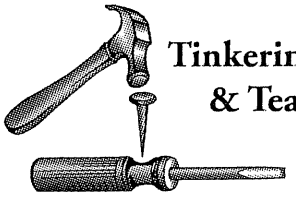
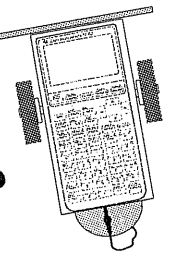


Tinkering, Toys, & Teaching Mathbots



by Jim Wilson

Many teachers and parents are concerned that calculators and computers will replace traditional teaching methods and materials. An effective way to address these concerns is to *begin* every learning activity (when conceptual foundations are being set) with traditional methods and materials and then *extend* the activity by using any available technologies.

Paper and pencil are hard to beat as conceptual development tools. It's been my experience that the technology-based tools often work best at the application or problem-solving level where concept mastery is assumed.

I've written a paper and pencil activity (see *Mathbot* in this issue) to illustrate this point. In the activity, a simple mathematical process is defined, understood, and explored at the paper and pencil level. I used a variation of this activity in my mathematics classroom during the 1970s and 80s when the only technologies we had were graph paper and pencil. I've updated the activity to include technology extensions. I use the activity and extensions in the classes I currently teach.

As described in the *Guiding Documents* section of the activity, *Mathbot* addresses several algebra standards listed by the National Council of Teachers of Mathematics for grades three and above. Also included in the algebra standards for grade three and above is the reasoning standard that students should *make and investigate mathematical conjectures.*

Exploring *Mathbot* with paper and pencil is relatively slow and prone to human error. Students look at a small number of patterns and conjecture that the patterns extend forever. At this point, the programmable graphing calculator can extend the activity to eliminate human error and greatly extend the number of examples that support the conjecture (see *Mathbot, Discussion 13).*

Both the paper and pencil and programmable calculator explorations take place in the abstract, perfect environment of mathematics. Friction and gravity are forces found in the real world and these forces tend to destroy mathematical perfection when mathematics is applied in the real world. Robotics is a popular way to teach students how to "engineer around" the problems created when abstract mathematics meets friction, gravity, temperature, and numerous other physical realities. It's one thing to command a robot to move forward two feet but quite a larger problem to get the robot to actually do it.

This chart summarizes the transitions made as students work through the three levels of *Mathbots.*

Paper & Pencil →	Programmable Calculator or Computer →	Robot
learn & practice new concepts	apply learned concepts use or write programs	
look at a few examples	look at a large number of examples	solve problems in an "engineering" environment

The *Mathbot* activity is the paper and pencil level shown in the chart. I will now describe how to extend the activity with a programmable calculator and a robot.

A Texas Instruments Calculator Mathbot

Graphing calculators are used in many middle and high school classrooms. These calculators are programmable. The popular Texas Instruments graphing calculators (models 73, 82, and 83 plus) can easily be programmed to draw the patterns explored in the *Mathbot* activity.

Key the following program into a Texas Instruments 73, 82, or 83 Plus calculator.

[Note: Press STO key to make the -> symbol.]

PROGRAM:FORT

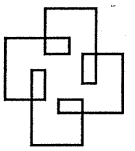
```
0->X:0->Y
-140->Xmin:140->Xmax
-80->Ymin:80->Ymax
0->Xscl:0->Yscl
Disp "ORDER"
Input N
Disp "SCALE"
Input Q
1->J:0->K
ClrDraw
Lbl 1
```

Continued

```
For(I,1,N)
For(V,1,Q*I)
X+J->X:Y+K->Y
Pt-On(X,Y)
End
J->T:K->J:-T->K
End
If X=0 and Y=0
Then
Stop
End
```

If you don't want to key in the program by hand, send an email to jawilson@fresno.edu and I will email you a copy of the program file and instruct you how to load it into your calculator.

This is the pattern the calculator program displays for an order 5, 4-cycle Mathbot (see the *Mathbot* activity for a definition of *order* and *cycle*). The scale factor was set at 15 (see the program) to enlarge the pattern.



I will also supply by email a detailed description of how my program works.

