SMITHSONIAN INSTITUTION

LEMELSON CENTER FOR THE STUDY OF INVENTION AND INNOVATION

Stephen (Steve) “Slug” Russell

Transcript of an interview
conducted by

Christopher Weaver

At

Computer History Museum
Mountain View, California, USA

on

8 January 2017

with subsequent additions and corrections

Preferred Citation:
Stephen Russell, "Interview with Stephen 'Slug' Russell," conducted by Christopher Weaver, January 8, 2017, Video Game Pioneers Oral History Collection, Archives Center, National Museum of American History, Smithsonian Institution, Washington, DC.

Acknowledgement:
The Smithsonian's Lemelson Center for the Study of Invention and Innovation gratefully acknowledges financial support from the Entertainment Software Association for this oral history project.

Abstract

Stephen Russell begins the oral history by describing early life and education leading up to his arrival at the Massachusetts institute of Technology. Russell then discusses development and evolution of the computer game *Spacewar!* for the Digital Equipment Corporation PDP-1 computer and his career path from MIT to Harvard to Stanford University. As the informal leader of this pioneering open source coding effort, Russell also highlights the roles of others in the overall development of *Spacewar!* and the organizational and technical conditions which allowed its development to occur. Additionally, Russell discusses his key influences for creating a physics-based space game and his inclinations towards programming and engineering.

About the Interviewer

Christopher Weaver is a Distinguished Research Scholar at the Smithsonian's Lemelson Center for the Study of Invention and Innovation, Distinguished Professor of Computational Media at Wesleyan University and Director of Interactive Simulation for MIT's AIM Photonics Academy. He has contributed to over twenty-five books and publications and holds patents in telecommunications, software methods, device security, and 3D graphics. The former Director of Technology Forecasting for ABC and Chief Engineer to the Subcommittee on Communications for the US Congress, he also founded the video game company Bethesda Softworks. Weaver is co-director of the Videogame Pioneers Initiative at the National Museum of American History, recording oral histories and developing new applications for interactive media and public education.

About the Editor

Justin S. Barber provided transcript audit-editing, emendations, and supplementary footnotes to this oral history as part of his broader work into video game history and digital museology.

Table of Contents

8 January 2017

Video Game Pioneers Oral History Collection

Interviewee:	Stephen (Steve) "Slug" Russell
Interviewer:	Christopher Weaver
Date:	8 January 2017
Location:	Computer History Museum, Mountain View, California, USA

Weaver: Steve, would you please, for the record, tell us your name?

Russell: My name is Steve Russell. Actually, my name is Stephen Rundlett Russell.

Weaver: Thank you.

Russell: And if I want to be *really* pompous, I could be S. Rundlett Russell.

Weaver: Sure, you could. We like that.

Russell: [Laughs.] When I was in college at Dartmouth my freshman year, I worked on the radio station, and the juniors and seniors on the radio station decided that I was "Slug." They would never tell me why, so it's lost in the mists of history.

Weaver: Got it. You're just identified that way, so I guess you have to live with it.

Russell: And when I escaped from college, my friends in Boston knew about it and they started calling me Slug on occasion. When I was writing *Spacewar!*, I was as much Slug as anybody else.

Weaver: Okay. So, given that this is an oral history, we want to start at the beginning and end when we come to the finish, to misquote Alice. What were some aspects of your early life history that you would normally talk about? In other words, where did you grow up? What did you do? That sort of thing.

Russell: I grew up—well, I was born in Hartford, Connecticut. I lived there until I was ready for high school in 1949. My folks moved to Mount Vernon, Washington, where my grandfather had a farm, and I went to Mount Vernon High School. I had actually visited them when I was about three. My uncle had a very tolerant wife who let him fill the living room with model railroads. I'm told I was very impressed. There are some pictures of me lying on the floor looking at the equipment. The next Christmas, my uncle gave me a Lionel train. I liked that, set it up, and over the years grew it larger and larger. I learned about basic electronics mostly from the Lionel train.

In 1949, just before we moved out to Washington, I went to visit my uncle George Washington Pierce, who was a professor at Harvard and had done a great deal of work in ultrasonics and acoustics. He took me around to interesting things on the campus, one of which was his attempt to build a clock, a mechanical clock, that was as good as the electronic clocks that he had built. One of the standard crystal circuits is [designated] the Pierce oscillator, and that was George Washington Pierce. That was one of his inventions. Incidentally, he did quite well with patents, and that was what sent me to Dartmouth.

Weaver: Talk about that a little more in the sense of what was your relationship with him and how influential was he, because didn't he also take you to see the Mark I?

Russell: Yes, and introduced me to Howard Aiken, who was very professorial and happily demonstrated to me the error detection on Mark I. The output device was several IBM [International Business Machines] electric typewriters, which were set up with contacts so that when the key actually got to the paper, it closed the contact and the electronics knew it had actually done the job. And he demonstrated by putting his fingers in front of the type bars, and an alarm went off and an operator sprang out of a chair, and Professor Aiken said, "It's all right." [Laughs.]

Weaver: How old were you then?

Russell: Twelve.

Weaver: That was pretty impressive.

Russell: Yes. I was impressed. The Mark I was a very impressive machine because it was all mechanical…electromechanical. The main power distribution was a propeller shaft about this big around. [Russell demonstrates a diameter of about 6 inches with his hands.] It went down the entire length of the machine, about 100 feet or so. I much later learned that the parts that IBM used for the Mark I were basically the parts they used for their high-performance tabulating machine, the IBM 407, so many of the things were the same. One of the features of the 407, and especially Mark I, was they had *lots* and lots and lots of cams with operating contacts, demonstrates because part of the design method was to have very fast relays that couldn't carry much current and have the current controlled by big cams so that the current only flowed when the relays weren't changing. This led to lots and lots of clocks. An impressive thing is look in the back of the 407 manual and you'll see a page of timing chart for inputs and another page of timing charts for outputs. It was a very strange style of relay design.

Weaver: You talked about when you were younger, getting model trains, so I'm assuming they were the old HO model trains[1].

Russell: Lionel O-Gauge.

Weaver: Yeah, O-Gauge. And that really started your interest in electronics?

Russell: Yeah, I got interested in making signals work and getting—Lionel had plenty of accessories that provided some sort of action, so I had a working a semaphore and a few other things.

Weaver: So that answers the logical question of why join the Model Railroad Club at MIT [Massachusetts Institute of Technology].

Russell: Yeah. Well, my short-form story is my uncle gave me a Lionel train when I was four, and I've never recovered. [Laughs.]

Weaver: Okay. Well, we're going to get back into Model Railroad Club in a minute, but I have a couple of other questions that I think are probably worth going into prior to MIT. You touched on it a little bit, which was "why Dartmouth?"

Russell: Well, I got accepted by Dartmouth and MIT, and Dartmouth sounded better. Later on, when I worked at MIT, I realized that if I had gone to MIT, I would have flunked out in my freshman year. I got four years of good education at Dartmouth. I only would have gotten one at MIT. [Laughs.]

Weaver: Got it. Okay. At Dartmouth, would it be fair to say that you found a mentor in John McCarthy?

Russell: I sort of found him, but we only had a relatively brief contact at Dartmouth because he was only teaching when I was around for about a year. He later on went off to MIT. I was in the theatre as an extracurricular activity and I got a student assistantship at the math department, and so I was the printer for the

[1] HO or H0 is a rail transport modelling scale using a 1:87 scale.

first edition of Kemeny and Kurtz *Introduction to Finite Mathematics* because it was done on the department's offset machine. McCarthy arranged to get Marvin Minsky's SNARC, and I attempted to restore it and get it working again and got a couple of units working. SNARC stands for Stochastic Neural Analog Computer.

So that was sort of my contact [with McCarthy] and the math department. John thought I was pretty good at mechanical stuff for a mathematician. So, between my junior and senior year, McCarthy had me down at MIT as a student assistant. I got to use the very first version of Fortran [Formula Translation] and learned about programming and took a machine-language programming course for the 704.[2] John seemed to be fairly happy with that, and so he offered me a job at the end of my senior year. It turned out I didn't finish my senior thesis, so I didn't graduate, but I got four years of excellent education. John didn't give a damn. He just wanted me to come and program. [Laughs.] So, after my senior year, I went down to Boston and worked for John for several years.

Weaver: Right at MIT?

Russell: Yeah. I was an employee of MIT. I was never a student and I never had to pay them a cent. They paid me.

Weaver: You're one of the few. [Laughs.]

Russell: Not at that time. There were a lot of people who didn't graduate.

Weaver: Well, no, what I meant was, is these days, it's rare that MIT just pays you.

Russell: Yeah. [Laughs.]

Weaver: One way or the other, they find a way to get you to pay them. But was it about this time when you were asked to come down to MIT that McCarthy had you start working to help on LISP [LISt Processor]?[3]

Russell: Yes. In fact, when I got there, I did summer stock, so I showed up there in September. At that point, John sort of had the idea of making something algebraic similar to Fortran, only useful for experimenting with symbolic representation. And over that fall, we went through making machine-language

[2] Fortran is a general-purpose programming language that served as a basis for many subsequent programing languages, including BASIC. ("Fifty Years of BASIC". *Time*. 29 April 2014)
[3] LISP is the second oldest high-level programming language, first being Fortran, invented by John McCarthy at the Massachusetts Institute of Technology (MIT) in 1958.

details, figuring out detailed representation on the 704. John went through figuring out what would make a neat algebraic representation.

Weaver: What about the non-recursive issues that you were dealing with in LISP?

Russell: Well, the computers didn't give a damn. You can write recursive algorithms in Fortran or anything else. You just have to handle the recursion in a sort of laborious, ugly way. And the idea of LISP was, well, you didn't have to do that. You could make the interpreter do that, or the compiler. And so, we went through that.

At one point, John came up with a one-page—well, a half-page universal expression which described how to interpret the language in the language. I looked at that and I—this was, oh, in October or November—I looked at that and understood it and said, "Oh!" I'd been hand-compiling all sorts of things like that for two or three months, so I was a real expert. I said, "Oh, I can do that," and I sat down and hand-compiled the function and debugged it for probably six weeks, maybe. By December, I had a working interpreter. We went charging off from there.

It turned out that one of the first graduate students to try to use it, Jim Slagle, who was blind but very smart started explaining in a seminar how he was going to do formal integration. He revealed a serious flaw in the interpreter. I spent January and February and maybe a little more rewriting the interpreter to remove the problem so it worked the way we people with more or less a mathematics background felt it should. Not everyone agreed with our judgment at the time. Anyway, we spent at least a year chasing most of the bugs out of the interpreter and putting in a few enhancements. I was working on that and various other similar projects when the PDP-1 [Programmed Data Processor One] arrived at MIT.

Weaver: Would you say that because of your involvement in LISP, from a practical standpoint, that really was the early way that you got into AI?

Russell: Yeah, and it was also the way I learned a lot about computer programming.

Weaver: So just briefly, in terms of the lab at the time, because it was a very early period, of course, in artificial intelligence, that meant that before [Marvin] Minsky was *Minsky*, he was an associate professor in the lab?[4]

[4] Marvin Minsky was a prominent early pioneer in computer artificial intelligence (A.I.) and was a co-founder of MIT's A.I. Laboratory.

Russell: Yeah. Well, he had been working on various sorts of artificial intelligence since he was, I think, an undergraduate at Harvard. The SNARC [Stochastic Neural Analog Reinforcement Computer] was a neural network simulator of sorts that tried to learn, so you'd give it something to do and it would do something. You'd see what the output was and you'd either reward it or punish it. And that would cause it to— [in either case] —adjust its parameters a little to make the good things happen more often.

Weaver: Got it.

Russell: But with only six units, it wasn't a very convincing demonstration.

Weaver: What about in terms of the lab itself, what about somebody like Jack Dennis? I'm jumping a tiny bit.

