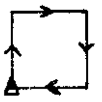
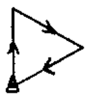
On Logo Squares, Triangles and Houses

JOEL HILLEL

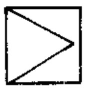
In his book *Mindstorms* [1980], Papert describes a "classical" LOGO story of a child trying to "construct a house". In this case the child, Pamela, after a few sessions of turtle-geometry, spontaneously decides to use her procedures SQUARE



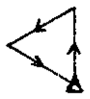
and TRIANGLE



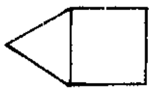
Her first attempt is SQUARE, TRIANGLE which results in



because of an *interface* bug, i.e. the two procedures are not properly linked. Pamela debugs her program by "playing turtle" and then changing her procedure TRIANGLE so as to produce



Her new procedure SQUARE, TRIANGLE now ends up with



and is fixed up by an initial RT 90 command.

Papert then discusses the significance of the above episode by pointing out how the use of procedures as "building blocks" is embedded in the idea of hierarchical organization, and how the activity of debugging in LOGO is often incorporated into the process of understanding.

I will be describing below four hour-long sessions of children working with the definition and use of a procedure for triangle. Now, I don't know how old Pamela is, since Papert doesn't tell us (indeed, throughout *Mindstorms,* it is not always obvious when Papert is talking about 5 year olds and when he is talking about 15 year olds). But for the 9 year olds that I have been observing, the sequence (program + bugs) → debugging → new understanding, portrayed above, does not even begin to capture the level of conceptual difficulties experienced by these children, nor, for that matter, does it capture the richness of the experience. The four children I'll be describing would be in their 26th to 30th session of turtle-geometric activities. Their "LOGO environment" is somewhat privileged both in having a complete access to computers (in the computer lab of the math department) and in being provided with a lot of help, ideas and challenges. On the other hand, since our research is focused on the acquisition of mathematical and programming concepts, we have often structured activities to suit *our* research agenda rather than the children's immediate interest (I recognize that, in the *extreme,* such an approach is antithetical to the whole LOGO philosophy. But the viewpoint that adults have nothing to offer to children is equally absurd.) For more details on the first 12 sessions, see Hillel [1984].

I would like to first examine the two important ideas mentioned by Papert, namely, the idea of "hierarchical organization" and that of "debugging". Clearly, defining and using LOGO procedures as "building blocks" is central to the programming activity. This involves both the synthetic activity of using existing procedures to construct more elaborate figures (as in the example recounted by Papert) and the analytic activity of making a "procedural-analysis" of, say, a complex figure in terms of identifiable "building blocks" (subprocedures). Both kinds of activity ought to foster the idea of "hierarchical organization". Yet the children we observe and those observed by others (see for example, Leron [1984]), do not spontaneously adopt a procedural viewpoint when writing a program. Despite the fact that our children have been provided with extensive experience of using procedures and the fact that we have discussed the relative merits of using procedures, they, surprisingly often, opt for writing *simple* programs (i.e., using only LOGO primitives). I think that a plausible explanation for this is the overwhelming identification by the children of programming as "drawing with the turtle". On the one hand, part of the "naturalness" of LOGO is due to the strong resonance of "instructing the turtle" with the children's drawing schema. On the other hand, linking programming with drawing stands in the way of making a "procedural-analysis" of a particular task. Writing a program to produce a geometric figure is viewed as tracing certain linear or curvilinear segments rather than putting together some, possibly not yet defined, building blocks.

