Main page
Community portal
Language list
Browse by category
Recent changes
Random page
Help

Tools

What links here
Related changes
Special pages
Printable version
Permanent link
Page information

Piet

Piet is a stack-based esoteric programming language in which programs look like abstract paintings. It uses 20 colors, of which 18 are related cyclically through a lightness cycle and a hue cycle. A single stack is used for data storage, together with some unusual operations.

Piet was invented by David Morgan-Mar and is named after geometric abstract art pioneer Piet Mondrian.

"Hello World" in Piet.

Contents [hide]

Computational class

If the interpreter for Piet allows for an arbitrarily large stack with each item holding an arbitrarily large number, then Piet is Turing-complete. In addition, a brainfuck interpreter has been created in Piet, and since brainfuck is Turing-complete, Piet is also Turing-complete.

Execution

Codels

A codel in Piet is like an image's pixel. Some Piet programs are upscaled, meaning that a codel might not always be equivalent to 1 pixel, but a codel *is* always a substitute for pixels.

Color blocks

A color block is any group of codels of the same color that are adjacent to each other. Note that codels only touching each other diagonally are *not* considered part of the same color block; they must be touching in one if the 4 cardinal directions to be part of the same color block.

Direction pointer

The direction pointer (DP) is what moves along the program to make it run. It can be in any one of the 4 cardinal directions. The direction pointer always starts at the color block containing the upper-left-most codel, and always starts facing right. After it has executed the proper command, it will move on to the next color block that is both:

a. adjacent to the current color block, and
b. is the farthest in the direction of the DP.

This continues until the program terminates (see below).

Codel chooser

The codel chooser (CC) is used when multiple color blocks meet the above two criteria for the next block to be executed. Its direction is always relative to the DP's direction, and starts out facing left. When there are more than one possible color blocks to be executed, the one farthest in the direction of the codel chooser (again, relative to the DP) is the one chosen. The codel chooser can only point left or right.

Colors

Piet uses 20 colors in its programs. Each of these colors (with the exceptions of white and black) have two properties, those being hue and lightness. All colors and their properties are shown in the table below.

Colorful

Light red (#FFC0C0)	Light yellow (#FFFFC0)	Light green (#C0FFC0)	Light cyan (#C0FFFF)	Light blue (#C0C0FF)	Light magenta (#FFC0FF)
Red (#FF0000)	Yellow (#FFFF00)	Green (#00FF00)	Cyan (#00FFFF)	Blue (#0000FF)	Magenta (#FF00FF)
Dark red (#C00000)	Dark yellow (#C0C000)	Dark green (#00C000)	Dark cyan (#00C0C0)	Dark blue (#0000C0)	Dark magenta (#C000C0)

Hue is shown going to the left and lightness is shown going down. Note that these properties are cycles, meaning that, in terms of hue, red comes after magenta. Hue always goes to the left, and lightness always goes down, meaning that going from yellow to red is 5 changes in hue, and vice versa.

White

White (#FFFFFF) is one of the two colors in Piet that doesn't fit into either cycle. White color blocks act like blank spaces. When the DP encounters a white block, it will simply go through it and move on to the next color block. No commands are executed when the DP goes through a white block.

Black

Black (#000000) is like the opposite of white in the sense that the DP cannot pass through it. If the DP tries to go to the next color block but fails because of a black block, it will switch the CC to its other state and try again. If it still can't get to the next color block, then the DP will be rotated one step clockwise. If the DP has gone through all possible states but it still can't get to the next color

block, it will conclude there is no way out and the program will terminate. This is the only way to terminate a Piet program.

Commands

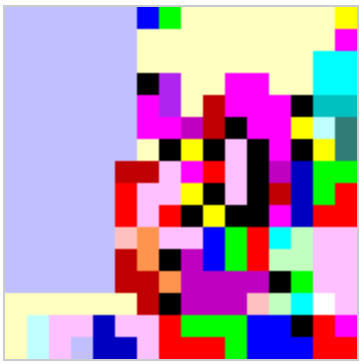
Piet commands aren't executed based on the color the DP is on, but instead based on the change in lightness and hue. Below is a table with all 17 commands in Piet and how they're executed.