Russell: Okay. Well, Jack Dennis was a different— [he was in] the laboratory down the hall which had a TX-0 [Transistorized Experimental Computer Zero] and some enthusiastic undergraduates. He also got the PDP-1. It was installed in his laboratory and he herded us around it. Really, there were a lot of people who contributed to the story behind *Spacewar!*, and the lack of any of them would have made this story much different.

Weaver: Can you tell me a little bit more about—because you were talking about the lab down the hall and the A.I. [Artificial Intelligence] lab— who were the people involved? In other words, set the stage.

Russell: Well, for management, such as it was, the A.I. lab had Minsky and McCarthy, and the A.I. lab at the time consisted of a room with a bunch of desks. Two of us were full-time employees, me and Clint Mailing, along with some undergraduates and graduate students.

Weaver: Do you remember the graduate students?

Russell: Well, let's see. Bob Braden, and [David] Luckham, and Jim Slagle.

Weaver: And what were the various projects going on in those early days?

Russell: Well, Braden and Luckham were trying to figure out how to make a LISP compiler. My LISP interpreter was very simple, but it was very slow.

Weaver: For the sake of the people who are watching this who are not particularly computer-literate, would you please explain what an interpreter is and what a compiler is?

Russell: Oh, all right. [Laughs.] I guess I have to start with "how do you program a computer?" Well, the computer, the 704, at least, takes binary numbers, and it can interpret them either as instructions or as data and it doesn't give a damn which it is. It just picks up a number by its rules and does something with it. So, very early, people just programmed as raw numbers. Very quickly, long before I got there—well, years before I got there, they built assemblers, which allowed you to use symbols for the operation codes and symbols for the data addresses. It was still your job to keep them straight, but at least you could use symbolic names rather than raw numbers.

In 1957, IBM released the first version of Fortran for the 704, and this allowed you to write things that looked like mathematical formulas and generated code which did something close to what the mathematical formula suggested. The original Fortran had very much rough edges. [Laughs.] Sometimes it would give you a literate diagnostic, well, semi-literate, you know, in terms of Fortran and the machine, but most of the time, or a lot of the time when you screwed up, it would just stop. You'd go to the system programmers, who were responsible for keeping Fortran more or less alive and getting it accessible, and you'd say, "What does this mean? It says it stopped here."

And they'd pick up a book, a listing book that was about this thick [Russell demonstrates an approximate size of 4 to 8 inches with his hands] called the *Stop Book*, and they would look up the location in the *Stop Book* and maybe tell you what Fortran didn't like and maybe not find it or maybe get nothing useful and say, "There's a bug in your program."

And at that point, if it gave you a clue, you'd scowl at the program for quite a while and see if you could figure out something was obviously wrong to fix. If you couldn't, you'd get involved in a very ugly process of dividing the program into two, which made it guaranteed not work, and then pasting on enough stuff so that you had a small program that was half the size of the original program. And then you'd try both halves and maybe get another clue. Doing a binary search through the space of undescribed bugs in Fortran was a pretty ugly operation, which I participated in a couple of times.

When I wrote the LISP interpreter, Fortran was clearly unsuited for writing the interpreter, so it was written in machine language, and we also anticipated that it would be slow, since the 704 was slow. As a rule of thumb, which is still pretty much true, if an interpreter picks up a symbol, or the text of the program picks up a piece of it, figures out what it's supposed to do and then does it, it saves the result and picks up the next piece. The figuring out what to do and doing it, over the years, what you could normally expect is about 60- to 100-to-1. [That means] it takes 60 to 100 instructions, machine instructions, to do one thing that the interpreter has to do. So, compared with the comparable program carefully written in machine language or in assembler, interpreters tend to be 60 to 100 times slower, which is—when you're dealing with a machine that takes five microseconds to do anything—I forget; I don't remember exactly what the cycle time was, but it was microseconds—that's *really* slow. What a compiler does is does the same thing as the interpreter where it picks things up and figures out what to do, but then it generates an assembler or a machine-language code to do the thing as part of a program. [For example], You give the interpreter a pile of code that's complete and does something, and it goes "twiddle, twiddle, twiddle," figuring out what to do and doing it at 1/50th to 200th speed. What the compiler does is take that much time or longer, but it only deals with each piece of the program once and converts it into the equivalent machine language, and then ties that into the rest of the thing. The output of a compiler is a complete machine-language program that does what the input described, and that does whatever it's supposed to do – gets the same results as the interpreter – but it does it 100 times faster. It's really worthwhile to do that, to have a compiler.

Weaver: Would you say that being forced to use equipment in those days that was so underpowered forced you as an early programmer to be as elegant as you could possibly be?

Russell: In some cases, it wasn't elegant, but it encouraged you to figure out things to do, and, in fact, with the 704, in a number of cases, the program was sufficient—you'd write the program and get it sort of working. While you were waiting for it to run again, you'd figure out ways to speed it up.

Weaver: Right. Something that, once learned, is something you remember.

Russell: Yes. Actually [Laughs.], my first few jobs, I was a programmer, that the job title was programmer or programmer analyst. My later jobs, my title was software engineer. I did more engineering when I was a programmer than I did when I was a software engineer because the machines were so slow, and it was important.

Now, with more power than a Cray-1 in your pocket, you can do programming for years before you ever have to wait for your program to finish.[5] [Laughs.]

Weaver: Right. Well, we'll get back to that, the point being, of course, that because you didn't have a Cray in your pocket ...

Russell: Didn't even have a Cray in sight.

Weaver: There wasn't even a Cray. The 704 is downstairs [at MIT] was a museum piece. I'm not sure it wasn't a museum piece when it was out originally. Given its lackluster speed, anybody who worked on those machines.

Russell: Well, but it was the fastest production machine at the time it was introduced, and it was a really good scientific machine because it had floating-point that worked. Now, another—Gene Amdahl was responsible for the 704 floating-point, among other things.

Weaver: Now I want to go back for a minute, because you're at MIT, and tell me a little more about the Tech Model Railroad Club. You've already established that you love trains.

Russell: Yeah.

Weaver: Give me some perspective about the club. Where was it. I'm not saying it's there anymore, but I'm just asking where it was?

Russell: The building has disappeared, finally.

Weaver: Right. What building was it?

Russell: Building 20.

Weaver: Building 20, which had a storied history in and of itself.

Russell: Yes. It was built at the beginning of World War II for the further development of radar and for the MIT Radiation Laboratory, which was a major contributor to radar. By 1958, development of radar had moved elsewhere, and it was sort of the least desirable space at MIT. It accumulated all sorts of things. It still had a machine shop and could do microwave stuff, but it wasn't the main place where

[5] The Cray-1 was a supercomputer built in 1975 by Cray Research.

it happened. The Electronics Club and the Model Railroad Club had all settled in Building 20.

Now, when I first got to MIT, I was kind of tied up in programming projects, which I thought was very interesting. After we started just chasing bugs, I met some of the undergraduates who were at the Research Laboratory for Electronics. Alan Kotok was working on a chess program. I naturally went over to see the Model Railroad Club and became a member and was a participant. In fact, many aspects of model railroading are games. You have this rather complicated machine to facilitate the games, but then when you run your models, that's sort of a game. You can do various sorts of simulations, like you can make a timetable that is plausible for a railroad that's built like the model. You can also arrange to send freight cars to different destinations in a sort of realistic way. That was one of the things that I got interested in there and worked on. We would have operating sessions where we would have a dispatcher and a bunch of engineers running trains. Sometimes we would use a timetable and sometimes we'd attempt to move cars around in a realistic way. Since we were dealing with something that's 1/100th or a 1/200th the size of the smallest railroad, it was sort of realistic, but it definitely wasn't like real railroading.

Weaver: Who were some other members of the club who were both friends of yours and influential, eventually, into what became later on to be *Spacewar!*?

Russell: Well, Pete Samson. Alan Kotok was—actually, he eventually became a computer designer at DEC [Digital Equipment Corporation], and he worked very hard on the PDP-6, PDP-10 and various others. John McNamara, who edited a book on computer engineering with Gordon Bell and various others. Let's see. Pete Samson, Bob Saunders. There are lots of people.

John McNamara and Alan [Kotok] were both interested in telephones and telephone communication. I picked up a bunch of stuff from them that subsequently got more or less useful. In particular, we got into telephone equipment design, because one of the MIT clubs had a rather fancy system to connect the engineers to their trains that didn't require the engineers to do much stuff other than moving the trains and watching the signals, which is more or less what a real engineer does. That was all built with relays from the Western Electric Educational Gift Program.

The Model Railroad Club, long before I got there, had made good friends with the guy who ran the electrical engineering stockroom. Every year, he got a catalog from Western Electric of all the things that Western Electric had that they didn't need that they wanted to donate. He would order things from them. Well, so as soon as the catalog came in, the Model Railroad Club signal experts would go over and look at the catalog and say, "Oh, I could use this," "We could use—," this, this, this, this, and Freddie would order the stuff. It would end up under the Model Railroad Club layout, because the layout was more or less waist-height, and so there was a fair amount of storage space available underneath. They had little carts that you could go around on this cart with casters and not bump your head, so it was fairly accessible. There was lots of stuff there and lots of relays and lots of switches and that sort of stuff. One of the activities of the Model Railroad Club was build new things using the relays. That led to a great deal of study of how you build slightly obsolete telephone exchange out of relays, since that was what was available.

Weaver: Would you say that it was also part of the beginning of the hacking culture?

Russell: Of the which?

Weaver: The hacking culture at MIT?

Russell: Oh, yes. I mean, at that time, "hack" was a verb indicating that you were playing around with something which might or might not work or might just disappear in smoke. [Laughs.]

Weaver: When did you first meet Wayne Wiitanen and Martin Graetz?

Russell: When I got to Boston, my cousin had been living in a co-op called Old Joe Clark's, which was basically a bunch of people. Five or ten people who got together and rented a big old house and then lived in it in more or less cooperative fashion. I moved into Old Joe Clark's and there was a strong bias toward folk singing and folk dancing.

Wayne and "Shag" [Martin Graetz] were associated with the Old Joe Clark people, and so I met them very early on. Several years later, Wayne and Shag and I got an apartment at 8 Hingham Street in Cambridge and lived there. It was fairly exciting, because it was a cheaply built nineteenth century tenement, which had not gotten better with age. At one point, the basement had been actually a residence, but they had given up on that. It was kind of porous for winter winds and also, it turned out, bedbugs. The two memorable features—it had many memorable features—it was remarkable. [Laughs.] The center had sagged about three or four inches more than the outside walls, so there was a definite slope to all the floors. As I say, it was rather porous, and so every now—I think it happened a couple times, anyway—the bedbugs would become obnoxious. The guy who lived downstairs—we had the top two floors—would saturate all of our part of the building with DDT and go off for a weekend, which dealt with the bedbug infestation.

Anyway, it was during that time that the PDP-1 arrived and we started talking about what you can do with it. I had seen Whirlwind and the bouncing ball demonstration on Whirlwind, so I knew you could do animation no problem.[6] I also remembered enough of physics to know that doing animation—calculating the equations of motion for something is very simple if you don't have gravity. What we thought about was, it would be nice to have a better demonstration. We talked a fair amount about what you could do. We concluded that a spaceship trainer, something that trained people who to fly a spaceship, would be a good thing to do. Simple equations. You could make a nice animation on the screen and you could probably teach people something.

Weaver: Was this discussion when you were still at MIT or were you now at Littauer at Harvard?

Russell: I think it started at MIT. I'm not entirely clear on the details or the exact dates.

Weaver: The reason I'm asking is that at one time didn't the three of you, besides Hingham Street, share an office at Littauer?

Russell: I'm not sure we all shared an office at the same time, but we all worked for Littauer at one point or another.

Weaver: When you talk about Hingham Street, is this where the famous Hingham Institute came from?

Russell: Why, yes. As a matter of fact, one of the reasons for calling it the Hingham Institute was we were impressed at Harvard's corporate pomposity and we felt it needed some parody.