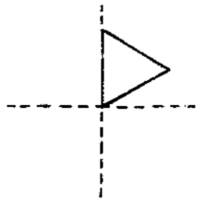
I do not dispute Papert's claim about the importance of debugging to the learning process, but I do question whether a lot of the children's attempts to fix their procedures should be called debugging rather than "adjustment strategy". In the example cited by Papert, Pamela debugs her procedure by retracing the *process* underlying the definition of SQUARE and TRIANGLE. I think that it is quite significant that the debugging activity ("playing turtle") is done *away* from the computer. More typical children's behaviour can be termed "debugging-in-action", i.e. using *direct-mode* (executing one command at a time) to, say, adjust the heading of the turtle until the triangle and square are properly linked. Such debugging stategy is generally successful, but precisely because it tends to avoid viewing procedures as *processes of production,* it rarely leads to new knowledge. The persistence of interface bugs, even by children older than those in the present study, is the result of a non-dynamic conception of procedures.

About LOGO triangles

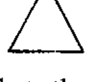
When I decided to initiate a series of activities for the children around the definition and use of a procedure for (an equilateral) triangle, I had several aims in mind. Aside from adding another geometric shape to the children's repertoire of procedures (which includes rectangles, squares, circles, "steps" etc.), the triangle offers several other possibilities, namely:

(i) its definition can be easily parametrized by length, hence fits in with the children's current activity with *general* procedures (procedures with inputs);
(ii) it involves angles which are multiples of 30° which could be used to break the "grip" of 45° and its multiples which have predominated in the children's work;
(iii) it could lead to a generalization of the procedure for a regular *n*-gon.

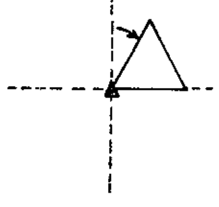
There is something else which is interesting about a triangle. Its most "elegant" LOGO definition, as a procedure: REPEAT 3 [FD X RT 120], when executed from the "normal" initial turtle-position is clearly in the "wrong" orientation, i.e.



(In fact, it is the only regular *n*-gon which is so obviously "not in place".) Since most people think of a triangle as



I was pretty certain that the children's spontaneous attempt to define the triangle would begin with a right turn, i.e., the triangle would be visualized as being located on the screen as



Now a procedure like RT 30, REPEAT 3 [FD X RT 120] is, of course, a perfectly good procedure to produce a triangle on the screen. In some particular situations it may actually be the most "economical" procedure. However, as a general procedure for a triangle, the command RT30 does not seem to belong to an "intrinsic" definition of a triangle. Its inclusion in the procedure mixes in the *location* question (the position and orientation of the triangle on the screen) with the process of defining a triangle. (From the turtle's perspective, it is equivalent to asking the question "How should I produce the figure relative to where I am?" rather than "Where ought I to be relative to the figure?") Furthermore, such a definition makes it more difficult to make the generalization to *n*-gons

It should be clear that these are my concerns and not the children's. They, most often, understand their task as that of trying to find an economical solution to a given problem without attaching importance to more long-term goals (since they hardly know what these goals might be). However, it has been my strategy, throughout the experience, to confront the chldren with issues that may lead them to reflect on the meaning of an "instrinsic" geometry. This explains why I decided to introduce the activity in the way which I elaborate below.

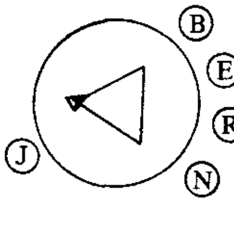
About the children

The four children, Norm, Bill, Elaine and Robert, are all around 9½ years of age. They have been coming to the computer lab for about a year and have already had 25 full hours of turtle-geometry work. For the last 10 sessions they have been working with procedures that take a single input. For the first 22 sessions, the children came a pair at a time and mostly shared a single computer (thus enabling us to audio-tape the sessions and save all the screen productions on a video cassette). They now come all together and have a computer available to each.

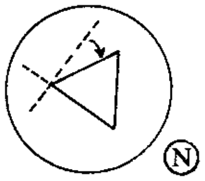
For the sessions described below I was assisted in my observations by Ernest Steingruber of Realschule, Matzingen, Switzerland, to whom I extend my thanks

Session 1: Defining a procedure for triangle

I place a cut-out shape of an equilateral triangle on top of a circular table. At one of the vertices, I put a small toy-turtle with the appropriate heading and I invite the children, who are gathered around the table, to tell me the instructions for tracing the triangle.