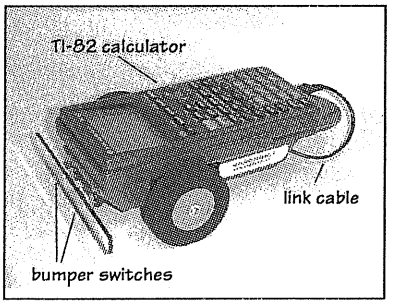
I spent several days tinkering with this program before getting it to work. You might know a simpler, better way to write this program. If so, please share it with me and other readers. I've also tinkered a program for a Mathbot that can make both right and left turns (see *Extension one* in the *Mathbot* activity). I call this Mathbot *Fortle*. Fortle patterns are more complex than FoRt patterns but they too can be analyzed and generalized. Again, I will supply this program and documentation upon request.

A Mathbot Robot

At a recent national science conference, I visited the Texas Instruments booth and watched a calculator scooting around inside a square-shaped corral that had been built on the floor.

The calculator was mounted on a chassis containing a pair of motor-driven wheels and the electronic circuit needed to control the robot. The front bumper of the robot activated one of two switches (or both) when the robot struck one of the walls of the corral. Depending upon which switch was depressed, the robot would back away from the barrier, turn left or right, and then start moving forward. Soon, it would hit the barrier again and repeat the process. The calculator was behaving just like a bug in a box (Volume XII, Number 5); spending most of its time near the edges of the square. If a door wide enough to let the robot pass through is opened in the wall, the robot will eventually escape from the corral.

I purchased one of the robots and have found it easy to use and maintain. The calculator slips easily in and out of the base mounted on the chassis of the robot.



The following program teaches the robot how to be a Mathbot. Be sure to use brackets { }, not parentheses ( ), where indicated in the program.

I've recently used the robot with students and found it easy to manage; students typed programs on their calculators and then took turns attaching their calculator to the robot and testing and debugging their programs.

[Note: Press STO key to make the -> symbol.]

prgmFORT1

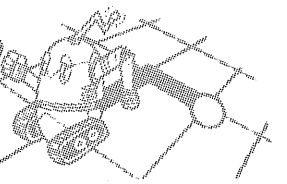
```
ClrHome                    prgmFORT1
Disp "ENTER ORDER "        Lbl 2
Input T                    T-> Q
Goto 2                     While Q ≥ 4
Lbl 1                      Q – 4->Q
For (P,1,W)                End
For (S,1,T)                If Q = 0
{122,100S}  -> L1          5 -> W
Disp 100*S                 If Q =1 or Q = 3
Send (L1)                  Then
Get (R)                    4 -> W
{121,140} -> L1            End
Send (L1)                  If Q = 2
Get(R)                     2 -> W
End                        Goto 1
End
Continued
```

I've also written the Fortle, left and right turn Mathbot for the calculator robot.

Norland Research sells the calculator robot and other robots over the Internet. For the latest pricing and product information visit their website at www.smallrobot.com/

In the next column I will tell the story of Leon Foucault and his famous pendulum. We will tinker two simple projects that demonstrate the principle that makes his pendulum work.

Mathbot



by Jim Wilson

Topic
Algebraic thinking

Key Question
What geometric and algebraic patterns are generated by a robot that follows a simple sequence of move-forward and turn-right commands?

Focus
Students will learn to convert a string of symbols into a geometric diagram. They will systematically collect, record, and analyze data to find the relationship between the number of symbols in the string and certain characteristics of the geometric pattern generated by the string.

Guiding Documents
Project 2061 Benchmark

- *Mathematics is the study of many kinds of patterns, including numbers and shapes and operations on them. Sometimes patterns are studied because they help to explain how the world works or how to solve practical problems, sometimes because they are interesting in themselves.*

*NCTM Standards 2000**

- *Describe, extend, and make generalizations about geometric and numeric patterns*
- *Represent the idea of a variable as an unknown quantity using a letter or a symbol*
- *Represent, analyze, and generalize a variety of patterns with tables, graphs, words, and, when possible, symbolic rules*
- *Relate and compare different forms of representation for a relationship*
- *Develop an initial conceptual understanding of different uses of variables*
- *Make and investigate mathematical conjectures*

