Introduction to LogoWorks

Marvin Minsky

March 14 1984

Published in *LogoWorks: Challenging Programs in Logo,* Cynthia Solomon, Margaret Minsky, and Brian Harvey, eds. McGraw-Hill 1986

Adults worry a lot these days. Especially, they worry about how to make other people learn more about computers. They want to make us all "computer-literate." Literacy means both reading and writing, but most books and courses about computers only tell you about writing programs. Worse, they only tell about commands and instructions and programming-language grammar rules. They hardly ever give examples. But real languages are more than words and grammar rules. There's also literature -- what people use the language for. No one ever learns a language from being told its grammar rules. We always start with stories about things that interest us. This book tells some good stories -- in LOGO.

The trouble is, people often try to explain computers the same ways they explain ordinary things -- the way they teach arithmetic by making you learn "tables" for adding and multiplying. So they start explaining computers by telling you how to make them add two numbers. Then they tell you how to make the computer add up a lot of numbers. The trouble is, that's boring. For one thing, most of us already hate adding up numbers. Besides, it's not a very interesting story.

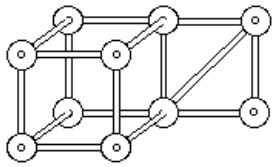
You can't blame teachers for trying to make numbers interesting. But -- let's face it -- numbers by themselves don't have much character. That's why mathematicians like them so much! They find something magical about things which have no interesting qualities at all. That sounds like a paradox. Yet, when you think about it, that's exactly why we can use numbers so many different ways! Why is it that we get the same kind of result when we count different kinds of things -- whether we're counting flowers or trees or cars or dinosaurs? Why do we always end up the same -- with a number? That's the magic of arithmetic. It wipes away all fine details. It strips things of their character. The qualities of what you count just disappear without a trace.

Programs do the opposite. They make things come to be, where nothing ever was before. Some people find a new experience in this, a feeling of freedom, a power to do anything you want. Not just a lot -- but anything. I don't mean like getting what you want by just wishing. I don't mean like having a faster-than-light spaceship, or a time machine. I mean like giving a child enough kindergarten-blocks to build a full-sized city without ever running out of them. You still have to decide what to do with the blocks. But there aren't any outside obstacles. The only limits are the ones inside you.

Myself, I first had that experience before I went to school. There weren't any programs yet, but we had toy construction-sets. One was called TinkerToy. To build with TinkerToy you only need two kinds of parts -- just Sticks and Spools. The Spools are little wooden wheels. Each has one hole through the middle, and eight holes drilled into the rim. The Sticks are just round little sticks of various lengths, which you can push into the spool-holes. The sticks have little slits cut in their ends, which make them springy when they're pressed into the holes, so they hold good and tight.

What's strange is that those spools and sticks are enough to make anything. Some spools are drilled with larger holes, so sticks through them can turn. You can make Towers, Bridges, Cars, and Bulldozers. Windmills. Giant animals. You can put wheels on your cars and make bearings for pulleys and gears, to make them do more interesting things. You have to make the gears yourself: just stick eight sticks into a spool. They work, though not too well, and always go click-click-click when they turn.

The sticks are cut to two kinds of lengths. One series of lengths come in the ratios One, Two, Four, and Eight. The other lengths are cut so that they fit across the diagonals of squares made from the first series of sticks. Thus, all the lengths are powers of the square root of two—and this means that you can also use the first kind as diagonals for squares made with the second series of sticks. You can use this to build sturdy cross-braced structures.



The secret is in finding out how much can come from so few kinds of parts. Once, when still a small child, I got quite a reputation. My family was visiting somewhere and I built a TinkerToy tower in the hotel lobby. I can't recall how high it was, but it must have been very high. To me it was just making triangles and cubes, and putting them together. But the grown-ups were terribly impressed that anyone so small could build anything so big. And I learned something, too -- that some adults just didn't understand how you can build whatever you want, so long as you don't run out of sticks and spools. And only just this minute while I'm writing this, I realize what all that meant. Those adults simply weren't spool-stick-literate!

When my friend, Seymour Papert, first invented LOGO, I had the same experience again. LOGO has some things like sticks—except that their computer commands: a stick 100 units long is called "FORWARD 100". LOGO also has things like spools: "RIGHT: 90" starts a second stick at right angles to the last one you drew. I recognized old building-friends at once.