Hue change	Lightness change		
	No change	**1 darker/2 lighter**	**2 darker/1 lighter**
No change	N/A	Push	Pop
1 step	Add	Subtract	Multiply
2 steps	Divide	Modulo	Not
3 steps	Greater	Pointer	Switch
4 steps	Duplicate	Roll	Input num
5 steps	Input char	Output num	Output char

- **Push:** Pushes the number of codels in the previous color block onto the stack.
- **Pop:** Pops the top value off the stack.
- **Add:** Pops the top two values off the stack, adds them up, and pushes the sum back onto the stack.
- **Subtract:** Pops the top two values off the stack, subtracts the top value from the second-top value, and pushes the difference back onto the stack. Note that if the top value is X and the next value Y, this means that Y - X will be pushed, not X - Y.
- **Multiply:** Pops the top two values off the stack, multiplies them together, and pushes the product back onto the stack.
- **Divide:** Pops the top two values off the stack, performs integer division (Python equivalent of //) on the second-top value divided by the top value, and pushes the quotient back onto the stack. This has the same X/Y property as subtraction.
- **Modulo:** Pops the top two values off the stack, divided the second-top value by the top value, and pushes the remainder back onto the stack. This has the same X/Y property as subtraction.
- **Not:** Pops the top value off the stack. If the value is 0, it pushes 1 onto the stack. Otherwise, it pushes 0.
- **Greater:** Pops the top two values off the stack. If the second-top value is greater than the top value, it pushes 1 onto the stack. Otherwise, it pushes 0. This has the same X/Y property as subtraction.
- **Pointer:** Pops the top value off the stack, then rotates the DP one step clockwise that many times (anti-clockwise if the value is negative).
- **Switch:** Pops the top value off the stack, then switches the state of the CC that many times (absolute value if the value is negative).
- **Duplicate:** Pushes a copy of the top value onto the stack.
- **Roll:** Pops the top two values off the stack, and then rotates the top Y values on the stack up by X, wrapping values that pass the top around to the bottom of the rolled portion, where X is the first value popped (top of the stack), and Y is the second value popped (second on the stack). (Example: If the stack is currently 1,2,3, with 3 at the top, and then you push 3 and then 1, and then roll, the new stack is 3,1,2.)

- **Input:** Takes an input, either as a character or a number. If the input is a number, that value is pushed onto the stack. If it's a character, its Unicode value is pushed onto the stack.
- **Output:** Pops the top value off the stack. If a number should be printed, the value itself will be printed. If a character should be printed, then its corresponding Unicode character will be printed.

Examples



A Hello World script that capitalizes the words of "Hello" and "World", as well as includes an exclamation point, unlike the more famous Hello World Piet program.

Notes

In 2022, a museum found out that it had a draft of a painting of Piet Mondrian hanging upside down. Unlike his finished artworks, the draft was not an oil painting, but made of colored strips of paper glued to a white canvas, so that Mondrian can experiment by moving them. Since the draft is mechanically fragile, the museum didn't reorient the piece.

The situation above is quite identical to the Piet language since it is almost invaried in case of turning programs upside down: the only time that the language notices is when it initializes the program counter. If a Piet program were found to somehow have turned upside down, you would not need to adjust its orientation. You'd just add a painted frame that initializes the program correctly, entering the bottom left codel through a white block with the direction set to left and the codel chooser set to left.

See also

- Piet++
- PIET
- Pietfood

External resources

- Piet website
- npiet - an interpreter and editor for Piet
- Piet interpreter & editor
- PietDev - An online Piet editor and debugger (Currently Unavailable)
- Encode Piet with ASCII
- Dana Connell made 99 bottles of beer in Piet
- Fizzbuzz in piet , produced using a code generator
- A puzzle about a Piet, of which the solution is produced by a program in Piet. Created by hand.
- Piet interpreter - An interpreter with GUI written in functional Python, by Jens Bouman
- PietPlus - A JavaScript-based IDE and interpreter. (GitHub repo)
- DMM's blog entry on a domain name that he does not own but used to have a HTTP redirect to his description of Piet

Categories: Languages | Stack-based | Turing complete | Two-dimensional languages | Implemented | Non-textual | Low-level | Thematic | 2001