Math
Algebra
- algebraic thinking
  - variable
  - generalization

Geometry and spatial sense
- right angles

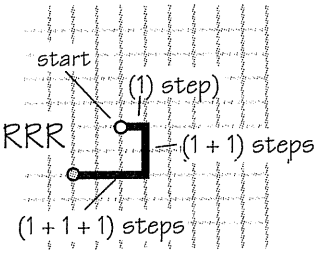
Number sense
- use of infinity (∞)

Integrated Processes
Observing
Collecting and recording data
Generalizing

Materials
For each student:
- pencil
- colored pencils, crayons, or makers
- student pages

Background Information
Consider an imaginary robot that can move forward 1, 2, 3, 4, 5, or *n* steps and then make a 90-degree right turn. A robot that can count and make right-angle turns is called a *Mathbot.* A simple programming language called FoRt controls a Mathbot. FoRt is short for **For**ward-**R**ight. The FoRt language is simply a string of Rs. The Mathbot executes the string of Rs, beginning at the left end of the string, incrementing step length by 1 unit, turning 90 degrees to the right, and after executing the last R of the string, looping back to the first R. The string RRR is interpreted as follows:

R means move forward 1 step, turn right 90°,
RR means move forward 1 step, turn right 90°, move forward (1 + 1) steps, turn right 90°.



RRR means move forward 1 step, turn right 90°, move forward (1 + 1) steps, turn right 90°, move forward (1 + 1 + 1) steps, turn right 90°.

The number of right turns in a string of Rs is called the *order.* The order of the string *RRR* is three. Executing the program one time is called a *cycle.* If the program is repeated two or more times (two or more cycles), the pattern may close with the Mathbot returning to its starting position. For some orders and cycles, the Mathbot wanders off towards a point at infinity, never to return to its starting position (see *Mathbot* page).

If the order *n* is a multiple of 4 *(n* divided by 4 leaves a remainder of 0), then the Mathbot wanders away from the origin never to return. If the remainder of *n* divided by 4 equals 1 or 3, the Mathbot returns to its starting point after four cycles of the program. If the remainder of *n* divided by 4 equals 2, the Mathbot returns to its starting point after two cycles.

The order and cycle data collected by students reveals a 4, 2, and 4, infinity pattern.

Order	1	2	3	4	5	6	7	8	9	10
Cycle	4	2	4	∞	4	2	4	∞	4	2

Management

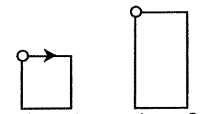
1. Pair students so that they can monitor and assist each other.
2. Encourage students to use a pencil to lightly sketch each pattern, check its accuracy, and then go over the pattern with a colored pencil or marker. A different color can be used for each cycle to highlight each pattern.

Procedure

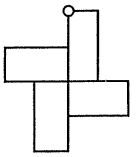
1. Make a transparency of the *Mathbot* page.
2. Using the transparency as a guide, explain the meaning of *order* and *cycle* for a FoRt program. Be sure to trace through the diagram for the program 4(RRR) at the bottom of the transparency.
3. Distribute three sheets of the *FoRt Patterns* graph paper to each student.
4. Distribute one *FoRt Patterns and Generalizations* page to each student.
5. Instruct the students to graph, in numerical order, the FoRt patterns for orders one through ten. Tell them to record the number of cycles that it takes to close each pattern. If any pattern doesn't close, tell them to leave that cycle entry blank until a class discussion takes place.

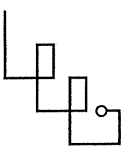
Discussion

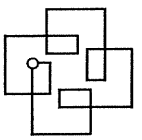
1. Have the students compare the patterns they drew with these patterns. (The patterns are not drawn to scale.)

