because the drum was 4000 words around, you could start the transfer at any point to the appropriate words, to the appropriate address and have it wrap around and end up in one revolution without any latency penalty. That was the basis for doing the PDP-1 timesharing system, the idea being that you'd get a 16K PDP-1, you'd have 12K for system, and 4K for user, and then you could swap users in and out. Part of that was there was a computer science department committee, which included Niklaus Wirth, sort of blessing the operation, the design, and so on. And Wirth was very insistent that the keyboard should include modifier keys. I remember that I resisted this idea; I believe John did, too. But Niklaus Wirth was very insistent.

Kossow: And what did he want to use the alternate sets for?

Russell: He felt that modifying the meaning of a keystroke would be a good additional control mechanism. I don't remember his exact detailed argument. And since they were just contacts, it wasn't very expensive to put them in so it was easier to put them in. And of course as soon as they started working, everybody started using them, and they've been on the keyboard ever since.

Kossow: So this was in addition to the normal control keys.

Russell: No, in addition to the normal shift key. These were essentially an ordinary typewriter keyboard.

Kossow: Right, because they weren't ASCII yet.

Russell: Yes.

Kossow: So why was he [Wirth] called "Bucky"?

Russell: He was very enthusiastic. He got the nickname Bucky Beaver. I don't know exactly the history of that.

Kossow: Just eager beaver.

Russell: Yes.

Kossow: So they were called “bucky bits”?

Russell: Informally.

Kossow: The modifier [keys]. So they were a separate bit in the encoding?

Russell: There was a space bar. There was a space bar, which was sort of at the bottom of the keyboard, and they were just square buttons, maybe just a key and half space off from the ends of the space bar, one on each side.

Kossow: And then when they came back from the display, they were just extra bits that got tacked onto the encoding.

Russell: And since the PDP-1 system— they seemed to be quite useful for editing and the like on the PDP-1 system, so when MIT AI was doing new keyboard system, they imitated it, and they may have added another one. But at any rate, they never fell off the keyboard after that.

Kossow: All right. Well let's take a break. It's a couple minutes to go on here, let's head on into tape number three.

END OF TAPE 2

Kossow: This is tape number three of the oral history of Steve Russell. You were just getting the PDP-1 time-sharing system going at Stanford**,** and you had just gotten the Philco displays.

Russell: Yeah, and we got them going. Philco did custom displays.

Kossow: Were the displays in a room adjacent to the 7090 and the PDP-1?

Russell: The displays had a central controller that was almost as big as the PDP-1. And they had big cables that went out to 12 displays. I think the cable length could be something like 100 to 150 feet. It was one off. It was done by Philco military people. It was very high performance for the time, not nearly as high performance as was originally promised, but high enough. It could do text much better because in addition to a high performance deflection system, it had an analog character generator system. So there was a digital memory for the strokes involved in making a character, but an analog system that produced high-speed but small deflection signals to actually draw the character on the screen at the current position. And being high-speed analog being pushed as hard as it would go, the printing— the character set, the first time people looked at it, was nearly unreadable. However, after a week or so, you get used to it just like getting used to somebody's handwriting, and so it was no particular problem, wasn't too satisfactory. And the overall speed of the system was not what was promised, and in fact the negotiations over the final price with Philco started out with, well, the system seems to be performing at about half the speed, so we think it's worth about half the price. <laughs> Things went on from there. But at any rate, they didn't make a terrific lot of money on it, that's for sure.

Kossow: Could it only draw characters or could it do graphics as well?

Russell: It could do graphics also. It had a vector generator which could cover the whole screen. The reason the graphics were fast was because they just covered a character position. And I think they had a separate deflection coil for the characters. So there was a gross deflection and then a character deflection. But we could run 12 users and the response was pretty good. And you could run 4K PDP-1 program. And it was six…

Kossow: What sorts of programming did you do on that system?

Russell: Other than working on pieces of the various operating systems that we tried to do for it and spending an unhappy summer evening— several— many, a couple of months of unhappy summer evenings debugging the IBM 1301 disk interface, the only thing I really did was a 3-D flight simulator. I didn't do all of it. I think, let's see— Bill McKeeman wrote a program he called GOGOL which was a compiler, a sort of ALGOL 58 compiler, which in fact fit in a 4K PDP-1 and compiled code that fit in a 4K PDP-1. And he provided the flight equations and we had a little— and I think I provided the scenery, which was lights. And so we had a program where you could fly a simulated, real plane. And the landscape consisted of a field with runway lighting and it was night, so there was a town with a grid street display next to the air field. So everything was done with sequences of lights on a line, uniformly spaced lights on a line with a perspective view so as that they gradually disappeared in the distance. And that was instructive but not the— as a game it wasn't too much fun because it was too much— it was very slow-paced. Now, that's realistic. The rate at which things go on in an airplane when it's going cross-country and not going very fast isn't very fast. But it's a lot more interesting when you can turn around and look at the countryside than when you're looking at a screen and it's night. <laughs> So, that was interesting, but not a good game.

Kossow: The start of the CRT-based text editors were on those though, right?

Russell: Yes, and in fact <laughs> we had— I didn't do the text editors but I was certainly in on the text format and the way things worked. It turned out that we had a pretty good text editor, eventually. We had one problem in that if inserted a lot in the middle of a text file or at the beginning of a long text file, it would provoke copying, recopying the whole rest of the text file to provide some space. The basic storage idea

was you had the standard read text routine, which would read a block from the disk. And the block always had the information about how much of the block was actually in use. So for text reading, you would read the block, pick up the word count or— and go that far. And you knew when you got that far you had to go to the next block. So the reading was reasonably as fast as you could do it, but you didn't have to fill the block completely. And so the meta-information, which was the pointers to all the line beginnings, was stored in the top of the block. And there was some more notes in the blocks so as it, I think it started, there was a pointer than said how far that extended at the top of the block. And then the space in between was for expansion of the text. And that way, by knowing where all the lines began, we could get reasonably fast searching. And by having the block normally starting out half full, there was room for a lot of editing without requiring any copying or anything. You could just read the block, edit it and write it back or maybe read the two adjacent blocks, edit them and write them back.

