COMPUTER RECREATIONS

Two-dimensional Turing machines and tur-mites make tracks on a plane

by A. K. Dewdney

"[Termites] . . . forge the complexion of a landscape like no other organism except man."

—WALTER LINSENMAIER, *Insects of the World*

Anyone who has ever seen a termite mound must have been impressed by the complex patterns of tunnels built by the industrious but mindless insects. Paradoxically, artificial forms of life that make termites look like geniuses can produce equally astounding creations. Take tur-mites, for example. They are squarish, cybernetic creatures that have the most rudimentary of brains. And yet as they move about on the infinite plane on which they live, they trace out strange patterns that appear to reflect an underlying intelligent design.

The tur-mites were inspired in part by Greg Turk, a graduate student at the University of North Carolina at Chapel Hill. For some time Turk has been experimenting with a special type of Turing machine, a construct that has long served as a basic model of computation. A Turing machine is usually assumed to operate on an infinite linear tape that is divided into cells. Turk, however, has studied Turing machines that operate on a kind of two-dimensional tape—essentially the same plane on which the tur-mites roam. Converting a two-dimensional Turing machine into a tur-mite is simple and painless: abstract rules are replaced straightforwardly by a neural network. Such a conversion highlights an important theme in the theory of computation: one computational scheme often turns out to be equivalent to another, seemingly unrelated, one.

Turing machines are named after the British mathematician Alan M. Turing, who first proposed them as a way to define computation. In effect, a Turing machine is the ultimate digital computing machine. It can compute anything that a modern computer can—as long as it is given enough time.

One can visualize a Turing machine as it is shown in the illustration on the opposite page: a black box equipped with a device that reads a symbol in a single cell of an infinitely long tape, writes a new symbol in the cell and moves the tape either forward or backward in order to examine the symbol in an adjacent cell. What is inside the black box? It does not really matter, as long as the box adheres strictly to a given table that lists what the Turing machine must do for every symbol read and for every one of the machine's possible "states." These may change with each cycle of operation. A cycle consists of the following three steps:

1. Read the symbol currently under the read/write device.
2. Look up the table entry given by the machine's current state and the symbol just read.
3. Write the symbol given by the table entry, move the tape in the direction indicated and enter the state shown.

Each table entry therefore has three parts: a symbol to be written on the current cell, a direction in which to move the tape and a state to enter.

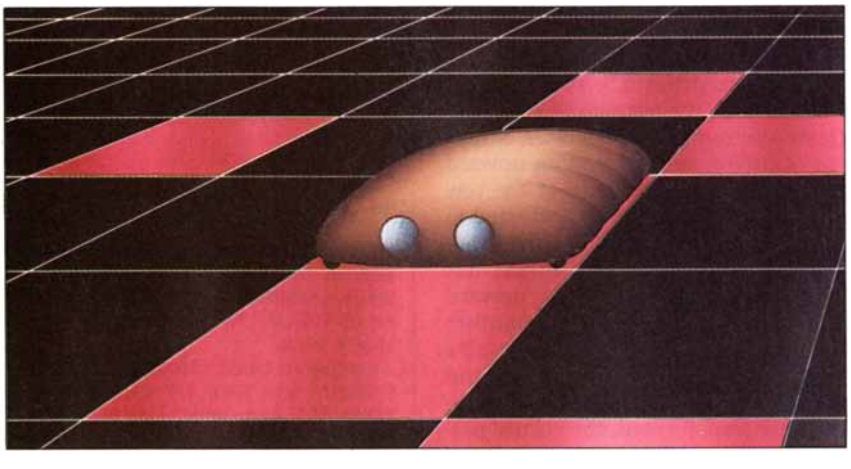
To a Turing machine the tape's motion is relative. One could just as easily arrange for the tape to remain fixed and the machine to move itself from cell to cell. In fact, once one contemplates the idea of moving the Turing machine and not its tape, it does not take much imagination to envision a two-dimensional "tape" on which the machine may move about freely.

Regardless of whether it has a one- or two-dimensional tape, a Turing machine's table is what ultimately determines its behavior. It is closely analogous to the program that controls a modern digital computer. In terms of computational capability, two-dimensional Turing machines are not more powerful than one-dimensional ones. They just have more interesting patterns of movement over the cells. The pattern shown in the illustration at the top of page 182, for example, was made by a single-state two-dimensional Turing machine. Its internal table is:

A	BLACK	RED
	(RED, LEFT, *A*)	(BLACK, RIGHT, *A*)

The machine's single state has been labeled *A*.

