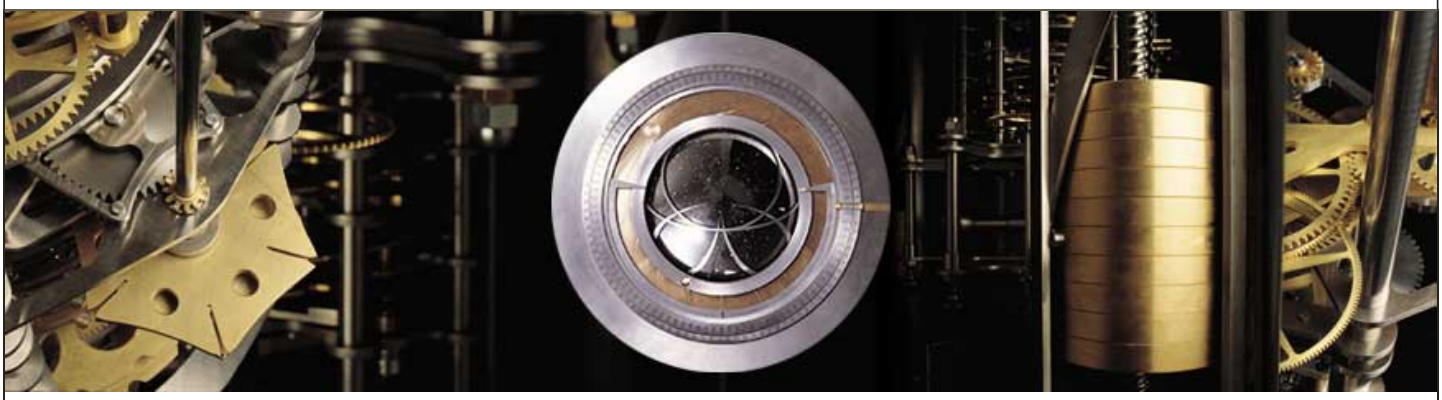

TEDxCaltech - Danny Hillis - Reminiscing about Richard Feynman

Richard Feynman and The Connection Machine

by W. Daniel Hillis for Physics Today

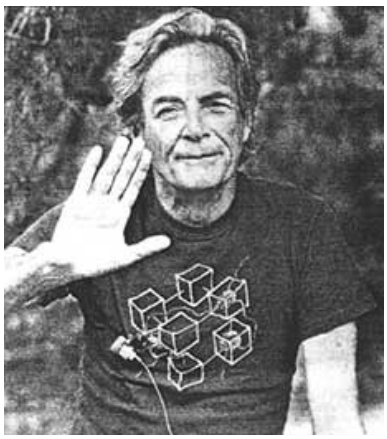

Photo by Faustin Bray

One day when I was having lunch with Richard Feynman, I mentioned to him that I was planning to start a company to build a parallel computer with a million processors. His reaction was unequivocal, "That is positively the dopiest idea I ever heard." For Richard a crazy idea was an opportunity to either prove it wrong or prove it right. Either way, he was interested. By the end of lunch he had agreed to spend the summer working at the company.

Richard's interest in computing went back to his days at Los Alamos, where he supervised the "computers," that is, the people who operated the mechanical calculators. There he was instrumental in setting up some of the first plug-programmable tabulating machines for physical simulation. His interest in the field was heightened in the late 1970's when his son, Carl, began studying computers at MIT.

I got to know Richard through his son. I was a graduate student at the MIT Artificial Intelligence Lab and Carl was one of the undergraduates helping me with my thesis project. I was trying to design a computer fast enough to solve common sense reasoning problems. The machine, as we envisioned it, would contain a million tiny computers, all connected by a communications network. We called it a "Connection Machine." Richard, always interested in his son's activities, followed the project closely. He was skeptical about the idea, but whenever we met at a conference or I visited CalTech, we would stay up until the early hours of the morning discussing details of the planned machine. The first time he ever seemed to believe that we were really going to try to build it was the lunchtime meeting.