Kossow: So these blocks were being swapped in from the 1301 disk?

Russell: The 1301 or— yeah.

Kossow: The drum was just used for swapping or was there a file system on there too?

Russell: I don't remember how it was distributed. There was some file system available on the drum. But it wasn't all that big. I forget, was it 32 files? I don't know. I think there's a brochure for it running around loose somewhere, but I don't remember. May have been just 16 files, but that's still a four files over for non-swap, not involved in swapping. <laughs> As that system got going and with— six displays were for computer science department, six were for Pat Suppes' programmed instruction studies. And when that system got going it had the <laughs> most trying computer bug I've ever had to deal with, which was "the drum with two beginnings". This drum, you know, had a counter which kept a position. And when the system got to be used heavily, what we discovered was every now and then it would— programs would get really trashed and looked like they'd just been relocated— we looked carefully and eventually figured out that it looked like they'd just been relocated in memory. So the right code was there, but it was in the wrong place. And a bunch further detective work indicated that the drum was sometimes screwing up the locations where it was swapping the addresses. And…

Kossow: Did it appear to be swapping on 4k boundaries?

Russell: It was still doing 4k boundaries, but it was just relocating the 4k that was swapped into different addresses, modulo the 4K. Well, there was much weeping and a great deal of study. Of course this only happened when it was in heavy use. So we had to whip up a diagnostic to show it, which we managed to do. And then we called in the DEC and Vermont Research who had made the drum and they determined, oh, it seems there are two beginnings to this drum sometimes. Turns out, the way they got the index mark was they had one track reserved for the index and one track reserved for clock. And what they do is write a single transition on the index track, which was the beginning of the drum, and then write 4,000 words, 4,000 pulses around the drum and twiddle the frequency of the writing when they were writing the clock track, so as that there was some gap, but not a very big gap between the end of the 4,000 words and the beginning. And they'd always start writing when the index results came around. And that was fine. But it turned out that when things were very busy and there was lots of writing, you were doing writes and reads at the same time, in some cases, there would be enough cross-talk to generate a spurious index pulse, which would zero the counter.

Kossow: How did they get around the problem?

Russell: They moved the twisted pair that carried the index pulse signal carefully away from everything else. But that was a long saga, because it took at least a week, I think it may have been more than a week, to get it all sorted out. And there were lots of people standing around, you know, when is it going to be fixed? When is it going to be fixed? And to debug it you needed the whole system.

Kossow: So you had to run it really hard and then you could put a scope on the index and every once in awhile see this erroneous pulse.

Russell: Yeah. And this is before the day, you know, with the modern storage scope there'd be no problem. But this was the days when you put a hood on the scope and twiddled… <laughs>

Kossow: And tried to see the extra little thing that isn't there very much. The learning group kept using it after John moved off to the PDP-6?

Russell: Right.

Kossow: So that was about 1965?

Russell: Something like that. I don't remember the times. You know, I was busy with setting up the PDP-1, herding the PDP-1 maintenance and at the time then, we got the D. C. Power Lab and there was a bunch of setting up with that and getting the PDP-6 to work. As I say, I don't remember the times, but the PDP-6 was more or less working, and I left in about '68 I believe.

Kossow: I was just reading John Markoff's book and the whole story about how the D. C. Power building happened, and that there's nothing inside of it, and there was no air conditioning…

Russell: Oh yes, there was air conditioning. There was the Great Mother Of All Air Conditioners. <laughs> No, the central air conditioning had been provided and there was a 300 ton, a BIG 300 ton air conditioning unit down in the basement. And that was always fun for us because— well, usually it was fun. When it didn't work, it wasn't fun. Anyway, it eventually, because it was so big, we started calling it the great mother of all air conditioners or GMOAC. <laughs>

Kossow: So was it a central unit supplying the whole building?

Russell: Yes, the whole building.

Kossow: But it wasn't enough air conditioning with the 6 in there and everything else?

Russell: Oh no. No, it was designed on the idea that it was offices. And a big computer room with that vintage computer was a lot more than offices. And of course offices, you know, had a desk and a light. It didn't have a computer on the desk. The neat thing about the basement was, it was a partial basement and was all concrete. And it was curved, the building was curved, so it was sort of a science fiction

location. And taking people down there and leading them into this room with this giant green machine was sort of surreal. We liked taking people on that part of the tour.

Kossow: So you ended up having to shuttle a lot between campus and there?

Russell: Yeah. And eventually I ended up at the, in the Power Lab.

Kossow: So you overlapped with Les [Ernest] for a couple of years?

Russell: Yes, yeah.

Kossow: What did you end up working on out at SAIL?

Russell: Mostly herding maintenance and connecting peripherals and that sort of stuff. There was plenty of maintenance to do on a PDP-6. We had, generally had one or two technicians to help with that but I was supervising that. You know, there were lots of things happening.

Kossow: Widgets being built and robots running around?

Russell: Well, and we were debugging hardware while the time-sharing system was running, which was periodically exciting for all concerned.

Kossow: Do you have any interesting Dave Poole stories?

Russell: <laughs> Let's see. I don't think I was there. I wasn't very aware of the tree house. What was it? I don't remember. <laughs> The PDP-6 was stretching the technology, it turned out. It wasn't planned to be, but it was. And so many things in the PDP-6 were marginal for one reason or another. One thing that was marginal was the memory bus, which was DC, marginal at DC at full length as— in hindsight what I say now is that the notion of noise margins was not well developed at the time those circuits were designed. I don't think they calculated noise margins AT ALL. At any rate, there was an additional problem in the memories that we had, which was that they had— they were built with the latest two-sided technology at the time, which involved, at the time they were designed, which involved eyelets instead of plated through holes because plated through holes weren't considered reliable and probably weren't reliable for quite a while. I wasn't exposed to early unreliable plated through holes. It turns out the eyelets were brass. And the idea was, you put the eyelet through the board and you just crunched down and flanged it out. So, on the side you had a eyelet that was flared out at about a 45 degree angle and a piece of etch running up to it. And when it got soldered, you had a solder fillet in the, between the eyelet and the etch. But it also turned out that the coefficient of thermal expansion for circuit board and brass was not the same. So as the system heated and cooled, there was always a little push and pull. Now, solder doesn't behave well under tension, and under repeated stress tends to fatigue and crack. And the cracks tend to corrode. And it turns out that lead oxide or tin oxide or some combination of them makes a poor but intermittent diode. And so <laughs> the essence of the thing was that the memories kept developing cold solder joints. And so we ended up with a program of doing what we called running mechanical margins. I revealed the method of running mechanical margins to the PDP-1 restoration group a year or so ago, and they were amazed. They just weren't familiar with this. Anyway, what we did was we would run the memory diagnostic and start tapping gently on all the boards in the memory that

