

LibreLogo



Logo: born to teach

The Logo language

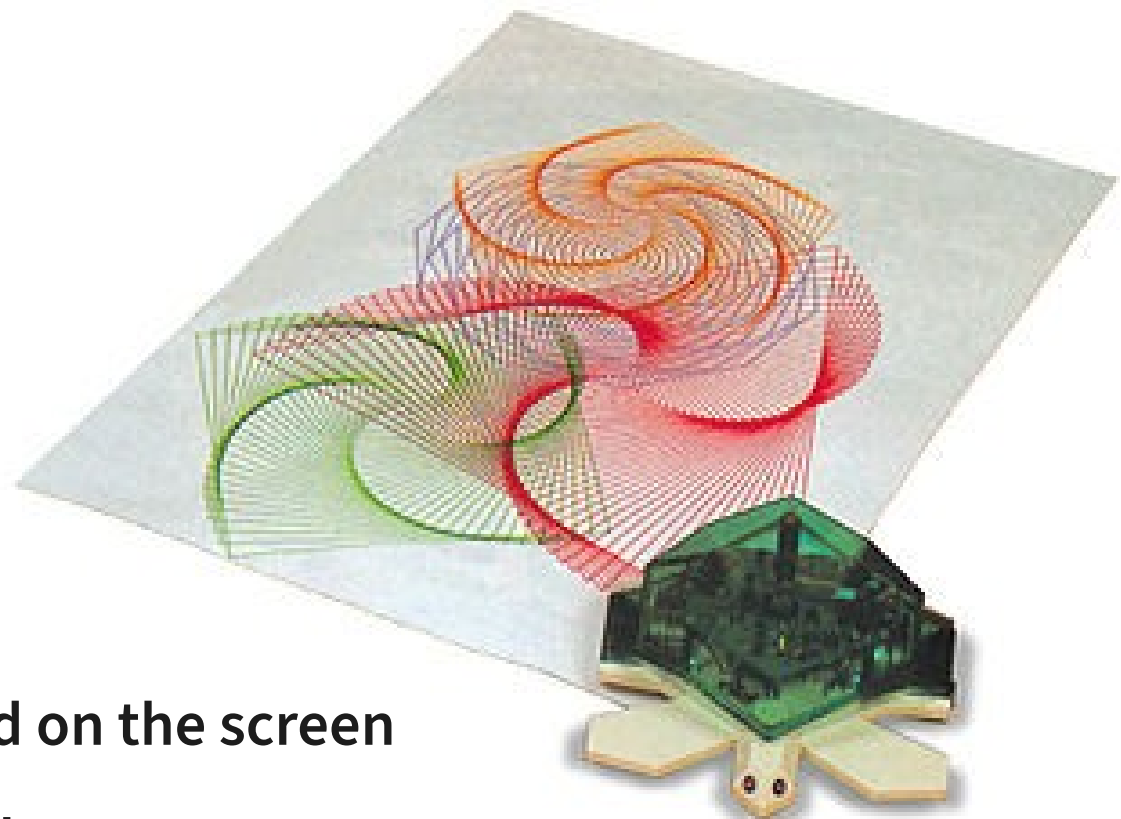
- born in '67
- initially without turtle,
later added by Papert in '70

as a physical robot, later simulated on the screen

Easy to write, inspired by the Lisp language,
created for numerical AND textual manipulation

Has inspired the Smalltalk language and the eToys system
(and now Scratch) and the Kojo system (in a future lesson)

Papert (one of the fathers of Constructivism) posed that by teaching how to
solve a problem to a computer, kids will learn how to think



Many Logo implementations

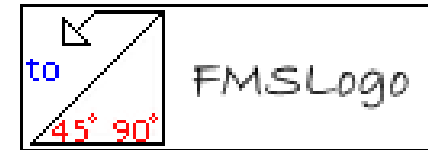
LibreLogo: a mini Logo in your text-editor (today)



NetLogo and NetLogo 3D (later)



FMSLogo: fmslogo.sourceforge.net



Browser-based:

- Papert: logo.twentygototen.org (broken)
- Malt2: etl.ppp.uoa.gr/malt2 (3D turtle)
- www.logointerpreter.com (broken)
- www.calormen.com/jslogo

QLogo: qlogo.org (QT-based)

...

