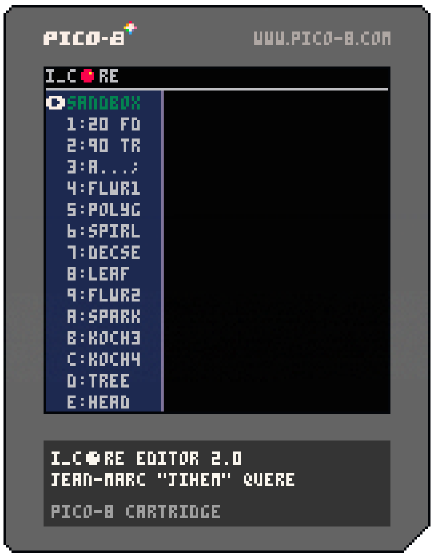
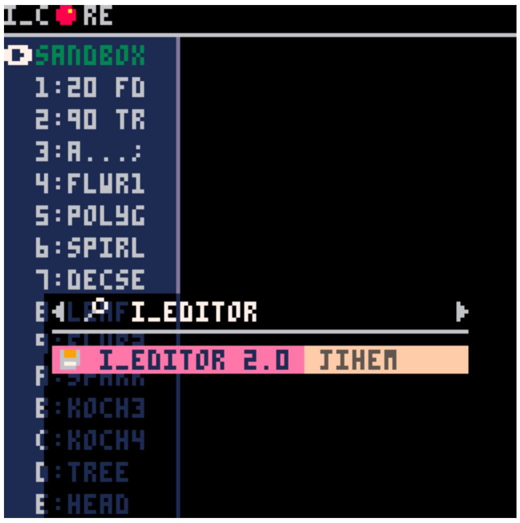


I_CORE is a programming game in which you write smart scripts to replicate drawing models.

I_EDITOR.P8.PNG

I_EDITOR is a programming game. The goal is to write simple scripts to reproduce a drawing on screen. The commands used in the scripts move a pointer ("turtle"). The pointer has a pen, which can be up or down. When the pen is down, each move lays a print. There are additional instructions to make calculus, evaluate conditions and call previous declared sequences. The syntax and semantic are organized in language called I_CORE. This documentation describes the main capabilities of this language and the editor to set properly the scripts.

STARTUP



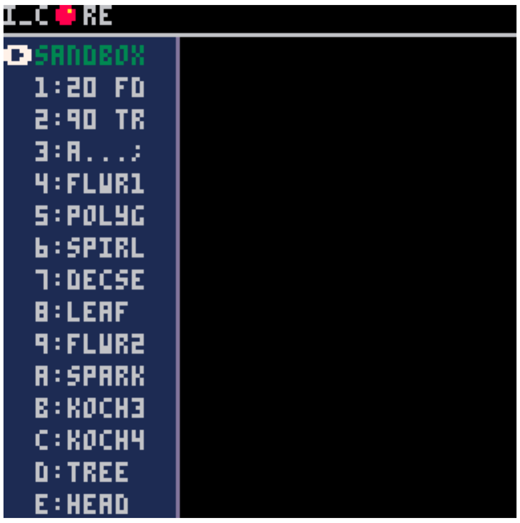
If you haven't the cartridge, the simplest way to download and start it is to use SPLORE, type:
> **SPLORE**
Navigate with ← and → to the SEARCH option (magnifying glass), select [SEARCH], type:
I_EDITOR
Select the entry "I_EDITOR n.m" (where n.m is the version number, i.e.: 2.0).

If you already have the cartridge or want to install your own (from a download on codyssea.com or lexaloffle.com web site), launch PICO-8 as usual, copy then cartridge in the carts folder, then load the cartridge:

> **LOAD I_EDITOR.P8.PNG**
and start it:
> **RUN**

☑ **YELLOW** text must be type in a PICO-8 shell.
☑To open the carts folder from PICO-8, type:
> **FOLDER**

MAIN MENU

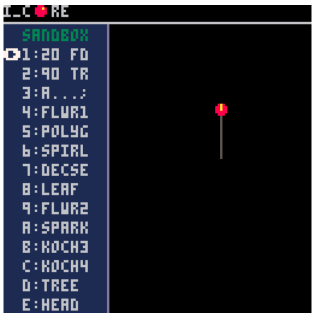


The menu options are listed on left. You can use ↑ and ↓ to select an option. To activate the selected option, press **X** button.