had this problem. When the memory diagnostic rang the bell to say there was a memory error, then we'd start localizing the tapping to try and locate the board. It turned out you could do this fairly reliably. And the alternative would have been to take the computer down for something over a week and redo all the joints, which was a really big job. And we concluded it was better to run mechanical margins for a couple hours once a week than do that. And what we'd do is we'd…

Kossow: So you just take the handle of the screwdriver or something, just tap on them?

Russell: Tap gently. Because, if you tapped on the back of the board that had the problem, you'd get pretty consistent failures and it you tapped on its neighbors, it wouldn't be consistent. So then we'd pull the board and redo the connections on that board. But it was kind of laborious because the fix we used was, we'd take a piece of copper wire, we'd clean out the solder in the eyelet, we'd take a piece of copper wire and run it through the eyelet and then solder it on both sides. And so then when the eyelet stopped conducting, it didn't matter. But there were like 40 of those on a board, so— and there were, you know, lots of boards on the machine, so it was ugly. And as part of this, the system programmer decided that I needed to be alerted whenever there was a memory problem. And so whenever the system detected— they fixed it so whenever the system detected a memory error, it rang the bell on every teletype in the building. <laughs> So there was no escape. And this was when all of the office terminals were teletypes. So it was…

Kossow: How long did it ring it? For, like, a minute or just once?

Russell: No, just once for each error. At any rate, because of this memory intermittence, there was— we did have problems on acceptance test too, or DEC had problems on acceptance test. And Bob Clements was the DEC engineer who herded the acceptance through. I don't know whether he was still working in field service. He started working in DEC field service and he went on to engineering fairly quickly. But anyway, he was herding the acceptance and because the memories were so intermittent, we kept finding memories problems and reasons not to. And eventually, the final acceptance involved Dave Poole stomping up and down on top of the memories and having the memory diagnostic continue to run without reporting any errors. <laughs> It was a nice high computer room, so you could easily stand on top of the boxes and jump up and down or stomp. I don't think he was jumping up and down, he was stomping.

Kossow: How big of a hardware crew was there back then, a dozen people? How many people were at SAIL when you left in '68?

Russell: I think there was like 25 or 30 offices. I'm not sure. You know, some of them were two or three students to an office, some of them were individual. Let's see. What was it? There was a technician for sure, and usually another engineer working on hardware and on some combination of maintenance and hooking up new stuff, because there were various arms and TV camera and other things getting hooked on. So…

Kossow: So by the time you left, you weren't really doing much programming at all?

Russell: No. And when I left, I went to work for a startup in Seattle called Computer Center Corporation. And there it was some amount of herding the DEC hardware people and a fair amount of customer training and some helping with applications-type programming, but not much programming.

Kossow: Were they KI10s by that point or KA10s.

Russell: No, KA10s. This was the first round of time-sharing startups, Applied Logic and CompuServe, and Computer Center was a little late in that but essentially the same group.

Kossow: So where did they come out of?

Russell: University of Washington computer people, mostly, and me and Dick Gruen from variously DEC and Stanford. The . . .

Kossow: Did they end up adding any special hardware to the KAs?

Russell: No. <laughs> The communications hardware was really special. I forget what it was called. It was the Straight-8 with the bit banging teletype hardware attachment instructions.

Kossow: The DC01? It wasn't the 680.

Russell: Maybe it was 680.

Kossow: Okay. So it was like a 32-channel teletype multiplexer?

Russell: But it was done by adding, in essence, a “receive one bit and transmit one bit” instruction to a PDP-8. And so the software had to change the base address, do the instruction and then change the next base address. And I forget the horrible details, but at one point I wallowed in them to try to put in five-channel teletype code, to mix five-channel and eight-channel in the same line, which was in principle possible, but in practice very hard. That was GENUINE PDP-8 programming, where you look the— <laughs> you got these…

Kossow: Count the cycles?

Russell: You had the assembler listing with all the timings and you kept adding them up and worrying about it. But the thing that was especially memorable about that was the console. The PDP-8 was a Straight-8, installed on a rack, but still a Straight-8. And we had a problem with static control in the computer room. I think they didn't put any humidification in originally, and we also had some cheap plastic chairs in the computer room. And at one point in the middle of winter, I remember sitting on the chair in front of the stopped 680 and getting up and having the chair go zap, zap, zap, zap to the strips at the side of the floor tile, and sitting down and having it go zap, zap, zap, zap again, and watching the PDP-8 program counter increment. So to keep things running, we had the— during the daytime we insisted on no shoes in the computer room. And <laughs> don't get anywhere near this PDP-8. We also had a giant Bryant. This was the big giant— the big Bryant moving head disk there, first moving head disk which had a hydraulic actuator and…

Kossow: All these vertically stacked platters.

Russell: Yeah. That was not as reliable as was desirable. But the main struggle was we just didn't get enough business. And so eventually they went out of business. And at that point, I guess I had annoyed DEC sufficiently that they thought maybe I would be useful in marketing. And so I started working for Digital.

Kossow: There's the one little story that you had a couple of testers up in Seattle.