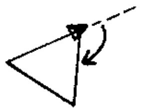


Norm says right away that we first need to turn the turtle, by about RT 45. I object by saying that the turtle is already lying on the triangle, but Norm explains that this is not "home". Even from his position, Norm "sees" the triangle as being aligned as follows:



I invite Norm to walk around the table, so as to view the triangle from different angles, and I ask if he is still sure where "home" is. I am not sure if my ploy is effective, but he decides not to contest.

After agreeing on an initial displacement, FD 30, the children debate how much to turn

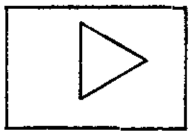


and offer both 90° and 135° as possibilities (they are still most comfortable with multiples of 45°). Since 135° has the most support, they proceed with it and with the rest of the instructions, which Ernest has been writing on the blackboard as a procedure

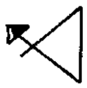
```
TO TRIANGLE
FD 30
RT 135
FD 30
RT 135
FD 30
RT 135
END
```

(The last RT 135 command which renders the procedure *state-transparent* i.e. leaves the turtle in the same position and orientation as when it started, was put in at my instigation.)

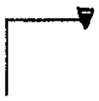
The children proceed to try to execute the procedure on the computers. Only one child types the instructions as they are listed on the blackboard. The other three children start with RT 45 prior to entering the other instructions. It seems that they will not accept a triangle located on the screen as



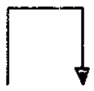
After executing the procedure and ending up with



they realize that 135° is too much, and fall back on 90° (Ernest changes the instruction accordingly). Curiously enough, despite their extensive experience with procedures for squares and rectangles, none of the children anticipates the outcome. There is no comment after the first four commands are executed and yield



(by now, the children have forgotten about an initial rotation of the turtle) and only after the final FD 30 is executed, producing

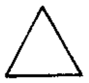


do they claim that "90 is too little".

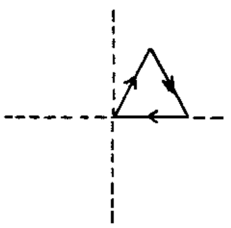
The children are now aware that the rotation has to be somewhere between 90° and 135°. They suggest angles of 100, 110, 120, 125 and they decide that each will try one of the values. Norm then says "I know, why don't we use REPEAT", something which they all readily accept. After trying out the different values, they finally end up with the procedure

```
TO TRIANGLE
REPEAT 3 [FD 30  RT 120]
END
```

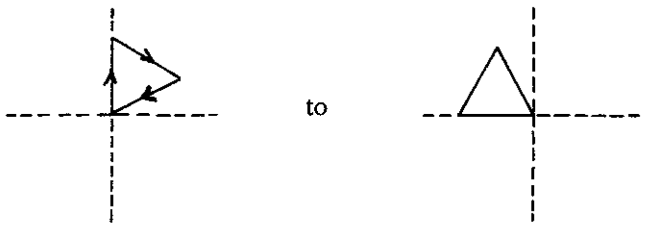
After all the children are satisfied that the procedure really works and have saved it on their disk, I ask them if they can use their procedure to draw a triangle in the "usual" position.



Of course, my aim here is to reinforce the idea that the location of the figure on the screen depends only on the initial *turtle-state* (the turtle's position and orientation). Three of the children begin with RT 45, TRIANGLE. This is almost an implicit assumption that the triangle should be to the right of "home" position and drawn in a clockwise direction, i.e.



(The tendency to work in the upper-half screen and to move in a clockwise direction is very consistent. RT's outnumber LT's by a healthy margin!) Robert, however, sees the possibility of going from



and starts with LT 90. Curiously, he does not then call the procedure TRIANGLE, but starts *planning-in-action* (i.e. planning as he goes along, one command at a time) a new set of instrucitons for the triangle (not even using REPEAT). Robert does not, on this occasion, see that his initial procedure TRIANGLE defines a whole class of congruent triangles.