Richard arrived in Boston the day after the company was incorporated. We had been busy raising the money, finding a place to rent, issuing stock, etc. We set up in an old mansion just outside of the city, and when Richard showed up we were still recovering from the shock of having the first few million dollars in the bank. No one had thought about anything technical for several months. We were arguing about what the name of the company should be when Richard walked in, saluted, and said, "Richard Feynman reporting for duty. OK, boss, what's my assignment?" The assembled group of not-quite-graduated MIT students was astounded.

After a hurried private discussion ("I don't know, you hired him..."), we informed Richard that his assignment would be to advise on the application of parallel processing to scientific problems.

"That sounds like a bunch of baloney," he said. "Give me something real to do."

So we sent him out to buy some office supplies. While he was gone, we decided that the part of the machine that we were most worried about was the router that delivered messages from one processor to another. We were not sure that our design was going to work. When Richard returned from buying pencils, we gave him the assignment of analyzing the router.

The Machine

The router of the Connection Machine was the part of the hardware that allowed the processors to communicate. It was a complicated device; by comparison, the processors themselves were simple. Connecting a separate communication wire between each pair of processors was impractical since a million processors would require 10^{12} wires. Instead, we planned to connect the processors in a 20-dimensional hypercube so that each processor would only need to talk to 20 others directly. Because many processors had to communicate simultaneously, many

messages would contend for the same wires. The router's job was to find a free path through this 20-dimensional traffic jam or, if it couldn't, to hold onto the message in a buffer until a path became free. Our question to Richard Feynman was whether we had allowed enough buffers for the router to operate efficiently.

During those first few months, Richard began studying the router circuit diagrams as if they were objects of nature. He was willing to listen to explanations of how and why things worked, but fundamentally he preferred to figure out everything himself by simulating the action of each of the circuits with pencil and paper.

In the meantime, the rest of us, happy to have found something to keep Richard occupied, went about the business of ordering the furniture and computers, hiring the first engineers, and arranging for the Defense Advanced Research Projects Agency (DARPA) to pay for the development of the first prototype. Richard did a remarkable job of focusing on his "assignment," stopping only occasionally to help wire the computer room, set up the machine shop, shake hands with the investors, install the telephones, and cheerfully remind us of how crazy we all were. When we finally picked the name of the company, Thinking Machines Corporation, Richard was delighted. "That's good. Now I don't have to explain to people that I work with a bunch of loonies. I can just tell them the name of the company."

The technical side of the project was definitely stretching our capacities. We had decided to simplify things by starting with only 64,000 processors, but even then the amount of work to do was overwhelming. We had to design our own silicon integrated circuits, with processors and a router. We also had to invent packaging and cooling mechanisms, write compilers and assemblers, devise ways of testing processors simultaneously, and so on. Even simple problems like wiring the boards together took on a whole new meaning when working with tens of thousands of processors. In retrospect, if we had had any understanding of how complicated the project was going to be, we never would have started.

'Get These Guys Organized'

I had never managed a large group before and I was clearly in over my head. Richard volunteered to help out. "We've got to get these guys organized," he told me. "Let me tell you how we did it at Los Alamos."

Every great man that I have known has had a certain time and place in their life that they use as a reference point; a time when things worked as they were supposed to and great things were accomplished. For Richard, that time was at Los Alamos during the Manhattan Project. Whenever things got "cockeyed," Richard would look back and try to understand how now was different than then. Using this approach, Richard decided we should pick an expert in each area of importance in the machine, such as software or packaging or electronics, to become the "group leader" in this area, analogous to the group leaders at Los Alamos.