Russell: Oh yes, yes. We did have a testing program. Was it TOPS-10 by then? I'm not sure. At any rate, the multi-user operating system for the PDP-10 was not a paragon of reliability. In fact, it started out with a pretty dismal mean time to failure. And so we had a couple of people working on basically fixing, full-time, fixing bugs in the operating system. And there was no real lack of bugs to fix. Monique Rona had— was one of the principles, and her son was going to Lakeside School. And so she arranged that we could— we give some time to the Lakeside School who had got a teletype. And we also had built a training room which had six or eight terminal teletypes in it. And what we would do is on Saturdays especially, we'd open that up to the Lakeside students to come in and load test the computer. We wanted lots of people. And the rules were, you can do anything you want, you can try to break the system if you want. But if you do manage to break the system, you've got to tell us what you did and don't do it again until we tell you. And this was quite fruitful. We collected heavy load bugs this way. And two of the students were Paul Allen and Bill Gates. And they had to be admonished several times about the “don't do it again until we tell you” part of the rules. But they certainly had a great deal of fun and they evidently learned quite a bit.

Kossow: You remember anything in particular that they did?

Russell: No. They asked lots of questions. And one of the things that we would do at Saturdays was, Dick Gruen or I or somebody would be in the terminal room, or periodically stick our nose in the terminal room, make sure that the pandemonium was under control, and answer questions.

Kossow: So how many terminals did you have in the terminal room?

Russell: I think it was six or eight. The idea was, you'd use it for doing training classes because we were training people and using— I think we had BASIC by then. There was one professor at the University of Washington was using computer time who was very annoying. And I had sort of heard of his reputation. I hadn't dealt with him directly when he started. But I got him a couple of times, and knowing his reputation, I had somehow set my mind that I was not going to be annoyed by him. And I managed not to be, which was a little surprising, and I tended to have relatively short patience. And so since I didn’t <laughs> run away, I'd swear never again when I talked to him, I got him for, as— you know, if he had a call it got to me very quickly. And eventually, we got to be quite good friends and he was— we got along well. But it was mostly because I had decided for some reason that I just wasn't going to let him annoy me. Now, I don't usually do that, and I've not done that a number of times since when it would have been a good idea. But…

Kossow: But you managed to get through it. So would this be 1972, 1973 then when you went to DEC?

Russell: Yeah. I started in DEC I think in late '69 or '70, something like that.

Kossow: So the time-sharing business didn't stay around very long then at all?

Russell: No. Well, it managed to operate, it turned, out nine months after its net worth was zero, which was rather amazing.

Kossow: Just ended up not paying the bills for…

Russell: Yes. Well, you know, those things like travel agents and…

Kossow: But you were out of there by then?

Russell: Well, no, I was around, sort of, as a spectator, as a bemused spectator, sort of preparing for the next paycheck not to appear.

Kossow: That's what I was wondering, if they stiffed you for your pay. You managed to get out of that.

<crew talk>

END OF TAPE 3

Kossow: All right. This is tape number four of the world history of Steve Russell. So we're launching into your career at DEC.

Russell: Well, let's see. I started out in PDP-10 marketing, which translated as maybe marketeer. And what that translated to was running benchmarks and explaining sales pitches of a certain amount of system configuration. And I started out in building 5 in the mill of 5/5. Let's…

Kossow: So this was still before the KI10 came out? They were still selling KAs?

Russell: I don't remember the exact phasing of which…

Kossow: Because it was right around then when the KI10 came out. Did you end up going on field trips a lot to customers?

Russell: Not a lot, but some. And one of the more memorable disasters was Bonneville Power Administration— actually it turned out quite well. But Bonneville Power Administration was looking for a central computer to gather together the data and to run predictions. Because Bonneville Power is the federal agency that runs most of the power dams in the Columbia Basin starting with the Grand Coulee. And one of the problems that they wanted to optimize was when they— the water storage is almost all upstream in Grand Coulee and a couple of other dams up at the headwaters. And so once they let water out of Grand Coulee or the other storage reservoirs, the water goes down to the sea sort of irresistibly over a period of about 48 hours. And what they want to do is optimize it so that they don't let any water go over the spillway and get wasted on the way, they want to go through turbines and generate power.

And so by regulating when they— how much water they use in which dams, they can try to optimize it so that the head is the maximum and yet none of the water goes over the spillway.

Kossow: Were they worrying about peak demand then, as well?

Russell: Yeah. They knew what their demand pattern was. But they also wanted to keep track of what was going on in real-time for reporting. And so they had a real-time benchmark. And I was one of the guys working on the benchmark. And one of the things we had to do was time a number of things. And so we got the benchmark running, the usual style, late at night, and came the big demonstration. And we started it, started running the benchmark, and things were all right for a bit. But then it started coming up with negative elapsed times for some of the tasks. And we scratched our heads a little and figured out oh, it was because one— we were triggering one of the start or stop, I forget which from one— from the internal crystal clock and the other one from the power line clock. And so when the power line clock was slightly off when they were making up for the power line having gotten slow, then we started getting the negative elapsed times. Well fortunately, Bonneville Power understood that explanation. And so as we were— the two of us were sort of commiserating after the long demo and trying to wind down, the salesman came bouncing in and said, "Hey boys, you're in real luck. I persuaded them to give us another chance," to which our reaction was, "Oh, shit," <laughs> because we got to work our ass off for ANOTHER two weeks.

Kossow: So you switched everything over to the crystal clock?

Russell: Yeah. We fixed the timing problem and ran the benchmark. The benchmark worked fine. They bought the system and much later the salesman gave me a tour of the Bonneville Power Control Center so that worked out all right after all. But it was memorable, and continued my education on benchmarking.

Kossow: So then you stayed in marketing support for a couple of years?

Russell: Yeah. Wasn't too much too memorable other than that, I think. I did herd a product through the process. It wasn't a very big product. There was a remote printer for the 10 systems, which had PDP-8 and a card reader, and a printer, and some Teletypes on it.

Kossow: Ran over a synchronous line?

Russell: Well, yes. And the printer was inexpensive and had a cam instead of a— this more or less standard paper tape— wide paper tape carriage control unit like the IBM printers have. And I wrote it up something like "a vertical format preset to 11-inch forms." And I guess I did a pretty good job because the salesman on the account that was complaining, that wanted to run checks or something like that called up and said, "Does this mean it does what the customer wants?" I said, "No. It's just describing what we have and it still doesn't do what the customer wants." But at least the salesman read it. Dick Gruen's [ph?] joke was, "He was never sure that the salesman read any of the technical material, but he's QUITE sure that they knew every detail of the commission plan." Of course the DEC commission plan was very simple. There wasn't any.