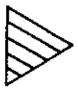
Defining a general procedure for triangle
Since the children have used the defined general procedures for several geometric figures, I ask if they can now write a single procedure for triangles of different sizes.

Elaine says right away, "Use length" (Elaine identifies the notion of a variable with the variable-name "length". She has used "length" in every general procedure that she has defined thus far.) The generalization of the procedure is done without any difficulties.

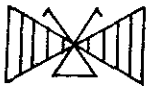
```
    TO TRIANGLE
    REPEAT 3 [FD :LENGTH RT 120]
    END
```

As with all previous generalizations of procedures, I have to remind the children that :LENGTH needs to be declared in the title line.

After trying out TRIANGLE with different inputs, two children end up with the pattern

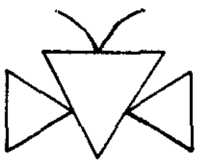


Since the session is coming to an end, I suggest that they might want to think about writing a procedure at home for a "butterfly", which I draw on the blackboard.



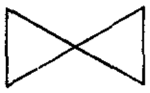
Session 2: The "butterfly" project
This turns out to be an over-ambitious project, partly because the children have not, as yet, made spontaneous "procedural-analysis" of a complex figure in terms of simpler "building-blocks". (The children are more likely to identify a "building-block" if they have already defined a procedure for it, e.g. TRIANGLE in the case of the "butterfly". They are less likely to define, say RIGHTWING and LEFTWING procedures.)

Bill has written a lengthy procedure for his simplified version of "butterfly" which he has drawn in his notebook

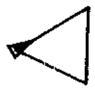


Despite the "obvious" appearance of three triangles in his drawing, he hasn't used TRIANGLE as a subprocedure but has written a "simple" program (using only LOGO primitives). His program is difficult to read because it is so long. Interestingly enough, I note that Bill tries to construct the antennae by using REPEAT 3 [FD 1 RT 1]. This connects to a very brief session of work on arcs (in which the number of repetitions was varied), which took place almost two months earlier.

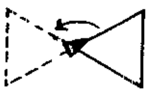
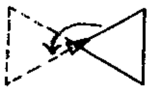
Since Bill becomes quickly aware that his procedure is full of bugs he ignores his written instructions and starts planning-in-action. I encourage him to use the TRIANGLE procedure and he ends up spending the rest of the session trying to construct



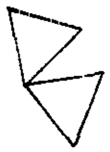
He has a lot of difficulty with the "interface" between the two "wings". Having succeeded in completing the right-wing



he seems to visualize the interface as corresponding to the turn

 rather than 

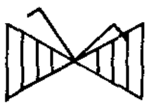
and ends up producing figures such as:



Bill's problem is precisely in not conceptualizing procedures as processes of production, but looking only at the resulting figures. He does not go back to examine his TRIANGLE procedure in order to debug his difficulties. Rather, he simply uses the "adjustment strategy" of changing the angle of rotation until he gets the desired effect.

Robert's construction relies heavily on the use of the command HOME as a way of dealing with intricate "interface" problems. His scheme is essentially based on right antenna/HOME/left antenna/HOME/rightwing/HOME/leftwing/HOME.

Except for a few minor modification which he makes while entering the instructions in the Editor, the procedure outputs

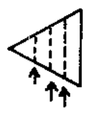


Robert is pleased with the results, though he does subsequently try several modifications in order to "straighten out" the right antenna, without success.

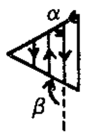
The use of the command HOME is very convenient and, in a sense, provides the most "economical" solution to the problem of defining a procedure for "butterfly". I have argued elsewhere [Hillel, 1984] that using HOME in a procedure tends to "sabotage" the intrinsic nature of "turtle-geometry", by implicitly tying the figure to a preferential point (the "origin" of the cartesian system) This takes away the flexibility from a procedure since it no longer describes a class of congruent figures but, essentially, a single figure.