Weaver: When you talked about the idea of a space simulator, go backwards for one minute, because you and Shag and Wayne did other things together. My understanding is, for instance, wasn't Wayne getting you into doing things like mountain climbing, so the three of you spent a lot of time—

[6] The Whirlwind was vacuum-tube computer developed by MIT at the request of the US Navy and constructed in 1948.

Russell: Oh, well, that was sort of something that a lot of people at Old Joe Clark's did, too, and, you know, it's nice. We also had fun with winter mountaineering. White Mountains get white, and you can go tramping around on the trails if you have snowshoes.

Weaver: Did the same thing apply to doing things like going to see Toho films?

Russell: We didn't do that together very much, but both Shag and I took advantage of opportunities to see bad science fiction.

Weaver: And what about reading bad science fiction?

Russell: Oh, Shag and I both liked to do that. I don't remember who introduced me to E.E. Smith's stuff, but I thought that was great. E.E. Smith was a science fiction writer, and he had been writing science fiction since the twenties, I guess, and was still—at least through the thirties. The characters were somewhat lacking in depth, but the colors were very bright. The typical thing was—Smith wasn't the only one who did it; this was also common science fiction. But the idea was there was this force of—you'd start out with a force of pure evil and some representatives of the force of pure good. The force of pure evil would almost defeat the source of pure good several times during the story, and finally the sources of pure good would succeed in vanquishing the forces of pure evil. End of book.

Next book, it turns out the force of pure evil was not completely vanquished and they're back! It gradually was revealed that the force of pure evil from the previous book was merely the incompetent minions of the *real* force of pure evil, which was even more evil and even more powerful. The whole thing would repeat, and you'd manage to get through two or three forces of pure evil that were even more powerful before the end of the series.

One of the vehicles that Smith was fond of using was the forces of pure good would be flying across the galaxy or the universe trying to get away from the force of pure evil, and in the process, they'd invent something new. Discover a whole new field of physics and generate a new ultimate weapon which would vanquish the forces of pure evil. You know, it was an exciting ride. That influenced what we wanted to do with the spaceship trainer.

Weaver: When you go to the spaceship trainer, set a context for a minute just before you get into it, because remember that you're in your twenties, you're living through this. What was going on in the world? Why was this so kind of interesting?

Russell: Well, in 1961 when the PDP-1 arrived in the fall. In January the Russians had gotten a man into orbit, and in I think it was April, the U.S. was still playing catch-up and managed to get a man into orbit. This was front-page news every time something happened. One of the reasons to do a trainer was because it was easy, and it would teach people something. We thought that was worthwhile and it would also make a good demonstration. You know, how can you lose?

Weaver: Did this stuff spring from nothing or was this part and parcel of the Hingham Institute Space Warfare Study Group?

Russell: The Hingham Institute Space Warfare Study Group was, in fact—this was the main product of the Hingham Institute Space Warfare Study Group.

Weaver: And who came up with the idea?

Russell: I don't know—I don't remember who came up with the idea of the name, but we all bought into it very quickly.

Weaver: Who came up with the idea of, for instance, an intergalactic space-fighting simulator?

Russell: If you take the simulator out of it, it's Doc Smith. [Laughs.]

Weaver: Were you guys giving rise to Doc Smith in the first—what was going to be a computer demonstration?

Russell: Yeah. We certainly understood the attraction of Doc Smith's stuff. Every once in a while, we'd consider how you would arrange to discover a whole new field of physics during the game, but we never did figure out anything close to how to do that. But we did want to teach, and we very quickly realized that if we added torpedoes and made it a two-person game, it would teach better because it was more interesting. And then once we started playing with it, we started adding things that would still fit.

Well, because of the PDP-1, which was even slower than the 704, and the display, which was very limited by any standard, one of the main things was can you execute all the code you have to execute in time to keep the display from flickering horribly. The primitive feature of the display was you gave it an address on the screen, and you said "Display," or you executed a display instruction. That gave you one spot. Fifty microseconds later, you could do it again. And, unfortunately, programming a display—or fortunately—was that simple. But what it meant was you had to arrange to come around twelve to twenty times a second to do the display.

The first version of *Spacewar!* didn't have any gravity because we didn't understand how to calculate it in time. Dan Edwards, who was also sometimes an undergraduate and also an employee of the AI project, figured out a way to speed up the code that I had written to make it run faster, which gave us enough time to calculate the effect of gravity on the two spaceships. We still didn't have enough time to calculate the effect of gravity on all the torpedoes, so we decided they were photon torpedoes not affected by gravity.

What Dan did was he wrote what apparently is the first example of a just-in-time compiler, which is something that is used for modern display a lot. He looked at the spaceship outline and generated exactly the right machine instructions to draw the spaceship outline and keep the display running at full speed, which my code did not do. That gave us enough time to calculate the effect of gravity. The problem with gravity was the calculation takes a square root. Doing a square root on the PDP-1 was even slower than just doing instructions. It also required some multiplies and divides. The PDP-1 at MIT had only the cheap version of multiply step and divide step, and so you needed to do eighteen of them to get a full word-by-word multiply or dividing a word-by-word. It was much later that my cousin was studying compilers and stuff and decided that was probably the first instance of a just-in-time compiler.

Weaver: Which was intended to speed up the process.

Russell: Yeah.

Weaver: Let's go back to the famed and storied Hingham Institute. So, we have three friends you and Shag and Wayne, and you're talking, as you said, about a concept. The concept ends up being *Spacewar!*.

Russell: Yeah.

Weaver: All right. How did the concept of *Spacewar!* get out of Hingham Street?

Russell: [Laughs.] By bicycle.

Weaver: Okay. Go ahead.

Russell: [Laughs.] I had a car. Did I have a car? No, I didn't have a car then, and so most of the time, I got around by bicycle or MTA, now MTBA [Massachusetts Bay

Transportation Authority]. One of the traditions, which was fairly strong at MIT and certainly very strong at the Model Railroad Club and with Shag and Wayne and myself, was you sort of looked at everything with a kind of engineering eye and say, "Why is it this way? What would be better? Is there anything you can improve anything in this?" And the answer was you could figure out things and you typically would discuss them and critique them. You didn't necessarily have to do anything about it, but just the exercise of "Could this be improved?" was an entertaining game and much participated. Discussing it was certainly easier than doing it, and so we got lots of practice in doing that in all sorts of things.

Weaver: You were discussing it at the Model Railroad Club?

Russell: Yeah, and we were discussing it at the Model Railroad Club at the same time we were discussing it at the Hingham Institute. Wayne and Shag didn't have anything particular to do with the Model Railroad Club. I was the only one there. But, you know, we talked about it and people would speculate on what would be good. In many cases, the various features, I don't remember exactly who suggested them originally, but the things that looked to be easy to do just sort of got adopted into the discussion. Then we went on to look at other things that were easy or hard to do.

When we actually got fairly late in the discussion, but before any code got written, the idea percolated that we really needed to minimize the number of sine and cosine calls we had to do. The sort of "Aha! Now I know how to do it" for me was realizing that I could get by with just calculating an initial heading vector for each spaceship. That's two sets of calls, one for each spaceship. Then by doing shifts and adds and subtracts, I could figure out the positions of all the dots in the spaceship outline, and that was crucial in getting through displaying two spaceships fast enough to not flicker.

Weaver: Before you got to that point, when you were going from the discussion to concept, there were some other things going on in parallel. For instance, you had a relatively new device, the PDP-1. Correct?

Russell: Yes.

Weaver: Who got the PDP-1 at MIT?

Russell: Jack Dennis. Jack Dennis was running the Research Laboratory for Electronics, which had TX-0 in one room. When the PDP-1 arrived, they put the PDP-1 in the next room. It wasn't quite the next room, because there was a very small room between them that had Flexowriters, which were the offline input device for both TX-0 and the PDP-1. Although you had to use different Flexowriters for each one because the coding schemes were different.

Actually, one of the things that I haven't seen demonstrated and I'd like to arrange a demonstration is a blow-by-blow account of how you actually generate a program for the PDP-1 – the mechanics of getting a program into the PDP-1 – but it's hard to find a working Flexowriter.

Weaver: But at the time, when the PDP-1 was there, why did, for instance, Professor Dennis allow you to use the PDP-1?

Russell: The standard way of allocating the PDP-1 and TX-0 was to have a signup sheet and some rules about who could sign up for what. The signup sheets typically were half-hour or hour divisions. If you were important, like a professor, you could sign up for maybe an hour a day. If you were less important, like a graduate student, you could sign up for maybe an hour or three hours a week or something like that.

Accounting for professor usage and graduate student usage was understood, but for hangers-on or undergraduates that didn't have any particular departmental blessing, it wasn't done. Professor Dennis decided or was convinced—I'm not exactly sure which—to let unsponsored projects use unused time. If nobody authorized signed up for the time, if you were an authorized but not supported project, you could sign up for it. And one of the allowed projects was doing something interesting. Peter Samson wanted to play music, somewhat encouraged by Professor Dennis, and so he wrote a music-generating program that would play tunes. That sort of gave us the first piece of the modern personal computer which could play music. And then we generated *Spacewar!*, and so it could play games. But, unlike the modern personal computer, it couldn't do them both at once.

Anyway, the idea that a hanger-on could get time on the machine encouraged hanging around hoping that somebody didn't show and also encouraged late-night computer hacking, a habit which I have never recovered from. I do remember that in the process of getting the release version of *Spacewar!* going, Bob Saunders and I did a lot of testing. It seemed like we never decided that a version was good enough to publish such as it was—that is, put on the console—before midnight. And frequently it was sunrise when we decided we were done.

Weaver: So how long would you say it was before you went from the concept of *Spacewar!* to actually committing yourself to programming *Spacewar!*?

Russell: Well, I thought the concept was neat and interesting, but I was hoping someone else would do the work. [Laughs.] I kind of wanted to do some PDP-1 programming, but I'd just as soon somebody else did the heavy lifting and I'd just play. I had been talking up the idea, especially at the Model Railroad Club, and getting the response of, "Yeah, that's really a good idea. Yeah, yeah, somebody ought to do that."

But then I'd get the next step, which is, "Somebody who really understands the idea really ought to do it. We need somebody who *really* understands the idea ought to do it. Why don't *you* do it?"

I looked for excuses, and one of them was I knew I needed sine and cosine routines, but I didn't know how to write them. I hadn't taken numerical analysis very seriously, and so I didn't understand that. Alan Kotok, who snuck out to Maynard, as it were, and picked up the library copies of the sine and cosine routines from the users' group. He then did a ceremonial presentation—I think it was at the Model Railroad Club, but somewhere where there were lots of bystanders—and said, "Okay, Russell, here's the sine and cosine routine. *Now* what's your excuse?"

I was somewhat embarrassed, and so I went off and started thinking about it. I figured out that I could get by with one call, one call per spaceship and figured out the vector scheme for drawing the outlines. That turned out to be good enough to be fast enough to keep the display going. At that point, I was done, I thought. And then Dan Edwards went off with his own copy of the sources and added the just-in-time compiler and that gave us the time to calculate the effect of gravity.

And then after that, essentially the spring of 1962, we spent a fair amount of time polishing it as a game. We did a lot of testing. In fact, we had to get the lab to introduce a policy—didn't take much effort, actually, but—the policy was that playing *Spacewar!*, absolutely the lowest-priority thing that the computer could do. Making a new version of *Spacewar!* was educational and therefore higher priority than merely playing *Spacewar!* In other words, if they were playing *Spacewar!*, I could kick them off.

Weaver: When you started doing this, how many people starting playing *Spacewar!*?

