Feature Article

Computer Modeling for Open-Ended Mathematics

by Reinhold Wappler

In my 6th grade class we sometimes devote a sequence of whole periods and assignments to computer modeling. We work with LogoWriter, equipped with four turtles and a page of 20 shapes. This paper describes a portion of one multi-period exercise in which there was deliberately no particular plan at the outset, but which went where the language and our imaginations took us. My aim in this modeling is the creation of dynamic and engaging situations to which traditional mathematics may be applied.

While our students know some Logo, I make it clear that they are not responsible for the language or for programming skills. They are to employ mathematical thinking to solve problems generated by the model. Some of the procedures we wrote are presented in this paper at the end, but since they constantly changed during the exercise, they are only "snapshots" of the process. Indeed, much of the exercise of mathematics occurs in the changing of procedure lines.

The kids take turns at the keyboard, and we start simply, drawing a road along which we will move an object. There will be other objects, moving and doing things, but initially there is no clear program goal. The kids think the road is easy, but I intrude to make it more challenging.

"What are the *properties* of the road? " I ask. "Its just a stupid line, let's draw it!" they retort. "Nope, you have to list the properties." I tell them. "One line of the ROAD procedure must be devoted to each property."

With some effort, I wheedle some properties out of them, knowing the idea of properties will be easier for the second and third objects.

ROAD's properties include starting x and y coordinates, heading, length, color, which turtle, shown or hidden, and pen up or down. We spend a lot of time on ROAD as the first object, anticipating a knowledge transfer exercise when the second and third objects come up. Finally, ROAD does what the kids want, in the form I want.

Argument arises over what object to send down the road. Many kids have strong feelings, but some less familiar with Logo space away from the unfamiliar and are doodling or starting tomorrow's homework. A tutor-student is assigned to work with each novitiate. A 4-line shape-displaying procedure DISPLAYSHAPE (see page 22) is written to refresh them about recursive procedures and variables. I explain carefully to those with limited Logo knowledge what causes the recursion and what those dotted variables do. The shape selector, which initially riffled thru shapes so fast as to be a blur, is adjusted to stop and show the next shape when a key is pressed. It's considered neat. By majority vote, a truck shape is selected.

Before the procedure TRUCK is written, I emphasize the importance of using a *pattern* which will suggest many or all of the properties of the truck. The pattern involves selecting and pasting a copy of ROAD, and the kids write TRUCK simply by making appropriate changes in the copy of ROAD. The discusson of properties for TRUCK is thus much shorter and more focussed. "But the truck's supposed to move!" Well, OK, let's do motion in a separate MOVE procedure. Within MOVE, the kids want to create motion with a repeat command, but I insist they *transfer* what they learned about recursion in the DISPLAYSHAPE procedure to move the truck. Someone explains, "You can make it repeat by putting the name of the procedure on the last line, before END". We write:

```
TO MOVE
TELL "TRUCK FD 3
MOVE    (there it is)
END
```

and try it out. Very simple, and the truck moves forward, taking 3 turtlesteps with each recursion of MOVE. Now, following the immutable laws of the Video Game Era, HELIcopter and MISSILE procedures are written. The girls parallel the boys in zeal for blow-em-up games. There's a big discussion of where precisely to hang the missile on the helicopter, leading to several adjustments on the precise starting coordinates of the missile. Then lines are added to MOVE which give motion to each of the new objects. TELL :HELI FD 5, and TELL :MISSILE FD 5 give the helicopter and missile the same speed, but faster than the truck. The game's purpose is now clear to everyone. Motivation and interest rise. We write a procedure called DOIT to sequence and run all the other procedures.

But with DOIT, we see only endless horizontal motion of all the objects. We need a way to aim and fire the missile. I show them how KEY? works, and put the line IF KEY? [FIRE] into MOVE. If a key is pressed KEY? outputs "true" and the procedure FIRE (which we haven't written yet) is allowed to run. This makes a big impression. In MOVE, the missile heading was set at 90 degrees. In FIRE we decide to make it 180 degrees, straight down. This works, but it's too simple. Three attempts and the precise lead for the helicopter to hit the truck every time is learned.

We discuss how to randomize the speed assignments to make the game harder. FD 3 for the TRUCK gives way, for example, to FD 3 + RANDOM 4. RANDOM 4 outputs a number from 0 to 3, so FD gets a randomly selected input number from 3 to 7. All the objects in MOVE are given ran-

Continued next page