The other art of computer programming. Milestone 2: LOGO.

1960s

Melanie Tarr

*Edith Cowan University, melanie.tarr@gmail.com*
Learn how Papert programmed a robot called a ‘turtle’

Inventing turtle graphics
Tell which shape is programmed by looking at a few lines of code

Program patterns in LOGO
Use code to make a Koch snowflake and the Hilbert pattern

1960s
ENGLISH MATHEMATICIAN SEYMOUR PAPERT CREATED AN EMBODIED UNDERSTANDING OF MATHEMATICS THROUGH THE LOGO LANGUAGE. THIS LANGUAGE, UNDERSTOOD BOTH BY HUMANS AND COMPUTERS, DESCRIBES DRAWINGS NOT IN AN ABSOLUTE CARTESIAN GRID, BUT FROM THE PERSPECTIVE OF THE PERSON THAT IS DRAWING. HE CALLED THIS TURTLE GRAPHICS.
# Table of Contents

- Turtle graphics .................................................. 3
- Letters and drawing ........................................... 4
- Procedures ........................................................ 5
- Recursion .......................................................... 6
- States ............................................................... 7
- Abstraction ........................................................ 8
- Gestalt ............................................................. 12
- Recursion .......................................................... 13
- Koch snowflake .................................................. 14
- Hilbert pattern ..................................................... 18

Note: This resource may be useful for the above DT curriculum codes and associated elaborations in the Australian National Curriculum.
ENGLISH MATHEMATICIAN SEYMOUR PAPERT CREATED AN EMBODIED UNDERSTANDING OF MATHEMATICS THROUGH HIS LOGO LANGUAGE. THIS LANGUAGE, UNDERSTOOD BOTH BY HUMANS AND COMPUTERS, DESCRIBES DRAWINGS NOT IN AN ABSOLUTE CARTESIAN GRID, BUT FROM THE PERSPECTIVE OF THE PERSON THAT IS DRAWING. HE CALLED THIS TURTLE GRAPHICS.

THE EMBODIED ROBOT TOOK COMMANDS FROM THE OPERATOR AND PERFORMED LINE ART WITH ITS OWN FORM OF GEOMETRY.

2 KINDS OF TURTLES

NOT EVERYONE COULD AFFORD A ROBOT TURTLE SO A SIMULATED TURTLE WAS USED AS WELL. THE SIMULATED TURTLE APPEARED IN THE MIDDLE OF THE COMPUTER SCREEN.

THE MECHANICAL TURTLE

THE DIGITAL TURTLE

360

THE MECHANICAL TURTLE WAS A SMALL, ROUND ROBOT THAT WAS CONNECTED BY WIRES TO THE COMPUTER. LOOKED LIKE A TURTLE.

THE DIGITAL TURTLE WAS A TURTLE DRAWN ON A COMPUTER SCREEN. THE COMMANDS TELL THE TURTLE TO DRAW LINES ON THE SCREEN.

A TURTLE ROTATED THROUGH A NUMBER OF STEPS TO DRAW A CIRCLE. THE NUMBER OF STEPS WAS 360.

TURTLE TRAILS ARE MERRY AND THE CLEAR-SCREEN COMMAND WOULD CLEAR THE SCREEN.

THE TURTLE LINE IS A TURTLE TAIL.

THE TURTLE HAS A NUMBER OF STEPS IT MOVES FORWARD.
Which trail is produced when the previous code is executed?

The above code produces the letter "N".

Turtle code words were called primitives. They could also be shortened, for example.

Sometimes it is important for the turtle not to draw.
Procedures are the steps you take to compute information from data. They are evolved from the 1950s subroutines and are also a form of abstraction.

Logo procedures are instructions you teach Logo to do. They include all the steps Logo must take to make something happen.

Abstraction
Subroutines were a very early form of abstraction within computation performed by machines.

TO CORNER

THE SUBROUTINE

the following example are steps that do convert into steps you can program or automate

non-programmable example

programmable example

the following example is one that you cannot program in LOGO however it is still a procedure.
recursion and steps to making a shape

visualising things that repeat, iterate or recurse

gestalt - closure

Closure is a psychological principle that says to our brain, “The square is drawn.”

a computer can not see

It doesn't know when to stop so it could keep drawing the square forever
states in computer science

in computer science
states work in a similar way

If a machine is in a certain state (1), it will execute a set of instructions

If a machine is in another state (2), it will execute a different set of instructions

so to program a square

we need to draw four lines and turn at a 90 degree angle each time.

