Ivan E. Sutherland 1988 A. M. Turing Award Recipient

Ivan E. Sutherland, Vice President and Technical Director of Sutherland, Sproull, and Associates, Inc., received the 1988 ACM A. M. Turing Award for his pioneering contributions to the field of computer graphics. The award, ACM's highest honor in computer science research, was presented at the association's annual Computer Science Conference in Louisville, Kentucky, this past February.

Sutherland is the inventor and developer of the interactive graphics program Sketchpad, which defined the interactive computer graphics field 25 years ago. Sketchpad's many innovations include a display file for screen refresh, a recursively traversed hierarchical structure for modeling graphical objects, recursive methods for geometric transformations, and an object-oriented programming style.

Born in Hastings, Nebraska, Sutherland received his B.S. from Carnegie Mellon University in 1959, his M.S. from California Institute of Technology in 1960, and his Ph.D. from Massachusetts Institute of Technology in 1963, all in electrical engineering. He continued his research in computer graphics as an associate professor at Harvard University, and later as professor of electrical engineering at the University of Utah. In 1968 he cofounded the Evans and Sutherland Computer Corporation where he was Vice President and Chief Scientist. He served as chairman of the Computer Science Department at the California Institute of Technology from 1976 to 1980. In 1980, he left Caltech to establish Sutherland, Sproull, and Associates, a consulting firm, and Advanced Technology Ventures, a venture capital firm, both located in Palo Alto, California.

AN INTERVIEW WITH IVAN SUTHERLAND

KAREN A. FRENKEL

KAF: How did you decide to go into computer science and who influenced you in the course of your training?
IES: One early influence was the man who named the ACM, Edmund Berkeley, who died last year. In the early 50s, my brother, Bert, and I had jobs working for him in New York City. We lived in the suburbs, in Scarsdale, and we used to come down to New York and meet with him, and did various tasks for him for some very nominal wage. I was about 12 at the time. Berkeley was very helpful and taught me quite a lot about computing.

He had a wonderful "personal computer" called Simon. Simon was a relay machine about the size of a suitcase; it weighed about 50 pounds. It had six words of memory of two bits each—so a total of 12 bits of memory. And it could do 2-bit arithmetic, which meant that using double precision arithmetic you could add numbers up to 15 to get sums up to 30.

I wrote the very first program for it to do division so that it divided numbers up to 15 by numbers up to 3. Division was hard because the machine did not have a conditional operation. It was programmed from a paper tape, and the tape just went through sequentially so there was no way for the program to branch. I had to rewire the machine a little bit to make a conditional stop, and that enabled me to get this division program working.

Berkeley introduced me to Claude Shannon, who at that time was at Bell Laboratories. Bert and I went down to visit Shannon and spent a very pleasant day with him. And many years later, before I went to MIT as a graduate student, I wrote a letter to Shannon who had joined the faculty there. He remembered our visit and invited me to come out to his house. He eventually became my thesis supervisor. So another influence on my life was Claude Shannon. He is a very remarkable man.

KAF: In what ways is he such a remarkable man?
IES: He had very broad interests, from how computers work to robots to gadgets of various sorts, and he was a balance nut. He rode a unicycle and could walk on a tightrope. He is quite amazingly skilled. In fact, I have a photograph of him that I took only two or three years ago when he was about 75. I asked him if he'd ride his unicycle for me. He said he wasn't sure he could, but if he put his hand on his wife's shoulder he thought he could manage, and I have a photograph to prove that he did.