A slightly more complicated two-dimensional Turing machine that was discovered by Turk has two states,



A tur-mite occupies one square at a time

designated *A* and *B*, and it follows this internal table:

	BLACK	GREEN
A	(GREEN, LEFT, *A*)	(BLACK, FORWARD, *B*)
	(GREEN, RIGHT, *A*)	(GREEN, RIGHT, *A*)

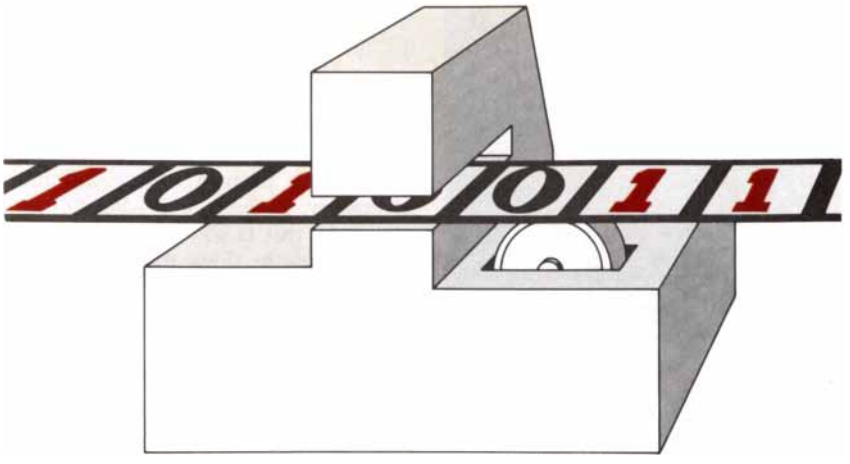
According to Turk, a two-dimensional Turing machine programmed with this table produces a marvelous spiral pattern. The machine creates "larger and larger patterned regions that are placed in orderly fashion around the starting point."

Any pattern generated by a two-dimensional Turing machine can be reproduced exactly by a tur-mite. A tur-mite's behavior, however, is not controlled by a mysterious black box. It is controlled by what can be loosely described as a brain. The fact that one can dissect and examine a tur-mite brain is what makes the creature so fascinating.

Judged only by its appearance and ethology, a tur-mite certainly is not fascinating. Its body is roughly square, so that it fits snugly into the squares that divide the infinite plane on which it lives. It has a flat bottom equipped with some form of locomotory apparatus. (I do not know what makes a tur-mite go, since I have never turned one over.) The apparatus enables the creature to rotate and to move exactly one square in the direction in which it happens to be facing. Actually, a tur-mite's face has no purpose except to let us know which way is forward; its "eyes" do not function. When a tur-mite changes direction, it merely swivels 90 degrees on its current square before moving to a new one.

Initially all the squares on the plane, including the one the tur-mite occupies, are black. Before it moves, however, a tur-mite may change the color of the square it currently occupies. (The tur-mite's color-changing organ is as mysterious as its locomotory apparatus.) A tur-mite that duplicates the pattern shown at the top of the next page, for example, must be capable of painting the square one of two colors (in this case, red or black). To produce the pattern shown at the bottom of the page, however, a tur-mite has to have more colors at its disposal.

How does a tur-mite know when to move or when to change the color of its square? Those actions are controlled by its brain, which consists of a collection of "neurodes," simplified versions of the neurons in our own brain. A neurode receives signals along fibers that originate at sensors (which are found on a tur-mite's underside) or at other neurodes and sends signals along fibers to effectors (such as the tur-mite's locomotory apparatus or its color-changing organ) or to other neurodes.



The standard visualization of a Turing machine

A neurode fires (sends a signal along its output fiber) if the number of incoming signals equals or exceeds the neurode's threshold, which is given by the number written on the neurode. Otherwise, it does not fire. Because time in the tur-mite's world proceeds in discrete steps, all excitatory and inhibitory signals are sent or received in discrete steps as well.

To illustrate how a tur-mite actually makes a decision, I shall dissect the brain of two specimens [*see left and middle drawings on page 183*], both of which produce exactly the same pattern as that generated by the single-state two-dimensional Turing machine described above. The brain on the left contains two neurodes that are not connected to each other. Each neurode has just one input fiber and one output fiber. When the tur-mite's color sensor detects red, it sends a single signal to the left neurode, causing the neurode to fire. The neurode's output fiber splits into two parts, one going to the color effector (which then colors the entire square) and the other going to the locomotory apparatus (which then swivels the creature 90 degrees to the right and advances it one square in the new direction). On the other hand, when the tur-mite's color sensor detects black, it sends a signal to the right neurode, causing it to fire. The neurode's output, in turn, causes the tur-mite to paint the square red before turning and heading to the square on its left.