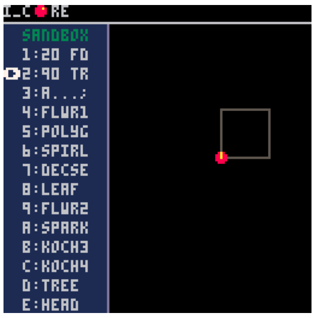
If you have forgotten, you current keyboard configuration, type:
> **KEYCONFIG**

The **SANDBOX** let you do what you want. There is no blueprint to reproduce. You can use this option to learn the language, test things or compute new puzzles. All the other options are puzzles (from 1: to E:). When you select a puzzle, his blueprint is displayed to give you an overview of the challenge. You can try to solve every puzzle when you want in the order you want. Keep in mind that the difficulty increases with the number. So, if you haven't solved a puzzle the next could be harder.

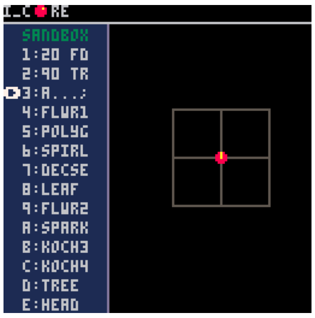
20 FD



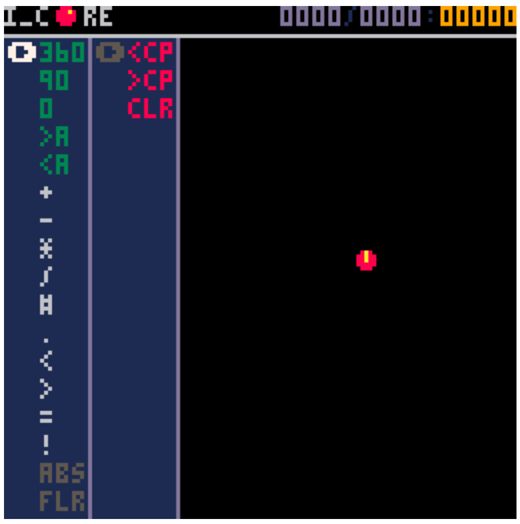
90 TR



A...;



EDITOR

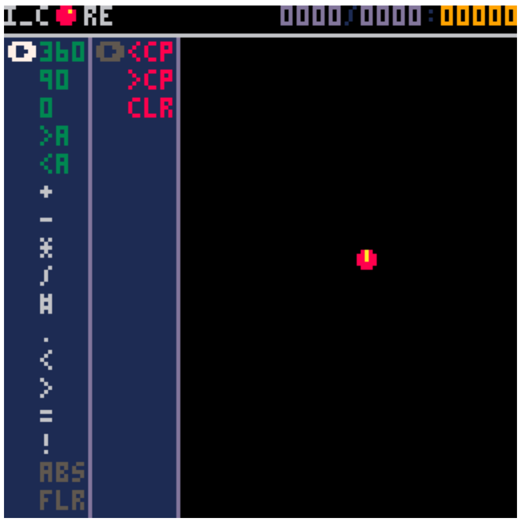


Once you have activated an option in the main menu, the editor is displayed. You can return to the MAIN MENU with the **O** button.

The editor has 4 areas (from top left, to right bottom):
- Title and statistics
- Commands list
- Current script
- Doodle area (with the pointer/turtle in the middle)

☑ The yellow line on the red turtle shows where it's heading: to the top of the screen in the picture.

TITLE AND STATISTICS



I_CORE statistics are displayed on the upper right of the screen. The violet numbers 0000:0000 indicates the current line number and the number of lines of the current script. The orange one shows the number of execution steps needed to terminate the script.

The 3 first items (in red) in the script column (2nd) are special editor commands:

<CP	Copy from clipboard
>CP	Copy to clipboard
CLR	Erase current script

☑ <CP and >CP use are restricted by PICO-8 (see pico-8.txt documentation) ; disabled in the HTML export.

EDITOR KEYS

<u>Commands list</u>

→	Go to script column
↑ ↓	Select a command
X	Insert select command after the current line in the script (marked with the cursor)
O	Go to MAIN MENU

☑ The content of the editor is saved when you exit to the main menu (for each previously chosen option). So, you can go back and forth.