Robert, of course, has solved very nicely the problem that I presented him. But I do take the opportunity to bring Robert to reflect on his procedure and its limitation. I ask Robert if he can now fill the screen with several "butterflies". Robert picks up the idea and proceeds with the obvious step of moving the turtle to a new position and typing BUTTERFLY. He is very puzzled at the outcome (which, aside from the right antenna, keeps producing a "butterfly" in "standard" location) I ask Robert to think what the command HOME is doing to his procedure. This is a strong enough hint for Robert to see the problem and he spends the rest of the session trying to rewrite his procedure without the use of HOME.

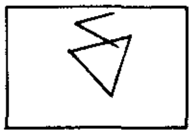
Norm's procedure includes TRIANGLE as a subprocedure. He does not view the wings as consisting of four embedded triangles, but rather as a single triangle with three vertical lines, i.e.



He tries to produce this as a sequence



This makes the procedure particularly difficult as it involves sorting out the interior angles ($\alpha = 60°$) and exterior angles ($\beta = 120°$). When he executes his initial procedure, it begins with something like:



Norm needs a lot of help from Ernest in figuring out the two relevant angles of rotation. He does not, however, want to change his initial plan of production even when it is suggested to him that he could use his TRIANGLE procedure to construct the vertical lines. (Norm is possible opting for an "economical" solution in the sense of not unnecessarily retracing the same lines, i.e. minimizing the total path travelled by the turtle!)

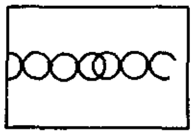
Elaine has the flu and has stayed home.

Session 3: Unstructured activity

Harbouring, perhaps, some guilty feelings about the last session, I decide to leave the children to "do their own thing".

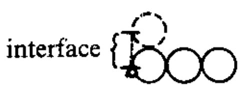
Norm is the only child to return to the "butterfly" project. He returns to his procedure and cleans up the last details. None of the other children touches the TRIANGLE procedure.

Bill returns to a previous project called BELT which he obtained by repeatedly using the sequence [CIRCLE 30 displacement] and by using the "wrap" feature of the screen.



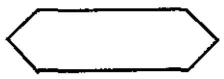
He is the only child who shows interest in the "wrap" feature since it became available to the children (the command FENCE was automatically in place for the first 24 sessions). The others, having been accustomed to working within the confines of the screen, seem to consider the "wrap" feature more of a nuisance.

Bill decides to fill the screen with an array of circles, using BELT as a "building block". He takes unusual (for him) care to determine the interface move:



which he figures out, correctly, to be PenUp, FD 30, PenDown (since BELT uses CIRCLE 30, where 30 is the diameter of the circle. It is not at all a trivial observation that the displacement of the turtle should equal the diameter of the circle!) Bill then defines BELTS as REPEAT 15 [BELT PU FD 30 PD] and he is very pleased with the outcome, which he shows to other children. He realizes that he does not need so many repetitions and, after counting the number of rows of circles on the screen, changes the number of repetitions to 8.

Both Elaine and Robert spend half the session in non-goal-oriented, direct-mode activity (using only multiples of 45° for rotation). This, at least in Elaine's case, leads to a fortuitous outcome, as she tries to convert her construction of the figure



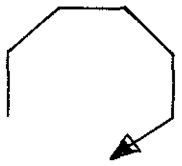
to what, she thinks, will be a hexagon. In writing out a procedure for hexagon, which she calls STOPSIGN, she spontaneously writes it as a general procedure (it is the first time that I see any of the children do this).

```
TO STOPSIGN :LENGTH
REPEAT 6 [FD :LENGTH RT 45].
```

She then asks me if there is a way to change the number of times "it (the turtle) repeats". I suggest to her that just as she replaced FD 30 by FD :LENGTH, she can replace REPEAT 6 by REPEAT :TIMES. She changes her procedure (after being reminded about the declarations of the variable-name) to

```
TO STOPSIGN :LENGTH :TIMES
REPEAT :TIMES [FD :LENGTH RT 45]
END
```

She now worries that the order in which the two variable-names are declared since they appear in the reverse order within the procedure. (This seems to me to be a very legitimate concern even though, in LOGO, the order of declaration is not important.) She changes to STOPSIGN :TIMES :LENGTH and executes STOPSIGN 6 30 which produces



She does not seem to be very pleased about the outcome. I think that her idea is to define a general procedure for regular *n*-gons of any size. She has not yet become aware that changing the number of repetitions necessarily calls for changing the angle of rotation.