KAF: Were you at MIT when Project MAC was going on?
IES: The MAC project got started just as I was leaving MIT. I got my Ph.D. in 1963 with a thesis called Sketchpad. I did Sketchpad on the Lincoln Laboratory's TX-2 computer, which was one of the biggest computers ever built; it filled a room. It was about a 20th the power of a Macintosh II, but it was at that time a very big machine. The man who built it, Wesley A. Clark, was also quite influential on my work. Wes put a number of features into the TX-2 for man-machine interaction that I was able to exercise and make use of in the thesis. It was important that I had that machine to use; it would have been difficult to do the kind of things that I did with graphics at that time without the very strong facilities that were available at Lincoln Laboratory. It had an especially rich input/output system, which included a point plotting display, switches, knobs, light pen, and a keyboard. Finally, it had 70,000 words of memory, each 36 bits long, more than twice as much memory as other large machines of the day.

KAF: How did you come up with the idea for Sketchpad? No one had ever done anything like that before.
IES: Where do ideas come from? I haven't a clue as to where ideas come from. Ideas just come around. It wasn't until years later that Chuck Seitz of Caltech put a common thread through everything I've done. He pointed out that everything I've done has something to do with geometry, and Sketchpad was of course a geometry thing. I had been interested in drawings, mechanical drawings in particular, since a very young age.

My father was a civil engineer, and I used to look at his blueprints and try to understand what they meant, and what was important in them and what wasn't. So I was able to read mechanical drawings before I ever entered high school. When the opportunity came to use a suitable computer, it seemed the most natural thing to make drawings with it.

KAF: Some computer scientists think of themselves as engineers, some as mathematicians, and some of them think that they're writers of code. Do you think that your interest in geometry puts you more firmly in any of those groups?
IES: I've always thought of myself as an engineer. On the other hand, good engineering depends on good science. If you know the fundamental limitations, what you can and can't do, or have some mathematical basis for choosing one design over another, you can build better things more easily. A little bit of mathematics saves an awful lot of engineering effort. The scientific underpinnings of engineering are so extremely valuable that they're to be sought at every turn. But my interest is in building things. I have always been interested in seeing something new actually working. Whereas knowledge for its own sake is valuable and I cherish it, it nevertheless is not the central focus of my life. It's the application of technical knowledge that's important to me.

COMPUTER GRAPHICS

KAF: Much of the work that resulted from your project at the University of Utah has had a tremendous influence on computer graphics. Two ways of thinking about computer graphics, photorealism versus fast image processing, seem to date back to the early days at Utah. Did you see that split happening there at the time?
IES: The period starting in about 1966 when the University of Utah was active in graphics, extended over a decade or more, and was influenced by a lot of people. But at the start, only a few computer pictures had ever been made of opaque objects. Larry Roberts had done work at Lincoln Laboratory, and some work had been done by one or two others to compute what parts of an object hid other parts. But talk about photographic realism—nobody had seen anything that was anywhere close to photographic realism.

The man who figured out the basic techniques that made modern graphics possible was David Evans. He had come out of a whole collection of early computing experiences in which incremental computing was used simply because that was all that could be done with the available equipment. He knew that incremental computing was a powerful way to get more results for less work. He suggested to a collection of his students, and to me, that we make use of all of the information that we'd computed for one pixel in doing the computations for the adjacent pixel—that we try never to recompute *ab initio*, but rather to compute on the basis of nearby changes. The basic principle of incremental computing is still the mainstay of much of high-performance computer graphics today.

In Utah there was a confluence of ideas and people brought together by Evans. Evans had come from Berkeley, I came from Boston, Tom Stockam, an expert on human vision, came from MIT, as did Chuck Seitz, an expert on hardware design. William Newman, who later coauthored an important book on computer graphics, came from England via my group at Harvard. Rich Riesenfeld brought in mathematics of curved surfaces. Rod Rougelot and Bob Schumacher came to the Evans and Sutherland company bringing an entirely different view of graphics that had developed at General Electric. So there were a number of people, who each brought their own collection of ideas and DARPA was very generous with its support, which made it possible to do very interesting research and teaching.

KAF: Do you lean in one direction or the other, regarding photorealsim and fast image processing?
IES: I don't even understand how to answer that question. I mean, you're making a dichotomy that I don't understand very well.