Part Two of Feynman's "Let's Get Organized" campaign was that we should begin a regular seminar series of invited speakers who might have interesting things to do with our machine. Richard's idea was that we should concentrate on people with new applications, because they would be less conservative about what kind of computer they would use. For our first seminar he invited John Hopfield, a friend of his from CalTech, to give us a talk on his scheme for building neural networks. In 1983, studying neural networks was about as fashionable as studying ESP, so some people considered John Hopfield a little bit crazy. Richard was certain he would fit right in at Thinking Machines Corporation.

What Hopfield had invented was a way of constructing an [associative memory], a device for remembering patterns. To use an associative memory, one trains it on a series of patterns, such as pictures of the letters of the alphabet. Later, when the memory is shown a new pattern it is able to recall a similar pattern that it has seen in the past. A new picture of the letter "A" will "remind" the memory of another "A" that it has seen previously. Hopfield had figured out how such a memory could be built from devices that were similar to biological neurons.

Not only did Hopfield's method seem to work, but it seemed to work well on the Connection Machine. Feynman figured out the details of how to use one processor to simulate each of Hopfield's neurons, with the strength of the connections represented as numbers in the processors' memory. Because of the parallel nature of Hopfield's algorithm, all of the processors could be used concurrently with 100\% efficiency, so the Connection Machine would be hundreds of times faster than any conventional computer.

An Algorithm For Logarithms

Feynman worked out the program for computing Hopfield's network on the Connection Machine in some detail. The part that he was proudest of was the subroutine for computing logarithms. I mention it here not only because it is a clever algorithm, but also because it is a specific contribution Richard made to the mainstream of computer science. He invented it at Los Alamos.

Consider the problem of finding the logarithm of a fractional number between 1.0 and 2.0 (the algorithm can be generalized without too much difficulty). Feynman observed that any such number can be uniquely represented as a product of numbers of the form $1 + 2^{-k}$, where k is an integer. Testing each of these factors in a binary number representation is simply a matter of a shift and a subtraction. Once the factors are determined, the logarithm can be computed by adding together the precomputed logarithms of the factors. The algorithm fit especially well on the Connection Machine, since the small table of the logarithms of $1 + 2^{-k}$ could be shared by all the processors. The entire computation took less time than division.

Concentrating on the algorithm for a basic arithmetic operation was typical of Richard's approach. He loved the details. In studying the router, he paid attention to the action of each individual gate and in writing a program he insisted on understanding the implementation of every instruction. He distrusted abstractions that could not be directly related to the facts. When several years later I wrote a general interest article on the Connection Machine for [Scientific American], he was disappointed that it left out too many details. He asked, "How is anyone supposed to know that this isn't just a bunch of crap?"

Feynman's insistence on looking at the details helped us discover the potential of the machine for numerical computing and physical simulation. We had convinced ourselves at the time that the Connection Machine would not be efficient at "number-crunching," because the first prototype had no special hardware for vectors or floating point arithmetic. Both of these were "known" to be requirements for number-crunching. Feynman decided to test this assumption on a problem that he was familiar with in detail: quantum chromodynamics.

Quantum chromodynamics is a theory of the internal workings of atomic particles such as protons. Using this theory it is possible, in principle, to compute the values of measurable physical quantities, such as a proton's mass. In practice, such a computation requires so much arithmetic that it could keep the fastest computers in the world busy for years. One way to do this calculation is to use a discrete four-dimensional lattice to model a section of space-time. Finding the solution involves adding up the contributions of all of the possible configurations of certain matrices on the links of the lattice, or at least some large representative sample. (This is essentially a Feynman path integral.) The thing that makes this so difficult is that calculating the contribution of even a single configuration involves multiplying the matrices around every little loop in the lattice, and the number of loops grows as the fourth power of the lattice size. Since all of these multiplications can take place concurrently, there is plenty of opportunity to keep all 64,000 processors busy.

To find out how well this would work in practice, Feynman had to write a computer program for QCD. Since the only computer language Richard was really familiar with was Basic, he made up a parallel version of Basic in which he wrote the program and then simulated it by hand to estimate how fast it would run on the Connection Machine.