Russell: I don't know. Some people started demonstrating it and playing it when I got the first version going with no gravity—well, with torpedoes—and when we got the final version going with—it had finite fuel, finite torpedoes, unreliable hyperspace. It turned out hyperspace got added deliberately to improve the learning aspect of the game. What we discovered was, first off, before hyperspace, people would play the game and if one player had a little experience, they could reliably kill off the other player every time. That meant the player who didn't understand, hadn't had any experience, was at a severe disadvantage and didn't learn very fast. We started out by adding hyperspace to allow the player, the beginner who noticed they were surrounded by torpedoes and didn't know what to do, could escape by pushing hyperspace and they would be taken out of where they were and put somewhere random else after a while. Then we discovered that people learned how to push hyperspace and then didn't learn anything more, so we added the finite resources and made hyperspace unreliable so that a player could only use it about seven times or maybe less. It was random. That gave us motivation to learn some more, and that worked quite well. It still does. One of the things I say to people who play *Spacewar!* after they've played a game or two is they now know more about navigating a spaceship than they did before, so it's still a learning tool, and people may not realize they're learning something, but they can't escape it.

Weaver: Would you say that at this point where you had started off with one base principle, a basic principle, other people changed it? By the way, before I forget, you talked about Alan Kotok, but you also mentioned Pete Samson. What did Samson do? Because remember that Edwards added gravity. What did Samson add?

Russell: My first version had a random-number generator to generate stars in the background, and Pete thought that was terribly unrealistic. He coded up a two-page program that took a compressed version of the star chart, of a real star chart, which he also encoded, and displayed it., A real star chart. It actually moved the stars very slowly—took about two and a half hours, I believe, to go through the entire seasons—and added that. He had actually just given me a program called *Expensive Planetarium*, which you could run a standalone and would display the stars. I spliced it into—I think I did; maybe he did—one of us spliced it into *Spacewar!*. It's typical Pete Samson code. It's very compact and very clever. [Laughs.] He frequently says that he knows an awful lot of ways to not quite play music right.

Weaver: Right. Because he wrote that music program with four-part harmony – basically creating oscillators, true digital oscillators, on the PDP-1.

Russell: And dealing with the initial complication that every time a voice has to change from zero to one to make the note, that takes up extra time because changing the note takes an extra instruction.

Weaver: And this is long before the Moog synthesizer.

Russell: Yes.

Weaver: Right. And, of course, the Moog synthesizer was analog, whereas what he was writing was digital.[7]

Russell: Yes. But the PDP-1 synthesizer was really lousy. It could just do a closed organ pipe.

Weaver: True, but still digital.

Russell: Yes.

Weaver: The interesting thing is that it sounds as if you were surrounded by people who were, I don't know, maybe a little obsessive about respective things. For instance, one person saying, "I don't like the fact that you don't have gravity," another person saying, "I don't like the fact that it's not a real star field." Jack Dennis basically getting the bureaucracy out of your way and giving you time on the machine, so you could iterate. What would happen if any one of those people simply had not been there?

Russell: Things would be different.

Weaver: As in?

Russell: I don't know how.

Weaver: Maybe no *Spacewar!*?

Russell: Well, yeah, quite possibly. I don't know if the Hingham Street bedbugs had any contribution, but a lot of other things did.

Weaver: Well, when you were writing it—just for perspective—were you still at MIT or were you, by this time, at Harvard?

[7] The Moog company, founded by Robert Moog, pioneered the commercial manufacture of modular voltage-controlled analog synthesizer systems in the mid-1960s.

Russell: I don't remember exactly. There were a bunch of things that happened in relatively quick succession. I went to Harvard, and because I no longer was working on the AI project, I didn't have a deferment. I enlisted in the Army Reserve and I had to go on six months of active duty sometime in 1961 and 1962, and so I was out of the picture for six months somewhere in there.

Then I got back, discovered that the management at Harvard had changed and learned that I didn't like the new management at all. John McCarthy decided to go to Stanford, and so when he found that I was more or less available, he said, "Why don't you come to Stanford," and I did. That happened in the summer of 1962.

Weaver: When was *Spacewar!*, for all intents and purposes, finished?

Russell: Sometime in the spring of 1962.

Weaver: Spring would have been April of 1962?

Russell: I'm not sure it was that early.

Weaver: Wasn't the MIT open house in May of 1962?

Russell: I think so. It definitely was ready in May. It was done in May, or we declared it finished in May.

Weaver: Right. In other words, just before the open house it magically was finished.

Russell: Well, it was playable before then, so it could be that the version that went into the users' group was a little later than the MIT open house version, but it wasn't much different.

Weaver: And what was DEC's position on this? I know that they had given Kotok the sine and cosine routines for you in terms of the tables, but how did DEC feel about what you were doing?

Russell: Well, they didn't tell me. [Laughs.]

Weaver: Well, but they ended up including it in every new PDP.

Russell: Well, presented with the evidence, they understood it was a good demo. Since it didn't cost them a cent, they used it. You must realize that this was before you could copyright software. It was before you could patent software. All that happened in the 1980s. It was very much a cooperative thing. There was no particular advantage to deciding it was proprietary. You know, so what? DEC was sponsoring a users' group and the users' group maintained the library, so it was relatively little effort on DEC's part. It was just a matter of you sent a description and whatever you wanted, the source or the binary, to the users' group library and they'd copy it for people who requested it.

Weaver: And wasn't it true that virtually every one of the PDP-1s had requests for *Spacewar!*?

Russell: It seems to be the case, but there was another thing that showed up a little later. That was that the checkout people in Maynard and the field service guys in the field agreed that they'd load *Spacewar!* just before a computer shipped. Now, loading *Spacewar!*, the PDP-1 had core memory and the core memory is based on a permanent magnet per bit. When the memory is working correctly, the data stays in memory forever, at least as far as we can tell. We know it stays for at least twenty years. The checkout people would load *Spacewar!*. The field service people would unpack the machine and make sure that the shipment hadn't done anything horrible like knock out modules or stuff. If things looked good, they'd plug it in, turn it on, and start *Spacewar!*. If *Spacewar!* ran, then they'd call the customer over and say, "See, it works," and teach the customer how to play *Spacewar!*, which got them out of there in a few hours rather than a few weeks, which was more the norm for installing a computer at the time. I'm sure a lot of people got it that way, and for quite a while, it was the only, or the best, demonstration program for the PDP-1 for people who weren't computer experts. Most people got a display and most people used it for demonstrating how they were up with the latest in computer stuff.

Weaver: When you went to Stanford, did *Spacewar!* follow you?

Russell: Well, very quickly, we ordered a fancy PDP-1 for a timesharing system and we got a not-so-fancy PDP-1 beforehand so we could start writing code. We certainly had *Spacewar!* running on that, and a lot of people saw it and remembered it. Then we eventually got the PDP-1 timesharing system going, which was a first, which interfered with playing *Spacewar!* We had twelve displays and a timesharing system, so we could run twelve users at the same time, not with super good performance, but twelve at a time for two or three times the price of a single machine wasn't a bad deal. People kept using the timesharing version. It had a display which was *much* faster than the Type 30. Eventually, once the system was running, we got an airplane simulator going and discovered that the pace of a 3D simulator is much, much different from the pace of *Spacewar!*. It's very slow because it's very hard to understand where you are in three-space relative to two-dimensional space right there in front of you.

Weaver: Not the least of which is the ongoing real-time calculation of vector graphics.

Russell: It turns out that wasn't too bad.

Weaver: Really? On the old machines?

Russell: The simulator was for a light plane, but it was in a universe that consisted of streetlights and horizon.

Weaver: Oh, okay. [Laughs.]

Russell: It was definitely dark.

Weaver: One more massive cheat, yes?

Russell: Yes.

Weaver: Okay.

Russell: Well, with a 50-microsecond—well, we were using the high-class display system, so we had a lot faster display.

Weaver: By the time you got to Stanford, this was, what, 1962, 1963?

Russell: Yeah.

Weaver: Did you know at that time what kind of a cult hit *Spacewar!* had started becoming?

Russell: I don't think I realized it was a cult, but I do remember we'd been working on something on the system until 10:00 or 11:00 at night and we decided it was time to go over to the Oasis to have a hamburger. The Oasis was a bar that was in Menlo Park, but was the closest bar to Stanford. We were sitting there talking among ourselves and having our hamburger, and there was a group of students playing the pinball machines, you know, regular old mechanical pinball machines with pins and balls. [The bar] closed and I had to go over and pick up something from Stanford, so I went back there. I looked in on the PDP-1 and there was the same bunch of guys playing *Spacewar!*. I said, "Oh. It's really a pinball machine." That was sort of the first time I recognized it, but I just thought it was an entertaining game like a pinball machine.

Weaver: When did you actually realize the longevity of *Spacewar!* on the DEC machines?

Russell: Oh, I guess I was always sort of bemused at the fact that it lasted so long, and then even more surprised when it turned out that almost everyone who wrote an arcade game had seen *Spacewar!* and remembered it. Okay. I didn't realize it was memorable until then.

Weaver: Well, so would it be fair to say that as far as the DEC Corporation was concerned, *Spacewar!* was their killer app?

Russell: No.

Weaver: No?

Russell: No. It took eight or ten years for computers and displays to get cheap enough to use in an arcade game, and arcade games were never a major seller for DEC computers.

Weaver: Understood.

Russell: Now, there were a few of them, but there weren't very many.

Weaver: Right. But at $120,000 in 1960s money, you weren't going to have a huge amount of the public buying PDP-1s.

Russell: Well, but eight years later, it was only ten or twenty thousand dollars.

Weaver: But the primary buyers initially were universities.

Russell: Yeah. If they didn't buy it—they never admitted that they were buying it for *Spacewar!*, and they probably never were.

Weaver: But with *Spacewar!* largely being, prior to being publicly known, "public" being the general public, wouldn't it be fair to say that *Spacewar!* was known to generations, plural "generations", of college students?

Russell: Yes. And there's lots—I don't know about lots, but there are certainly plenty of examples of people seeing *Spacewar!*, remembering something about it, discovering that they were stranded on a desert island with a different computer which had a display, and writing Spacewar! for the different computer and different display. *Spacewar!* itself is about two thousand lines of code, so if you know what you're doing, that's not a horrible—that's a reasonable spare-time project for a couple of months. If you don't know what you're doing, it still looks like a reasonable spare-time project.

One visitor at the [Computer History] museum was talking about *Spacewar!*, and his contact with it was he was a graduate student in a physics lab and a bunch of people in the lab had written various versions of *Spacewar!* They were all buggy, but differently buggy. His only contact was to take all the versions and gather them together and chase out most of the bugs, so it worked better. Spare-time project.

I think another thing that showed up in the arcade games and a lot of the subsequent development of games, in general, was the hardware kept changing for various reasons. You got a different, faster computer and maybe a different, faster display system or a display system with color. There was a strong motivation to write a game that was at least as good as *Spacewar!* and worked on the available hardware, and so there were lots and lots of variations. A lot of the early hardware had various sprite display systems, and they aren't very good for doing *Spacewar!*.

Also, something I realized much later is *Spacewar!*, as tuned, is really dependent on a relatively high-resolution display. If you have a different display with lower resolution and attempt to do *Spacewar!*, it's almost unplayable unless you do major tinkers to all the parameters. There have been a number of unplayable versions published.

Weaver: Well, it's interesting, because if you think about it on a larger scale, do you feel that *Spacewar!* as an educational entry into programming affected or influenced a large number of people who might not otherwise have gone into programming?

Russell: I don't know how to evaluate that, but it certainly influenced a number of people. I wouldn't say it was necessarily large, although how you determine large—when I started, there were only about a thousand programmers in the world. There are considerably more now.

Weaver: Right. I think that's probably true.

Russell: Well, there are considerably more now on any commercial game or computer-generated movie. Notice that's one of the things I didn't have to worry about: artwork. Actually, I was surprised to realize recently that I feel kind of proprietary about the original *Spacewar!* outlines. They're just thirty or forty dots per each, which isn't a very big claim to fame. As soon as you get color and stuff, then there's a lot more artwork to do, and in spite of everything, doing artwork is not much easier or simpler with computers than just doing it with a pen, pencil, or brush.