Kossow: So the salesmen were always straight salary?

Russell: Yeah, which has benefits but had its disadvantages, too. The DEC sales force was clearly not in the IBM league, and the IBM sales force certainly had a commission plan. But one of the things that I've noticed is that IBM was very careful to use their very best salesmen to sell this year's commission plan to the sales force. Because they always changed it every year to emphasize the things they wanted to push and deemphasize the things they didn't want to push or would sell themselves, they felt. And they had very good salesmen in sales management to sell this— the opportunities in the new plan to the sales force. So it gave them an opportunity to fire up the sales force again at least once a year. And if you don't change this commission plan, or don't have one, you have to try and invent other opportunities.

Kossow: So then you ended up in Computer Special Systems after that?

Russell: Yeah. One of the things that I ended up doing was configuring special systems. And at one point they decided to put selling standard systems in one group and selling special systems with one-off configurations and special peripherals in another group, and I ended in that group and ended up doing a lot of figuring and some amount of sketching out solutions for, you know, listening to customer problems and sketching out solutions. Oh, I guess it was before then, one of— this— because I worked on the remote printer thing I got also involved in the software. And the first version of software had a problem that it was written by a guy who was somewhere around the world's worse software designer, but unfortunately one of the world's best debuggers. And so he had debugged this software almost into operation. And he left, and I had inherited it. And it was in a race with one of the other guys in the group who was rewriting it reasonably from the ground up. Unfortunately he won the race, but I was only two weeks late, but it was a real slog because it was all PDP and assembly language. And it was a full PDP-8 and it was full standard PDP-8 program, out of time and space. And to fix bugs, what the original perpetrator had done was say “oh, it fails under this condition, this condition, this condition. Okay, I'll recognize that and then get around the failing code.” So the code kind of grew, and ran out of space, of course. So a fair amount of the time what I would do is discover I didn't have enough space for the next patch that needed to be put in. So I'd run through and find all the flags that indicated the various horrible conditions that were no longer used because I had removed the problem, and eliminate the code that set and reset those flags, then run a cross-reference again and discover some more code that that eliminated. And so I do this algorithmic simplification for about a week and then I'd have enough space in most of the pages so I could do some more patching.

Kossow: The standard PDP-8 programming, where it doesn't fit, and then you squish the subroutines until it fits. And then you repeat.

Russell: Repeat until done, yes. It was educational but not particularly rewarding. And as I say, it took just as long for another guy to rewrite the code from scratch with a little discipline and get it working as it did to fix the broken code. And, of course, the final result was infinitely better because there was still some space to add features.

Kossow: But somebody in management give this guy the task of doing it.

Russell: Yeah. Ah, yes, there was COBOL. One of the things that was going on was Digital was writing COBOL for the PDP-10, or was getting a COBOL for the PDP-10. I don't remember if it was all in-house or external. And so I tried using it. I did a TMRC timetable in COBOL. Well, I'd never written COBOL, so I was sort of using the manual and what I knew. And so I ended up using multiple subscripts on an array. I forget what else, but basically what I was doing was writing FORTRAN in COBOL. Well, apparently

nobody had tested those things, and so, in particular, the double subscripted array just was just plain a bug. It just started out not working, so I filed a software performance report. Funny how they always report bad performance and never— very seldom saw one with good performance. At any rate that got fixed and the program eventually got working. And it turned out one of the things that we needed was indexed sequential access. This was an IBM buzz word that was around and it's basically an indexed file. And it turns out to be a— it was very useful for random access files because you could look them up on the main index and you could get to essentially any size file that you could hold in about three disk accesses, which is wonderful compared with doing it sequentially or almost anything else you can do. And in fact, it's about as good as you can do. And so I ended up writing some ISAM routines for COBOL. And I hadn't particularly looked at other— the IBM implementations were sufficiently surrounded by jargon that it was hard to penetrate what was going on. So I sort of winged it and ended up with essentially the right solution. And that worked and sold a system, I believe. And unfortunately there was one bug which I never got fixed, which was if you deleted— if a block got completely deleted and went to zero, somehow it— the pointers got confused and the software crashed. Didn't damage the file but it did crash and you had to restart it. But that was interesting, and the solution was very neat and very nice. I used something like the Stanford trick where in the block there was some metadata hidden up at the top of a block. But at least you didn't have to copy the block forward, you just split— when a block filled up you just split the block into two new blocks. And then one write of the index, change the index from pointing from the old one to the new one. So if you had atomic writes, which the hardware usually did, the file would be in good order no matter what.

Kossow: So I guess this is right around the time when the Magnavox lawsuit happened and you had to dig up all the old Spacewar! code?

Russell: Yeah. The time Magnavox lawsuit was happening, I don't remember exactly what the timing on that was but I had to— they wanted copies of Spacewar!, and so I dredged up my copes of Spacewar!. I made some copies for myself and gave them the originals which had the date of my writing on them, and which eventually I think came into the computer history collection, and did a deposition, which didn't go very well. That was about the most reward I ever got for Spacewar! was I got a free lunches from— free dinners from the lawyers. I guess I didn't mention that when Spacewar! was written there was— in principle we were not allowed to patent software and you couldn't copyright it. So there was basically no protection for programs at all. Later on that got changed, but that was the situation with Spacewar!. The only choice you had was never revealing the sources or making them open.

Kossow: Which was hard to do in a university.

Russell: Yes.

Kossow: How long did you end up staying at DEC?

Russell: Eight or nine years. After Special Systems, I went off on a consulting contract to New Zealand which was fun for me, but worked out really poorly. And so I ended up getting shunted off into running the message switching system, the internal Teletype message switching system and herding that for a while, while a new one was being written.

Kossow: So it was like an internal TWX system?

