Filling Squares: Variations on a Theme

ALAIN SENTENI

Many Logo dialects provide a FILL primitive that children are very fond of: the turtle can fill in whole areas of the screen with colour. To try doing the same thing when Logo does not have this feature leads to the discovery of a whole family of procedures.

This set of procedures covers a very wide range of Logo understanding: from very basic techniques used by 9 to 11 year old children to recursive procedures using coordinates [Ross, 1983; Solomon et al, 1986] or to complex procedures with many recursive calls [space filling curves in Abelson and diSessa, 1981].

One of the goals of this article is to analyze the mathematical or programming concepts underlying a few simple algorithms that can be used to fill in squares. Exploring such themes can be quite helpful for teachers using Logo in a classroom. It gives a first idea of the different directions that children can take when given the same starting instruction. It enables the teacher to know what is or could be happening without having to be restrictively directive. Finally, we may say that it gives the teacher a chance to anticipate the nature of the children's discovery process as in a pre-planned adventure and to act as a resource person, helping when necessary.

During the Fall of 1985 two different groups of children were given the task of filling a simple square with colour. The second purpose of this article is to describe and analyze the solutions given by these children.

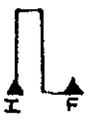
The first group of ten fifth and sixth graders belongs to a multi-age class, involved for two years in a research project about learning styles in a Logo environment (GREL: Groupe de Recherche sur les Environnememts Logo, UQAM, 1985). Filling a square was a task preliminary to drawing a checker board. No constraint was imposed on the dimension of the square, so the variable aspect is ignored. Children worked alone on the computer with an adult observer whose interventions were aimed at helping to achieve the task rather than teaching a specific content. We used French Apple Logo.

The second group of four 10-11 year old children had already been introduced to the notion of variable [Hillel and Samurçay, 1985]. The children were asked to fill a square of any size, but they could start with a fixed dimension and, once a filling algorithm had been found, generalize the process by introducing a variable. Children worked in pairs, as they are used to, and the interventions were more specifically focused on the use of the variable in REPEAT loops. The dialect used was Apple Logo II, without, of course, mentioning the existence of the FILL primitive.

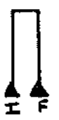
No attempt was made to introduce or to encourage the use of tail recursion. The syntax of the stop rule was likely beyond the reach of most of the children involved. Nevertheless the question was not avoided. Recursive templates were given, with some success, to four children who had been in the first group the year before as they were starting their third year of Logo. (The results of this experiment are not described in this paper.)

Thus only four methods of filling a square are discussed here. Comments on the children's solutions which use one of the first two methods are added to show the variations that may arise.

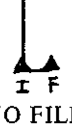
Method 1: "Back and forth"
The most natural process for filling a square seems to be the one often used by children when they do it by hand. The turtle can be told as follows:



```
TO FILLSQ.1
REPEAT 25 (FD 50 RT 90 FD 1 RT 90 FD 50 LT 90 FD 1
LT 90)
END
```



```
TO FILLSQ.2
REPEAT 50 (FD 50 RT 90 FD 1 RT 90 FD 50 RT 180)
END
```



```
TO FILLSQ.3
REPEAT 50 (FD 50 BK 50 RT 90 FD 1 LT 90)
END
```

The three procedures are almost equivalent. The constraint of filling squares of variable dimensions brings out the articulation between the input to REPEAT, the amount of FORWARD and the variable in the title. Filling a square with side 20, one with side 30, one with side 100, may be a good progression to introduce variables. The procedures could be called FILLSQ20, FILLSQ30, FILLSQ100, and children can explore the relation between respectively 20, 30, 100 and the numbers in the procedures. These fixed procedures can then be turned quite naturally into a general one by making the implicit variable in the title an explicit one:

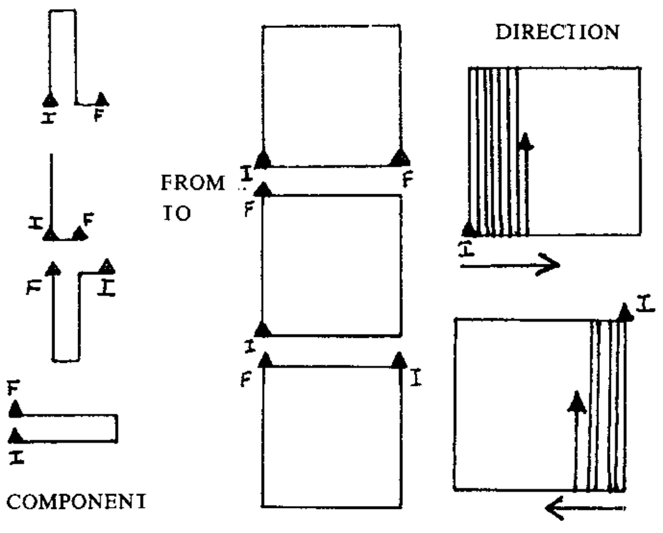
```
TO FILLSQ.4 :LENGTH
REPEAT :LENGTH (FD :LENGTH BK :LENGTH RT
90 FD 1 LT 90)
END
```

EXAMPLES OF CHILDREN'S ALGORITHMS

Eight out of ten children from the first group used a "back and forth" method. But a closer look at their work shows variations in the algorithms produced and in the way they were generated.

The initial square to be filled is usually absent in the final product. The frame is generated by the filling algorithm. This brings forward a "saving" dimension in the solution—that is, differences in decisions about what is worth saving: time spent typing, time spent planning on the paper what to type, the turtle's efforts. It is interesting to note that, in this case, the less the children have to type, the more the turtle has to travel. This can be generalized to many programming shortcuts such as the REPEAT command or the use of recursion. In general, children dislike to impose superfluous work on the turtle but they do not mind typing. The use of procedures as saving devices makes programming refinements unnecessary. Once a procedure has been defined, the effort in using it stays the same whether it is a long and awkward procedure or a very short, "elegant" one (the idea of elegant programming might not be the same for adults and children). Therefore, teaching programming is a task that differs from teaching *through* programming and it would require a different type of situation to create the need for programming subtleties.

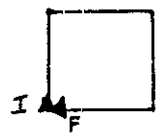
What was repeated to fill in the square
in the children's solutions
(Initial turtle-state, Final turtle-state)



The turtle's path can be vertical or horizontal and it can move from either left to right or right to left. The initial and final turtle-states are always different. The position of the filled square relative to the intial turtle state is always "on top" except in one case. These differences are related to the building steps and to the accidents that occur during the building process. A "crenel" can become a "hook" because some commands were changed when included in the REPEAT loop. A "crenel" which was horizontal in direct mode became vertical after editing because the initial turtle-state was not checked.

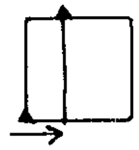
B's solution is a good example of how planning can be mixed in with adapting to accidents and patching. B prefers to give up his plan rather than erase and start anew.

First B draws a square that is not state transparent:



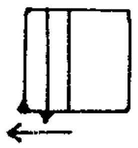
```
FD 10 FD 30 RT 90
FD 40 RT 90
FD 40 RT 90
FD 40
```

then he backs up, making a line that is not close enough to the side:



```
BK 5 RT 90 FD 40
```

then he comes back to fill in the space between the last line and the side:

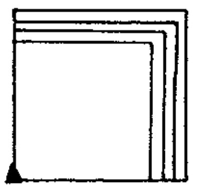


```
LT 90 FD 2
LT 90 FD 40
```

When he realized that this solution was actually working, he adopted it for the rest of the task. He will go a bit too far, will come back to fill in the leftover space, and so on. Here, fantasy and originality can be interpreted as an effort to get back in balance. Logo allows for patching, which is using accidents, successful or unsuccessful attempts, as part of the building process.