TO SQUARE
REPEAT 4 [FC 20 FT 90] END

a state in logo

every machine has a start and an ending state

every machine has a start and an ending state. In this example it counts to 4 and then stops

an ending condition satisfies the machine and it stops after repeating the corner procedure 4 times

eg. I'm in a great mood so I will visit the beach

I'm feeling tired so I will have a sleep
abstraction

symbolic abstraction

One of the first examples of abstraction was Alan Turing’s universal machine

“The essence of computational thinking is abstraction. In computing, we abstract notions beyond the physical dimensions of time and space. Our abstractions are extremely general because they are symbolic.”
ABSTRACTING A CHECKERBOARD

TO DRAW 8 SQUARES IN A COLUMN YOU COULD CODE

REPEAT 8 [SQUARE FD 20]

RT 90 FD 20 RT 90

THE SQUARE PROCEDURE IS AN ABSTRACTION THAT DRAWS A COLUMN

TO TURN RIGHT AT THE FIRST TOP COLUMN TO START DRAWING THE SECOND COLUMN

REPEAT 8 [SQUARE FD 20] RT 90 FD 20 RT 90

TO ABSTRACT A COLUMN PROCEDURE

REPEAT 8 [FD 20 RT 90] RT 90 FD 20 RT 90

90

MORE ABSTRACTION

TO COLUMNS

REPEAT 8 [SQUARE FD 20] RT 90 FD 40 RT 90

REPEAT 8 [SQUARE FD 20] LT 180

END

TO CHECKERBOARD

REPEAT 4 [COLUMNS] END

1. move forward once and left once
2. repeat 360 times

IDEA OR DESCRIPTION

algorithm/pseudocode

code

shapes

This next section shows the programming of a flower drawing. The programming language in this comic is called LOGO and was created for children in the 1960s. Although LOGO can seem very basic, it is also powerful enough to use in Artificial Intelligence Systems. The instruction panels should be read down from picture to escription to algorithm to code. LOGO was written by Marvin Minsky and Seymour Papert. Marvin Minsky is also known as the father of Artificial Intelligence.

TO CONSTRUCT A CIRCLE

1. move forward once and left once
2. repeat 360 times

REPEAT 360 [FD 1 LT 1]
1. move forward once and left once
2. repeat 90 times

REPEAT 90 [FD 1 LT 1]

TO CONSTRUCT AN ARC
AND THIS IS JUST PART OF A CIRCLE
1. move forward once and left once
2. repeat 90 times

NOW GO BACK THE OTHER WAY
HINT TO TURN IS NO AS YOUR TURTLE IS ALREADY FACING WEST
3. turn left 90
4. move forward once and left once
5. repeat 90 times

LT 90
REPEAT 90 [FD 1 LT 1]

REPEAT 2 [REPEAT 90 [FD 1 RT 1] RT 90]

TO LEAF
REPEAT 2 [REPEAT 90 [FD 1 RT 1] RT 90]
END

15°
(360/24)

TO CRYSANTHEMUM
REPEAT 24 [LEAF RT 15]
END

1. run the leaf pattern
2. turn the turtle increasing it by 360/24=15 degrees
3. repeat 24 times

TO LEAF
REPEAT 2
[REPEAT 90 [FD 1 RT 1]] RT 90
END

TO CRYSANTHEMUM
REPEAT 24 [LEAF RT 15]
END

1. turn 180 degrees
2. forward 200 steps

TO STEM
RT 180
FD 200
END

circle the shape produced by the code

REPEAT 6 [FD 100 RIGHT 360/6]

REPEAT 360 [FD 1 LT 1]
gestalt - parts of a whole

The unified whole is greater than the sum of its parts.
Recursion in computer science is when a procedure calls itself as part of the solution through recursion we discover the emergence of pattern.

What steps repeat?