He was excited by the results. "Hey Danny, you're not going to believe this, but that machine of yours can actually do something [useful]!" According to Feynman's calculations, the Connection Machine, even without any special hardware for floating point arithmetic, would outperform a machine that CalTech was building for doing QCD calculations. From that point on, Richard pushed us more and more toward looking at numerical applications of the machine.

By the end of that summer of 1983, Richard had completed his analysis of the behavior of the router, and much to our surprise and amusement, he presented his answer in the form of a set of partial differential equations. To a physicist this may seem natural, but to a computer designer, treating a set of boolean circuits as a continuous, differentiable system is a bit strange. Feynman's router equations were in terms of variables representing continuous quantities such as "the average number of 1 bits in a message address." I was much more accustomed to seeing analysis in terms of inductive proof and case analysis than taking the derivative of "the number of 1's" with respect to time. Our discrete analysis said we needed seven buffers per chip; Feynman's equations suggested that we only needed five. We decided to play it safe and ignore Feynman.

The decision to ignore Feynman's analysis was made in September, but by next spring we were up against a wall. The chips that we had designed were slightly too big to manufacture and the only way to solve the problem was to cut the number of buffers per chip back to five. Since Feynman's equations claimed we could do this safely, his unconventional methods of analysis started looking better and better to us. We decided to go ahead and make the chips with the smaller number of buffers.

Fortunately, he was right. When we put together the chips the machine worked. The first program run on the machine in April of 1985 was Conway's game of Life.

Cellular Automata

The game of Life is an example of a class of computations that interested Feynman called [cellular automata]. Like many physicists who had spent their lives going to successively lower and lower levels of atomic detail, Feynman often wondered what was at the bottom. One possible answer was a cellular automaton. The notion is that the "continuum" might, at its lowest levels, be discrete in both space and time, and that the laws of physics might simply be a macro-consequence of the average behavior of tiny cells. Each cell could be a simple automaton that obeys a small set of rules and communicates only with its nearest neighbors, like the lattice calculation for QCD. If the universe in fact worked this way, then it presumably would have testable consequences, such as an upper limit on the density of information per cubic meter of space.

The notion of cellular automata goes back to von Neumann and Ulam, whom Feynman had known at Los Alamos. Richard's recent interest in the subject was motivated by his friends Ed Fredkin and Stephen Wolfram, both of whom were fascinated by cellular automata models of physics. Feynman was always quick to point out to them that he considered their specific models "kooky," but like the Connection Machine, he considered the subject sufficiently crazy to put some energy into.

There are many potential problems with cellular automata as a model of physical space and time; for example, finding a set of rules that obeys special relativity. One of the simplest problems is just making the physics so that things look the same in every direction. The most obvious pattern of cellular automata, such as a fixed three-dimensional grid, have preferred directions along the axes of the grid. Is it possible to implement even Newtonian physics on a fixed lattice of automata?

Feynman had a proposed solution to the anisotropy problem which he attempted (without success) to work out in detail. His notion was that the underlying automata, rather than being connected in a regular lattice like a grid or a pattern of hexagons, might be randomly connected. Waves propagating through this medium would, on the average, propagate at the same rate in every direction.

Cellular automata started getting attention at Thinking Machines when Stephen Wolfram, who was also spending time at the company, suggested that we should use such automata not as a model of physics, but as a practical method of simulating physical systems. Specifically, we could use one processor to simulate each cell and rules that were chosen to model something useful, like fluid dynamics. For two-dimensional problems there was a neat solution to the anisotropy problem since [Frisch, Hasslacher, Pomeau] had shown that a hexagonal lattice with a simple set of rules produced isotropic behavior at the macro scale. Wolfram used this method on the Connection Machine to produce a beautiful movie of a turbulent fluid flow in two dimensions. Watching the movie got all of us, especially Feynman, excited about physical simulation. We all started planning additions to the hardware, such as support of floating point arithmetic that would make it possible for us to perform and display a variety of simulations in real time.