The definition of a procedure with two variables is rather a remarkable generalization. Both Elaine and Robert were shown a single example of a two-variable procedure about two months earlier. It was a brief 5 minute episode which we felt, at the time, was a complete disaster. The children seemed very confused about the idea that we were trying to show. Perhaps that session is, nevertheless, the catalyst for Elaine's idea.

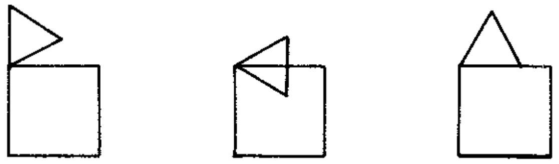
Session 4: Building a house
At the end of Session 3, I suggested to the children that since they had a general procedure for a triangle, and they knew how to write one for a square (though there is no such procedure on their current disks), they could plan at home a procedure for one or several houses. All the children worked on the project at home.

Robert (who left his work book at home for the first time ever) starts by defining

```
TO SQ :LENGTH
REPEAT [ :LENGTH RT 90]
```

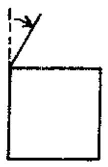
Since the variable LENGTH is not passed to the FD command, his attempt to execute SQ 40 ends in an error message. Robert returns to his procedure and I ask him to execute SQ 40 mentally, but before he even begins, Elaine quickly scans the definition of SQ and picks up the bug.

Robert now begins to plan-in-action the roof of the house without using TRIANGLE, until I remind him that the procedure is available to him. He now tries to use SQ 20 followed by TRIANGLE 10:



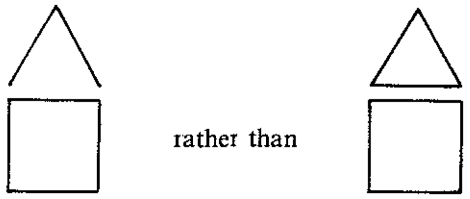
First attempt Second attempt Third attempt

Having apparently solved the interface problem, he reverts to using FD and RT to construct the roof. I ask him why he has abandoned the procedure TRIANGLE and he answers that though he figured out the turn



as RT 30, he doesn't know how large the triangle should be.

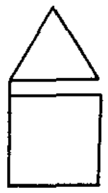
I am at first very puzzled that he should miss the "obvious" until it strikes me that Robert "sees" his house as



When I draw the latter, he immediately writes the commands SQ 30, FD 30, RT 30, TRIANGLE 30 and is very pleased at the outcome. He then, on his own, writes the following in the Editor:

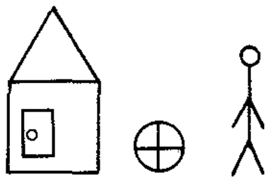
```
TO HOUSE :LENGTH :SIZE
ST
SQ :LENGTH
FD 30          ← bug
RT 30
TRIANGLE :SIZE
END
```

Robert, who works next to Elaine, may have picked up the two-variable procedure idea from her STOPSIGN. Even though there is no real need for two variables in this case, it is interesting that Robert has changed the variable-name of TRIANGLE procedure from LENGTH to SIZE. I don't think he did this consciously, but, rather, that it was imposed on him by the way he wanted to structure his procedure. Nevertheless, he may be, implicitly, accepting that the variable-name is not by itself of any consequence. His procedure contains a "classical" variable bug [see Hillel & Samurçay, 1985], namely the failure to make part of the interface (FD 30) between the square and the triangle, input-dependent. He executes HOUSE 20 20 which outputs



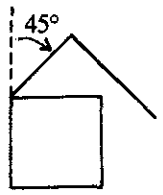
The bug is fixed up only after several interventions on my part. Robert then proceeds to make a "village".

