

CoffeeScript: Pencilcode.net



Andrea Sterbini – sterbini@di.uniroma1.it

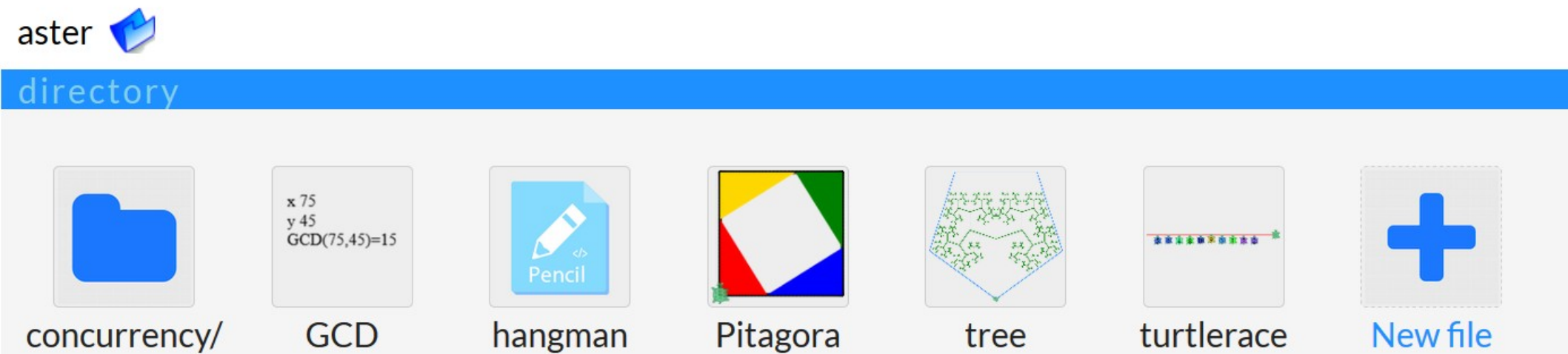
Pencilcode: CoffeeScript language (aka Javascript)

Editor with both **textual** and **block-based** editing

Turtle graphics, music, speech (and also the Processing.js lib!)

Input, print, picture display

Your personal web site (e.g. <http://aster.pencilcode.net>)
showing/running your programs



CoffeeScript = Readable Javascript

CoffeeScript translates to Javascript

Adds some features from Perl/Python/Ruby:

- indentation instead than curlies {} and semicolons ; Python
- list comprehension Python
- pattern matching (multiple assignment) Python
- argument packing/unpacking Python
- postfix syntax available for if/for/switch Perl
- interval comparison Python
- literate programming using Markdown

Iced CoffeeScript adds async interactions with ‘await/defer’

Easy-enough interaction with JS libs (Jquery, Processing, D3 ...)

Function definition with '->'

All functions are primitive objects and return their last value

Iterative version of GCD

```
GCD = (x, y) ->  
  # multiple assignment + postfix conditional loop  
  [x, y] = [y, x%y] until y is 0  
  # the last value computed is returned  
  x
```

Recursive version

```
GCD = (x, y) ->  
  # inline if + recursion + return last value  
  if y!=0 then GCD(y, x%y) else x
```

All function calls have at least 1 argument (use 'do' when 0-args)

Lists, arrays and dictionaries (and generators)

```
song = ["do", "re", "mi", "fa", "sol"]
```

```
singers = {Jagger: "Rock", Elvis: "Roll"}
```

```
Bitlist = [
```

```
  1, 0, 1
```

```
  0, 0, 1
```

```
  1, 1, 0
```

```
]
```

Generators using the
Pythonic **yield** syntax

dictionary/object in **YAML syntax**

```
Kids =
```

```
  brother:
```

```
    name: "Max"
```

```
    age: 11
```

```
  sister:
```

```
    name: "Ida"
```

```
    age: 9
```

More

Lexical scoping (var scope = same block/indentation) Python

Splats (...) allows for

- Variable args functions * in Python

```
# "others" gets the remaining args
```

```
LOSERS = (gold, silver, bronze, others...) ->  
others
```

- List unpacking * in Python

```
all_elements = [ group_1..., group_2... ]
```

- Object/Dictionary unpacking "update" in Python

```
currentUser = { user..., status='logged' }
```

Asynchronous code with await/defer

‘await’ wraps a call and waits for completion ‘defer’ring assignment

Example:

search for 'keywords' then callback 'cb' with an array of the results

SERIAL SEARCH

```
serialSearch = (keywords, cb) ->  
  out = []  
  for k,i in keywords  
    await search k, defer out[i]  
    # each waits for prev. compl.  
  cb out
```

PARALLEL SEARCH

```
parallelSearch = (keywords, cb) ->  
  out = []  
  await  
    for k,i in keywords  
      search k, defer out[i]  
  # cp wait for completion of all  
  cb out
```

Programming styles

Programming style:

- procedural? not so much
- functional? YES
 - all procedures return something (their last value)
 - functions can be passed as values and used in map/filter...
- object oriented? YES (with prototypes like in JavaScript)
- concurrent
 - “await” execution / “defer” control to assignment of the result
 - e.g. sync between animation “plans”

(DEMO)

Activities: **Pencilcode Gym**

DRAW: draw turtle graphics

JAM: play music with keyboard/piano interaction
generate music or new sounds


IMAGINE: write interactive fiction (multiple-ended stories)

Using other Javascript libraries ...