Feynman the Explainer

In the meantime, we were having a lot of trouble explaining to people what we were doing with cellular automata. Eyes tended to glaze over when we started talking about state transition diagrams and finite state machines. Finally Feynman told us to explain it like this,

"We have noticed in nature that the behavior of a fluid depends very little on the nature of the individual particles in that fluid. For example, the flow of sand is very similar to the flow of water or the flow of a pile of ball bearings. We have therefore taken advantage of this fact to invent a type of imaginary particle that is especially simple for us to simulate. This particle is a perfect ball bearing that can move at a single speed in one of six directions. The flow of these particles on a large enough scale is very similar to the flow of natural fluids."

This was a typical Richard Feynman explanation. On the one hand, it infuriated the experts who had worked on the problem because it neglected to even mention all of the clever problems that they had solved. On the other hand, it delighted the listeners since they could walk away from it with a real understanding of the phenomenon and how it was connected to physical reality.

We tried to take advantage of Richard's talent for clarity by getting him to critique the technical presentations that we made in our product introductions. Before the commercial announcement of the Connection Machine CM-1 and all of our future products, Richard would give a sentence-by-sentence critique of the planned presentation. "Don't say `reflected acoustic wave.' Say [echo]." Or, "Forget all that `local minima' stuff. Just say there's a bubble caught in the crystal and you have to shake it out." Nothing made him angrier than making something simple sound complicated.

Getting Richard to give advice like that was sometimes tricky. He pretended not to like working on any problem that was outside his claimed area of expertise. Often, at Thinking Machines when he was asked for advice he would gruffly refuse with "That's not my department." I could never figure out just what his department was, but it did not matter anyway, since he spent most of his time working on those "not-my-department" problems. Sometimes he really would give up, but more often than not he would come back a few days after his refusal and remark, "I've been thinking about what you asked the other day and it seems to me..." This worked best if you were careful not to expect it.

I do not mean to imply that Richard was hesitant to do the "dirty work." In fact, he was always volunteering for it. Many a visitor at Thinking Machines was shocked to see that we had a Nobel Laureate soldering circuit boards or painting walls. But what Richard hated, or at least pretended to hate, was being asked to give advice. So why were people always asking him for it? Because even when Richard didn't understand, he always seemed to understand better than the rest of us. And whatever he understood, he could make others understand as well. Richard made people feel like a child does, when a grown-up first treats him as an adult. He was never afraid of telling the truth, and however foolish your question was, he never made you feel like a fool.

The charming side of Richard helped people forgive him for his uncharming characteristics. For example, in many ways Richard was a sexist. Whenever it came time for his daily bowl of soup he would look around for the nearest "girl" and ask if she would fetch it to him. It did not matter if she was the cook, an engineer, or the president of the company. I once asked a female engineer who had just been a victim of this if it bothered her. "Yes, it really annoys me," she said. "On the other hand, he is the only one who ever explained quantum mechanics to me as if I could understand it." That was the essence of Richard's charm.

A Kind Of Game

Richard worked at the company on and off for the next five years. Floating point hardware was eventually added to the machine, and as the machine and its successors went into commercial production, they were being used more and more for the kind of numerical simulation problems that Richard had pioneered with his QCD program. Richard's interest shifted from the construction of the machine to its applications. As it turned out, building a big computer is a good excuse to talk to people who are working on some of the most exciting problems in science. We started working with physicists, astronomers, geologists, biologists, chemists --- everyone of them trying to solve some problem that it had never been possible to solve before. Figuring out how to do these calculations on a parallel machine requires understanding of the details of the application, which was exactly the kind of thing that Richard loved to do.