Weaver: When you went with McCarthy from MIT to Stanford, was the uptake in the interest in *Spacewar!* equivalent to what it had been at MIT?

Russell: I don't know. At MIT, I had contact with the machine, because I was scrounging time. I sort of knew how much of the time people were playing *Spacewar!*. At Stanford, I didn't have to worry about scrounging time, and I was working on other things, so I wasn't particularly aware of how much people were using it or not.

Weaver: Did you have anything to do later on with the ports, say, for instance, of the PDP-6?

Russell: No.

Weaver: Nobody consulted you?

Russell: No. Why should they?

Weaver: Well, you mentioned that there were no software patents, there was no copyright—you never went into this with an intention to create a commercial enterprise, did you?

Russell: We did think about it for about a week and we realized that the only possible buyer would be DEC, and we knew that DEC was cheap, and since they didn't have to pay for it, they wouldn't.

Weaver: So, it was, as you said, maybe a Hingham Institute concept, but never went anywhere.

Russell: Yeah.

Weaver: Now that you're the stuff of urban legend [Russell Laughs.], what do you think the biggest misconceptions are that have found their way into the public's perception or misperceptions of *Spacewar!*?

Russell: Well, the most prominent one for me is that I was a student at MIT, and I realize that that legend is firmly in place. There's nothing I can do to kill it. I don't think there's anything anyone can do to kill it.

Weaver: Understood. Anything else?

Russell: Well, I guess I've been fairly successful at pointing out that *Spacewar!* was a collaboration that depended on a lot of people. I haven't worked particularly hard at reminding people that there was a lot that went before. A lot of times, people don't understand that the Whirlwind project and the group of engineers on the Whirlwind project are really crucial to starting that whole business, because they worked on Memory Test Computer and Whirlwind and then they worked on the SAGE system, which was bigger than any computer effort that had been done before. They learned from that. One of the things they learned was that working on a government project wasn't necessarily something they liked to do. And then working on TX-2 and TX-0. At DEC, there was plenty of expertise on designing and debugging computers and a strong prejudice toward interaction and a strong prejudice for cheap, or at least no more expensive than absolutely necessary. It's hard to tell the difference sometimes. Cheap made it accessible, more accessible. And interactive, as long as IBM was stubborn about keeping batch processing and not doing anything better, was a great way to sell computers.

Weaver: You touched on two, I think, very important things, so let's go backwards for a second and address them. From the standpoint of the collaboration, this is the opportunity to sort of cement the point that you've been trying to make, which is as best you can remember, who would you consider the collaborators on *Spacewar!*? Who were the people who were the criticals and then the people who were the peripherals and the people with whom you felt added the finishing touches?

Russell: Well, *Spacewar!* certainly wouldn't have been as good without gravity, and that is almost all Dan Edwards' fault. It wouldn't be quite as good without stars and without explosions, and it turned out explosions were a major problem that we felt. I tinkered with the explosions for a while and got thoroughly disgusted and gave up. Shag tinkered with the explosions for a while and got them better. I don't remember who else—I think other people may have touched the explosions. But anyway, getting the explosions to look satisfying was a real problem. There were some very early unsatisfying versions, which I did. Bob Saunders didn't, at the time, contribute any code, I believe, but he and I ended up spending up a lot of time playing each other to get the balance to a point that we liked and to make sure that the residual bugs were chased out. In retrospect, that turns out to have been surprisingly effective, because now I can boast that there are no outstanding user reports of crashes. There are no user complaints outstanding. It's fifty years old, it's still running, and support is still available. [Laughs.] And I think it's going to be hard for anyone else to make that claim.

The no bugs outstanding is an amazing achievement when I look back on it. There are two things that made it possible, I think. One is, except for the places where we needed to be very clever, it's really dumb, straightforward code. Almost everything is in one big loop and gets tested every display cycle. And almost all the rest of the code is tested at least once per game. Everything got tested, and that wasn't necessarily a deliberate plan, but it is, in fact, true that everything got tested very frequently, and so it's hard for bugs to hide, it turns out. Some of it was I had some experience, so—I and the other people all had experience, so we wrote things in a way that we felt was less error-prone. It was very conscious. But the main thing is we did a surprising amount of testing.

Much later, when I was briefly involved in a game startup—this was in the eighties—the rule of thumb in the industry was that you needed something like an hour of testing time for every hour of software engineering time that went into writing the code. I don't know whether it was really a good rule of thumb, but, anyway, that was the rule. I thought about it, and for *Spacewar!* during the development period, we had relatively few hours involved in engineering and we had a lot of MIT undergraduates doing the testing, so I think we got a thousand times more hours of testing than we had hours of engineering and we got very good reporting because the only way they could get things tinkered to be better was to be on me, so I heard about all the problems with testing. The combination of really thorough testing compared with the engineering time, some experience and a relatively good test environment is the reason that there are no outstanding bugs. All the modern stuff is much more complicated, which is why the—

Weaver: Did Alan Kotok do something that was contributory other than or in addition to shaming you by basically bringing you the DEC cosine routine?

Russell: Oh, I'm sure he contributed in the conversations about what to do and that sort of stuff, but, no, he didn't have any direct involvement. I don't believe he wrote any code in *Spacewar!*.

Weaver: And what about Steve Piner?

Russell: I think he may have been involved in some of the explosion polishing, but I'm not sure. His main contribution was the *Expensive Typewriter*. There were a lot of expensive programs.

Weaver: Why don't you explain that?

Russell: I guess the first expensive program was *Expensive Desk Calculator*. What happened was one of the undergraduates was taking a numerical analysis course, was also in the Research Laboratory for Electronics. He was taking a numerical analysis course, and the calculators that they had to use for the numerical analysis course were on the opposite side of campus. So, he wrote a program that he called *Expensive Desk Calculator*, which did what a desk calculator did on the PDP-1. The reason it was expensive was a desk calculator was only $5,000 and a PDP-1 was $120,000. So, there was *Expensive Desk Calculator*.

Then Steve Piner wrote a program that allowed you to edit paper tape on the PDP-1 so you didn't have to use the Flexowriter. This was great because if you weren't changing anything, copying tape was very fast on the PDP-1 and was *very slow* on the Flexowriter. Then Pete Samson wrote *Expensive Planetarium*. There the claim of expensive was not clear, because a real planetarium projector was pretty expensive at the time, so it was up in the PDP-1 range. Anyway, it got called *Expensive Planetarium*. I don't remember what other expensive programs there were, but basically the idea was something that does something relatively prosaic that is done on a cheap machine, and here with a $120,000 PDP-1 you can reproduce the same effect as the cheap machine.

Weaver: And you said that Peter Samson put in the star field, but do you remember whether or not he gave you the star field, if he gave you *EP*, you know, the *Expensive Planetarium*, or did he actually implement some of that code too?

Russell: No, he gave me the full package, that is, *Expensive Planetarium* with the code. All I did was splice in the code. I'm not sure even I did it. He might have done it. But the code for the standalone version of the program just got spliced into the main loop of the—no, he must have done it, because there's some cleverness. [Laughs.] Well, the standalone version you just run, and it has the machine all to itself and does its own timing. In the case of the *Spacewar!* version, each one of the very dim stars get displayed in alternate loops, so the dim stars twinkle because they aren't getting displayed fast enough.

Weaver: Right. Okay, although you also took advantage, at the time, of the screen's persistence.

Russell: Yes.

Weaver: That's not inconsequential from the standpoint of one more little trick that you were able to utilize.

Russell: That was not a conscious trick. We only realized how much that saved us when we tried to do it on displays that didn't have the dual phosphor [screen]. In fact, a number of features of the display we didn't realize mattered because that was all that was available at the time. Much later when we attempted to reproduce it on a different display or watch someone else's effort to use a different display, it became clear that the resolution and appearance of the display were really important to the experience.

Weaver: Looking back—it's easy to try and look backwards, isn't it—what did you think you were doing at the time?

Russell: Having fun.

Weaver: And what did you actually do?

Russell: Have fun. And incidentally, we learned a lot, and some of what we learned was totally unconscious and we didn't realize we'd learned a damn thing.

Weaver: Isn't that the best kind of learning?

Russell: One of the things that I think the industry is not very conscious of is how much we've learned and how subtle it is. One of my last real jobs was a contractor at Intel, and I was dealing with tinkering a big project that had 9,500 source files in it. It turned out, given a herd of Intel machines, I could rebuild the entire project and run the basic test suite in about three hours.

Weaver: Impressive.

Russell: The system was usable, but buggy, not super buggy, but buggy. That's spectacularly larger than two thousand lines of machine language. The fact that it's tolerable and usable is not something I would ever have imagined back in the 1960s. I'm not sure anyone can explain exactly how that happened. I know various pieces of it, like better style of programming and better compiler diagnostics, no more stop books. Some of it is style, like one of the few lessons of academic computer science that got adopted fairly easily was don't use "Go To".[8] A lot of other pieces haven't been adopted. Also object-oriented programming in some disguise, which helps with testability a lot. But no single thing explains why a bug hidden somewhere in 9,500 source files is something that's only a little more work than debugging a new version of *Spacewar!* and produces satisfactory results.

Weaver: Since 1962, other than the cult phenomenon that *Spacewar!* has obviously become, and, of course, you along with it, other than the kudos and being a docent at CHM [Computer History Museum], have you ever gotten a dollar for *Spacewar!*?

Russell: No, I never got a dollar for *Spacewar!*. I got some pretty good steaks from lawyers who were dealing with litigation on computer game patents. Atari versus Magnavox was [one of them.]

Weaver: Right. In other words, as a professional witness?

Russell: Yes.

Weaver: Got it.

Russell: Well, actually, no—yeah, I was a professional witness. I was deposed as a professional witness, but most of the time—I *hate* reading patents! [Laughs.]

Weaver: Well, and, of course, until recently, you couldn't file a software patent anyway, right?

Russell: Well, relatively recently.

Weaver: Right. Looking back now, you've had a number of years, almost fifty years, to look back at sort of what you got into by three twenty-something-year-olds reading Lensman novels and talking about what intergalactic space war might be. What do you think about the industry that largely has been spawned from what you did?

Russell: [Laughs.] I'm very doubtful that I would enjoy participating in generating a modern game. It's an awfully big mob. I still haven't seen anyone deal with a usable version of discovering a whole new field of physics while escaping across

[8] "goto" or Go To is a file command found in many programming languages.

the galaxy. You can shoot at lots of things and you can have some really neat monsters to attempt to slay and that sort of stuff, but no invention.

Weaver: That's interesting.

Russell: There's actually an approach to that in *Clash of Clans*, which is one of the games that I enjoy playing now, and that is they have a laboratory [in the game]. If you spend money in the laboratory, they will discover ways to make your characters better. But it's still not inventing a whole new field of physics.

Weaver: Well, don't you have Doc to thank for that?

Russell: Yeah. Well, I have lots of people to thank for it, you know.

Weaver: What games are you playing now?

Russell: *Clash of Clans*, with a peculiar restraint because *Clash of Clans* works on a cell phone, and you can buy various things to make things better, but there's sort of a basic level that doesn't cost you anything. For some stupid reason or other, on the cell phone games, I've always taken the attitude that "I ain't gonna pay for it if it's free." I put up with the slow pace and every now and then think about spending ninety-nine cents for an enhancement, but, no, that's my rule and I'm sticking with it. And *Klondike*.

Weaver: And what does *Klondike* have that's so attractive?

Russell: Exactly the same stuff that made it attractive as a card game. It requires a little thought and a little memory work, and the pace is pretty fast.

Weaver: Is there anything you would have changed if you had the benefit of hindsight?

Russell: I haven't thought of that. I haven't really studied that. I would have liked to have figured out a way to invent new miraculous fields of physics while fleeing across the galaxy, but I don't think I will.

Weaver: For those people who are interested in getting into programming but it's a concept to them, and knowing what you know now about the relative simplicity of high-level languages compared to working at the machine level, what kind of advice would you give to someone who's interested starting out?

