POETRY IN MOTION

In this article in our LOGO series, we turn our attention to list processing, which is central to the way the language works. We also take another look at recursion (and revisit the psychoanalyst) before using LOGO to write a little poetry.

A list is a collection of objects in order, and is identified in LOGO by using square brackets; so that [CEYLON MADRAS VINDALOO] is a list. We have encountered lists several times in this series. In fact, we can't escape from them in LOGO, because the language is based on lists. We've already seen how a definition for a square — REPEAT 4 [FD 50 RT 90] — has a list of instructions (within the square brackets) as its second input. Similarly, MAKE "INP REQUEST assigns to INP a list consisting of the input from the keyboard.

Lists can be assigned to global variables — for example, MAKE "CURRY [CEYLON MADRAS VINDALOO]. The command PRINT :CURRY prints the list without the square brackets: that is, CEYLON MADRAS VINDALOO.

A LOGO object can be a number, a word or a list; and a list is defined as simply a collection of objects. This is, of course, a recursive definition; a list can contain another list, or a list of lists, and so on. [[CHICKEN TIKKA] NAN SALAD] is a valid list, with a list as its first element ([CHICKEN TIKKA]). Recursive procedures are often needed to process lists, precisely because lists are recursive objects.

The majority of our programming in LOGO has until now been concerned with one number or one word at a time. When we want to process groups of objects at the same time, we need to organise these simple objects into a single unit. LOGO takes the list as its basic method of grouping simple objects. The list is chosen because it is extremely versatile — you can construct any complex data organisation by starting from a list.

The two fundamental list operations are FIRST and BUTFIRST. FIRST [CEYLON MADRAS VINDALOO] outputs CEYLON — that is, it gives us the first element of the list. BUTFIRST [CEYLON MADRAS VINDALOO] outputs MADRAS VINDALOO; in other words, it gives us the list without its first element.

Here's a procedure that prints the elements of the list, one below the other:

```
TO PRINTOUT :LIST
  PRINT FIRST:LIST
  PRINTOUT BUTFIRST :LIST
END
```

So PRINTOUT [CEYLON MADRAS VINDALOO] gives:

```
CEYLON
MADRAS
VINDALOO
```

The first command prints the first element of the list and then passes the task of printing the rest of the input list to another copy of PRINTOUT. When you run this procedure you'll get an error message when it runs out of data. Here's a more elegant way of finishing:

```
TO PRINTOUT :LIST
  IF EMPTY? :LIST THEN STOP
  PRINT FIRST:LIST
  PRINTOUT BUTFIRST :LIST
END
```

EMPTY? checks to see if its input is the 'empty list' — []. Some MIT versions do not have the primitive EMPTY?, but you can always define it as follows:

```
TO EMPTY? :LIST
  IF :LIST = [] THEN OUTPUT "TRUE
  OUTPUT "FALSE
END
```

Similar to FIRST and BUTFIRST are LAST and BUTLAST. LAST [CEYLON MADRAS VINDALOO] outputs VINDALOO, and BUTLAST [CEYLON MADRAS VINDALOO] outputs CEYLON MADRAS.

BABBLING

For our first exploration in list processing, we'll try to mimic some random babblings on the psychoanalyst's couch. First we'll assign all the words we know to the variable WORDS:

```
MAKE "WORDS [MOTHER FATHER SEX MURDER
JEALOUSY FIRE SEA DEATH DREAM]
```

We want to produce a constant random stream of these words, for experience has shown us that these are the words that are always successful in attracting our psychoanalyst's attention. To get a random element of the list we need to select a random number, n, between one and the length of the list (nine in this case) and then select the nth element of the list.

```
TO NTH :NO :LIST
  IF :N = 1 THEN OUTPUT FIRST :LIST
  OUTPUT NTH :NO - 1 BUTFIRST :LIST
END
```

Let's use this procedure with a few examples to see how it works. Say you type NTH 1 :WORDS. The condition in the first line is true, so the procedure outputs FIRST :WORDS, which in our example is MOTHER.

Try NTH 2 :WORDS — now the condition is false