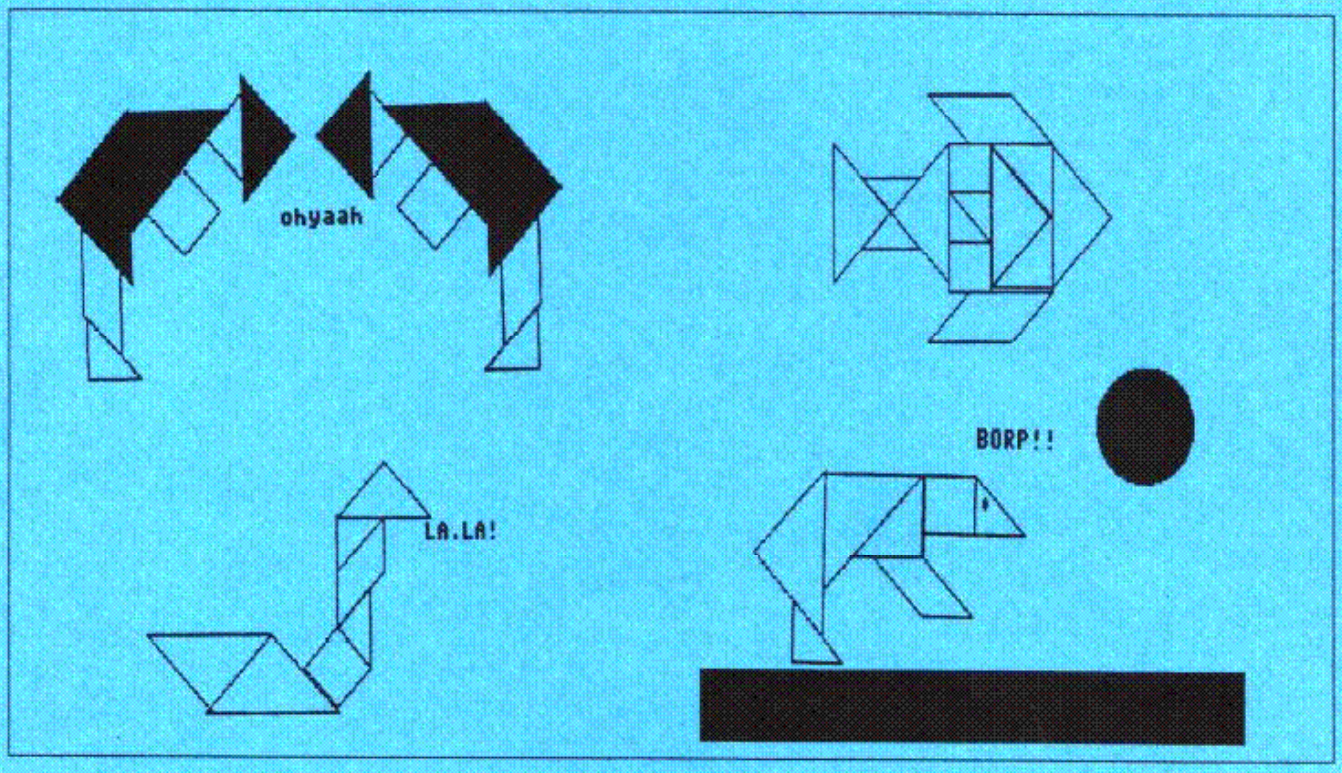
Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

March 1990 Volume 8 Number 7



International Society for Technology in Education

ISTE Publications

LOGO EXCHANGE

Volume 8 Number 7 Journal of the ISTE Special Interest Group for Logo-Using Educators March 1990

Logo Exchange is the journal of the International Society for Technology in Education Special Interest Group for Logo-using Educators (SIGLogo), published monthly September through May by ISTE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-9905, USA; 503/346-4414.

POSTMASTER: Send address changes to Logo Exchange, U of O, 1787 Agate St., Eugene, OR 97403. Second-class postage paid at Eugene OR. USPS #000-554.

Contents

ISTE Membership

	U.S.	Non-U.S.
	28.50	36.00

SIGLogo Membership (includes *The Logo Exchange*)

	U.S.	Non-U.S.
ISTE Member Price	25.00	30.00
Non-ISTE Member Price	30.00	35.00

Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA and Mastercard accepted. Add $18.00 for airmail shipping.

A Random Toolkit for LogoWriter

by Charles E. Crume, B.S.

The RANDOM primitive of LogoWriter accepts a single input and reports an integer between zero and that number. For example, the command:

```
RANDOM 6
```

would report a number between zero and five, not a number between one and six as might be expected. Therefore, this specific command would not be appropriate to simulate the rolling of a die.