<u>Script</u>

←	Go to commands list column
↑ ↓	Select a line of the script or one of the special editor commands
X	Activate Edit mode one the current script line (available only on specific commands)
O	Erase line

<u>Edit</u>

← →	Increase/Decrease value
X O	Validate

Numbers	**Variables**	**Sequences**
360	<A	:A
90	>A	A
0		?A

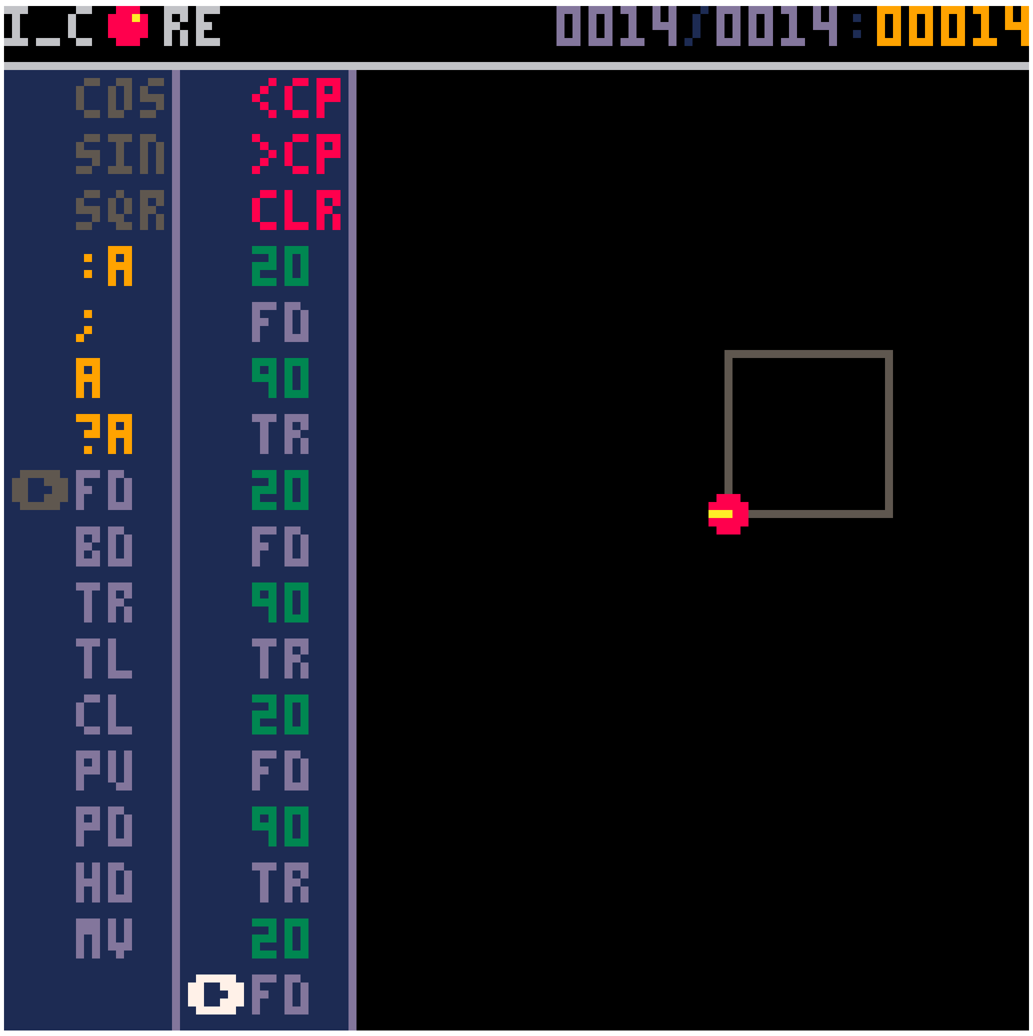
Numbers can be edited to change the value (+/-1). Variables and sequences names can be edited from A to C (and C to A).

DOODLE AREA

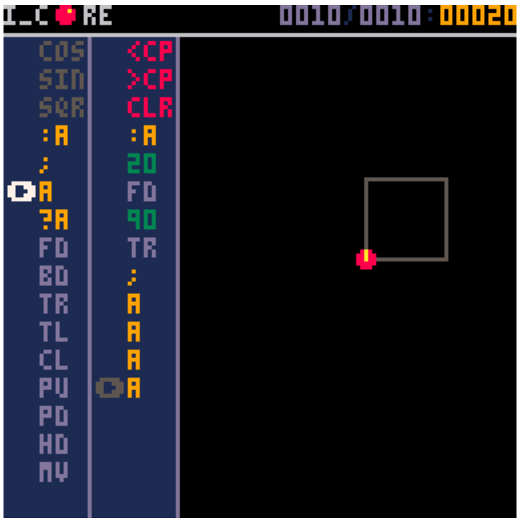
Each time the script is modified, it is restarted (real time). When drawing command are performed, the result is displayed (live) in the doodle area. So, you see exactly what happens when you program.