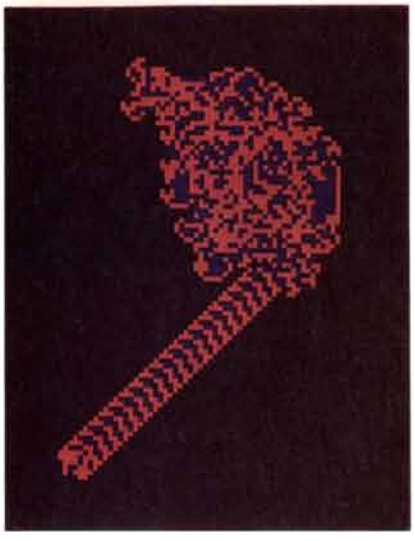
In short, when the tur-mite finds itself on a red square, it colors the square black and then moves one square to the right. And when the tur-mite occupies a black square, it changes the color of the square to red, then turns left and advances one square in its new direction.

The second tur-mite brain is more complicated, but it does exactly the same job as the first. It was derived by a method I shall presently describe. The two neurodes both have threshold 2; neither will fire unless it receives two input signals during the same time increment. Once the brain is set in motion, one neurode will always fire at each step in time.

The simple behavior embodied in the two neurode circuits just described results in the complicated image at the top of the next page: a red cloud of tiny squares from which an intricate structure extends straight to infinity. What causes that sudden sense of purpose in the tur-mite after what seems a great deal of pointless meandering? The answer has to do with the pattern of colored squares in the cloud. At a certain point, part of that pattern, in combination with the tur-mite's neurode-based rules, locks the creature into a repetitive sequence of moves that weaves the structure. (I wonder if any readers can discover the triggering pattern.)

Life is like that for tur-mites. Sometimes a seemingly random meandering turns into an almost deadly determinism. Of course, the appearance of randomness is purely illusory. All tur-mites are decidedly deterministic at all times.

Nonetheless, there are mysteries to be found in the tur-mite's world. Consider, for example, the pattern shown in the bottom illustration on the next page. The tur-mite that made that pattern is outfitted with four effectors that change the color of a square to



One of Turk's patterns

black, red, yellow or green. It abides by the following rules:

Square Color	Action
black	paint red, turn right
red	paint yellow, turn right
yellow	paint green, turn left
green	paint black, turn left

This tur-mite also has a very simple brain. It consists of four neurodes that are not interconnected. Each neurode executes one of the four behavioral rules in the manner of the first tur-mite's simplest brain. Turk is puzzled by the fact that this particular tur-mite produces a pattern having bilateral symmetry. Perhaps a reader can explain why this is so.

How exactly does one get a tur-mite from a particular two-dimensional Turing machine? The technique is actually quite simple. One merely replaces each entry of the machine's internal table with a threshold-2 neurode that receives input signals from a sensor for the color corresponding to the entry's column and perhaps from other neurodes as well. Each neurode's output fibers go to the effectors necessary to execute the moves and color changes listed in the table entry.

For example, suppose that a certain neurode corresponds to a table entry in a column labeled "red" and a row labeled "*B*." According to the conversion scheme, the neurode would have an input fiber from the sensor that detects red. If the table entry happened to be (black, left, *B*), then the neurode would send an output fiber to the effector that colors the occupied square black and to the effector that enables the tur-mite to execute left turns.

The various states of a particular Turing machine are realized by the connections between neurodes in a tur-mite brain. Because the table entry in the example requires the tur-mite to adopt state *B*, the neurode representing that entry would extend output fibers to each of the neurodes making up row *B* of the table.

In this context, such a neural network is nothing more (and nothing less) than a form of hardware embodying a behavioral table. Sample conversions are displayed schematically in the drawings in the middle and at the right on the opposite page. The one on the right shows how one would go about constructing the brain of the tur-mite that mimics the behavior of Turk's spiraling two-dimensional Turing machine.

It is fun to watch a tur-mite (or a two-dimensional Turing machine) wander about on a cellular plane. To follow the action, however, the reader must write a program that simulates the tur-mite's movements. How does one go from a table to a program?

Luckily, the process is nearly as simple as designing a tur-mite's brain. A program that I call TURMITE consults a Turing-machine table in the form of three separate arrays: *color, motion* and *state*. Each array is indexed by two variables, *c* and *s*. The variable *c* indexes the color of the present square, and the variable *s* indexes the Turing machine's (or the equivalent tur-mite's) current state. Because the indexes have to be assigned integer values, the colors and states used in the simulation must be numbered.