1. Draw a line forward a number of steps
2. Turning 90 degrees to start the next line
3. Add 5 steps to the number of steps

We also need to increase the line length by 5 steps each time we draw it, so we need to add another command in that adds 5 to the step number. This would be step 3 so our instructions to Logo now reads:

1. Draw a line forward a number of steps
2. Turning 90 degrees to start the next line
3. Add 5 steps to the number of steps

Logo remembers the number of steps it takes forward from the variable n. Variables are covered in chapter 10. The person operating the program inputs the first number of steps then Logo generates the rest by adding 5 each repeat to the number of steps. The command

\[ \text{FD :N RT 90} \]

1. FD :N
2. RT 90
3. IN +5
4. GO TO MAZE :N

Convert instructions to code

The format for this recursive procedure is as follows:

\[
\text{procedure name}<\text{arguments}> \\
\text{statements} \\
\text{repeat procedure} \\
\text{end}
\]

So fitting the code into the above format we have:

\[
\text{TO MAZE :N} \\
\text{FD :N RT 90} \\
\text{MAZE :N +5} \\
\text{END}
\]
koch snowflake

1. Start with a line
2. Divide the line into 3 equal parts
3. Draw an equilateral triangle using the middle segment as a base
4. Erase the base of the equilateral triangle
5. Repeat steps 2 - 4 until you have finished the flake

1960s Milestone two
Melanie Tarr
koch snowflake

snowflake 100 1

[repeat 3 [rt 120 side :size :level]]
(side 100 1 )
[if :level = 0 [fd :size stop]] >>>

[sidem :size / 3 :level - 1 lt 60] >>>
(side 33.33333333333333333 0 )
[if :level = 0 [fd :size stop]] >>>
side stops

[sidem :size / 3 :level - 1 lt 60] >>>
(side 33.33333333333333333 0 )
[if :level = 0 [fd :size stop]] >>>
side stops

[sidem :size /3 :level - 1 lt 60] >>>
(side 33.33333333333333333 0 )
[if :level = 0 [fd :size stop]] >>>
side stops

[side]
[if :level = 0 [fd :size stop]] >>>
side stops
(side 100 1 )

(if :level = 0 [fd :size stop]) >>>
(side :size / 3 :level - 1 lt 60) >>>
(side 33.33333333333333 0 )

(side stops
(side 100 1 )
(if :level = 0 [fd :size stop]) >>>
(side :size / 3 :level - 1 lt 60) >>>
(side 33.33333333333333 0 )
(if :level = 0 [fd :size stop]) >>>
(side :size / 3 :level - 1 lt 120) >>>
(side 33.33333333333333 0 )
[if :level = 0 [fd :size stop]] >>>
(side :size /3 :level - 1] >>>
(side 33.33333333333333 0 )
[if :level = 0 [fd :size stop]] >>>
(side stops
side stops
(side 100 1 )
hilbert pattern

hilbert 20 2

[h :size :level 1]
  ( h 20 2 1 )
  [if :lev = 0 [stop]]
  [lt :par * 90]
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 1 )
  [if :lev = 0 [stop]]
  h stops
  [fd :size]

[h :size :lev - 1 :par]
  ( h 20 0 -1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 -1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 -1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 -1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]

[h :size :lev - 1 :par]
  ( h 20 0 1 )
  [if :lev = 0 [stop]]
  h stops
  [rt :par * 90]
hilbert pattern

hilbert 20 2
hilbert pattern

hilbert 20 2
hilbert pattern

hilbert pattern

[475x544][h :size :lev - 1 0 - :par]( h 20 0 -1 )
[if :lev = 0 [stop]]
h stops
[fd :size]

[819x544][h :size :lev - 1 :par]( h 20 0 -1 )
[if :lev = 0 [stop]]
h stops
[fd :size]

[479x347][h :size :lev - 1 :par]( h 20 0 -1 )
[if :lev = 0 [stop]]
h stops
[fd :size]

[819x351][h :size :lev - 1 :par]( h 20 0 -1 )
[if :lev = 0 [stop]]
h stops
[fd :size]

[487x139][l :par * 90]
h stops
hilbert stops

[819x303][l :par * 90]
h stops
hilbert stops

[487x127][l :par * 90]
h stops
hilbert stops

[819x303][l :par * 90]
h stops
hilbert stops

[487x115][l :par * 90]
h stops
hilbert stops