LANGUAGE REFERENCE

The turtle can move forwards (FD) or backward (BD). The distance unit is the pixel, i.e.: **20 FD** move the turtle 20 pixels forward. The turtle can turn left (TL) or right (TR) from an angle (in degree), i.e.: **90 TR** makes the turtle turn 90° right (right angle).

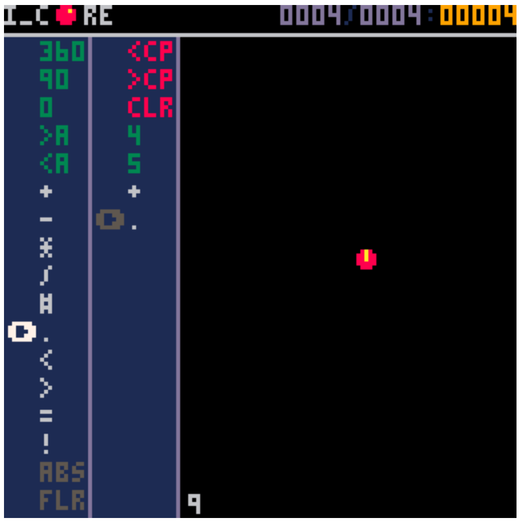


```
<CP
>CP
CLR
20
FD
90
TR
20
FD
90
TR
20
FD
90
TR
20
FD
```



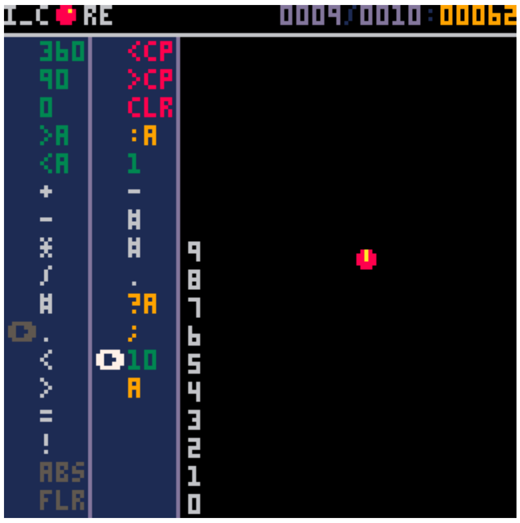
```
I_C●RE      0010/0010:00020
COS   <CP
SIN   >CP
SQR   CLR
:A    :A
;     20
A     FD
?A    90
FD    TR
BD    ;
TR    A
TL    A
CL    A
PU    A
PD
HD
MV
```

When a sequence of instructions needs to be repeated, a name can be associated to it (A, B or C). The set of instructions begins with the prefix **:A**, **:B** or **:C**. The character **;** ends the set. The letters **A**, **B** and **C** execute the set (declared with **:A**, **:B** or **:C**). The length of the script to draw the square can be optimized… but it costs more execute steps (20 ⇔ 14).



```
I_C●RE      0004/0004:00004
360   <CP
90    >CP
0     CLR
>A    4
<A    5
+     +
-     .
*
/
#
.
<
>
=
!
ABS
FLR
                9
```

The language I_CORE can handle numbers and basic operators: **+ - * /**. I_CORE is a tiny stack based programming language. The program are made with sequences of instructions separated by a space, i.e: **4 5 + .** The two first numbers are pushed in the stack, then the operator **+** takes them and compute the addition and pushes the result in the stack. The last instruction **.** takes the last value pushed and show it on the console.



```
I_C●RE      0009/0010:00062
360   <CP
90    >CP
0     CLR
>A    :A
<A    1
+     -
-     #
*     #
/     .
#     ?A
.     ;
<     10
>     A
=
!
ABS
FLR
                9
                8
                7
                6
                5
                4
                3
                2
                1
                0
```

?A, **?B** and **?C** are conditional calls. If the last value entered in the stack is not zero, the sequence (A, B or C) is executed. The stack is LIFO (last in first out). **#** duplicates the last entry (in the stack).