GlowScript - 3D shapes

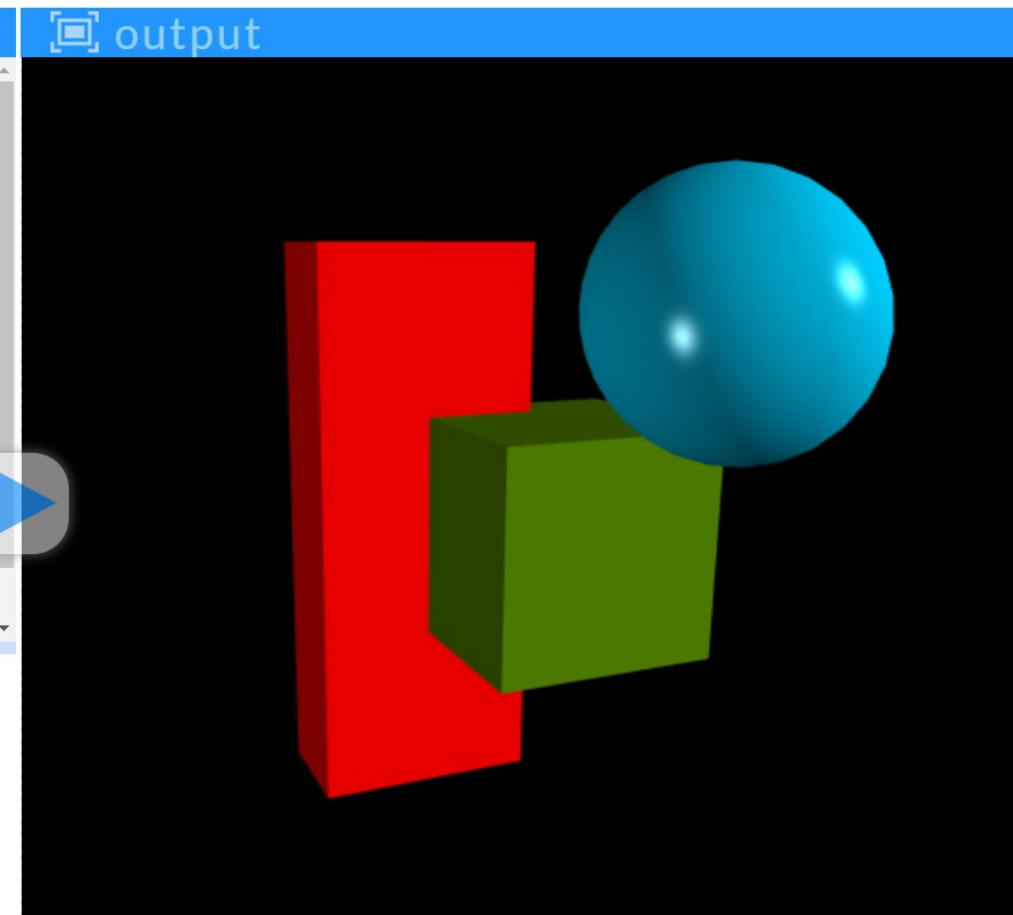
aster  3D

 Save  Share  New  Log out  ?  Guide

```
{ } code 
```

```
1 # to write 3D code you need also the javascript libs
2 # jQuery, jQueryUI, glow (GlowScript)
3 red = vec(1, 0, 0) # RGB red = 1, 0, 0
4 b = box # build a box
5   pos: vec(0, 0, 0) # at the origin
6   size: vec(2, 5, 1) # with sides 2, 5, 1
7   color: red # and red color
8
9 aqua = vec(0, 0.8, 1) # RGB aqua = 0, 0.8, 1
10 s = sphere # make an hellipsoid
11   pos: vec(1, 2, 3) # here
12   size: vec(1, 2, 3) # with radii 1, 1, 1
13   color: aqua
14 green = vec(0.3, 0.5, 0) # RGB green = 0.3, 0.5, 0
15 c = box # build a box
16   pos: vec(1, 0, 1) # in this position
17   size: vec(2, 2, 2) # cube with side 2
18   color: green # green
```


```
1 <!DOCTYPE html>
2 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4
  .1/jquery.min.js" type="text/javascript"></script>
3 <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1
  .12.1/jquery-ui.min.js" type="text/javascript"></script>
4 <script src="https://rawgit.com/davidbau/glowjs/master/dist
  /glow.js" type="text/javascript"></script>
```