KAF: You don't see the dichotomy.
IES: People use different kinds of pictures for different purposes, and their choices of what kinds of pictures they want to use and for what is a choice that they make. I think that the ability to construct representative pictures has gotten to be pretty good. There are some hanging on the wall right behind you. Some are very pretty, some are technically excellent but not beautiful. Some carry a great deal of underlying meaning. Some deal with trivial subjects. There's an apple and an orange blended together, and that's a classic example of a *tour de force* of image processing, but it is neither beautiful art nor a deeply meaningful subject (Figure 1). It's a demonstration of a technique that is very interesting. Pictures that are scientifically important are often quite abstract. How do I make a beautifully realistic photograph of an atom? Well, atoms are smaller than the wavelength of light, so no one's ever seen one. So I don't know what an atom looks like. Any picture that I make of an atom is going to be a representative picture that may be beautifully rendered with the most gorgeous techniques and have shiny globular parts, shiny electrons and shiny neutrons. That may be very impressive, but it may not correspond to good physics. The important underlying question is, does it convey a meaning that's of some importance? So I don't see the realism *per se* as being controversial.

One of the 10 unsolved problems in computer graphics that I mentioned in one of my articles [3] was the problem of abstraction. Much progress in science is a matter of capturing some new idea, getting some new concept of what something is. For example, Niels Bohr invented a model of the atom, the so-called Bohr atom, which is, as I recall, the model that says electrons whiz around in orbit around the nucleus. And that was a

very controversial idea when he first suggested it—that an atom was not a hard nut, but was mostly empty space. It turns out he had it wrong. Schroedinger later showed that electrons don't whiz around the nucleus. They have some very complicated probabilistic kind of orbit. Somebody else may show Schroedinger to be wrong. But each of these new concepts carries with it a set of pictures that you could make—a set of representations that lets you think about the world according to the model that you've developed. The most exciting power of computer graphics is to make abstract concepts real, and that's also the hardest because it involves visualizing new ideas.

KAF: You said in the conclusion of that article that the really big gains in scientific areas will come when someone invents new abstractions that can be represented only in computer graphical form. You wrote that in 1966. Has that happened?
IES: I've seen some examples of interesting visualizations of abstractions, some of them using moving pictures in ways that you wouldn't do without computer graphics. I don't know whether there have been any great scientific breakthroughs that have been helped because of computer pictures. But it is certain that the scientific and technical community routinely uses computer-processed pictures these days. The financial community routinely uses graphs and financial materials prepared by computer, because they're more understandable. So there's widespread use of computer graphics for precisely these kinds of abstract representation.

ON PROGRAMMING

KAF: You also wrote about the signatures that programmers leave on their code. And you asked in this article, "Have you ever tried to pick up somebody else's computer program and understand what it does? It ought to be possible to display whatever it is about a computer program that makes it unique. It ought to be possible to represent whatever it is that makes an individual computer program individual. I want to look at a display of a computer program and recognize that this part must go with that part just as I can match up the gears of a clock" [3].
IES: I think programming has been treated largely as an alphanumeric task, and most of the work in programming languages and most of the theory of programming and so on has been abstract and alphanumeric. People simply have not made pictures of programs very much, probably because nobody has ideas as to what a program should look like. I think it might be interesting to do, but I don't have any suggestions as to what they look like either.

KAF: It seems that there's a crisis of programmer productivity. I wonder if your hope is that if we can look at how programs work, that might be a way to solve the problem.