LibreLogo: a small Logo in your word-processor

Turtle graphics

Global and local variables

Full **recursive functions**

Data types: word, list, array, number (but no static typing)

...

Adds: (it's converted to Python and runs in pyUNO) (**HELP**)

- interface to Python (code, sets, dicts, lists, tuples, sorted ...)

Removes:

- list-based functional programming with anonymous functions

Some LibreLogo syntax

TO function_name arg1 arg2 arg3

instructions

OUTPUT return_value

END

IF test

[code if true]

[code if false]

NOTICE:

lists use [] WITHOUT space

programs use [] WITH space

REPEAT N [

code

]

FOR var IN [list] [

code

]

WHILE test [

code

]

CONTINUE, BREAK, REPCOUNT

can be used in loops

Programming style

Imperative/procedural single-threaded

(but other Logo implementations have concurrent agents)

Functional application of anonymous functions to lists (in full Logo)

map/filter/accumulate/reduce/...

Very readable syntax (no parentheses if unambiguous)

- the parser looks for function calls FROM RIGHT TO LEFT

E.g. a b c d e = a(b(c(d(e))))

The functional style allows for very readable code (see also Scala)

Why?

Yes, it's limited, but still useful. You could:

- **Generate drawings** just in your editor (with turtle graphics)
- Show how to manipulate texts/poems in your editor
- Implement grammar rules
- Generate texts/poems/limericks (next)
- ...

Demo 1

Create a Limerick generator

A limerick is a humorous poem (often dirty) consisting of five lines

- A 7-10 syllables, same verbal rhythm A, same rhyme A
- A 7-10 syllables, same verbal rhythm A, same rhyme A
- B 5- 7 syllables, same verbal rhythm B, same rhyme B
- B 5- 7 syllables, same verbal rhythm B, same rhyme B
- A 7-10 syllables, same verbal rhythm A, same rhyme A

There was a small boy of Quebec, A (8)
Who was buried in snow to his neck; A (9)
 When they said. “Are you friz?” B (6)
 He replied, “Yes, I is— B (6)
But we don’t call this cold in Quebec” A (9) (by R. Kipling)

A limerick often:

(DEMO)

- Speaks about somebody (person)
- With some strange characteristics (adjective)
- From a place/city (origin)
- Who at a certain time (when)
- Wanted to do something (desire)
- But something else happens (event)
- Then a different outcome arise (outcome)
- “For that (person) from (origin)” closing verse

IDEA: randomly choose the needed parts from lists for each verse

BUT: we should handle agreement of person and origin
between verses (and rhyme structure)

(DEMO 2)

A limerick generator: example output

person adjective “from” origin

when desire ‘in’ place

“but” event

outcome

“that” person adjective “from” origin

A **red-headed** surgeon from Milan
Yesterday fell asleep on the Dome
But after 3 hours
He remained aside
That **small** surgeon from Milan

There is still some incoherence ... we didn't handle agreement of adjective

Demo 2

choosing the correct article for an italian word

Type: definite/indefinite (**determinativo/indeterminativo**)

Gender: male/female

Number: singular/plural

1) deduce the word gender from final char (very rough approximation)

2) select proper gender/number from final char “

3) handle Normality and Exceptions (here for indefinite male singular only)

N - starts with vowel → ”un”

N - starts with consonant → “un”

E - starts with 2 special vowels (‘ia’, ‘ie’, ‘io’, ‘iu’) → “uno”

E - starts with 1 or 2 special consonants → “uno”

(“x”, “y”, “z”, “gn”, “pt”, “ps”, “pn”, “sc”, “sf”, “sq”, “st”)

Demo

DEMO