Making LOGO programs is a lot like building with construction toys -- but it's even better. You can make drawings of things and structures, but you can make procedures, too. You can make them use words. You can make things change their forms. And you can make them interact: just make the properties of some of your objects depend on some features of other objects. As toys, those programs have their faults: you can't take LOGO cars outside and roll them down a real hill -- but, in exchange, their parts don't get loose and fall out and get lost. And the basic experience is still there: to see how simple things can interact to make more wonderful things.

LOGO started many years ago and several writers of this book were children when they helped us find new things to do with it. I'm very pleased to write this introduction now, recalling what a great adventure this has been and knowing, too, that it has just

begun.

There were other good construction toys, like Erector-Set and Meccano. Erector sets were just great. They have many kinds of parts, but the basic ones are metal strips with many holes, and different kinds of angle brackets. You get a millions little screws and nuts to put them together with. They also give you long, steel shafts, which fit through the holes just large enough to let them turn freely. And there are gears and pulleys to attach to the metal shafts, so you can make complicated things that really work.

The Meccano sets, made in England, were even better. It had real brass gears that could more smoothly mesh. When I was older, I built one of the very first modern, remote-controlled robots, using parts of a Number 10 Meccano set, and using ideas invented at MIT's first computer research laboratory in the 1940's. And, speaking of building computers, some of the people in this book once built a real, honest-to-goodness computer out of TinkerToy parts. It could actually compute the moves to play the game called Tic-Tac-Toe. It actually worked, and is now in a museum somewhere. It was made of spools and sticks—and also some string and, since the truth must be told, they hammered in some little brass nails to keep the sticks from falling out. It took about 100 sets, and was too big to fit in your room.

The golden age of construction-sets came to its end in the 1960's. Most newer sets have changed to using gross, shabby, plastic parts, too bulky to make fine machinery. Meccano went out of business. That made me very sad. You can still buy Erector, but insist on the metal versions. Today the most popular construction set seems to be LEGO -- a set of little plastic bricks that snap together. LEGO, too, is like LOGO -- except that you only get RIGHT 90. It is probably easier for children, at first, but it spans a less interesting universe, and doesn't quite give that sense of being able to build "anything." Another new construction toy is FischerTechnik, which has good strong parts and fasteners. It is so well made that engineers can use it. But because it has so many different kinds of parts, it doesn't quite give you that LOGO-like sense of being able to build your own imaginary world.

About the time that building-toys went out of style, so did many other things that clever kids could do. Cars got too hard to take apart -- and radios, impossible. No one learned to build much any more, except to snap-together useless plastic toys. And no one seemed to notice this, since sports and drugs and television-crime came just in time. Perhaps computers can help bring us back.

After you've built something with a "real" construction-set, you have to take it all apart again -- or you won't have enough parts for the next project. With programs, you can keep them on your disc and later get them out and build them into bigger ones. This year, you might run out of memory -- but that won't be a problem for your future children, because memory will soon be very cheap. What's more, you can share your programs with your friends -- and still have them yourself! No emperor of ancient times could even dream of that much wealth. Still, many adults just don't have the right kinds of words to talk about such things – or the kinds of ideas that could help them think of them. They just do not know what to think when little kids converse about "representations" and "simulations" and "recursive procedures." Be tolerant. Adults have enough problems of their own.

To understand what computers are, and what they do, you shouldn't listen to what people say about those "bits" and "bytes" and binary decisions. I don't mean that it isn't true. Computers are indeed mostly made of little two-way switches. But everyone is simply wrong, who says that this is what you need to understand what computers can do. It's just as true that houses can be made from sticks and stones -- but that won't tell you much about architecture. It's just as true that animals are mostly made of Hydrogen, Carbon, Oxygen, and Nitrogen -- but that won't tell you much about Biology.

> *A Martian szneech once mindlinked me; it wanted to know what literature was. I told it how we make sentences by putting words together, and words by putting letters together, and how we put bigger spaces between words so that you can tell where they start and stop. "Aha," it said, "but what about the letters?" I explained that all you need are little dots since, if you have enough of them, you can make anything.*
>
> *The next time, it called to ask what tigers were. I explained that tigers were mostly composed of hydrogen and oxygen. "Aha," it said, "I wondered why they burned so bright." The last time it called, it had to know about computers. I told it all about bits and binary decisions. "Aha," it said, "I understand."*

COMPUTER PROGRAMS ARE SOCIETIES.