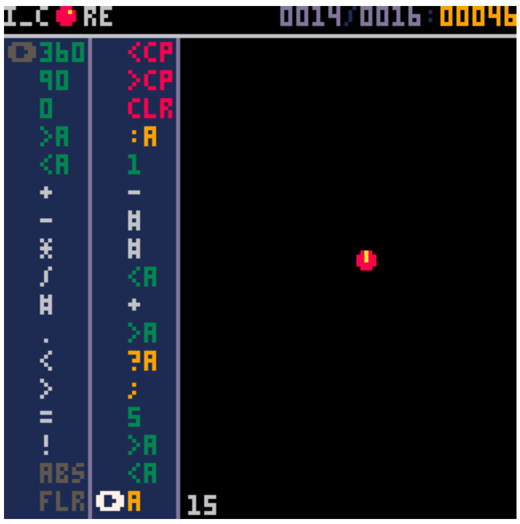
10 A pushes 10 in the stack and calls A.

:A (stack before → stack after):

1 -	subtracts 1 to the last values (10 → 9),
# #	duplicates 2 times the last value (9 ➔ 9 9 9)
.	shows the last value (9 9 9 → 9 9)
?A	calls A if last value <> 0 (9 9 → 9) ⇔ 9 A

>A	put the last value (from the stack) in the variable A
<A	put the content of the variable A into the stack

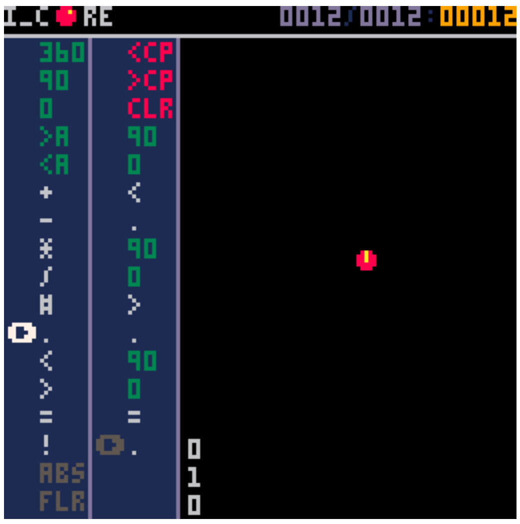
☑ Variables need to be initialized (not equal to 0 by default).



```
I_C●RE              0014/0016:00046
   360    <CP
    90    >CP
     0    CLR
    >A     :A
    <A     1
     +     -
     -     #
     *     #
     /     <A
     #     +
     .     >A
     <     ?A
     >     ;
     =     5
     !     >A
   ABS     <A
   FLR     A    15
```

`:a 1 - # # <a + >a ?a ; 5 >a <a a <a .`

evals 5 + 4 + 3 + 2 + 1 ➔ 15



```
I_C●RE              0012/0012:00012
   360    <CP
    90    >CP
     0    CLR
    >A    90
    <A    0
     +    <
     -    .
     *    90
     /    0
     #    >
     .    .
     <    90
     >    0
     =    =
     !    .     0
   ABS          1
   FLR          0
```

`90 0 < .`

Is 90 lesser than 0 ? No (0)

`90 0 > .`

Is 90 greater than 0 ? Yes (1 ⇔ not 0)

`90 0 = .`

Is 90 equal to 0 ? No (0)

`!` is the unary operator ***not*** : `90 0 = ! .`

Is 90 not equal to 0 ?

Hum… Let me think about it… Yes ?

MATH

ABS	Absolute value, `- 9 ABS .` → 9
FLR	Floor, `10 3 / FLR .` → 3
COS	Cosinus, `0 COS .` → 1
SIN	Sinus, `90 SIN .` → -1
SQR	Square root, `9 SQR .` → 3

☑ Trigonometric circle upside down because y axis grows from top to bottom.

TURTLE

CL	Change Color (0 → 15), `9 CL` I like orange color ;-)
PD	Pen Down: turtle prints when it moves, `PD`
PU	Pen Up: turtle doesn't print when it moves, `PU`
HD	Heading: set turtle direction (0 → 360), `90 HD`
MV	Move turtle to position (x, y), `0 0 MV`

The doodle area is a 116x80 pixels grid: 0 0 top left, 79 115 bottom right. You can draw outside this area but you will not see some part of the result.

There are a lot to do with I_CORE language. Now, it's your turn to explore… In the hope you will have a good journey ;-)

SCORES

20 FD	0002:00002	90 TR	0012:00034	A… ;	0020:00148
FLWR1	0026:00422	POLYG	0031:00438	SPIRL	0014:00442
DECSE	0026:00849	LEAF	0016:00198	FLWR2	0026:01226
SPARK	0031:01069	KOCH3	0067:00439	KOCH4	0067:01783
TREE	0020:02034	HEAD	0092:04176		

Less is beautiful

- * -

Many thanks to Zep for doing PICO-8
and to the others who have survived my coding addiction.
Kind regards

Jihem