Method 2: Using the subprocedure SQUARE :N

```
TO FILLSQ.4
SQUARE 50
SQUARE 49
SQUARE 48
SQUARE 47
....
SQUARE 2
SQUARE 1
END
```



This works very well. Unfortunately Logo does not understand repeated dots. It can become a very fastidious endeavour unless one really enjoys typing.

In fact techniques are missing if one wants to take a shortcut by using a REPEAT command. It is rather tempting to ask:

```
TO FILLSQ 5
REPEAT 50 (SQUARE 50
could you please do the next square a little smaller?)
END
```

But it isn't easy to go from this kind of "wishful" thinking to the correct formalization of the idea:

```
TO FILLSQ.4 :LENGTH
REPEAT 50 (SQUARE :LENGTH MAKE "LENGTH
:LENGTH — 1)
END
```

The MAKE command is a difficult one. This will be the first time a child will be talking in Logo to something abstract, that isn't the turtle. Furthermore, the syntax is complex.

AN EXAMPLE OF THE CHILDREN'S SOLUTIONS
Only one child in the first group used this method, and he discovered it by chance. When A had to fill a square, he was really discouraged. He didn't know how to get started, and it was suggested he start by drawing a square. He had very little energy that day and the square was a small one:

```
REPEAT 4 (FD 3 RT 90)
```

The thickness of the line made this square look almost full making A feel a little closer to the target. He kept drawing squares around this core, the way somebody might roll a ball of thread:

```
TO VYW130 (secret cypher for full square)
REPEAT 4 (FD 2 RT 90)
REPEAT 4 (FD 3 RT 90)
REPEAT 4 (FD 4 RT 90)
........
REPEAT 4 (FD 31 RT 90)
END
```

The typing was the same throughout and shows the pleasure he had in repeating forever something he had discovered by himself.

In the second group, B and R used the same process but started with the largest square.

They spent Session 1 writing:

```
TO FSQ
SQ 100
SQ 99
SQ 98
.....
SQ 1
END
```

They had been asked to write a general procedure that would fill a square of any dimension, but not time was left for generalization.

As Session 2 was a week later, they had likely forgotten most of what they had already done They began with:

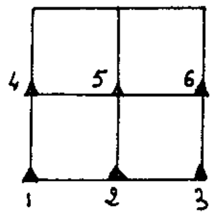
```
TO SQWQ
REPEAT 100 (FD 50 RT 90 FD 2)
RT 90
END
```

which makes the turtle turn 25 times around the same 52×52 square. They wanted to stick to the same idea (many squares in the bottom left corner are filling the big square) but make it shorter. The problem is that one can use a REPEAT command that automates the initial process only when one has a technique to make the squares decrease

After discussing it, they changed to:

```
REPEAT 40 (FD 40 BK 40 RT 90 FD 1 RT 90)
```

This fills a square using a back and forth method and leaves the turtle at the bottom right corner when starting from home. It is a lucky ending because when repeated it makes a long filled rectangle. R suggests reusing this process inside a procedure which would draw bigger squares They agree on:



```
TO FSQ1

two squares dimension 40

REPEAT 40 (FD 40 BK 40 RT 90 FD 1 LT 90)
REPEAT 40 (FD 40 BK 40 RT 90 FD 1 LT 90)

interface to the upper part

FD 40 LT 90 FD 80 RT 90

two squares dimension 40

REPEAT 40 (FD 40 BK 40 RT 90 FD 1 LT 90)
REPEAT 40 (FD 40 BK 40 RT 90 FD 14LT 90)

END
```

Building an 80×80 square is the first step towards controlling dimensions The intuition of a recursive process is also present. B wants to use the FSQ1 procedure inside another one, R mentions that it could be done in the same one. The result is:

```
TO FSQ
REPEAT 40 (FD 40 BK 40 RT 90 FD 1 LT 90)
REPEAT 60 (FSQ)
END
```

Let's assume that REPEAT 60 is there just to make sure. They realize it is too much and change 60 to 6. But it does not change the final result, which is a long strip.