For example, the colors black and green can be assigned to the variable *c* by the numbers 1 and 2, respectively. Similarly, the states *A* and *B* can be designated respectively by the values 1 and 2 of the variable *s*. In this case, a simulation of a spiraling tur-mite would require the following arrays:

s \ c	1	2
1	2	1
2	2	2

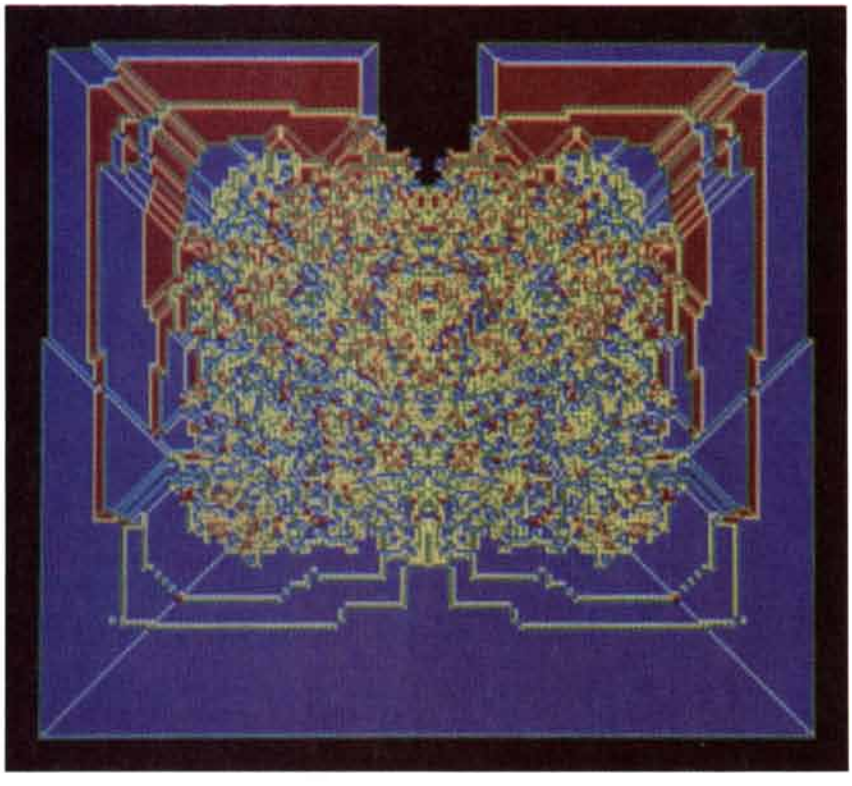
COLOR

s \ c	1	2
1	1	2
2	1	1

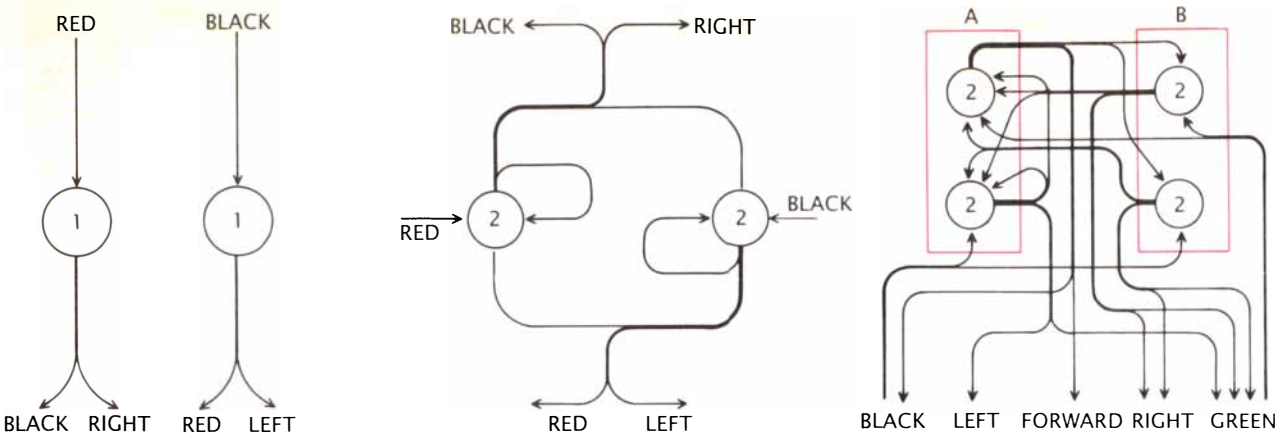
STATE

Directions of motion must also be coded in terms of numbers. Hence, forward, backward, left and right could be indicated respectively by the numbers 1, 2, 3 and 4, which are contained in the array *motion*.