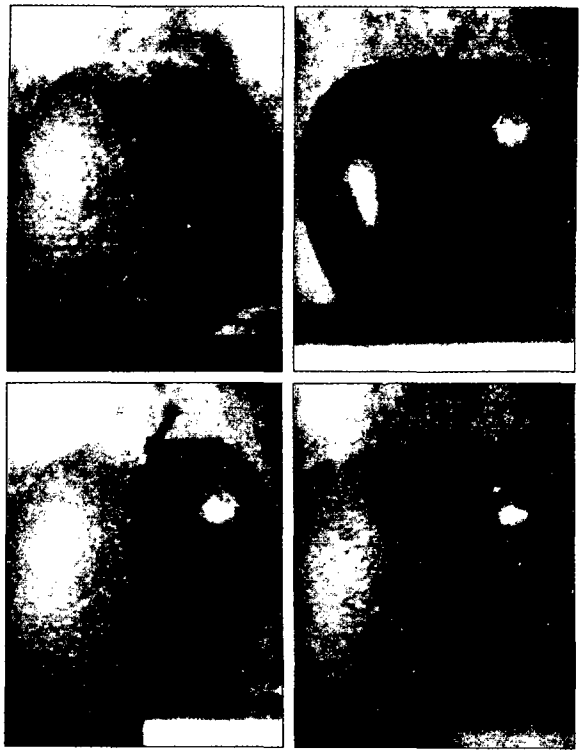

FIGURE 1. A "*Tour de Force* of Image Processing" [1]

IES: It's a fond hope to think that programming is going to become easy. The hard part of programming is understanding what the computer is supposed to do. If you want to write a program to fill out people's income tax, the first thing you have to do is understand what the income tax law is. It's hard for me to believe that any programming language is going to make that task easy. The hard parts of programming are generally hard because it's *understanding* what the problem is that's hard. You're a journalist, so you know about writing. Writing an article is hard not because putting one word after another is hard, but because you have to know what the subject is. You have first to do the research to learn what to talk about. I think programming is the same. It's doing the research to learn what the program's supposed to do that's hard, and I don't think that's going to get easy any more than a fancy word processor makes journalism a lot easier.

KAF: But perhaps if you could examine it in this visual way you're talking about, if it can't become easier, maybe it could become a little bit more efficient.
IES: I suggested that a visual display of a program makes going through an existing body of code a little easier, the same way that a road map makes getting through an existing body of territory easier. I don't have any suggestions as to what programs look like, however, so I leave that as an unsolved problem. I can't sit here and tell you that I think it's the world's most important unsolved problem.

I once said to Alan Perlis, "You know, computers and the information revolution that we're going through today may be the most important thing that man's ever discovered or invented." And Perlis looked me in the eye and said, "How does it compare with fire?"

CALTECH AND VLSI DESIGN

KAF: You founded the computer science department with Carver Mead at Caltech. And you wrote a paper with him that appeared in Scientific American on microelectronics [4]. Could you tell me about your transition from graphics to VLSI?

IES: In the graphics business, we built equipment, starting at Harvard, and then at Evans and Sutherland. We basically built special-purpose computers to do graphics, and so I had been face-to-face with using electronics during all of my career. In 1975, I became heavily involved in integrated circuit work with Carver. He and I had been commissioned by DARPA to do a study on integrated circuit fabrication. Carver knew quite a lot about how integrated circuits worked and how to design them. In the course of our study he taught me quite a bit. It had a healthy dose of geometry, which is something that I like, so I was intrigued. When we started the computer science department, it was clear it would never be very large. Caltech is a tiny little school; it has only 800 undergraduates. That's only 200 undergraduates in every class. So we decided to specialize and try and do one thing well, rather than do lots of things badly. Since Carver had built up some expertise and had ideas as to what to do in the integrated circuit arena, we decided we would focus on that. I think that was a wise choice. It was the only way to provide enough critical mass in some area to do some interesting things that involved basically two beliefs.

Carver believed (and I adopted his belief and was instrumental in getting support put behind it), that integrated circuit design was not black magic—ordinary people could do it following a fairly simple set of rules. He subsequently published that [idea] with Lynn Conway in a well-known VLSI book [2]. In our *Scientific American* article, we pointed out that there was a place where computer science folks were counting the wrong thing. When you build a computer, most of what's in the computer is wires, and very small fraction of the computer is switches. This is truer and truer as transistors get smaller. Most of the failures, most of the delay in the integrated circuit, most of the difficulty of laying it out, has to do with the wires.