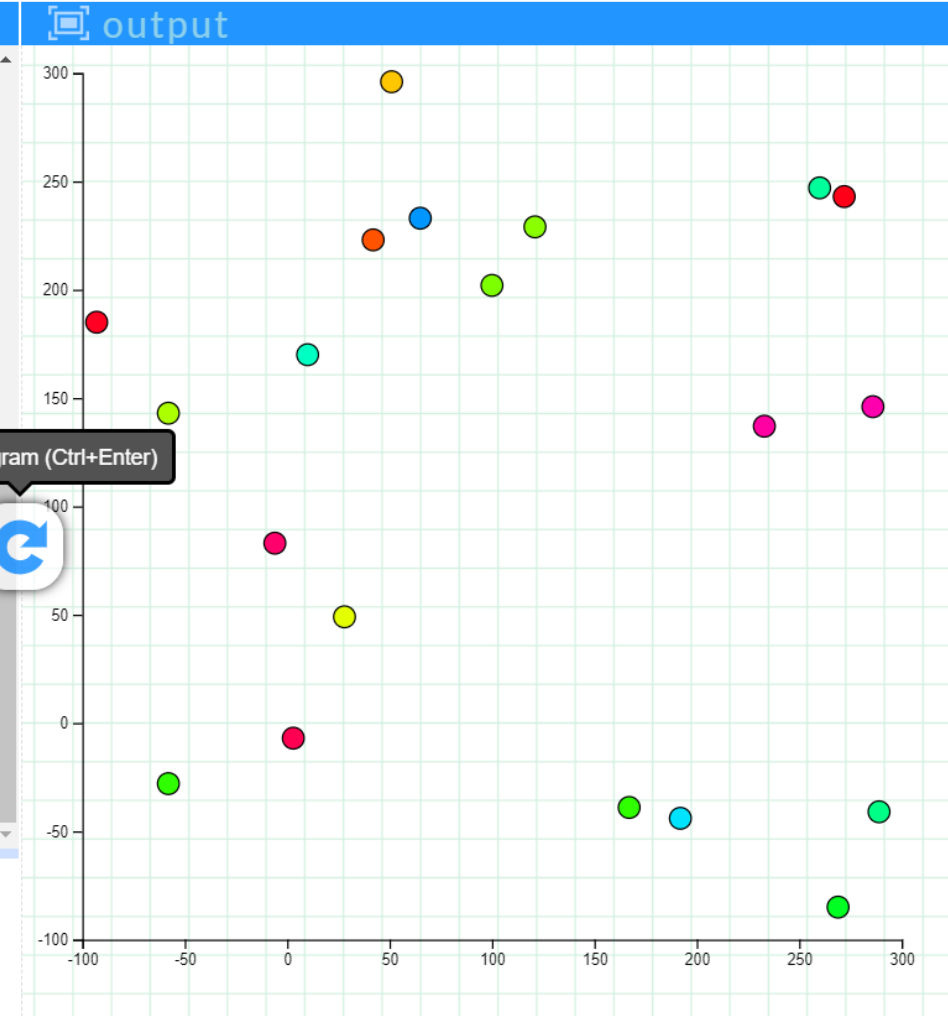
... other JavaScript libraries

D3.js - data visualization

aster  D3js

 Save  Share  New  Logout

```
code
30
31 svg
32   .append('g')
33   .attr("transform", "translate(0," + height + ")")
34   .call(d3.axisBottom(x));
35
36 # X scale and Axis
37 y = d3.scaleLinear()
38   .domain([min, max])      # This is the min and the max of the data: 0
39   .range([height, 0]);    # This is the corresponding value I want in
40   Pixel
41 svg
42   .append('g')
43   .call(d3.axisLeft(y));
44 # Add 3 dots for 0, 50 and 100%
45 svg
46   .selectAll("whatever")
47   .data(data)
48   .enter()
49   .append("circle")
50   .attr("cx", (d) -> x(d.x) )      # scale wrt X axis
51   .attr("cy", (d) -> y(d.y) )      # scale wrt Y axis
52   .attr("r", 7)                    # dot size
53   .style("stroke", (d) -> "black")  # dot border
54   .style("fill", (d) -> d.color)   # dot color
55
56
57
1 <!DOCTYPE html>
2 <!-- Add a svg area, empty -->
3 <div id="scatter_area"></div>
4 <!-- Load d3.js -->
<script src="https://d3js.org/d3.v4.js"></script>
```



HTML

Demo

DEMO