Russell: Do stuff. There are a number of things to do. One is there are an awful lot of bad interfaces available. A few years ago, I was investigating, or actually my cousin was investigating bicycle lights. Some of them are very clever, but some of them have truly lousy interfaces—well, a lot of them have truly lousy interfaces that are complicated to understand and complicated to operate. One depended on the delay, how long you held the button down, but it was done in terms of seconds and not in terms of human perception. Doing the difference between a ten-second delay and a twelve-second delay, nobody's going to do that, or at least not very successfully. They're going to be very peeved.

Another thing, there's one thing missing on my toaster oven. You can set different shades of toast, which is basically just time, but it won't tell you how long until your toast is done. It just, "Don't worry. You'll get it when I—I'll tell you when I'm done," but it doesn't give you a clue as to whether it's going to be in a minute or five. Also, when you discover that it's not done enough and you'd like it a little more, it doesn't have anything to say, "Give me another 10 percent." You have to figure out what 10 percent or 20 percent is and dial it in again. Now, given the computer they have in there, they should be able to do that with very little extra code, but they didn't.

Weaver: So, it's a user interface problem.

Russell: Yeah. Writing a user interface for either of those situations is a simple programming exercise, but you'll learn something.

Weaver: Do you think that *Spacewar!* was arguably one of the, if not the first—and that would be highly arguable—let's say it was one of the earliest human-machine interactions that was approachable or touchable by a relatively large number of people?

Russell: I think you can try as an exercise to build a definition, but it's not the first computer game, it's not the first computer game that was displayed on a display, it's not the first two-person computer game. Attempting to make it a first requires an awful lot of fine print, and there are much better things to do.

Weaver: Well, I guess my point is part of Douglas' thesis at Oxford only ran on two computers in the world at the time, and that was part of a Ph.D. thesis and it was nothing like *Spacewar!*. And Higginbotham's, of course, was done for Brookhaven National Laboratory. That's kind of where I'm coming from, which is yours wasn't military. Yours was not strictly a theoretical part of a Ph.D. thesis. Yours was a game. It was a game that was approachable by people on an admittedly expensive machine, but it was playable by a large number of people.

From that standpoint, wasn't yours the most approachable video game on a computer of the time?

Russell: Well, I have a problem with "video game," but that—

Weaver: Okay. You use your language. You know where I'm going.

Russell: Yeah. What you're saying is true, but I think the thing to do is say it was a very inspiring early game. We know now that a lot of people saw it. We know now that they remembered it. We know now that some of them tried to do something better. The other thing is I can't defend it because there are no facts around one way or the other, as far as I know, but I think it was the first computer game where people wanted to come back and play it again, and that's certainly true of *Spacewar!*.

Weaver: Given that you sit in a very unique position, how would you describe *Spacewar!*?

Russell: Very influential early computer game.

Weaver: Fair point.

Russell: Or the grandfather or maybe great-grandfather of the modern computer game.

Weaver: And maybe the modern computer industry.

Russell: No.

Weaver: No? Computer *game* industry.

Russell: The computer game industry. Of course, you could argue as to whether the computer game industry isn't the main part of the computer industry now.

Weaver: True. But the computer industry, as evidenced by the 704, is not something that influenced the computer game—

Russell: No.

Weaver: —whereas I think *Spacewar!* would be.

Russell: Yeah.

Weaver: With that in mind, you have a phrase of how you'd describe it relative to the industry?

Russell: I'll stick with "very influential."

Weaver: Okay. All right. Steve, as best you can recollect, when did you know you really had something?

Russell: [Laughs.] I was satisfied with the original version with no gravity in the sense that I had gotten it—I had done what we had talked about. It was all working and it wasn't too buggy. That was an achievement. That was a milestone. That was okay. Then when the other contributions happened, and Bob Saunders and I spent some time testing thoroughly, that was another milestone, and that [iteration] was obviously better. That's what we're exhibiting now [at the Computer History Museum].

Weaver: Okay. Was it calm?

Russell: I wasn't expecting to be inspiring to other people, particularly. It was just that was a project that reached a stopping point.

Weaver: Was there yelling and screaming involved?

Russell: No.

Weaver: No? Even when one guy downed another guy and he went down in flames, there was no fist- pumping?

Russell: Oh, yeah, there was that but not relative to the software. That was just part of the game. Oh, yeah. That was one of the reasons that the game is fun, is because it's two-person and there's plenty of chatter between the players. One of the things that we show in the software exhibit here is—especially for *World of Warcraft* and similar things—having a back channel or side channel to talk to your friends and enemies is very important. It makes it social and interesting, and that, of course, is something that board games do.

Weaver: I know that tomorrow we're going to do an actual demo [of *Spacewar!*], and I know that you have sort of one of the standard spiels of the way that you explain it to the audience downstairs, but because we're in a protected environment here and we have everything set up, I want to ask something that you might normally address downstairs, but here. Would you explain a PDP-1? Explain a little bit of the history of the PDP-1 and what the PDP-1 is and why it was so important. In other words, what made it the right tool for you to do *Spacewar!* on?

Russell: Well, the first and foremost thing was it was handy and available. The second thing was it was good enough. In some ways, its restrictions made it possible to get *Spacewar!* done, because with a modern computer, simply bringing up the display is more work than all of *Spacewar!*. So, since it was slow, there weren't too many bells and whistles that could be put on, and you really had to work to get the bells and whistles on. And since the display was limited, you had to have everybody in the same room. Those things all contributed to making *Spacewar!* relatively simple and relatively easy to do.

Weaver: Well, you bring up an interesting point with what you just said, because when you talk about the testing, the innovations, the iterations, etc., you brought up the point about you had two people in the same room. Obviously, you had the chatter between the two of you. But when you first had *Spacewar!* come out, you didn't have your controllers, did you?

Russell: Well, they were always—the first version ran for the console switches.[9]

Weaver: Exactly.

Russell: So, yeah, but you still had to be in the same room, and, in fact, you had to be right next to each other.

Weaver: Well, but the person who was closer to the display had a slight advantage visually.

Russell: I don't think that ever turned out—

Weaver: Somebody's on the left, somebody's on the right.

Russell: Yeah, I don't think that ever turned out to be recognized as significant.

Weaver: So why did you build—not that you built the controller. And, by the way, who built the controller, from what parts and where, and why was it built?

Russell: Oh, well, as previously mentioned, there was a big stash of telephone equipment underneath the Model Railroad Club layout, and so a bunch of switches were tested by hand to see if they felt suitable, and then the control box was built. I don't know for sure who built the controllers. There were actually a couple of iterations. One which I remember and no one else does was a little four-button box that Western Electric built for buzzers for telephones, for office telephones

[9] In a follow up query, Steve Russell describes the original control interface as "the 18 'test word' switches on the front panel." (Email dated Aug 22, 2019)

where the receptionist has a buzzer to tell who to pick up. That one failed fairly quickly because it was designed to be pushed by a receptionist a few times an hour, whereas playing *Spacewar!*, they fatigued and fell apart and became useless pretty quickly. That was the point at which the first control box got built. Nobody remembers the push-buttons, so I suspect they failed fairly early—or no one else remembers them. It was some combination of people from the Model Railroad Club, but I don't remember who.

Weaver: And the reason that it was built, as I remember, was because it was not as playable or not as enjoyable from the console because you didn't have the specialized controls. The mother of invention here was giving you the little joysticks you were talking about and a button.

Russell: Actually, the main motivation was physical. To play from the console—the buttons were out here—you pretty much had to rest your elbows on the table. After half an hour, that got pretty painful. The other problem was that you also ended up spending a half an hour or more looking to the side because you were facing the console, but the display was over there. You had to turn your head and keep it turned for a long time, and that was painful too. Something that allowed you to sit directly in front of the display and sit was much less painful than playing from the console. That was part of the motivation for having the special controllers.

Another feature with possibly some unconscious foresight from DEC was that the basic computer came with enough spare sections and a custom wiring panel for input/output gear. By merely adding wires, you could add the eight buttons as input, any eight buttons you wanted. You provide them. So that made it very easy, you know, a sort of two-day project, maybe, to add the buttons. Essentially every installation ended up doing that for some reason. I think they frequently justified it as interactive input.

Weaver: Well, even today, especially for two-player games in front of a screen, it's not all that different, is it?

Russell: Yeah.

Weaver: Would you explain—I mean, we've been talking about it a lot, but we really have not started from ground zero. What is *Spacewar!* and how does it work?

Russell: Well, I'm sorry, but I've got to give you the straight-man answer. It works pretty well. [Laughs.]

Weaver: Okay. Now the docent's answer.

Russell: Well, the structure of the program is two loops.

Weaver: No, I'm talking about for the general public; in other words, what you would do downstairs to explain—

Russell: Okay. You have this program. If it was on your cell phone, we now call it an app. You load it into the memory of the computer and start it. It displays two spaceships in sort of standard starting positions, stars in the background. They happen to be real stars, a real star map, but you don't have to worry about that. There's a star in the center, and the star in the center has gravity and gradually pulls the spaceships in toward the center. The goal is to maneuver the spaceship so it doesn't fall into the star and use the torpedoes—you can maneuver the spaceship by turning it and firing a rocket, and the rocket accelerates the spaceship in the direction it's pointed. That's the way most rockets currently work. You then can turn, and there's a torpedo tube in the very front of your spaceship, so you can fire a torpedo out the nose of the spaceship. The torpedoes will blow up other torpedoes. They'll blow up spaceships. You can even blow yourself up if you work at it hard, but it takes some work.

The goal is to destroy your opponent before he destroys you. You can run out of torpedoes, you can run out of fuel, you can use up your hyperspace jumps. The hyperspace generators were rushed into the field and they're not very reliable. No ship has been known to survive more than seven jumps into hyperspace. They just blow up and that's that. When the game reaches a conclusion, like nobody has any torpedoes left or only one person is left standing, then it restarts. As we normally set it up, after there's a conclusive winner, the game stops, displays the scores of both players. After a while, it waits for somebody to push a button to restart. That's what the user sees. You can play it that way for hours and hours.

As to what's going on, what you load into the computer is a small, by today's standards, program that loops through all of the colliding objects—that's all the torpedoes and all the spaceships—and it looks to see whether any of them are close enough to each other to explode. If they are, it causes both of them to turn into explosions and that's that. At the end of the loop, it looks to see whether anything else can happen, and if it can't, then it goes and restarts, and keeps score. Inside that loop, for each colliding object, it goes through a display routine which looks to see if there's any change in the status and updates the status and displays the object. Now, in the case of a torpedo, it's just a dot, so that takes no effort. In the case of a spaceship, it has to look at the controls for that spaceship and adjust its position if it's turning or accelerating. Then it takes its current velocity and updates its position. Displaying the spaceships is complicated, and that is one of the main time consumers. Then it goes on and calculates the next spaceship.

It turns out that the program is an example of what we now call an object orientation, because there's a common code for each spaceship, but there's separate data for each spaceship. The position and the acceleration and which buttons to use and the outline is peculiar to the individual spaceship, but it's the same code manipulating all these things for both spaceships. There's only one copy of that code. Similarly, for the torpedoes, there's a subset of that code that gets used only by torpedoes. The result is that the colliding objects—and my comment in the original code calls them colliding objects—are polymorphic objects. There's the display moving object, same code for all torpedoes and spaceships, same code for all spaceships, but different outlines for the spaceships. So, I was using a very advanced method of programming which seemed obvious to me and wasn't invented until ten years later.

Weaver: [Laughs.] Okay.

Robertson: How do you actually load the program physically into the machine?