To circumvent this inconvenience of LogoWriter's RANDOM primitive, the command:

```
1 + RANDOM 6
```

could be used. The RANDOM 6 portion of the command reports a value between zero and five. Then, one is added to that value giving a final result between one and six. If however, a child writes the above command as:

```
RANDOM 6 + 1
```

it will not work properly. This form of the command reports a value between zero and six (not between one and six as would be expected). This is because the operation of addition (+) takes precedence over the primitive RANDOM. Writing the command in the above manner requires parentheses, as shown below:

```
(RANDOM 6) + 1
```

The parentheses are needed so that the RANDOM function is performed before the addition. In either case, both forms of the command are probably more complex and confusing than they need to be.

This article presents three useful procedures along with some sample programs that may make working with random numbers easier.

The first procedure, called RND, reports a positive random integer within a user specified range. The code is shown below:

```
TO RND :LOW :HIGH
OUTPUT (RANDOM :HIGH - :LOW + 1) +
   :LOW
END
```

This procedure requires the inputs LOW and HIGH (where LOW is the lower limit and HIGH is the upper limit). The procedure first computes the difference between the upper and lower limits (subtraction takes precedence over the RANDOM primitive) and then adds one to this value (addition takes precedence over the RANDOM primitive). Next, the procedure computes a random number between zero and this value. Finally, the procedure adds the lower limit to the random number. For example, the command:

```
RND 1 6
```

will always return a number between one and six inclusive. Changing the range is easy. For example, to simulate rolling a pair of die (whose output will always be a number between two and twelve), the command:

```
RND 2 12
```

is used. To obtain a random number in the range of 100 through 999, the command:

```
RND 100 999
```

is used. Compare the example above to

```
100 + RANDOM 900
```

or the form requiring parentheses

```
(RANDOM 900) + 100
```

neither of which have the value 999 (the upper limit) in them.

A sample program that uses the RND procedure plays a number guessing game. The child chooses a range of numbers, the program selects a number at random from that range, and then the child tries to guess the number. The code is shown below:

```
TO GUESS
RG
HT
CT
PRINT []
INSERT [PLEASE ENTER LOWER LIMIT...]
MAKE "LOWER FIRST READLIST
INSERT [PLEASE ENTER UPPER LIMIT...]
MAKE "UPPER FIRST READLIST
MAKE "NUMBER RND :LOWER :UPPER
CT
```

```
MAKE "GUESS.LIST []
CHECK
PRINT []
PRINT []
(PRINT [IT TOOK YOU] 1 + COUNT
   :GUESS.LIST [TRIES TO GUESS THE
   NUMBER.])
END

TO CHECK
INSERT [WHAT NUMBER DO YOU GUESS?]
MAKE "GUESS FIRST READLIST
IFELSE MEMBER? :GUESS :GUESS.LIST
   [PRINT [YOU ALREADY GUESSED THAT
   NUMBER!] CHECK STOP]  [MAKE
   "GUESS.LIST LPUT :GUESS
   :GUESS.LIST]
IF :GUESS < :LOWER [PRINT [YOUR GUESS
   IS BELOW THE LOWER  LIMIT] CHECK
   STOP]
IF :GUESS > :UPPER [PRINT [YOUR GUESS
   IS HIGHER THAN THE  UPPER LIMIT]
   CHECK STOP]
IF :GUESS = :NUMBER [PRINT [YEA, YEA,
   YEA. YOU GOT IT!]  STOP]
CHECK
END
```

Whereas the command RND 2 12 simulates the rolling of a pair of die, it returns only one value — the total of both die. A procedure that returns two numbers (each representing one of the die) is called DICEROLL. This procedure can be useful in analyzing how often each number appears on each die and for knowing when doubles have been rolled. The code is shown below:

```
TO DICEROLL
OUTPUT LIST RND 1 6 RND 1 6
END
```

The output from the command:

```
PRINT DICEROLL
```

can be any of the following:

```
[1 4] [2 2] [6 4]
```

A sample program that rolls the dice electronically is shown below:

```
TO ROLL
RG
HT
CT
LOOP
END

TO LOOP
PRINT []
INSERT [PRESS RETURN TO ROLL THE
   DICE...]
IGNORE READLIST
(PRINT [YOU ROLLED:] DICEROLL)
LOOP
END

TO IGNORE :RETURNKEY
END
```

Sometimes, a list of random numbers is needed. Instead of using the RND procedure in a REPEAT statement or recursive procedure, the procedure below can be used:

```
TO RNDLIST :LOW :HIGH :HOWMANY
IFELSE :HOWMANY = 1  [OUTPUT RND :LOW
   :HIGH]  [OUTPUT SENTENCE RND :LOW
   :HIGH RNDLIST :LOW :HIGH
   :HOWMANY - 1]
END
```

The procedure RNDLIST requires three inputs. The first two are the lower and upper limits as described for the procedure RND. The third input specifies how many random integers you want in the list. For example, the output of the command:

```
RNDLIST 1 6 10
```

can report any of the following:

```
[3 4 2 3 5 4 1 1 3 5]
[1 1 4 3 2 5 6 4 5 2]
[4 3 5 2 1 5 1 5 4 3]
```

Each list contains 10 integers, each between one and six inclusive.

Charles E. Crume, B.S.
Technical Consultant
University of Nevada System Computing Services
University of Nevada-Reno