[In other words] an adequate theory to deal with computing has to involve the communication of numbers from one place to another. If you simply count the number of multiplications that are required to do some operation and try to reduce them, you have missed the point. You may not have developed the minimal machine to do the job, because by doing that you may have increased the amount of communication and thus the number of wires. Since communication is expensive and arithmetic is relatively cheap, you may have minimized the wrong thing.

So the set of ideas that came out of Caltech [is], Let the physics tell you what's expensive and what's cheap in computing and then develop your theories to match that the physical constraint. The result is a little unnatural for people with a mathematical bent, because in the ordinary notions of mathematics, getting a variable from one place to another on a page is free. If you just write the same symbol somewhere else on the page, its value appears by magic. The required communication is implicit; it is not written down anywhere and consequently doesn't reach the focus of your attention. The operation, on the other hand, is explicit; you write a multiplication sign and you write an addition sign, and so the operations seem important whereas the physics of circuits makes the actual costs just the reverse.

TRANSITION SIGNALLING, FLIP FLOPS, AND METASTABILITY

KAF: How did this lead to your interest in transition signalling?

IES: Throughout that period I had been interested in transition signalling and I tried various designs. Some worked and some didn't. The basic idea that works is to design a set of modules that can then be composed into systems. At that time, the late 70s, the techniques for designing those modules just weren't available. It was just too hard to do. A little bit later on, in about 1983 or 84, after I'd left Caltech, I learned that Charlie Molnar and his associates at Washington University developed a very simple way of designing little asynchronous modules and we have made quite a bit of use of it since. That new design technique, to make the modules possible, is a very big help in making transition signalling practical.

KAF: Are there other reasons why people don't use transition signalling?

IES: Yes. There is the chicken and egg phenomenon; there aren't any components available to do transition signalling because nobody uses it. And nobody uses it because there aren't any components available. And there aren't any components available because nobody teaches how to do it. And nobody would bother teaching how to do it because if you knew how to do it you couldn't buy any parts anyhow. Now, to some extent that chicken and egg phenomenon gets broken by VLSI, because when you do a custom integrated circuit design you start with a blank piece of silicon, and you can build your own components to do whatever you want. The same, however, does not apply to memories, because the merchant semiconductor people can produce memories cheaper than anybody. Nor does it apply to microprocessors, because you can buy off-the-shelf microprocessors of a power and capability that you couldn't possibly build as custom circuits. So there is still some problem in getting started in using transition

signalling.

One of the reasons I chose transition signalling as the topic for my Turing lecture is in the hope of attacking that chicken and egg problem by getting enough folks interested and active in the field. That way they can begin getting some components available so that people can build useful systems. That was a long answer to a short question.

KAF: It's a fascinating answer because the initial question was why it was too hard. I see it was too hard because it involved different ways of thinking, physics versus mathematical. But can you elaborate on modules?

IES: The early designers built computers using events that went through delays. That style of design was abandoned in favor of clocked systems because of the inability to design modules, as I already mentioned. Another reason had to do with how you debug the resulting system. It was an era in which computers were built with wire-wrapped back panels and each machine had a "personality" brought about by the exact ways in which the wires lay together. Different, peculiar problems would show up in individual machines and people recognized that by using clocked systems they could reduce substantially the task of actually checking that the machine was correct, because not only was the information quantized, so was the time.

TECHNOLOGY TRANSFER

KAF: Can you generalize from the fact that transition signalling has been in development for 20 years to why technology transfer takes so long?

