Modeling...continued from previous page

dom speed assignments. We take time with RANDOM because of its obvious mathematical importance. But when we try MOVE, jerky motions result. We're not sure why, but we persist. Finally we see that in a recursive procedure like MOVE which repeats itself over and over, the input to FD changes randomly with each recursion. I explain that each object should be given a randomly selected speed as one of its initial properties, way before the program gets to MOVE. This doesn't mean much to the kids until we go back to TRUCK, HELI, and MISSILE, and actually add a command to each naming a speed variables and assigning a random speed as a value. We print the speeds on the screen so we know what they are. These speeds are understood as additional properties of each object.

We go to MOVE and change the inputs to the FD commands for each object to the names of the speed variables named in TRUCK, HELI and MISSILE. Voila! The jerky motion disappears, but there is a lot of experimentation with the random speeds before the helicopter always moves faster than the truck. To make the game even harder, and to get more practice naming and giving values to variables, the heading at which the missile is to be fired is randomly chosen somewhere between 135 degrees and 180 degrees. The screen is now as shown in Figure 1.

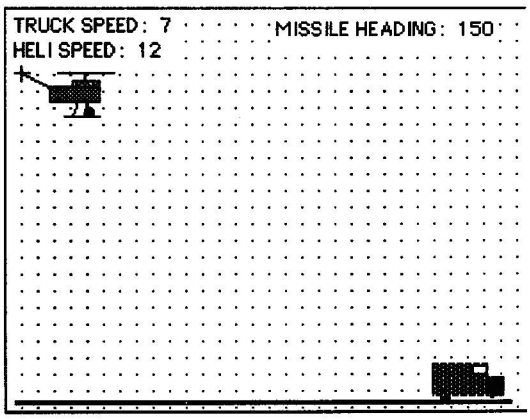

Fig. 1

Although crude, this is now a working and very interesting game, and there are immediate dismaying consequences. Boys crowd around the keyboard, jostling for position and a turn. Some quarrel and shove. Shouts of "Shoot now! Shoot now!" *All* girls move back to the tables and sit down. Too bad, it was going well. But we proceeded again after establishing some rules about taking turns.

Another unexpected outcome: nobody can hit the target! The game is too hard, too many variables. Surprisingly, there's no desire to simplify anything. It seems they'd rather play the game unsuccessfully than work on the game's construction. I'm reminded how hard it is to teach.

Then an idea forms. Instead of simplifying and cutting down on the variables, we'll accept the difficulties and develop a data table and firing plan that will guarantee a hit every time. This goal has wide appeal. Kids like to measure and collect data. It is clear to all players that the key is how far ahead of or behind the truck the helicopter must be at the instant of missile firing. This judgement is too difficult with all the different speeds and missile headings. I suggest a table of forward vectors of the missile path for each 5 degrees of missile angle. Nobody has any idea what I'm talking about, so I rough out the data table on the board, but even with much explaining, comprehension is just not there. So we just start the table with the simplest case, 180 degrees, and find the missile's "forward vector" is 0.

MSL HEADING	180	175	170	165	160	155	150	145	140	135
FWD VECTOR	0	1								

The dot grid helps with measurements. At 175 degrees, the forward vector is about 1 interval on the dot grid. Do we need any more measurements? Isn't every 5 degrees 1 more interval on the grid? Good discussion ensues, and we extrapolate values up to 140 degrees. To check we measure the forward vector for 140 degrees, as per Figure 2, and find it larger than our estimate. Impressed by how smart we

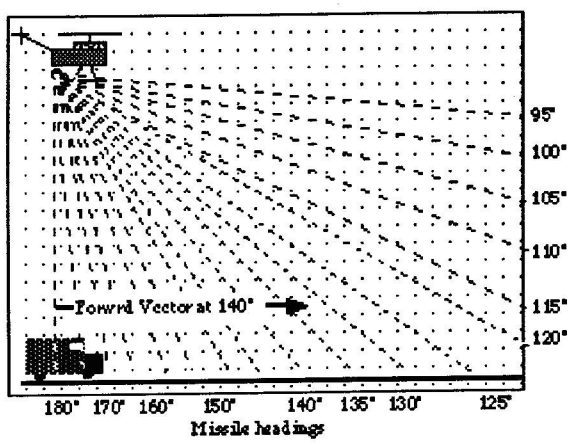

Fig. 2

get by making mistakes, we measure at all 5 degree intervals, with roughly these results.

MSL HEADING	180	175	170	165	160	155	150	145	140	135
FWRD VECTOR	0	1	2	3	5	6	7	9	11	13

Now the kids catch the idea, and sensing large numbers they want the forward vectors for headings closer to 90°. When we get to about 120°, they are chanting 91! 91! I talk them into 95°. Several screen wraps go by, and everyone is

*See **Modeling**...page 21*