Russell: Oh, it's on a piece of paper tape. Built into the machine is what's called read-in mode. Read-in mode reads characters from the paper tape and interprets them as variously alternating instructions and data. As long as the instructions are deposited into the I/O register, it keeps reading paper tape. When the instruction turns out to be a jump, it gets out of read-in mode and transfers to that location. That's the basic hardware. This was a convention that was established long before the PDP-1. You write a small loader which goes on the front of the paper tape or the first few cards of the card deck. That program typically checksums itself to make sure it got loaded right, and then starts reading in blocks, which give a loading address, a pile of data, and a checksum at the end. There's also one that says, "Start the program here." By the time you finish reading in the paper tape, you know you have a good copy of the program in memory and it's ready to go. It looks neat because the paper tape reader is fanfold, and so it's flapping.

Weaver: Give us a few stats. For instance, how much money was a 704? This is the 1950s.

Russell: I don't remember. Millions.

Weaver: But it was far in excess of millions.

Russell: Yeah.

Weaver: Yeah. In 1960s money, how much was a PDP-1?

Russell: A hundred and twenty thousand dollars with a display.

Weaver: So, it was cheap.

Russell: Yes. It's DEC. [Laughs.]

Weaver: Of course. It's DEC. Why did they call it a PDP and not a computer?

Russell: Ah! When Harlan Anderson and Ken Olsen were shopping around the idea of building transistor modules similar to what they had done with TX-2, they went to the only venture capitalist around in Boston at the time, General Doriot. Suggested that they start this company and call it the Digital Computer Corporation, because they knew they wanted to do that eventually. And they said, "No, do not mention computers. Everybody in finance knows that nobody except IBM has ever made any money with computers, so don't mention computers. It will scare off investors and, incidentally, give IBM a target." So, okay, Digital Equipment Corporation.

Things go along and the module business turns out to be very good for them. They're high-margin and they're attractive and they sell nicely. They have a cookbook which allows you to build high-speed logic without having to do any circuit design and without having to know much about electronics. Real good business. They start building some core testing machines for some of their customers. This is a few extra pieces of electronics and you can plug in a new stack of core memory and test it. You have confidence that it's all working, and it doesn't take very long.

It's kind of a mystery exactly how Digital decided it was time to build a computer, but they decided it's time. They set an engineer to designing a computer. Well, what does a computer look like? It looks like what they're used to. It's a cost-reduced version of Whirlwind. It's much, much cheaper because it's using transistors and it's eight or ten years later. At any rate, so they build the computer. Well, what are they going to call it? Well, they're not going to call it a computer because that's the kiss of death, sort of, so they decide to call it a Programmed Data Processor. Now, this is good enough to fool purchasing. "Okay, we don't have to refer it to the Computer Selection Committee. It's an expensive piece of laboratory equipment, and they think they need it." So, they call it a Programmed Data Processor.

Now, anyone who knows anything about computers looks at it and says, "Oh, that's a cheap computer." And some people say, "I can afford one of those, but I can't afford a big IBM system," or, "I want a special instruction and a special bubble chamber picture reader," or something to attach to it. That's appalling to do with an IBM system. Digital is quite happy to design a special instruction for me and quite happy to let me hook whatever I damn please onto it. Furthermore, it's easy.

That's what they introduce in December of 1959. A small number of people, but enough, look at it and say, "Hey, yeah, that is a good deal. I could use that."

They get a satisfying number of orders and they decide to donate serial number 5 to MIT. Now, it turns out this is a good deal for them. Any interesting programs that MIT writes, because they're exposing it to MIT grad students and undergraduates. They get MIT turning out a crop of graduates who already know about PDP-1s and would probably buy more—and they did—and they can write it off their income tax.

Now, forty years later, we discover, or we were told, that it turned out they wrote off the list price. The list price was $120,000 off their profit and they paid $50,000 less on income tax and it cost them $40,000 to build. They were still ahead $10,000. Not like they had sold it at full list, but still not a bad deal. And the IRS sort of figured this out eventually and changed the rules the next year, so that was the last time they got to do that.

[So, what kind of computer is the PDP-1 comparable to?] It looks like Whirlwind. There's an improvement. People have figured out indirect addressing and talked about it since then, so they add indirect addressing. They notice that they can save a register something like six flip-flops by not having a shift counter and changing the closed subroutine calls. They give you multiply step and divide step, and you have to do them eighteen times to get a full multiply or divide done. And instead of having a register that holds the place you came from after every transfer of control, they have close subroutine calls that simply store the old program counter in the accumulator. The rest of it is basically like Whirlwind. You can look through and compare the order codes and see that Whirlwind was a little more numerical oriented, because they knew they were doing coordinate transformations. The PDP-1 is a little cheaper because it has fewer registers.

Weaver: As best you can recollect, what was the purpose of DEC building the PDP-1? What was its intended utilization?

Russell: I'm not qualified to say. Try Ken Olsen's oral history and Harlan Anderson's book.

Weaver: In your experience, we already know that PDP-1s early on were basically primarily university.

Russell: A lot of people doing research stuff and a lot of the early research was on graphics. The conclusion of the early PDP-1 research on graphics was we needed a faster computer, a faster display, and bigger memory, and further study is indicated.

Weaver: Very good. Steve, did you ever design a board game?

Russell: Not singlehandedly. Shag and Wayne and I periodically worked on a game that we called Graft, which was based on Monopoly. Instead of properties, you had figures in the news, like being the mayor's cousin who got a lot of graft. It was intended as a game, but part of the entertainment was that the stories that came out of it were very much like what was getting reporting in Boston politics, which at the time had some graft exposed.

Weaver: In other words, colorful characters.

Russell: Colorful characters doing things that weren't necessarily legal.

Weaver: I don't think anything's changed, even till today.

Russell: [Laughs.] Well, there's always the strange climate in Massachusetts, which you notice when you cross into New Hampshire. The Massachusetts climate is very hard on the roads and the roads tend to fall apart a lot, but as soon as you get into New Hampshire, the different climate makes the roads much smoother.

Weaver: Very interesting. [Laughs.] Are you familiar with a game called D&D, Dungeons & Dragons?

Russell: Yes. I played it for several years when I was working for Digital Equipment, and it was a great deal of fun, because two of the people in the group were would-be authors. They were very good at dreaming up things to do. Some of the other players were colorful characters, too, so it was fun. There it's sort of a merge between gameplay and table talk. A lot of the stuff you talk about is related to playing the game right now, but there's a certain amount of side remarks about the individuals rather than the characters.

Weaver: When Wayne left for military service, did you and Shag end up as roommates?

Russell: Yeah. Well, the three of us had rented 8 Hingham Street, and when Wayne went off for military service, it was only for six months. He was in the Army Reserve and they had six months of active duty, so he was going to come back fairly soon.

Weaver: Okay. But you stayed in close proximity?

Russell: Oh, yeah.

Weaver: Yeah. Okay. You spoke about the idea of altering physics in a game, so would you like to talk a little bit about the Bergenholm space drive?

Russell: No, that was Doc Smith's department. Other than the idea of hyperspace, which wasn't peculiar to Doc Smith—it was pretty common in science fiction at the time—I don't claim that it's anything very specific. It's mostly the name and its use in the game, which is not in the classical tales, exactly.

Weaver: I thought that when you were talking about it before in terms of altering physics, I instantly thought of the Bergenholm drive, of course, which did a good job of that.

Russell: There was a long tradition of altering physics to make the plot work, including the transporter, you know.

Weaver: I get it. All right. Let's go back to MIT for a minute. I want to just put a few things—get a few answers on the record. Were you involved in TMRC hacking on the TX-0?

Russell: No. Well, I didn't do anything. I would periodically be an admiring audience.

Weaver: Got it. Okay.

Russell: The TX-0 was really very influential because they inherited from the Lincoln Labs people a bunch of applications. A bunch of sort of seeds of applications, like the type justifier, which would do word breaks, that sort of stuff, and the text editing and the debugging tools. All those, again, most of those had showed up in some form at Lincoln Labs, but needed some work to translate to the MIT installation, which had less memory and very quickly grew extra instructions.

Weaver: Understood. When you were at Harvard, were you still part of the MIT hacking scene?

Russell: Oh, sure.

Weaver: And when you were employed by Harvard, were you programming *Spacewar!* at MIT?

Russell: I was working on it. *Spacewar!* was always a spare-time project.

Weaver: Right. No, no. What I was asking is was it a project both when you were at MIT and also when you went to Littauer? [That you] just kept working on it?

Russell: At MIT, management sort of knew what I was doing; at Harvard, they didn't.

Weaver: Got it. Okay. Did you have any formal process for programming *Spacewar!*?

Russell: Well, sort of. One of the things that I did for LISP, I had an algebraic specification, which was John's universal m-expression. In the case of *Spacewar!*, I wrote the physics part in imitation—before I coded it, I wrote it in imitation Fortran. It was a very loose version of Fortran that suited me, and it allowed me to look at the formulas and do some simplification that was based on the formulas rather than having to write all the code and then figuring out some of it was redundant.

Weaver: Trying to put it within the construct of what followed in terms of more methodological rigorous approaches as computer science sort of developed, would you say that it was a waterfall-type development? Was it more of just an iterative development system? It certainly wasn't agile, right?

Russell: I don't know if I could fit it easily into any of the formal systems. I had brief exposure to a relatively rigorous machine language development system. When I was working for the Artificial Intelligence project, I took a spare-time job working for Dick Bennett , who had contracted to do some things with the 709.What I ended up with was a program that managed to keep the tape drives running full-time on the 709, which was a bit of an achievement. That was all done with machine language macros, and he had a very structured way of writing things, which I put up with but didn't really like. And I've noticed that a lot of people have methodologies which will solve all problems, and I've never met one that seemed to work for all the problems I knew about.

Weaver: Okay. I have a list of people who were mentioned regarding some contribution to *Spacewar!*, and I'd like to read the names that I have, and if you don't mind, just give me a quick piece or portion of a sentence of what that person contributed so that I have it for the record. And if I'm missing anybody, please feel free to add anybody I'm missing. The first person on my list is yours truly, Steve Russell.

Russell: Okay. Yes, indeed, I worked on *Spacewar!*.

Weaver: From a high-level standpoint, if you were going to try and describe what your contribution was, how would you describe yourself in terms of the product?

Russell: Project manager. I wrote some of the initial code and I kept what I considered to be a good copy, and as other people got things, I added those things, and so I had a version that I thought was better than any of the individual versions.

Weaver: So, were you also the keeper of the gate? When people wanted to add things, it was something where they either gave it to you or it went through you so there was some sort of a methodologic—

Russell: No. I gave the source files for the current version to anyone who asked, so they didn't have to return anything to me.

Weaver: What about the group that was immediately around you? In other words, you mentioned that—and I'm going to get to him, but you mentioned that Peter Samson, for instance, you believed, had put in his own magnitude-reduced star field. Was that something that you had both discussed or was it just something that he did willy-nilly?

Russell: I don't know as we discussed it before the fact, but he certainly did the tinkering to make it work. He had to make it different for being in—as part of *Spacewar!* rather than a standalone program.

Weaver: Right. But he did that even though you were program manager. You didn't necessarily oversee it?

Russell: Oh, no.

Weaver: Fine. Okay.

Russell: As project manager, about all I did was talk about the project and try to keep track of what interesting things were getting worked on.

Weaver: Got it. What did Wayne Wiitanen do?

Russell: He didn't contribute any actual code, but he was part of the original Hingham Institute discussions. There was a great deal of discussion before I started working on the prototype, and so we had a pretty clear idea of the basics. Then I was shamed into writing the prototype, and at that point, there were all sorts of suggestions pouring in from everywhere. But Wayne and Shag and I were definitely in serious discussion at great length.

Weaver: Got it. All right. So, speaking of Shag, so what did Martin "Shag" Graetz do on the project?

Russell: Well, my remembrance is that he was relatively sort of the advocate of the teaching part. We all bought into the idea fairly quickly, but he was the advocate of the teaching part. He made the explosions look much better. Getting the explosions to look right was a sort of continuing effort. A compromise between appearance and keeping the display and that sort of stuff. He was also one of the chief promoters after it was working. He wrote the article for *Decuscope* and gave a talk at the DEC Users' Meeting.

Weaver: What did Dan Edwards do?