IES: One hypothesis is that [a technology] is wrong. It's taken an awfully long time for the U.S. automobile industry to adopt steam engines. So one position one can take is that when things take a long time, they're not right. Another position that one can take is that, for good and sufficient reasons that were probably correct at the time, an industry chose to go in one direction. For example, nearly all typewriters have a certain arrangement of keys, and that choice was made long ago. Changing that choice is next to impossible, even though that arrangement of keys may not be the best. So a second hypothesis is that the computing industry has chosen to use clocked systems for reasons that may have been good at that time. Getting that choice rethought may be very difficult. It may be very difficult because there is such a mass of production behind the conventional clocked system.

On the other hand, design costs for today's computing machinery are very high. In fact, a large fraction of the overall cost may be the design. It seems to me that anything that can help speed the design along may be very valuable. Moreover, systems that are built with clocks tend to be monolithic. They tend to resist modification. They tend to resist incremental improvement. The transition signalling alternative that I presented is a nice way to provide intermediate upgrades partway through a system life. You can simply unplug one circuit part, plug in another one, and expect to obtain some benefit of improved performance. Those gains may outweigh the difficulties that at one time frightened people away. The intellectual difficulties of designing transition signalling devices have now largely been wiped out. That's the progress of the technology.

KAF: Your present company, Sutherland, Sproull, and Associates, is in the venture capital business and so you spot technologies or companies that you think have promise. Can you tell me about those that you've financed in the last few years that you're most pleased to have foreseen would be promising?

IES: Sutherland, Sproull does three things. We do management consulting for a variety of clients, usually associated with research management. We do technical work, of which the Turing lecture subject is one example. And we do venture capital. I don't think it would be appropriate for me to discuss specific investments here, but we've backed a number of firms that we think are interesting.

WALKING MACHINES AND ROBOTICS

KAF: Do you have a favorite project that you thought was the most challenging?

IES: The project that was the most fun was the walking machine. In about 1979, I got interested in legged locomotion. Mark Raibert came into my office and said, "Ivan, I think we can make a machine that hops on one leg." Hopping can be very hard, but I encouraged him. I had a little research kitty and I gave him a little research money and I got involved. I thought, "If you're going to build a machine that walks, it wouldn't be any fun if you couldn't ride on it. So it ought to be big enough to sit on and ride." It was about half a ton of steel. It had hydraulic cylinders to move six legs, an 18-horsepower gas engine to run the hydraulic pumps, and of course, a computer to control it. And we had wonderful fun. When it was nice weather, we took it out and ran it around the parking lot, and it would get in trouble. Something would break down and then we'd have this half ton of inert iron in the parking lot and we'd have to figure out how to get back home again. The kids at CMU named it the Trojan Cockroach. College students are irreverent, if anything.

Claude Shannon came down for a visit. He always liked robots, and he wrote a little ditty about it:

Higgledy piggledy
Ivan E. Sutherland
Built a huge cockroach
Twelve horsepower clout.

But the roach
Waxing vengeful
From previous roach genocide
Hexapodiacally stamped Ivan out.

Claude and his wife stayed at my house, and when he came home the evening after we'd run this machine, he said to her, "Well, Betty, I saw the robot do a pi-

rouette." Imagine half a ton of iron turning 180 degrees, a process which took it at least 30 seconds, and describing that as a pirouette.

I have another story. Mark Donner, a graduate student, did much of the programming for the walking machine. And Mike Ullner, a post-doc who got his Ph.D. at Caltech, came to work for me. Each time we made a run with the walking machine we had to put some water in its cylinders. Once, we had just finished filling them up with a siphon from a bucket that we hung up high, and for some reason I ended up with my thumb over the end of the syphon tube. Now, I'd been at some pains to stick the tubing into the bucket and to prime the siphon. If I let go, water would go all over the floor, but I didn't want to just pull the tube out of the bucket because it would be a nuisance to have to put it back in. So what was I going to do? And I thought, "Well, I'll just blow in the end of the tube and blow the water back up in the bucket." I started to do this when Mark Donner said, "Oh, Ivan, you're never going to be able to blow that water back up." And Mike Ulner, who had known me somewhat longer, said to Mark, "Mark, never underestimate the power of the professorial windbag."