At the beginning of session 3, the initial square:

```
TO FSQ
REPEAT 40 (FD 40 BK 40 RT 90 FD 1 LT 90)
END
```

was easily changed into:

```
TO FSQ :LENGTH
REPEAT :LENGTH (FD :LENGTH BK :LENGTH
RT 90 FD 1 LT 90)
END
```

The computers are close to one another and each pair of children can see what the other one is doing. There is some competition in the air about who will be the first to fill in the whole screen. B and R are puzzled by N and E using a MAKE command they were taught. They return to the first FSQ procedure and try:

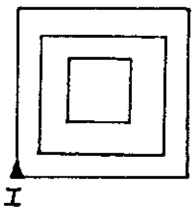
```
TO FSQ :LENGTH
REPEAT :LENGTH (SQ :LENGTH)
END
```

but it does not work, so they ask for a tool to make the squares decrease, as the other pair seems to have one. The final procedure is:

```
TO FSQ4 :LENGTH
REPEAT :LENGTH (SQ :LENGTH MAKE "LENGTH
:LENGTH —1)
END
```

This use of the MAKE command came after the problem was already solved by another method; it was not a very spontaneous discovery. Aside from that, the MAKE command was applied a few weeks later to another task and the children got quite mixed up with the syntax and the manipulation of variables.

Method 3: Concentric squares and spirals
The children didn't use either of these two methods. The difficulties are not greater than those in the above method and it is interesting to wonder why they were not discovered



```
TO FILLSQ :LENGTH
REPEAT :LENGTH (SQUARE :LENGTH
```

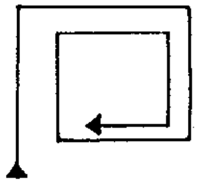
interface

```
FD 1 RT 90 FD 1 LT 90
```

do the next one a bit smaller

```
MAKE "LENGTH :LENGTH —1)
END
```

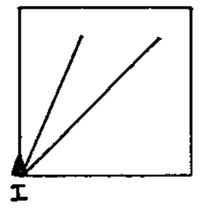
This procedure makes concentric squares; it is very satisfying to the eye. When asked to fill a square several children first think of this solution, or even draw it on paper. In trying to write the program they forget the interface and write a procedure that works but always draws squares in the bottom left corner. Then the initial goal—fill in a square—is reached and there is no real interest in improving the solution.



```
TO FILLSQ :LENGTH
REPEAT :LENGTH (FD :LENGTH RT 90 MAKE
"LENGTH :LENGTH —1)
END
```

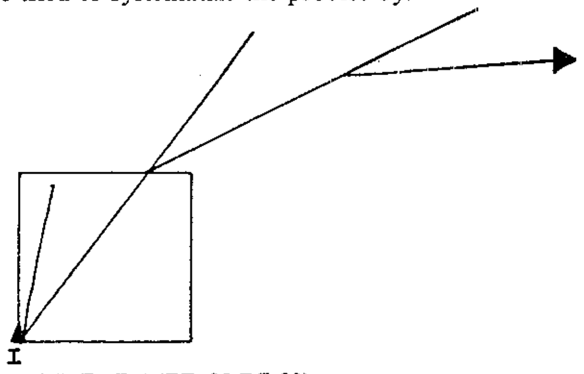
This procedure uses a spiral to fill in the square. It is very short, but a new process has to be conceived, different from the method that reuses the squares. It may be more comfortable for children to build on something that they know is working.