Russell: Yeah. And then eventually I got disgusted with that and tried to fit in another couple of places and that didn't work. So I got laid off and after a while started working for Chase Interactive Services, which was a interactive money transfer subsidiary of Chase Manhattan Bank, which was located in Waltham and was doing money transfer on IBM, big IBM 360s. And I did software on that in COBOL, and also ended up dealing with a worldwide communication system. One of the disadvantages of being in Boston and having the bosses in New York was you've got to have meetings in New York. And Boston wasn't far enough away so as you could do it overnight. So you had to get up at some God-awful hour in the morning, get to the airport, take the shuttle and get into New York about 10:00 and have your— have a day full of meetings and then go back, which made a long and ugly day. I learned a certain amount about expectation control, I guess, from that. There were two pieces to the system. One was a domestic money transfer system where people would deposit— typically branches, subsidiaries, licensees would deposit money in— at various remote locations into some kind of bank account and would call in what they had deposited. And Chase would accumulate it all and make it available as a usable balance for the customer. And so that involved a lot of phone calls of people calling in. And they would typically just call in the dollar— report an account number and a dollar amount. And we had a phone center that took those calls and typed them in on various types of terminals. And the thing I learned from that was that even trained and skilled operators who did that for a living still had a 1% error rate or so. And that's really the best you can hope for, for any sort of keying operation or most any sort of thing that people do. It was constrained, they got plenty of practice. They were reasonably motivated. They weren't dumb. And that's the best you can do. And you still have a lot of people who think they don't make mistakes, and they do. They make at last one percent. And if they're really bad they make 15%. And that's what you have to deal with in designing a human interface.

Kossow: So there had to be all this audit stuff to reconcile the errors?

Russell: Yeah, banks have been running with people for centuries. So the controlling that and dealing with fixing it up was a well-understood and routine operation. The worldwide money transfer system had another problem. It was a state of the art system with PCs, out in the far corners of the world, and it handled big money transfers. Unfortunately the software on both our end and the PC end was not perfectly reliable, it turned out. And the standard sort of problem would be you'd get an operations vice president from Tokyo, or London calling in and saying, "We've got to close the books in 15 minutes and we can't close the books. You've got to do something." Now, unfortunately, closing the books in London, New York, and Tokyo and Sidney is nicely distributed around the clock. So these calls would come in at 6:00 in the morning or midnight. And so it was an unpleasant support nightmare which I started out handling with three or four other people and eventually it got down to two. And when I announced I was leaving, the other person stopped speaking to me and never spoke to me again.

Kossow: Did you just end up digging through the database trying to find out…

Russell: Well, that was what I learned out about expectation control. It turned out the right thing to do was to do a very quick look or a quiz of the reporter to find out about what the problem looked like and then call my boss, so my boss got the word before it came down— before the shit started coming down from above. And so it ended up spending about half-time telling my management what was going on and the other half of the time actually working the problem. And sometimes the problem wasn't soluble, and there were times when we were told we had to close the books in 15 minutes and the books didn't get closed.

Kossow: So they can't close them till they balance?

Russell: Yeah. And if the system won't give them any numbers they don't even have the raw material to balance with.

Kossow: So this is the early '80s then, so pretty much everything is 370s so you're not working with batch anymore?

Russell: No, this was CP/CMS. Just sit at a terminal and correct the COBOL and then tell it to try it, and look to see what happened.

Kossow: So then at some point you moved back to the West Coast? Well, I guess so because you're out here now.

Russell: Yes. <laughs> Tom Zito was a…

Kossow: Well, he's ex-Atari.

Russell: He was promoting the idea of doing video games on VCR— on TV sets. He was starting an operation as part of Nolan Bushnell's Atari. Actually I forget, only very briefly I believe we were part of Atari.

Kossow: So they were still in Borregas in Sunnyvale?

Russell: We started out in the— where was it? Maybe it was always Axilon. There's…

Kossow: That's where Androbot and…

Russell: Was it currently— well it's not clear. There's this rusty building on— just off Lawrence [ph?] by 237.

Kossow: The triangle building?

Russell: No, it's not triangle, it's square. But the COR-TEN self-protecting steel siding, so it's a dark rust color, and that's where we started out. The building's still there. The enterprises have changed at last once.

Kossow: I was thinking of 280. The triangle building's 280. Yeah, 237.

Russell: Like the Mill, buildings stay through an awful lot of enterprises.

Kossow: Yeah, it's like that whole corner down 237 in Sunnyvale where CDC used to have all their software operations. It's down in there.

Russell: Anyway, that was the start…

Kossow: How did you end up getting linked up with Zito and the people on the West Coast?

Russell: Zito knew of me because of Spacewar!, and thought that since he was in the game business I would be an asset. I guess I was, but, you know, the whole enterprise didn't work out perfectly. But the idea was to record special material on a VCR tape and play the VCR, use your VCR, to play into a special box which would have game controllers on it and allow you to play a game on your TV set with TV-quality pictures. Now at the time you couldn't afford to synthesize them. There wasn't enough computing around to do that.

Kossow: And it would have been cheaper than a video disk.

Russell: There was electronics for doing frame storing. So with a frame store that held a couple of frames and a control box, you could selectively pick frames off the interlaced video on— from the VCR and so you could switch around. We developed the electronics. And the way it finally worked out was you could have four live video streams and periodic other updates. So what you could do is do a mass design which had the game control stuff on it and you could pick one of four video streams to show live at almost video quality in the picture window that was let through by the mask. And so you could do various sorts of games. And the two kinds of games that we ended up developing were either sports games like football where you got to choose what move to make and then it played out, or a surveillance camera sort of thing. There was a title, Night Trap, which got transmogrified through a couple of different iterations but started out in the video version. The idea was that you had a surveillance camera that looked at a bunch of places in a house. And the house was being used for a weekend party by a bunch of teenagers and was invaded by some evil monsters called Auger, which look like human beings in black leather. Look amazingly like human beings in black leather. And the idea was that the house had traps, trap doors in it. And you looked around and figured where the monsters were coming in. And if you saw one, if you hit the trap door at the right time it would disappear and you'd get a score. And if you didn't get them all then the teenagers were carried off by the monsters and you lost. But if you did get them all then you'd save them and so on.

Kossow: What year was this?

Russell: Not quite sure.

Kossow: Mid-'80s?