KAF: Some people would question the point of your walking machine. Can you tell me what was learned from that project?
IES: The reason I embarked on it was that it was an attention focuser to learn about locomotion, and to learn about a particular technology that I didn't know anything about, namely hydraulics. I learned basically two things: Number one, I learned a great deal about what technology is available for building hydraulic systems. I bought myself an education. Second, we learned that six is the wrong number of legs for a machine of that size. And that message I think is fairly interesting. Depending on the scale of an object, the relative importance of weight and inertia are different. For example, an ant has no difficulty at all lifting four or five times its own weight, and to an ant, a wall is as good a surface to walk on as a floor. So is a ceiling. An ant cares relatively little about gravity and that just has to do with the fact that it's a very small-scale animal. People operate in a whole different regime. We can't walk on walls. If you've ever watched a football player running for a touchdown, after he crosses the goal line he stops by leaning back at about a 45 degree angle. It's quite clear that the inertial forces, the momentum, that people and horses have are a very important part of their actions. A walking machine that's in the same scale as people or horses or dogs must have a control that takes inertial forces into account. The objective of having six legs is to not have to worry about inertial forces, but you just cannot ignore the inertial forces in a big machine. So six is the wrong number of legs to have. It should have four. One, two or four seem to be the right numbers and if you look around the world you'll find existence proofs of big and little things that have one, two, and four legs and little things with six or eight or more legs.

KAF: Are you continuing to look at that research now, or have you funded that kind of research since?
IES: No, not in any very serious way. I thought of the walking machine project as kind of fun. It was a wonderfully fun research project. We learned some things; we had a number of interesting and educational experiences. There was the day the hydraulic manifold sprung a leak and 2,000 PSI hot hydraulic fluid [gushed] all over the room. We had a lot of excitement. It was an attention focuser. Mark Donner used the six-legged control mechanism for his thesis and used the six legs as an exercise in independent control. His thesis was about how to make six separately operating algorithms, one for each leg, communicate only very general information and yet get reliable walking behavior. He used it as an attention focuser about independent processes running in a separate way. For that kind of purpose it was a wonderful project. It wasn't practical. It was never intended to be practical.

It seems to me the secret to success in research activities is to pick something easy enough to do. It's very nice if you can find something that you think is easy and everybody else thinks is hard. That's wonderful.

KAF: This occurred at a time when there was a flurry of research in robotics, and everyone was very excited about industrial robots. People thought of robotics as a way to combat the Rust Belt and general production problems that we saw in heavy industry, where factory activities were hazardous to humans. But that seems to have faded away. Can if you comment on the direction of robotics today?
IES: I don't know what you mean by faded away. I wonder how many welding robots General Motors runs. I wonder how many painting robots there are in service today. There's probably more than there were in 1984. Probably be more next year.

KAF: Yes, but it's no booming industry.
IES: Oh, I don't know what you mean by that.

KAF: What I mean by that is that instead of thinking in terms of individual robots being installed in factories that exist, people found that that was too hard, and now they're looking at automating whole new factories from scratch. And I just was wondering if you wanted to comment on the direction of the industry.
IES: The robot business is a capital investment business, and changing capital investments takes a long time. It just does. If I have a factory that's been in existence for a dozen years and it's all paid off, but the equipment in it is doing a job, there's no reason to replace it. The robot industry is working into an eco-

nomic system in which changes to the capital base come relatively slowly. Another thing about industrial robots is that programming them is a fairly expensive process. What it is that the robot is supposed to do needs to be established. It takes some skill and understanding to get that right. You can't just stamp robots out and turn them loose. You have to explain and train them. You can't just produce people and turn them loose and expect them to take factory jobs either. They have to be trained, too.