Method 4: Using a radius
In the first group F made a try which deserves to be mentioned:



```
REPEAT 4 (FD 25 RT 90)
RT 45
FD 25 BK 25
RT 23
FD 25 BK 25
```

She tried to systematize the process by:



```
REPEAT 4 (FD 25 RT 90)
REPEAT 10 (FD 25 BK 25 RT 23 FD 27)
```

Then she gave up and went for a "back and forth" method. This intuition was interesting. A procedure that uses this algorithm to fill in a square (the turtle always goes back to the bottom left corner) involves a manipulation of trigonometric functions that is beyond the reach of 9 year old children but could be a good exercise for high school children

In conclusion, most of the problems that children are confronted with in Logo can be solved in direct mode by successive approximations, with no real need for more sophisticated programming techniques. The refinement of technique or "style consciousness" [Howe, 1978] will come from the environment, that is the choice of working situations, and from the interventions of the teachers much more than it comes from the child's spontaneous interest. Teaching strategies and pedagogical progressions are not coming from the magic of Logo but from the exploration and the understanding of potential situations that Logo can provide.

A similar question arises about what mathematical knowledge is acquired by children using Logo It is sometimes difficult to evaluate what children are learning while doing Logo in terms of classical mathematical formalism. If one can understand mathematical formalism as a language to talk about mathematics, then Turtle geometry should be understood as a means of learning how to realize things that have a mathematical structure. It is, in this sense, a primitive way of doing math.

The relationship between formal "process-oriented" and informal "product-oriented" knowledge can be seen as conflictual but it can also be seen as complementary. When children are producing sophisticated drawings without any concern for the way these drawings are done, they might be concerned essentially with feeling able to produce such drawings With the help of teachers, this know-how might later become a core for the development of a more elaborate mathematical language.

References

Abelson, H and diSessa, A , *Turtle Geometry*. MIT Press, 1981
Hillel, J. and Samurçay, R., Analysis of a Logo environment for learning the concept of procedures with variable. Research Report No. 2, Concordia University, 1985
Howe, J , Developmental stages in learning to program DAI Research Paper No 119, University of Edinburgh, 1978
Papert, S., New theories for new learning, *School Psychology Review*, Oct. 1984
Ross, P., *Introducing Logo*. Addison-Wesley, 1983

Acknowledgment

This paper is based on material gathered from two research projects, one at Université du Québec à Montréal and the other at Concordia University, Montreal I would like to thank Dr Tamara Lemerise, Psychology Department, GREL UQAM, Prof Hélène Kayler, Mathematics Department, GREL UQAM, and Dr Joel Hillel, Mathematics Department, Concordia University, for allowing me to use this material and for their helpful advice

Continued from page 12

colonized peoples—was capable of developing mathematics in the past, and therefore—regaining cultural confidence [Gerdes, 1982, 1985a]—will be capable, now and in the future, of developing and using mathematics creatively

Defrosting frozen mathematics can serve as a starting point for doing and elaborating mathematics in the classroom, as we showed in the geometrically-related examples we gave.

At the same time "unfreezing frozen mathematics" forces mathematicians and philosophers to reflect on the relationship between geometrical thinking and material production, between doing mathematics and technology Where do (early) geometrical ideas come from? [Gerdes, 1985b]

Editor's note

The Proceedings of the conference at which Professor Gerdes delivered a fuller version of the above paper are published under the title, "Mathematics and Culture: a Seminar Report," edited by M. Johnsen Hoines and S. Mellin-Olsen. The report can be ordered from Caspar Forlag, 5046 Radal, Norway.

References

D'Ambrosio, U [1984] The intercultural transmission of mathematical knowledge: effects on mathematical education UNICAMP, Campinas
D'Ambrosio, U [1985] Ethnomathematics and its place in the history and pedagogy of mathematics. *For the Learning of Mathematics* 5, 1, 44-48
Gerdes, P. [1982] Mathematics for the benefit of the people CARIMATHS, Paramaribo
Gerdes, P [1985a] Conditions and strategies for emancipatory mathematics education in underdeveloped countries *For the Learning of Mathematics* 5, 1, 15-21
Gerdes, P [1985b] Zum erwachenden geometrischen Denken PH Dresden
Howson, A.; Nebres, B ; Wilson, B [1985] School mathematics in the 1990s. ICMI, Southampton