For Richard, figuring out these problems was a kind of a game. He always started by asking very basic questions like, "What is the simplest example?" or "How can you tell if the answer is right?" He asked questions until he reduced the problem to some essential puzzle that he thought he would be able to solve. Then he would set to work, scribbling on a pad of paper and staring at the results. While he was in the middle of this kind of puzzle solving he was impossible to interrupt. "Don't bug me. I'm busy," he would say without even looking up. Eventually he would either decide the

problem was too hard (in which case he lost interest), or he would find a solution (in which case he spent the next day or two explaining it to anyone who listened). In this way he worked on problems in database searches, geophysical modeling, protein folding, analyzing images, and reading insurance forms.

The last project that I worked on with Richard was in simulated evolution. I had written a program that simulated the evolution of populations of sexually reproducing creatures over hundreds of thousands of generations. The results were surprising in that the fitness of the population made progress in sudden leaps rather than by the expected steady improvement. The fossil record shows some evidence that real biological evolution might also exhibit such "punctuated equilibrium," so Richard and I decided to look more closely at why it happened. He was feeling ill by that time, so I went out and spent the week with him in Pasadena, and we worked out a model of evolution of finite populations based on the Fokker Planck equations. When I got back to Boston I went to the library and discovered a book by Kimura on the subject, and much to my disappointment, all of our "discoveries" were covered in the first few pages. When I called back and told Richard what I had found, he was elated. "Hey, we got it right!" he said. "Not bad for amateurs."

In retrospect I realize that in almost everything that we worked on together, we were both amateurs. In digital physics, neural networks, even parallel computing, we never really knew what we were doing. But the things that we studied were so new that no one else knew exactly what they were doing either. It was amateurs who made the progress.

Telling The Good Stuff You Know

Actually, I doubt that it was "progress" that most interested Richard. He was always searching for patterns, for connections, for a new way of looking at something, but I suspect his motivation was not so much to understand the world as it was to find new ideas to explain. The act of discovery was not complete for him until he had taught it to someone else.

I remember a conversation we had a year or so before his death, walking in the hills above Pasadena. We were exploring an unfamiliar trail and Richard, recovering from a major operation for the cancer, was walking more slowly than usual. He was telling a long and funny story about how he had been reading up on his disease and surprising his doctors by predicting their diagnosis and his chances of survival. I was hearing for the first time how far his cancer had progressed, so the jokes did not seem so funny. He must have noticed my mood, because he suddenly stopped the story and asked, "Hey, what's the matter?"

I hesitated. "I'm sad because you're going to die."

"Yeah," he sighed, "that bugs me sometimes too. But not so much as you think." And after a few more steps, "When you get as old as I am, you start to realize that you've told most of the good stuff you know to other people anyway."

We walked along in silence for a few minutes. Then we came to a place where another trail crossed and Richard stopped to look around at the surroundings. Suddenly a grin lit up his face. "Hey," he said, all trace of sadness forgotten, "I bet I can show you a better way home."

And so he did.

Visit the Front Page or Subscribe to our Blog

NEXT SEMINAR

Coco Krumme - The False Promise of Optimiza...

PREVIOUS SEMINARS

Bette Adriaanse, Chelsea T. Hicks - Radical Sha...
Members of Long Now - Long Now Member Ig...
- The Climate Parables: Reporting from the Fut...
Becky Chambers, Annalee Newitz - Resisting D...
Ryan Phelan - Bringing Biotech to Wildlife Cons...
Jenny Odell - Saving Time: Discovering a Life B...

LATEST BLOG POSTS

The Three-Century Lifespan of the Modern Bee
The Commodification of Air
[A Composition of Matter]
Getting in Touch with "Your Future Self"
The Future Of Invasive Species Mitigation
Shining a Light on the Digital Dark Age
The River Twice
Digital Avatars and Our Refusal to Die
Three Poems

ABOUT LONG NOW

The Long Now Foundation was established in 01996* to foster long-term thinking and responsibility in the framework of the next 10,000 years. More »

Twitter Facebook Flickr RSS

 Top of Page