The main purpose of TURMITE is to color small squares (perhaps individual pixels) on the computer's display screen that highlight a tur-mite's peripatetics. The program keeps track of



A multicolored tur-mite pattern



Three tur-mite brains, two of which* (left and middle) *do the same job

the displayed squares' colors in a two-dimensional array called *pattern.* Initially only one square is lit—the one lying at the center of the screen.

The value of *c* at any given time is provided by the entry in the array *pattern* corresponding to the turmite's current coordinates on the screen, say *i* and *j*. With *c* and *s* in hand, the program simply looks up the array entries *color*(*c,s*), *motion*(*c,s*) and *state*(*c,s*).

The program changes the color encoded in the entry in *pattern*(*i,j*) and then alters either *i* or *j,* depending on the value of *motion*(*c,s*). Here the program must translate the relative movement encoded in *motion* into an absolute movement by consulting another variable, *dir,* which contains the last direction moved: up, down, left or right. The final step in TURMITE's operating cycle consists merely of changing *s* to the number given by *state*(*c,s*). The rest can be left to the imagination and inventive skill of those readers who like to write their own programs.

While one is constructing a turmite's brain or simulating its behavior on a computer, it is interesting to reflect on the fact that, since tur-mites can carry out any computation a Turing machine is capable of executing, tur-mites can be just as powerful as some computers. If, as some claim, the human brain amounts to nothing more than a kind of digital computer, then some tur-mites could be just as smart as we are—if not smarter!

Simulated Evolution, the subject of the May column, drew an unusually heavy mail response. Several hundred readers requested copies of a detailed algorithm on which they could base their own version of the program. After all, it is not often that one has a chance to see pretend protozoa evolve into bacteria banqueters in an hour or less.

Michael Palmiter, the California high school teacher who developed the program, deserves the palm leaf for his creation. It was evidently an idea whose time had come. Swept into the simulated-evolution zeitgeist were a few readers who had independently developed programs that were remarkably similar to Palmiter's Simulated Evolution.

High school freshman Máté Sztipánovíts of Nashville, Tenn., won accolades at science fairs with a program that simulated oval bugs roaming on a two-dimensional space, looking for randomly distributed food. In Sztipánovíts's program, the medium surrounding the creatures exerts a drag that can be minimized through the evolution of streamlined shapes.

Christopher O'Haver of College Park, Md., also submitted a simulated-evolution program as a science-fair project. Unlike Palmiter's bugs, the organisms in O'Haver's program are stationary (more like algae than protozoa), absorb food continuously, grow and suffer predation.

Paul H. Deal of Moriarty, N.M., has developed a rather sophisticated evolution program that he has been distributing to educators. The genome of his creatures includes 13 genes that govern such characteristics as a creature's ability to feed on organic substrates, to absorb energy and to move (albeit somewhat feebly). Readers wanting to experiment with Deal's program may obtain it as shareware by writing Deal at P.O. Box 1398, Moriarty, N.M. 87035.

Among the readers who were able to set BUGS, my simplified version of Palmiter's program, in motion with only the spare description given in the column were Lewis V. Glavina of Burnaby, British Columbia, Ken Sheller of Bellevue, Neb., Jim Henry of De Kalb, Ill., and Albert H. Behnke of Boston, Va. Glavina, bothered by the amount of energy that bugs sometimes waste at the screen's boundary, gave his protozoa the power to bounce off the screen's sides. Sheller explicitly rewarded the gene that made the greatest contribution to food-gathering behavior. Finding it difficult to distinguish advanced bugs from their less evolved cohorts, Henry colored a bug according to its tendency to stay in the same place. Behnke endowed his bugs with a similar feature, causing a bug to change color as it changes directions.

Finally, a postscript on the April-fool anagram tangram gold scam. In the April column I quoted extensively from my correspondence with a shadowy character by the name of Arlo Lipof. Lipof maintained that he had driven down the price of gold by applying the Banach-Tarski theorem to make gold out of nothing. Recently I received an angry letter from the so-called International Gold Council in New York City in which I am held accountable for the "turmoil" and the "collapse of civilization" that might now result from the divulgence of Lipof's secret.

"For years the I.G.C. has made the Banach-Tarski paradox inaccessible to the general public.... We have always known that the apocalyptic reality of making more gold from less gold would have dire consequences for the international balance of world monetary systems."

FURTHER READING

MATHEMATICAL GAMES. Martin Gardner in *Scientific American,* Vol. 216, No. 3, pages 124–129; March, 1967.

MATHEMATICAL GAMES. Martin Gardner in *Scientific American,* Vol. 229, No. 5, pages 116–123; November, 1973.