Table of Contents

Quick Reference

Logo commands, or primitives, and system names, or variables, are the basic building blocks of Logo programs and control of the Logo environment. Use Logo primitives to move the turtle, control the appearance of your computer screen, and to write procedures for your Logo projects. This chapter provides a brief definition of each Logo primitive and variable as a guide for Logo programming.

Logo Language Overview

Logo is a language that is easy to understand and quick to learn. For people used to computer programming, Logo presents features that are not common to modern programming languages like C++ or JavaScript. This chapter provides a brief overview over the Logo programming language for programmers.

The information does not replace a thorough tutorial. It does not even Please see our own Getting Started tutorial for more details. Brian Harvey's excellent three-volume book set "Computer Science Logo Style" is available as a free download on his web site at https://www.cs.berkeley.edu/~bh. This is highly recommended reading if you would like to become more familiar with Logo.

Data Types

Logo is an untyped language; variables and procedure parameters may contain any data type. Logo numbers are in 64-bit floating point format with up to 15 digits of decimal precision. Strings are 8-bit encoded; Unicode is not supported. Booleans are represented with the constants `TRUE` and `FALSE`. Logo converts these data types to the required types on the fly, just like JavaScript does.

The main data type is the list. Several commands are available for list access and manipulation. Arrays are separate objects with their own commands.

Property lists are also an important part of the language. Commands like `GPROP` or `PPROP` read and set properties. Property lists can also be converted into lists and back.

Turtles, bitmaps, arrays, and controls are property lists. You can, for example, read or set a turtle's position with property list commands. Some properties are callable; these properties correspond to C++ methods. Use commands like ASK or `CPROP` to execute these properties.

Constants

Numbers: Numbers are floating-point. Use the prefix #B, #O, #D and #H to enter binary, octal, decimal, and hex values like, for example, #HFA or #B1101.

Strings: A single leading double quote defines a string constant like `"HELLO`. These constants are converted to upper case unless the `:CASE` variable is set to FALSE. Strings enclosed in vertical bars are not converted; such a string constant needs to be quoted like `"|Hello|`. You can use backquotes to combine vertical bars and the double quote as in `Hello`.

Lists: Lists constants are enclosed in square brackets as in [HELLO |Joe|]. Note the second constant, which is a mixed-case string. No double quote is necessary.

Variables

The MAKE command stores a value into a variable as in `MAKE "NAME "JACK`. A single colon in front of a name loads the variable. Example:

```
MAKE "CENTER [0 0]
SETXY :CENTER
```

Variables are global except for procedure parameters and local variables, which need to be declared inside a procedure with the LOCAL command. Procedures have access to all variables in the calling procedure chain.

Parentheses

The () parentheses have a special meaning in Logo:

- If used with a procedure call, Logo evaluates all arguments and feeds them into the procedure which is the list's first element. This makes it possible to call a procedure with a variable list of arguments. The PRINT command, for example, takes one argument, but if used with parentheses, PRINT accepts any number of arguments. `(PRINT)` prints a newline only, while `(PRINT “A “B :CENTER)` prints three values and a newline.
- In arithmetic expressions, parentheses are used to group expressions as in any other programming language.

Procedures

The TO command defines a Logo procedure. The formal parameters (called “inputs” in Logo) are written as variables. the word END ends a procedure declaration. Procedures are lists; they can be declared dynamically with the DEFINE command, or extracted with the TEXT command. Self-modifying code is, therefore, possible.

A procedure input is written as `:NAME`. Optional inputs are written as two-element lists as in [:NAME value]. Note that Terrapin Logo does not evaluate optional inputs. Optional inputs must follow regular inputs. If the last input is single-element list like [:NAME], it acts as a catch-all input that collects all remaining inputs as a list. Finally, a procedure declaration may have a number as its last input that defines the number of default inputs. If no number is given, that number is the number of regular inputs.

Example:

```
TO FOO :NAME [:VALUE 5] [:REST]
```

Defines a procedure FOO with a regular input, a second, optional input that defaults to the value 5, and a last input that collects remaining values. If FOO is used without parentheses, it needs one input; it requires at least one input.

Examples:

```
  FOO 1
    :NAME = 1
    :VALUE = 5
    :REST = []
  (FOO 1 2 3 4)
    :NAME = 1
    :VALUE = 2
    :REST = [3 4]
```

Recursion is a basic Logo concept; use it whenever possible. Logo detects tail recursion properly.

The STOP command lets you exit a procedure, and the OUTPUT command lets you return a value.

Program Flow

Logo offers the usual set of program flow commands like IF, REPEAT, FOR, or WHILE. The FOREACH command iterates of a list or a word, and the TELL, ASK, and EACH commands iterate over turtles, bitmaps, and controls.

CATCH, THROW and TOPLEVEL are available for nonlocal program flow.

You can execute lists that you have assembled with the RUN ad EVAL commands.

Input and Output

Terrapin Logo supports files and folders. The `OPEN` command opens a file and outputs a channel number. You can assign this channel number to the global variables `:STANDARD.INPUT` and `:STANDARD.OUTPUT` to make all input/output commands work with this channel number, or you can supply the channel number to most I/O commands as an additional, optional input.

Organization

This list of Logo primitives and system names is organized by subject area. Commands that work in the same area, such as controlling the Logo turtle, are grouped together. Within each topic area commands are again organized by function, such as moving the turtle or changing the turtle's state. Within each subheading the commands are listed alphabetically. If you are interested in a particular function of Logo, you can go to the topic area and review the commands that are useful for achieving that purpose.

Format

The command listing is in two columns. The first column provides the syntax for the command with its name followed by any required inputs. Syntax for primitives that are often used with optional inputs is also listed. Note that when Logo primitives that are used with more or fewer than the default number of inputs, both the primitive and its inputs must be enclosed in parentheses as illustrated. The second column provides the abbreviation for the Logo command, if any. The full name and the abbreviation may be used interchangeably for any Logo primitives that have abbreviations. The following paragraph provides a brief definition of the function of the primitive or system name.

Terminology

The following words are used in the definitions of the Logo primitives and system names that are listed in this section (a dotted underline indicates the term is in this list):

:	precedes a system name or variable to access it, and is equivalent to THING
”	precedes a special Logo word, such as a name
#	precedes a command that changes the base of a number
…	indicates that any number of similar arguments may follow
bitmap	graphics image that may be stored on disk or on the clipboard
character	an alphanumeric character
channel	an input or output stream identification number
color	a color number, name, or 3-element RGB list
command	one or more instructions to Logo
control	a graphical Logo object used to display or get information, such as a checkbox, text edit field or static text.
current entity/turtle	the first entity/turtle in the TELL list
descr	a word containing file type descriptors
entity	a Logo object that can be placed in a Graphics window, such as a turtle, bitmap or control
expression	a statement whose output could be a word, a list, a number, or TRUE or FALSE
filename	the name of a disk file
identifier	a word, either a number or a name, that identifies an object
integer	a whole number

list	an ordered set of Logo objects surrounded by brackets
mode	format for file reading and writing
name	word naming a procedure, variable, or other Logo object
number	an integer or decimal number
object	a Logo object, such as a word, list, procedure, window or turtle
objecttype	the name of a Logo object type
port	an input/output port
primitive	name of a Logo primitive (a built-in procedure)
procedure	name of a Logo procedure
property	name of a Logo property
shape	a turtle shape name
TRUE	Logo true object
FALSE	Logo false object
[x y]	a list specifying Cartesian x and y coordinates, or width/height

Commands

Properties