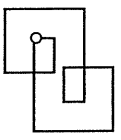

order = 1
cycle = 4

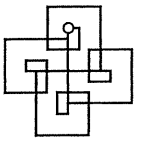
order = 2
cycle = 2

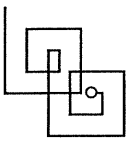

order = 3
cycle = 4

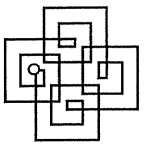

order = 4
cycle = ∞

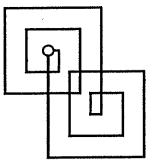

order = 5
cycle = 4


order = 6
cycle = 2


order = 7
cycle = 4


order = 8
cycle = ∞


order = 9
cycle = 4


order = 10
cycle = 2

2. Ask the students to compare the cycle data for orders 1, 2, 3, 5, 6, 7, 9, and 10 in their tables with the data in this table.

Order	1	2	3	4	5	6	7	8	9	10
Cycle	4	2	4		4	2	4		4	2

3. How would you describe the odd-numbered orders greater than one? [The patterns close and have four spiral loops.]
4. How would you describe the even-numbered orders greater than two? [The patterns close and have two spiral loops.]
5. How would you describe the even-numbered orders that are multiples of 4? [The patterns do not close. The Mathbot wanders off to infinity.]
6. The geometric patterns for orders 4 and 8 clearly indicate that these patterns do not close. What numbers do you think should be entered in those columns? What are your reasons for choosing the numbers? [It would not be incorrect to enter any whole number greater than two in the "cycle" columns for these orders. The pattern simply ends after that number of cycles. (This is one of those cases in mathematics where a unique, whole number answer can't be counted or computed. Try to convince the students that the symbol for infinity (∞) is the best answer for the "order" columns that are a multiple of four. One reason for using ∞ is that the pattern is only "completed" after an infinite number of loops. Another reason for the use of ∞ is that it makes it possible to describe the pattern as a "4, 2, 4, ∞" pattern.)]
7. Write this table on the board or overhead projector. Ask students to predict the cycles for each order in the table.

Order	11	12	13	14	15	16	17	18	19	20
Cycle										

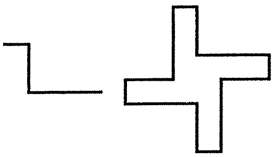
8. How can your predictions can be verified? [Graph each one to determine its pattern and cycle.] (The tedium of checking higher orders can stimulate writing a calculator or computer program (see *Extensions 2).*)
9. What are the variables for a Mathbot? [Let the symbol *n* represent the order and the symbol *C* represent the cycle of a FoRt pattern.]

10. How would you write an equation for *P*, the FoRt pattern? [$P = C \times n$]
11. How is a FoRt program like "place value?" How is it different? [The number of forward steps R represents depends on the place the R appears in the string of Rs.]
12. Can you find an order that requires 5 cycles to close? Explain.
13. Do the Mathbots that wander off to infinity all go in the same direction? Explain. [Yes, as long as every program starts in the same direction. (Draw spiral of 4, 8, and 12 steps to verify this conclusion.)]
14. In mathematics, a *conjecture* is a statement that is consistent with the known data, but has never been proven. Neither has it ever been shown to be false. Discuss why the results of this activity actually make a conjecture about the behavior of FoRt patterns. Challenge students to find one counterexample. If any student can do so, the conjecture is disproved.
15. The *irobot* corporation (www.irobot.com) has just introduced a new home robotic vacuum cleaner called *Roomba*. Roomba costs $199 and is the first robot designed for home use that actually performs a useful job. Roomba's vision is limited so it starts at the center of a room and moves in widening spirals until it reaches a wall. If Roomba strikes a chair leg or other obstacle, its program will help it navigate around the obstacle. Roomba contains a memory that tells it when it has covered every part of the room. At that point, it shuts itself down. Other leading vacuum cleaner manufacturers are also working on robotic models.

    Which mathbot orders would make good Roomba programs? Explain. [Answers will vary.]

Extensions

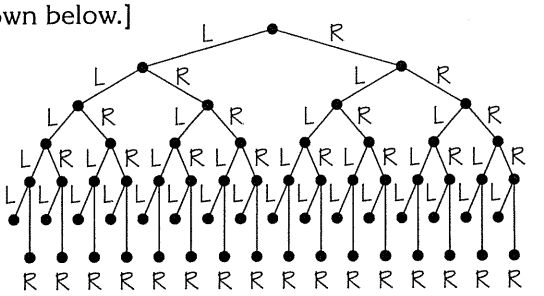
1. What patterns would be generated by a Mathbot that could make both right and left turns? For example, the Mathbot program RLR generates the figure on the left and four cycles (4(RLR)) closes the geometric figure.