You really need two other facts to understand what computations do.

The first idea is that making a big computer program is building a larger process out of smaller ones. I suppose you could truthfully say that sculptors make large shapes from stuck-together grains of clay. But that shows what's wrong with the bits-and-bytes approach. No sculptor ever thinks that way, nor scientist, or programmer. An architect first thinks of shapes and forms, then walls and floors, and only last about how those will be made. And that's the second important idea that most people don't really understand-but people like LOGO users can:It doesn't matter very much what kinds of parts you start with! Even if we start with different kinds of computers, with different kinds of parts inside -- still, they mostly can be made to do the same things at the top level. In the same way, you can build a big windmill with either wooden sticks or metal beams. When you look closely, each part will be quite different. But at the top level, both windmills will have a base, a tower, and a propeller. The same is true for computers!

> *Any computer can be programmed to do anything which any other computer can do -- or which any other kind of "Society of Processes" can do.*

Most people find this unbelievable. How could it be that what computers do does not depend on how they're made? This was discovered about 50 years ago by an English scientist named Alan Turing. He showed that, just as sculptors needn't care about how grains of clay are shaped, programmers needn't care how all those small computer parts work! For example, LOGO programs start with simple "true-false" choices, which depend on one thing at a time. But then we can write a LOGO program to choose from a list of things -- and we could make that list include a thousand million billion things!

What Alan Turing showed would take too long to explain here and you can find the details in any good book about the theory of computers. But here's the general idea. When Turing was quite young, he realized that what a computer does only depends on the States of its parts -- and on the laws that change their states. Except for that, it doesn't matter how the parts are made. Then Turing asked what programs are -- and realized that you could think of programs as just sets of states -- or rather, ways to pre-

arrange how a computer will, later, change its States.

Next, Turing thought, suppose you have one computer X but wish that you had a different kind of computer, Y. Then why not make a program for X that re-arranges its States so that, from then on, they will act just like the States of the other computer, Y? If that were done then, from now on, X's behavior will look exactly like Y's -- to anyone watching from outside. Today, a programmer would say that X is "simulating" Y. Of course you have to pay a price for this. It won't work at all unless X has enough memory to hold a description of Y. And if X and Y are very different, then the simulated programs will run very slowly. But aside from that, Turing showed, any kind of computer can be programmed to simulate any other kind! That is why we can write special programs to make the same LOGO programs run on all the different computers in the world.

In fact, every Atari LOGO system contains just such a program. It was written by Brian Silverman and his friends. I'm sorry that you can't read it. The trouble is, it's not written in LOGO but in a machine-language buried deep inside your machine. But it's there, hiding out of sight, making your computer simulate a LOGO computer. The strange thing is that Alan Turing figured out how to do such things 50 years ago, in 1936, long before computers were even invented! How could he do that? He simulated them inside his head.

This must be the secret of those magical experiences I had, first with those construction sets and, later, with languages like LOGO. There's something "universal" about the ways that big things don't depend so much on what's inside their little parts. What matters is more how the parts affect each other – and less about what they are, themselves. That's why it doesn't matter much if money's made of paper or of gold, or houses out of boards or bricks. Similarly, it probably won't matter much if aliens from outer space had golden bones instead of ones of stones, like ours. People are missing something important, who don't appreciate how simple things can grow into entire worlds. They find it hard to understand Science, because they find it hard to see how all the different things we know could be made of just a few kinds of atoms. They find it hard to understand Evolution because they find it hard to see how different things like birds and bees and bears could come from boring, lifeless chemicals -- by testing trillions of procedures. The trick, of course, is doing it by many steps, each using procedures which have been debugged already, in the same way, but on smaller scales.

Why don't our teachers tell us that computers have such glorious concerns? Because most adults still believe the only things that computers can do are huge, fast, stupid, but useful calculations of arithmetic. And so those dreary practicalities of billion-dollar industries crowd out our dreams and fantasies of building giant mind-machines. yksnimnivram