THE DIRECTION OF COMPUTER SCIENCE AND RESEARCH TOPICS

KAF: Getting back to more general matters in computer science, how do you think the field is progressing as a whole? A Business Week cover story last March on the computer industry emphasized the difficulty of software catching up with hardware and said there may be a lull in the industry. Is it as exciting and fruitful a time in computer science as it's ever been? Have things leveled off in terms of the problems there are to face?

IES: This is the kind of question the answer to which requires hundreds of thousands of interviews. I don't have a window on computer science. I have a little tiny peephole that I look out at some corner of the field that I'm interested in and have been able to keep track of. I find lots of interesting things to do.

Over a long period of time I've tried to pick projects that seemed to me to be easy. It seems to me the secret to success in research activities is to pick something easy enough to do. It's very nice if you can find something that you think is easy and everybody else thinks it's hard. That's wonderful, but a lot of people tend to pick projects that are too hard. I've tried to pick easy ones to work on. And there are plenty of easy ones going begging.

I think the notion of whether it is an exciting time, is very much a personal business. It's a matter of whether you are excited by what you're doing. I find interesting things to do. I'm not seeking some other field to go into. One of the nice things about computing is that it has many aspects, and what I've chosen to do is move from one aspect to another; every five years or so, when I get tired of one thing, there's always some other aspect of the field that's available to look at.

KAF: Can you tell me how it feels to have gotten the award?

IES: I was very honored to be selected as the Turing awardee. I was delighted that my two grown children, who are also in computer science, elected to come to the lecture.

KAF: What will you do next ?

IES: I'm working on transition signalling logic now. If we can get more people involved in it so that parts are available, standards developed, and people know how to use them, it may simplify the design of computing systems. Moreover, the systems would be more malleable, and better fit the changing needs and the changing demands of the marketplace. That's a big task. It's a task that's bigger than any one company can undertake. It's going to take the cooperative efforts of a number of companies, a number of academic institutions and a number of people to bring that about.

KAF: Is there anything I didn't ask that you want to bring up?

IES: Yes. There are two things that I want to emphasize. One addresses a sense that I have of the importance of young people. My experience being a college professor is that I always learned more from my students than I felt the students learned from me. Graduate education is a process of mutually learning things that aren't known beforehand. My message to young people is that developing new technology literally takes courage; one has to risk exposing one's ideas to the criticism of one's peers. That act takes some courage. Young people don't recognize that when they embark on a research career they have embarked on something that has emotional content, and that has personal impact, as well as technical content. There are not only financial but also emotional and personal risks involved. To me, failures of courage have often felt like something else. I did some research when I was getting a master's degree, and it wasn't until 10 years later that I published the paper. What it felt like to me was that no one would be interested. But in fact what had happened was, I'd had a failure of courage. It turned out to be a perfectly good paper and people accepted it.

I also want to emphasize that choice of problems to work on is very much a personal thing. It matters less what problem someone works on than that he works on it well. It does not matter what problems a person picks as long as they're problems that interest him. What I've tried to do, is find something that's interesting in a field where not too many other people are working, but above all something that's interesting, that I can get excited about and involved in. Because you get the best results when you're most deeply involved. People work most effectively at jobs that they like doing, and in some sense, in the research world, if it isn't fun, you're doing something wrong. Somehow, advanced technology is exciting. It's a thrill of some sort. I've enjoyed that thrill over a long time. I wouldn't give it up for the world.

REFERENCES

1. Burt, P.J., and Adelson, E.H. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics,* 2, 4 (Oct. 1983), pp. 217–236.
2. Mead, C., and Conway, L. *Introduction to VSLI Systems.* Addison-Wesley, Reading, MA, 1980.
3. Sutherland, I.E. Computer graphics: Ten unsolved problems. *Datamation* (May 1966), 22–27.
4. Sutherland, I.E., and Mead, C. Microelectronics and computer science. *Sci. Am.* (Sept. 1977), 210–228.