RLR 4(RLR)

[Note: This Mathbot is called *Fortle*. Fortle programs also generate interesting geometric patterns. There is also a relationship (albeit one different from FoRt) between the order and cycle of a particular program.

There isn't just one Fortle program for order five; there are 32 since there are 32 distinct ways a string containing five Rs and Ls can be formed. To identify all 32 strings, build a tree like the one shown below.]



2. The following program is for the Texas Instruments programmable graphing calculators, Models 82 and 83.

[Note: Press STO key to make the -> symbol.]

PROGRAM:FORT

```
0->X:0->Y                    For(I,1,N)
-140->Xmin:140->Xmax         For(V,1,Q*I)
-80->Ymin:80->Ymax           X+J->X:Y+K->Y
0->Xscl:0->Yscl              Pt-On(X,Y)
Disp "ORDER"                 End
Input N                      J->T:K->J:-T->K
Disp "SCALE"                 End
Input Q                      If X=0 and Y=0
1->J:0->K                    Then
ClrDraw                      Stop
Lbl 1                        End
```

Continued

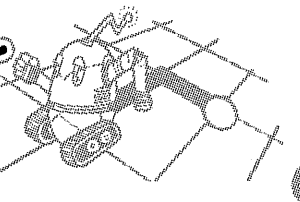
Enter this program into one calculator and then use the link cable (supplied with each calculator) to load the program into the other calculators.

(COMBINATION, cont. from page 27.)

After students have played several rounds in groups, close the activity with a time of class discussion in which students share about their experiences. Encourage them to describe any strategies they developed, and have them share the questions that eliminated several numbers at once. Determine which student(s) were able to discover a combination in the fewest number of guesses and have them share about their games with the class.

I hope you and your students find this experience valuable. As always, I welcome any input you have on this, or other *Primarily Problem Solving* activities. You can contact me at AIMS or by email: meyoungs@fresno.edu.

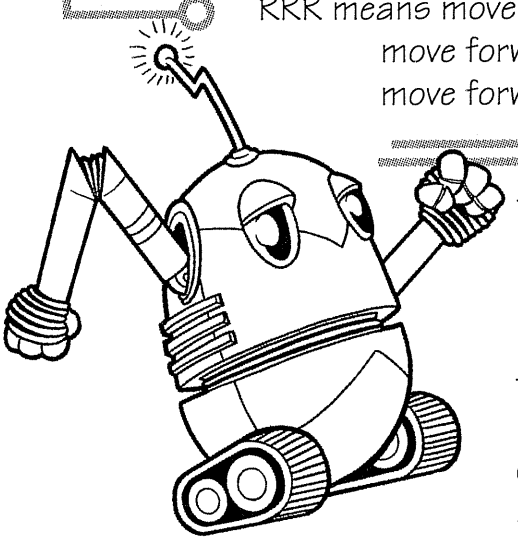
Mathbot



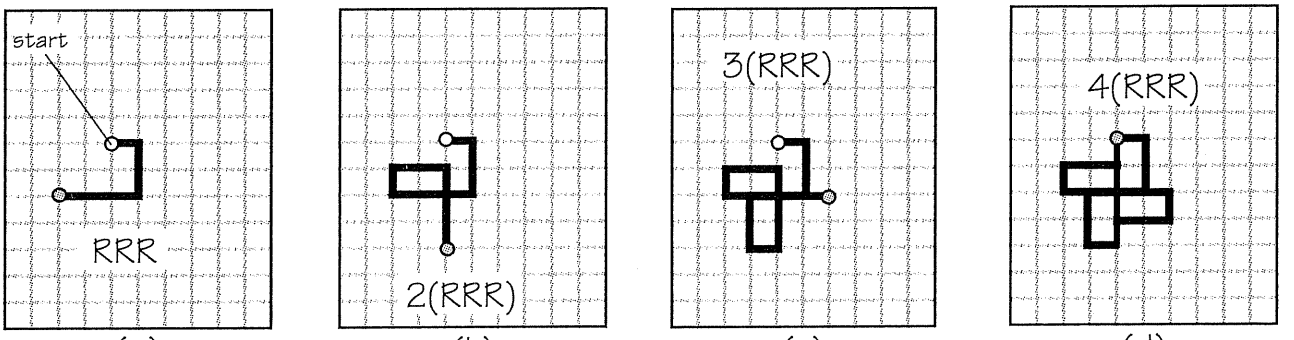
Imagine a Mathbot that can move forward any whole number of steps and can then turn 90 degrees to the right. The Mathbot is controlled by a simple *programming language* called FoRt. (FoRt is short for **Fo**rward-Righ**t**).

