CoffeeScript: Pencilcode.net

Andrea Sterbini – sterbini@di.uniroma1.it

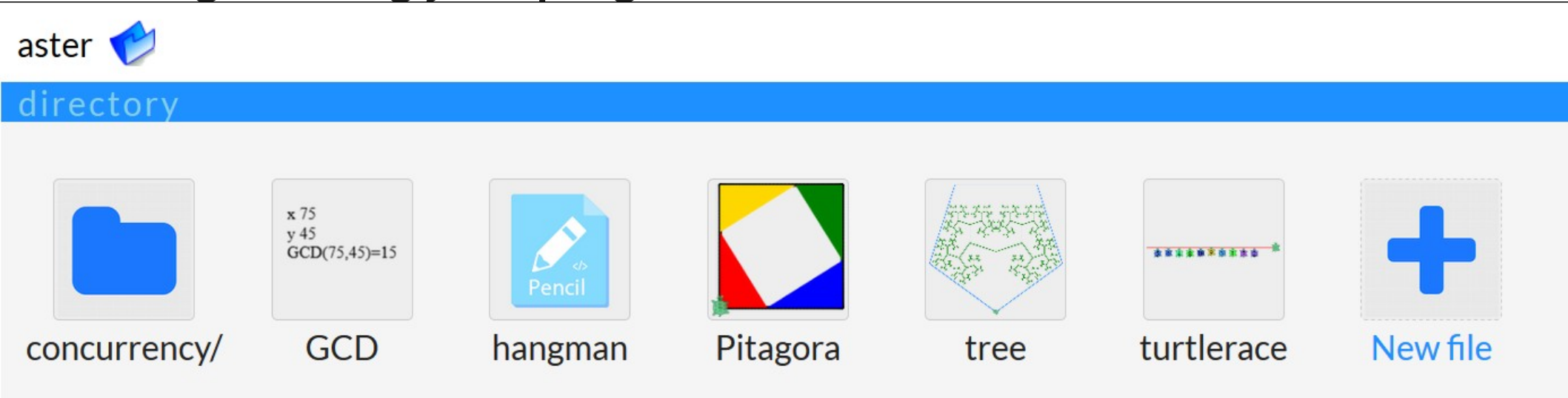
Pencilcode: CoffeeScript language (aka Javascript)

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 (Python)

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)

Data structures:
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

# ranges using double dot
[start .. end]
```

Loops and list comprehensions

Loops with automatic enumeration and guard clause

```
for element, index in list when index % 2        # print odd index values
    print element, index

# select odd index elements (list comprehension)
OddPos = [ element for element, index in list if index%2 ]
```

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”or ** in Python

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

Iced Coffescript

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

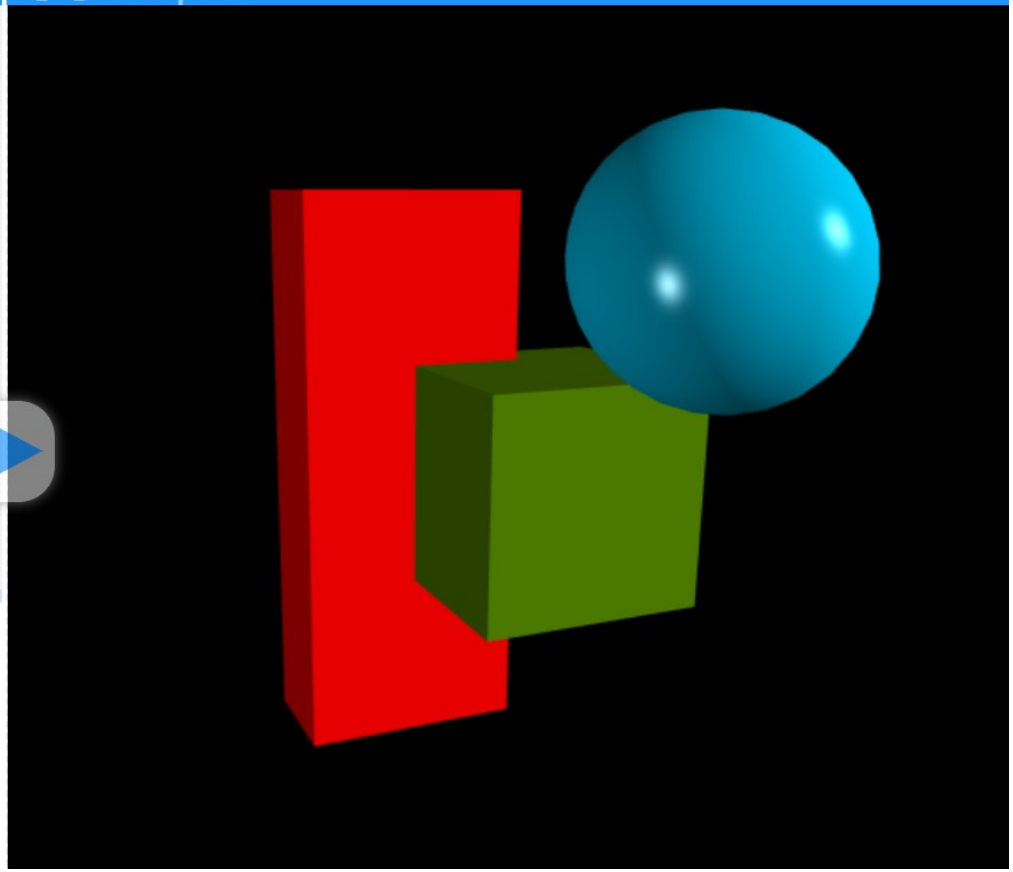
{} code