Elaine has a lengthy program for her project "PLAY" which was drawn in her notebook.

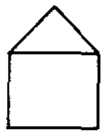


Her procedure is *simple*, using no sub-procedures at all, not even SQ and TRIANGLE. Elaine, who at times has shown the most subtle grasp of the use of procedures among all the children, seems to have relied completely on her "drawing" schema rather than making a "procedural-analysis" of the whole figure.

She executes her procedure in *direct-mode* (rather than first entering it in the Editor and then just calling the procedure, which is more her usual style). There are actually very few bugs in her lengthy list of instructions, which she debugs "in-action". Her initial house involves the familiar 45°, 90° schema, i.e.



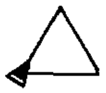
which she adjusts, by decreasing the length of the sides to



When she has complete the project, I discuss with her the merits of defining separate procedures for house, ball and person, instead of writing a long procedure. She seems to understand the explanation that I give.

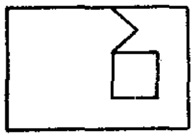
A little later Elaine asks if it is possible to write a single procedure which will produce either a "left square" or a "right square". She has certainly gone beyond her initial conception that one only varies LENGTH in the procedure. When I answer that I don't know how to do it (I don't!), she defines two general procedures RTSQ and LTSQ. It is only during the next session (the fifth session) that I get a chance to ask her what she intends to do. She explains that she wants to draw squares in "both direction" and she volunteers that it is not really necessary to define two procedures because "I could just turn the turtle around". (There is, of course, a subtle way in which the two definitions of SQ are different, namely in their orientation and their initial and final turtle-states.)

She returns to defining HOUSE without using TRIANGLE as a sub-procedure and says emphatically that she cannot use the procedure because the triangle is not in the "right position". (Her perception of a minute ago, that the location of the square depends on the turtle-state is not extended to the triangle.) I work with her for several minutes with the cut-out triangle and I discuss with her how it can be rotated by any angle. Elaine returns to the computer and after verifying that she needs RT 30 to "straighten out" the triangle, she *incorporates* RT 30 into the procedure TRIANGLE so that it produces

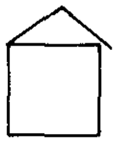


She now easily writes a general procedure HOUSE. (Later on in the session, Elaine executes TRIANGLE 40 and then decides to erase it. She writes PE TRIANGLE 40 and gets a surprise when it does not erase at all: only state-transparent procedures are "self-erasing".)

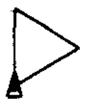
Bill's program for "house" uses SQ as a subprocedure, but not TRIANGLE. He enters his instructions in the Editor and then executes HOUSE which, because of a *lateralisation* error, produces



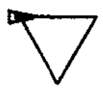
Bill's programs nearly always contain several lateralisation bugs. While these bugs are not really the result of conceptual problems, they often render the output so different than the intended figure, that Bill ends up at a loss what to do. This time, however, the bug is identified and fixed, resulting in



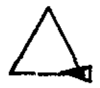
Bill then continues to modify his procedure until I inquire why he doesn't use TRIANGLE. His reaction can be taken as, "Oh, I forgot", since he quickly picks up the idea. He very intelligently renames his procedure TRI, since he has been very frustrated before in having to type TRIANGLE every time. He tries TRI,



says it needs a turn and writes RT 90, TRI

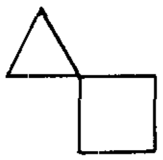


(lateralisation error), then LT 90, TRI



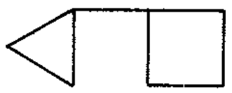
He now thinks that he can fix his HOUSE procedure, by using SQ and TRI as subprocedures. His first attempt is

```
SQ 40
FD 40    (incomplete interface)
LT 90
TRI 40
```