Russell: Gravity—well, not gravity. Made the time for the gravity calculation. He may also have done most of the gravity calculation. I know I had my finger in it a little, but the main thing was the run-time compiler trick, which, as far as we know, that was the first time it was done. You must understand that a lot of these things which have turned out to be serious concepts of computer science at the time were viewed more as despicable programming tricks. Despicable in the sense that they were hard to understand, and they were outside of the mainstream programming practice. They were tricks and they managed to make the program work. In the culture, getting the program to work was a good thing, even though you didn't do it entirely in the generally accepted practice or make it completely transparent.

Weaver: And what about Peter Samson?

Russell: Well, he built the standalone star chart, *Expensive Planetarium*, and he made it work. I think he later on, after I left, did the scorekeeping, some of the scorekeeping, anyway, and that's a little obscure.

Weaver: Bob Saunders?

Russell: He may have tinkered some of the stuff, but I don't think he did much. He helped me test the final version. We spent many hours playing each other and considering whether something needed to be tinkered. There was sort of a history of tinkering. What happened was as soon as the prototype was going, there were lots of suggestions about, "Make it faster," "Make it slower," "Change the rockets," "Give them more torpedoes," things like that. When we were making the final version, we had these various tuning things, and Bob and I ended up playing each other a lot and doing the tuning to our satisfaction. Because there were all these suggestions about tinkering the parameters, as part of the final version, I gathered together all the parameters and put them in the first page of the program. Anyone with the first page of the program could figure out how to adjust the parameters to their satisfaction. Apparently hardly anyone did. [Laughs.]

Weaver: Today if you probably have been following the industry, we call that play balancing.

Russell: Yes.

Weaver: That's what you were doing.

Russell: We were conscious that we were balancing things, but we were balancing it to our satisfaction. The play testing was just us two.

Weaver: Did Bob Saunders also have a hand—no pun intended—in constructing the controllers, the separate controllers?

Russell: He may have. I don't remember. He subsequently wrote a version for, I think, PDP-9. I think, either the 7 or the 9.

Weaver: And what about Alan Kotok, [pronounce] "Kah-tok"?

Russell: [pronounce] "Koh-tok." He was certainly one of the kibitzers. He was also responsible for provoking me into actually writing the prototype by presenting me publicly with the sine and cosine routines and saying, "Now what's your excuse?" I was embarrassed, a little, and wrote the prototype. But he was certainly one of the serious commentators and observers.

Weaver: Do you remember if Alan and Bob actually were the two who scrounged parts at TMRC to build those controllers?

Russell: Well, where else would you scrounge parts? [Laughter.]

Weaver: Okay.

Russell: Remember TMRC had accumulated obsolete telephone components for years and years because a lot of the projects of the railroad used those components, and so there was an ample supply of those things hiding under the layout.

Weaver: And what about Steve Piner?

Russell: I don't remember that he had his thumb in the code particularly, but he wrote *Expensive Typewriter*, which was one of the tools that we used and cursed at. It was, it turns out, one of many editors which dealt with character codes with explicit case shifts. It turned out there was a really rich bug mine there because when you backspaced, it was supposed to do the right thing. But when you backspaced over a case shift, it was hard to do the right thing in all cases because you had to know what case the device was in. You had to know what case the device had been before the character that got erased by the backspace. You had to get everything right. It turned out that very few people chased out all of the bugs associated with that problem. Certainly, *Expensive Typewriter* was infamous for having that problem.

Weaver: Are there any others that you remember I've not mentioned?

Russell: Not offhand, but remember this was an open-source project. Furthermore, it was tinkered with after I left for Stanford, so there's lots of things that people did to it that I wasn't aware of.

Weaver: So that would account for things like the additional features and also some subtracted features as well?

Russell: Yes, and also different tuning.

Weaver: How was *Spacewar!* development impacted by your own call-up to active duty?

Russell: Well, I didn't do it for six months. I think that Dan Edwards did most of the display improvements while I was on active duty, but I don't remember for sure.

Weaver: If scholars of the future wanted to uncover the various influences that went into creating *Spacewar!*, what would be the key areas or elements that they should research?

Russell: Well, they haven't asked me any questions yet. [Laughter.] Well, certainly the Lensman series.

Weaver: The political times? The Cold War?

Russell: Well, there was certainly the news because of the space race, and that provided "Here's this computer. Can we do anything with it?"

"Well, the *Minskytron* is kind of boring after a while."[10]

Then it was, "Oh, look. Here's spaceships. Nobody knows how to fly—most people don't realize about momentum and how you have to control a spaceship. Well, that's simple. The physics is very straightforward. Can we make that work?" And the answer turned out to be yes.

Weaver: Would it be fair to say that from the times, you were less interested in the Cold War part and more interested in the space part?

Russell: Yes.

Weaver: And then other than the Lensman series, were you reading things like Disney's *Man in Space*?

Russell: Yeah. And I believe I was getting science fiction magazines every month, so I had plenty of experience—I certainly had no problem understanding what hyperspace meant.

Weaver: Let's go to Stanford for just a moment. Was play of *Spacewar!* restricted at SAIL to off hours?[11]

Russell: Well, SAIL didn't exist, as such, when we were working on the PDP-1 timesharing system, so it was the AI project of the time. With the first machine, which didn't have a working timesharing system, it was essentially the same rules as MIT; that is, playing *Spacewar!*'s absolutely the lowest priority and debugging *Spacewar!* is one step above that. If anybody had anything serious to do—and I had a lot of serious stuff to do with the timesharing system—that was more important. Once we got the timesharing hardware and a timesharing system going, then if it was outside normal hours and you could manage to chase

[10] The Minskytron was a computer graphics only display demo for the PDP-1.
[11] Stanford Artificial Intelligence Laboratory

everyone else off the system and you knew how to load *Spacewar!*, then you could do it, but that made it much more restrictive.

Weaver: What was *Spacewar!* mode, and who programmed it?

Russell: I didn't program it, and I don't remember which system—I think that was more a PDP-6 thing that was done after I left.

Weaver: You mean for timeshare?

Russell: Yeah. The idea was that *Spacewar!* needed a consistent time slot, but on the PDP-6 it didn't need the whole machine like it did on the PDP-1. The idea was that you would give one user, one of the many timeshared users, a regular time slot and a restriction so they only got their allocation, but that happened regularly with the clock. You could keep a more or less consistent display going. And it turned out that *Spacewar!* mode had lots of other applications.

Weaver: Such as?

Russell: You have a piece of hardware like an arm or a camera looking at a scene, and especially when you're trying to debug the hardware, getting uniform access gave you the possibility of getting a consistent scope trace. For debugging the hardware and running the software, it was very useful to have a consistent time slot.

Weaver: Did the cult success of *Spacewar!* surprise you?

Russell: Not especially, because it gradually grew up over the years. Around MIT, there was this group of *Spacewar!* players that existed that I periodically had to shoo off because I wanted to make a better version, and so there had been some of that. Then I didn't pay any attention to it over the years, and I got interested again when personal computers were around, and you could play games on them, but it was a very gradual thing.

Weaver: Where do you place *Spacewar!* in human-computer interaction?

Russell: Well, I guess you could say the bouncing ball program on Whirlwind could be manually tinkered so that the ball got pushed with different vigor. The Minskytron was similar in the sense that it was a display where you set the parameters and then started it and it did what it did. I think *Spacewar!* was the first thing of any distribution that actually had interaction in the sense of you see something on the screen, you do something to change it, it changes, and you decide what to do next and tell it to do it and it responds. The crucial point there was that the response time was comparable to human response time rather than once every few minutes.

Weaver: [Recording equipment error and pause.] Yeah. Sometimes this stuff is beyond understanding.

Russell: Yes. Well, I always have fun with the kids these days. Nowadays most of them have never had the hardware routinely crash out from under them. Back in the vacuum tube days, that was a weekly occurrence, and with the early tape drives and disk files, it was embarrassingly common. Then I started working where I was dealing with new hardware, and let me tell you, when the hardware passes the smoke test, it ain't crash-proof. [Laughter]

Weaver: How did *Spacewar!* sensitize you to the importance of testing?

Russell: Not that much, because several years before, I did the first LISP interpreter, and that had its own set of bugs. I rewrote it, which had a different set of bugs. Then we attempted to debug the garbage collector. Now, the garbage collector is more or less—now is a very standard sort of thing. It was a crucial part of LISP because the idea was you didn't have to think about storage management. You'd just treat storage as though it was infinite. Because we were clever in writing the programming conventions, we had handles that pointed to all the list structure that mattered. If you couldn't get it through those handles, then it was obviously surplus, and you could reuse it. Now, it's a very simple idea. The keeping track of all the handles turned out to take I would say at least a year and a half, if not longer, to debug. The symptom was everything would be fine for a while and then all of a sudden some of the list structure would get garbaged. Typically, the way it got garbaged was it got connected to the very long list of free storage when it shouldn't have been.

Because the garbage collection was asynchronous to what the program was doing, it was just you do things and then you run out of storage, you call the garbage collector, wait a while, you get storage back and you go on for a while. Because of that, it was relatively hard to reproduce problems. You'd find a problem or you'd get an idea of what it was by the circumstances, but you typically couldn't exactly reproduce it right there and then. You'd have to either speculate on the problem and fix what you thought the problem might have been or you'd read the code in a highly motivated fashion and find some other stupid mistakes, which happened not to be the one that was causing the problem at hand but was definitely a stupid mistake.

Partly because the 704 or 709 debugging was a one-experiment-per-day, maybe, sort of thing, that dragged on for a long time. We were doing a great deal of just trying to get the garbage collector to collect exactly what it was supposed to and nothing else. The problem was in either case, if it was collecting too much, that was sort of obvious, although the fix wasn't, and the detailed circumstances might not be. If it was collecting more than it needed to, that was even worse because what you'd discover is you ran completely out of free storage sooner than you expected. But typically, the programs that were running, like Jim Slagle's symbolic integration program, were things where once it was working on simple examples, you tried more complicated examples. It turned out that a limitation on the complicated examples was the total amount of free storage available. If that happened sooner than you expected, then there was the question of are you sure you don't have a bug lurking in there. Figuring that out was a bit complicated.

Weaver: And what about the importance of user interface?

Russell: Well, a crucial part of making the prototype work was noticing that you didn't need much control. In fact, I'm sure that we figured this out in the preliminary discussions. You didn't need much in the way of controls, and four bits of user input was just fine.

Weaver: Did you end up modifying the interface over time or did you find that your original concept worked pretty well from the start?

Russell: It worked pretty well from the start. The modification was deciding that if you said turn left and turn right both at the same time, that meant you wanted to go into hyperspace. When you're playing from the switches, that's the way you get into hyperspace. When you wire up a control box, you can typically arrange it so that's done with a couple of diodes in the control box and you can give a separate hyperspace button, even though you still are only using four bits of input.

Weaver: Got it. Have I missed anything that was on your list that was important?

Russell: No, I don't think so.

Weaver: Well, so just before we finish, what is it that you would like to tell young programmers who will be looking at the archive now and in the future and the website of the Smithsonian? You've had a lot of years now suffering at the hands of computers, hardware, and engineers. So, looking back, sort of summing it up, what pithy statements would you give aspiring young programmers who are looking at your speaking to them?

Russell: Well, don't be afraid to try something. Pick up something you've coded a year or two earlier—anyway, let something sit for at least a year and then pick it up and look at it and see if you understand it. If you don't, think about what kind of comments to add would have helped you in understanding it.

Also, pay attention to keeping things as simple as possible, except when you *really, really* need something that's hard. I personally have found that it's much easier to train myself to write straightforward, simple code and have it work as intended than it is to do something quick and dirty which doesn't take into account all of the funny things that would happen. I have spent years of my professional life cleaning up sloppy code that other people got almost working—well, got working enough to pass the demo, but not to survive real users.

Weaver: Excellent. Thank you. Appreciate it.

[End of interview.]