```
# to write 3D code you need also the javascript libs
# jQuery, jQueryUI, glow (GlowScript)
red = vec(1, 0, 0)            # RGB red = 1, 0, 0
b = box                       # build a box
  pos: vec(0, 0, 0)           # at the origin
  size: vec(2, 5, 1)          # with sides 2, 5, 1
  color: red                  # and red color

aqua = vec(0, 0.8, 1)         # RGB aqua = 0, 0.8, 1
s = sphere                    # make an hellipsoid
  pos: vec(1, 2, 3)           # here
  size: vec(1, 2, 3)          # with radii 1, 1, 1
  color: aqua
green = vec(0.3, 0.5, 0)      # RGB green = 0.3, 0.5, 0
c = box                       # build a box
  pos: vec(1, 0, 1)           # in this position
  size: vec(2, 2, 2)          # cube with side 2
  color: green                # green
```

HTML

```
<!DOCTYPE html>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js" type="text/javascript"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js" type="text/javascript"></script>
<script src="https://rawgit.com/davidbau/glowjs/master/dist/glow.js" type="text/javascript"></script>
```

output



… other JavaScript libraries

D3.js - data visualization

aster D3js

Save | Share | New | Log out

{} code

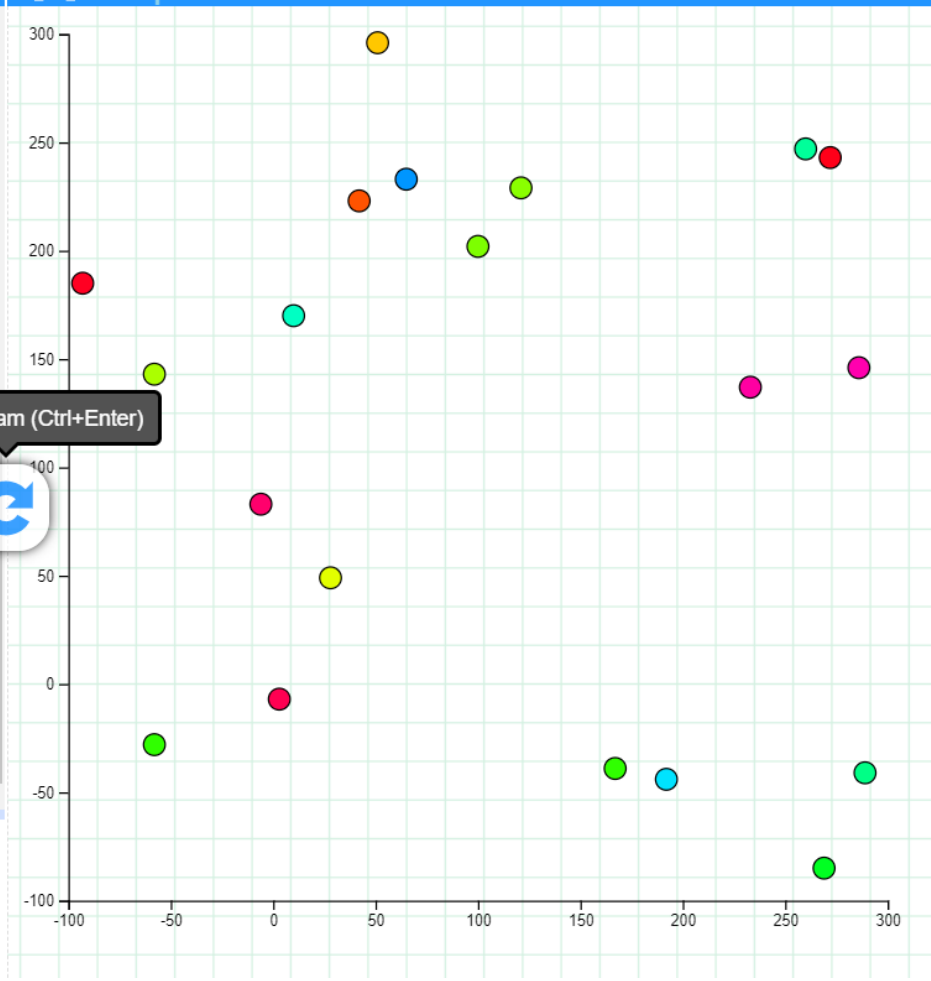
```
svG
  .append('g')
  .attr("transform", "translate(0," + height + ")")
  .call(d3.axisBottom(x));

# X scale and Axis
y = d3.scaleLinear()
    .domain([min, max])       # This is the min and the max of the data: 0 to 100 if percentages
    .range([height, 0]);      # This is the corresponding value I want in Pixel
svG
  .append('g')
  .call(d3.axisLeft(y));

# Add 3 dots for 0, 50 and 100%
svG
  .selectAll("whatever")
  .data(data)
  .enter()
  .append("circle")
    .attr("cx", (d) -> x(d.x) )          # scale wrt X axis
    .attr("cy", (d) -> y(d.y) )          # scale wrt Y axis
    .attr("r", 7)                        # dot size
    .style("stroke", (d) -> "black")     # dot border
    .style("fill",   (d) -> d.color)     # dot color
```

Restart program (Ctrl+Enter)

```
<!DOCTYPE html>
<!-- Add a svg area, empty -->
<div id="scatter_area"></div>
<!-- Load d3.js -->
<script src="https://d3js.org/d3.v4.js"></script>
```

HTML

output



Demo

DEMO