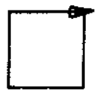


then

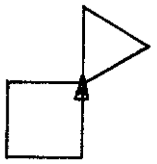
```
SQ 40
FD 40
LT 90   (lateralisation bug)
FD 40
LT 90
TRI 40
```



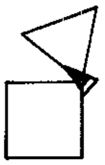
Bill now decides to plan-in-action. He brings the turtle to the top right hand corner of the square



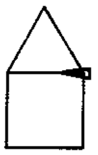
and then adds LT 90, TRI 40. However, the turtle's initial heading is not as before, and it produces



Bill tries to compensate for, rather than really debug, the errors in his instructions. He erases the triangle and enters RT 45, TRI 40

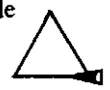


and, finally, with one more modification, he ends up with

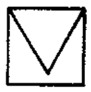


He recognizes now that he should have turned the turtle 180 prior to executing TRI and rewrites his procedure HOUSE, which now produces the right figure.

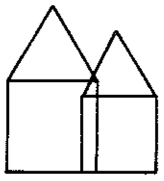
Norm has written a procedure for two different-sized houses. He is the only child to use both SQ and TRIANGLE as subprocedures (though he doesn't define a separate procedure HOUSE). In fact, his program is basically correct, including the correct angle for "putting on the roof" (RT 30). However, since he works next to Bill, he notices that Bill's triangle



involves a 90° turn. This leads Norm to question his own procedure and he makes a quick decision to modify his program by changing RT 30 to RT 90. He does this without analyzing the effect of the change on his procedure and he simply executes it. The output is



which leads Norm to return to his initial procedure. This produces



and the one bug is quickly fixed up. Norm, more than any of the other children, always wants to start and return to the "home" position. He uses the command HOME both before starting the second house and after finishing it.

Some concluding remarks

These observations, as with past ones that I have made, bring out to me the fact that there are rather difficult and subtle notions underlying turtle-geometric activity (at least for children in the 8-10 age range). Some of the difficulties have to do with understanding the essence of procedure use, the meaning of "intrinsic" geometry, and the relation between procedures and the objects they produce. There are also cognitive difficulties in making the shift from the predominant "drawing" schema for analyzing the structure of plane figures to that of making a "procedural-analysis" of such figures. The persistence of "interface" errors reveals that children do not easily and spontaneously adopt a process-oriented view of procedures.

The above comments should not be construed as a criticism of LOGO nor as arguments against introducing turtle-geometry to young children in schools. On the contrary, my observations should be taken as a case against those who claim that turtle-geometry is just a trivial activity and a waste of time. Now I agree that turtle-geometry can be trivialized because it is very easy to *gloss over all the difficulties* which I have described. This is achieved by always letting the children plan-in-action, by letting children return over and over again to doing the same kind of constructions, by overemphasizing the production of complex patterns at the expense of understanding the functioning of very simple procedures, and by failing to challenge the children to plan and to anticipate the outcome of their programs. I already hear questions such as "What should we do now that the children have learnt everything about turtle-geometry?" being asked after a month or two of minimum exposure to LOGO activities. I argue that, with a little bit of thought and effort, there is enough content in turtle-geometry to keep children busy throughout their elementary schooling

References

Hillel, J. [1984] Research Report No 1. Mathematical and programming concepts acquired by children, aged 8-9, in a restricted LOGO environment. To appear in *Recherches en Didactique des Mathématiques*

Hillel, J. and Samurçay, R. [1985] Research Report No 2. Analysis of a LOGO environment for learning the concept of procedures with variables. In preparation

Leron, U. [1983] Some problems in children's LOGO learning. *Proceedings of the 7th International Conference of the PME*, Shoresh, Israel

Papert, S. [1980] *Mindstorms*. New York: Basic Books