Russell: Yeah. So that wandered on through development in a couple of demos, and it got sold to Hasbro. So it was Hasbro Electronics for awhile. And we got that developed to the point where it was a go/no-go production decision, and Hasbro decided no-go, probably correctly. And so that folded up and I went off and worked for— did various sorts of consulting and ended up working for X-Ray Instrumentation Associates for a while doing gadgetry for x-ray work. And then it revived itself as digital pictures which was taking the same video material and more, and fixing it so it would play on PCs and Macs. I think that was around '85, '86. It was just when we were going from Windows— from DOS to Windows, I believe. No, I think it was going— yeah. And so I ended up taking the code that had— the C code that had worked on the Macintosh and bashing that into submission working on a PC for Night Trap. And that just

barely got out and got published. And we did not have a successful Christmas season, and that folded up. Then I was looking around and started working for Nohau who made microprocessor emulators. And what I was working on there was software which was interacting debugging for C programs. So this is back to making DDT. But the machine was considerably more complicated and it was more C stuff, and so it was considerably more complicated. And the disassembler actually was worse than the PDP-1 disassembler, but I never— it was never the worse problem that we had to deal with so I never ended up…

Kossow: So it was for different microprocessor targets?

Russell: Yes, 8051s was our staple business and it's changed hands, but that business is still sort of alive. And I was working on Motorola 68000s and 6800s mostly.

<crew talk>

END OF TAPE 4

Kossow: This is Tape No. 5, the oral history of Steve Russell. So you had switched then from mainframe work, through games, into tools and debuggers and things like that. So we're kind of into the early '90s.

Russell: Yes.

Kossow: So you've been working on tools since then?

Russell: Yes. Tools and, you know, board and microprocessors, board support type, but debugging and board support type tools.

Kossow: Like embedded systems?

Russell: Yes.

Kossow: And you were, I'm trying to remember, the company that you're with.

Russell: Nohau.

Kossow: Nohau. Oh, those are the little, white boxes that?

Russell: Yes.

Kossow: Ah, okay.

Russell: The original scheme was you have an 8051 on a board, and a tentacle so that you can plug it into an 8051 socket. And you can run the 8051 programs in the one in the box at full 8051, full, old 8051 speed, and operate your board as if the processor, as if it was plugged into the board, but you get debugging. You get visibility on all the memory references and a reasonable debugger, so that you could do source-level debugging.

Kossow: So it had like overlay RAM and then address traps and the usual sorts of things that in-circuit emulators have.

Russell: And that whole business kept getting transmogrified as the chips got faster and faster and denser and denser. The original scheme played a lot of games which depended on the original 8051 architecture. And as it evolved, and other people started doing 8051's, that kept having to get tinkered to deal with the latest hardware. And they also did ones for the Motorola 6800 family. There was a 8-bit 6809 derivative family that was system on the chip, and 68000 derivatives that were systems on the chip. And there the interface was considerably different, because they had different memory interface, and they had background debug mode, which was a not many wire debug interface to the processor.

Kossow: So these were the CPU32 [Motorola CPU 32 processor family] things. Little ten-pin connectors.

Russell: Those got used a lot. The Motorola systems got used a lot in the automotive business for ignition control, engine control, and miscellaneous function control. So they could afford relatively luxurious debugging facilities, although they eventually got over it.

Kossow: And so did you end up working on the firmware that was in the boxes, or the tools that ran on the PC's that controlled the boxes?

Russell: Both. It turned out I was primarily working on the Motorola or Freescale systems, and there you didn't have to do much in the box, but you had the single wire, or not many wire debugging interface, which is basically a serial half duplex interface to talk to the memory and do run control on the processor.

Kossow: So were they bought out by Arium?

Russell: No. Nohau went bankrupt a couple of years ago, and the business got bought up by ICE Tech [ICE Technology], and ICE Tech is still more or less a business. They aren't doing much software development, and I don't believe they're doing much new design, but they're selling the old products and supporting them. And there's still quite a bit of 8051 business apparently, and there's some Motorola business still, too.

Kossow: Did you ever work with ARM at all?

Russell: Only very, very peripherally. Nohau did support ARM, and occasionally I had to help with it, but I didn't have to get into all the details of ARM.

Kossow: Okay. So when did you get involved with the Museum? Were you involved much at all with the Museum in Boston?

Russell: I was involved in the Museum in Boston only in coming to the Spacewar!, the PDP-1 revivals, or one of them for 15 through 20th, or something, anniversary of Spacewar!. That must have been when I was working for DEC. There's a picture from that on the wall somewhere. And then I was working. Let's see. When I got out here at one point I got vaguely involved in the Computer History Museum in that I unloaded all my back issues of Datamation on them. Not a terribly unique contribution, but at any rate it cleared out my garage a little. And then I kept an occasional contact. They had a Spacewar! revival at Moffett where Bill Pitts had a computer, had Galaxy working. So, you know, they knew more or less where I was. And so when the PDP-1 restoration project got started, they got in contact with me, and I sort of said, "Yeah, yeah, do it." And then for the first, I think, two years, mostly what I did was monitor the email, and when they were saying, oh, my God, the display doesn't work, or the paper tape reader doesn't work, I'd say, "oh, yeah, they used to fail."

Kossow: And you've been getting more and more involved ever since.

Russell: Yes. Well, now I get a new audience every couple of weeks, so I can tell the same old stories.

Kossow: Bill Pitts.. that just reminded me. Did they have much contact with you while that was all going on? Did they ever talk to you about Spacewar! back then?

Russell: Back when?

Kossow: When they were working on Galaxy Game?

Russell: No. Bill Pitts, you know, he was hanging around the AI project when I was working there, and he certainly knew what Spacewar! was. And shortly after I left, he got the idea that he could build a multi-user Space War PDP-11 and collect quarters, and he did. And he and Ted Panofsky did a fair amount of work restoring that system. And he has grumbled at me several times since that the Museum doesn't keep it working.

Kossow: Yeah. Do you have any other thoughts?