The FoRt language is simply a string of Rs. The Mathbot executes the string of Rs, beginning at the left end of the string, incrementing step length by 1 unit, and loops back to the first R after executing the last R of the string. Here is an example.

R means move forward 1 step, turn right 90 degrees.
RR means move forward 1 step, turn right 90 degrees,
move forward (1 + 1) steps, turn right 90 degrees.
RRR means move forward 1 step, turn right 90 degrees,
move forward (1 + 1) steps, turn right 90 degrees,
move forward (1 + 1 + 1) steps, turn right 90 degrees.



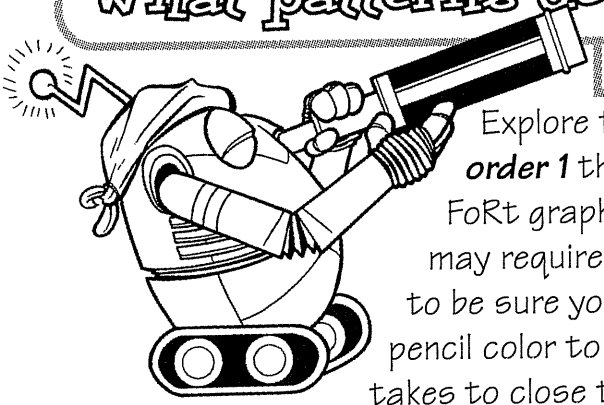
The path the Mathbot follows as it executes the FoRt program RRR is shown in diagram (a) below. If the program (RRR) is run two times, the path the Mathbot traces is shown in diagram (b). Diagram (c) shows the path for three program runs. After the fourth run (d), the Mathbot is back at its start position and the path has closed. The number of Rs in a FoRt program is called its *order*. The FoRt program 4(RRR) generates a closed pattern. The number of times a FoRt program is run to close its pattern is called its *cycle*. Do all FoRt programs generate closed patterns? This, and other questions, can be answered by exploring with graph paper and a pencil.



(a) (b) (c) (d)

FoRt Patterns and Generalizations

What patterns do FoRt programs produce?



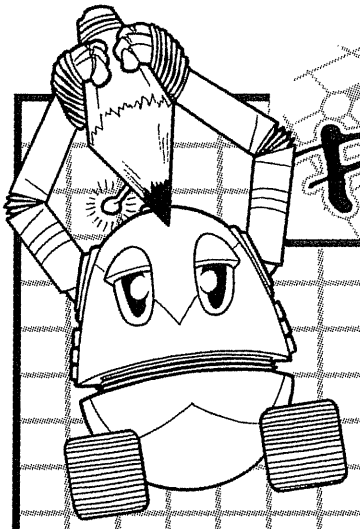
Explore the patterns generated by a Mathbot running ***order 1*** through ***order 10*** FoRt programs. Several sheets of FoRt graph paper will be needed. The higher order programs may require a single sheet. Trace each path lightly at first to be sure you've correctly drawn the pattern. Use a different pencil color to graph each cycle. Record the number of cycles it takes to close the pattern for each order.

Order	1	2	3	4	5	6	7	8	9	10
Cycle										

What's my pattern?

What's the general relationship between the order of the FoRt program and the number of cycles in the pattern generates?

What's different about the patterns for order 4 and order 8?



Felt Patterns