Russell: Well, there is a pitch for a pet project or two I'd like to do. One of the things which I believe I can still do. We don't have working Flexowriters, and I can't demonstrate it. But I believe I can show how, in the thrilling days of yesteryear, how we used to edit programs on a Flexowriter and give some impression of how slow it was but how it was still possible to get work done. And I think a mercifully short video can be made of that, and I think we can also do something that's longer and is a useful full explanation. But I think somebody, probably me, needs to do some writing. Because my impression is that something like that, the mercifully short demonstration especially isn't going to fly if it isn't scripted. Not that I'm good at following scripts, but writing the script is a good exercise and it allows you to time it. And if you rephrase it on the fly, then it usually doesn't upset the timing too much. Actually, I think it would also be useful to get a similar thing for a keypunch, as there some of the details of card drifting and stuff that are important to getting things done and make it a little less painful. But actually I was thinking that one of the things— Remember I mentioned with Lisp how, if we had segregated the storage, it would

have been a lot easier to debug, and there would have been a lot fewer garbage collector problems? That is something which nowadays with good editors, and programmable editors, and easy copy and paste, I wouldn't hesitate to think about if I had a program like Lisp stated without the garbage collector, I wouldn't hesitate about going back and segregating all the storage and doing those changes. But in the days of cards, when doing this to a card deck involved a great deal, no checking and a great deal of grief, it was just sort of something that was just too horrible, and it's now something which I would easily consider doing.

Kossow: Yeah. Just the state of tools that have. People complain that the tools really haven't evolved very much, but, right. Yeah.

Russell: <laughs> Well, one of the noted but forgotten, but unread pieces of literature is this Mythical Man Month. At Juniper, where I'm working now, we're dealing with a version of FreeBSD and applications, which is huge. At last report there were something, there were claimed to be 6,000 software engineers working on it. And they're complaining about we aren't getting features done on time. It takes too long to do a compilation. You know, all of the tools are just too slow.

Kossow: Maybe the code is just too big.

Russell: Yeah. But the code is probably at least one, and probably two orders of magnitude bigger pile of code than the system 360, which provoked Mythical Man Month.

Kossow: Right. And in 2008 the size of systems that we're trying to deal with are just so much bigger than was being done in the '60s. Right?

Russell: Yeah. And one of the other things is almost none of those software engineers are writing— are dealing with brand new designs. They are dealing with an extension or a modification of a current design.

Kossow: Or they're implementing something that's a standard.

Russell: Yeah.

Kossow: That you have to deal with what is the—

Russell: Or taking the standard thing and it's got a new. Well, okay, this worked on last year's hardware, but now we have this year's hardware and the interface is different. And the system starts out on a dual-core control system, and that actually deals with a third of maybe four to 32 separate sub-control systems that actually shovel packets. And those are now being implemented on N-Core MIPS systems where N is greater or equal eight.

Kossow: It's a lot of processors.

Russell: Yeah.

Kossow: So then you have all the synchronization nightmares.

Russell: Yeah. Well, there's a sort of old curmudgeon lunch group that periodically marvels at the state of things. And their sort of consensus is there are probably still a bunch of one-liners in the operating system. It's just that nobody particularly excites them. And probably if the load were radically different there'd be a whole bunch of new bugs that show up. But because the load isn't radically different, and the environment is reasonably settled and reasonably stable, it's possible to produce what looks like very reliable software, which is what is in practice very reliable software.

Kossow: You just never tickle the bugs.

Russell: Yeah. But we think they're there. It's okay since we know they're there. But it's still, I don't know the counts, but I think FreeBSD is probably spectacularly larger than System 360. And I know that what we distribute is spectacularly larger than FreeBSD, 'cause there's a whole pile of high performance applications in there. And so it's amazing. And we managed to make it work, but it's easy to prophesize gloom and doom.

Kossow: So would it have been better if it would have been implemented in Lisp instead of C or C++ or whatever it's being implemented in?

Russell: C damn it. C++ has been dragged in with kicking and screaming. But most of the code is still being implemented in C, and there is no— Well, I guess some of that you can blame on the free software movement, too, which likes C because it's more portable than C++, or five years ago, it was more portable than C++. One of the things that I did do when I was working at Nohau was every now and then when I had to do major surgery on a module, I would spend the extra two hours or so it took to make it compile in C without errors and then compile in C++ without errors. And when I did that, then the C, the somewhat better type checking in C++ would find my errors better. And even though it was still C, it was still written as C, we were was still ahead compiling it in C++. But Juniper isn't to that point yet.

Kossow: Now, it's the issue of having to turn products and just push out the next product after this one is done.

Russell: And, well, there are some pieces of discipline that Juniper has fairly well. They do insist on reviews before check in. And a lot of the reviews are thoughtful reviews. There are some disparaged groups that seem to do a lot of rubber stamp reviews. However, when they integrate with the full system, they seem to have a lot more problems. No, Lisp would not be a good language to implement something like a high performance router like Juniper is doing. There's a basic problem with garbage collection. If you do it simply, it causes everything to stop when it happens. And if you do something to avoid that, the overhead on the garbage collection becomes a lot more, and you still end up writing code with the illusion that free storage is infinite. And there's no way to guarantee— Free storage is not infinite. And reviewing code to establish or prove that it uses a bounded amount of storage, is really, really hard, and probably in a complicated system, a system with anywhere near that many lines, you don't stand a chance. And so it's questionable whether you should even start on using Lisp for something like that.

Kossow: What I was thinking was just a higher level language other than C.

Russell: I think there are a bunch of places where C++ or Modula 2 or 3 or, you know, something like that, which allows you to have objects, and encourages some discipline, would be a better way to do things from scratch but, you know, how you would get there from here is a little hard to say. One of the things which— if you can constrain the set of objects available, the types of objects available, to the class program so that you can say these system routines can only use these objects. The device drivers can only use these kinds of objects, and you can handle all— You could arrange to handle all of the inter-processor communication or inter-thread communication through a constrained set of objects where you have to use the high priority version if you're running a high priority thread, and you have to use the low priority version if you're running a low priority thread. I think you could build a structure that is a lot more—a lot better at gracefully dealing with errors and a lot better at not encouraging stupid programming errors. But I ain't see one, and I'm not sure I'm prepared to design one.

Kossow: So there it is, the state of programming in 2008.

Russell: The state of one corner of programming. These are embedded systems, by the way.

Kossow: Right, where things break. Your car stops working or, your packets don't get routed.

Russell: Or the Internet gets stolen.

Kossow: We haven't talked about security at all. Oh, well, thank you very much for this marathon session.

Russell: You're welcome

END OF INTERVIEW

