Hybrid programming

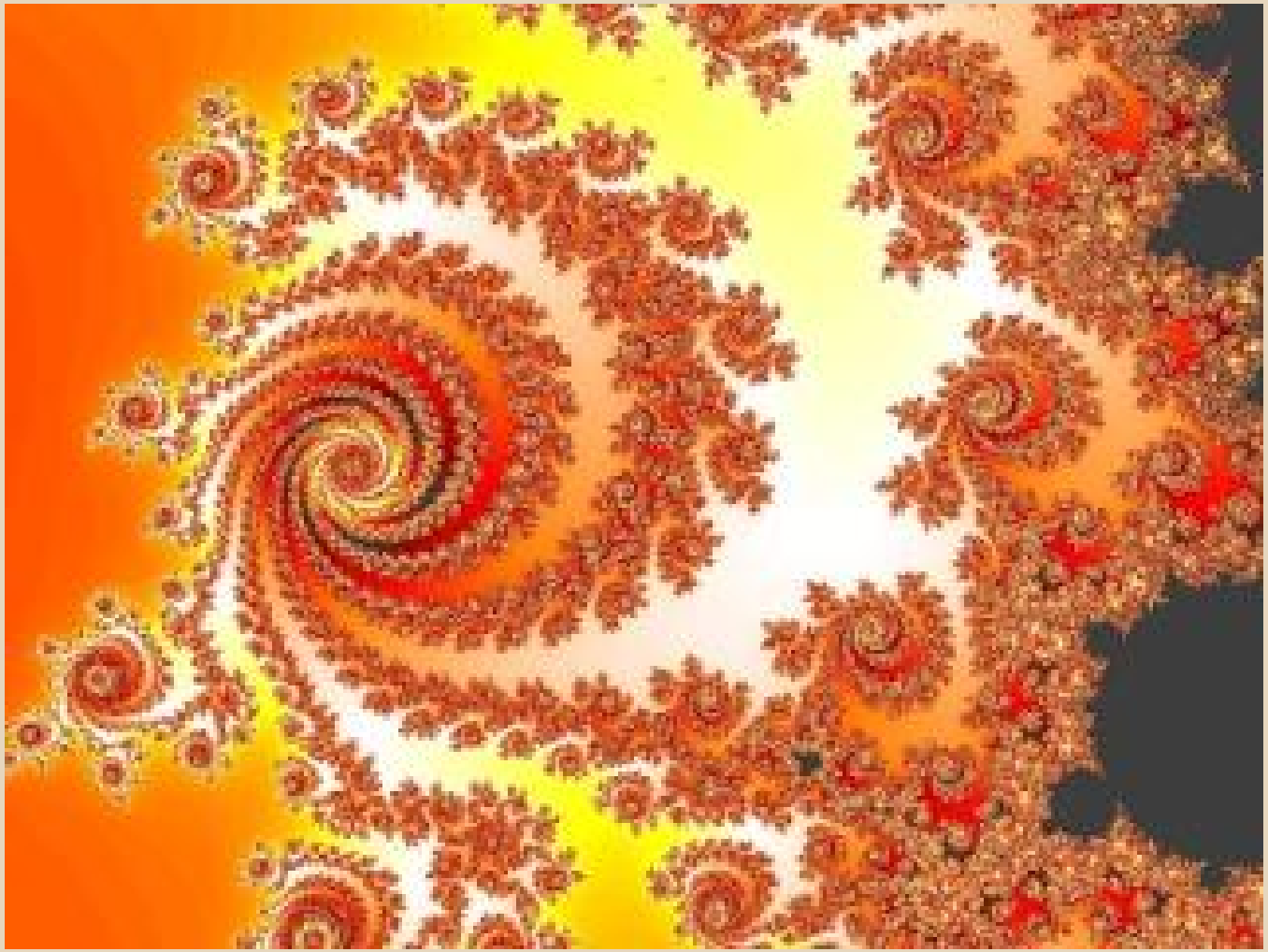
Aleksandra (Ola) Mierzejewska

aleksandra.mierzejewska@gmail.com

Agenda

1. Discussion about choice of language
2. Introduction to Lua
3. Example of languages mixing

A new project

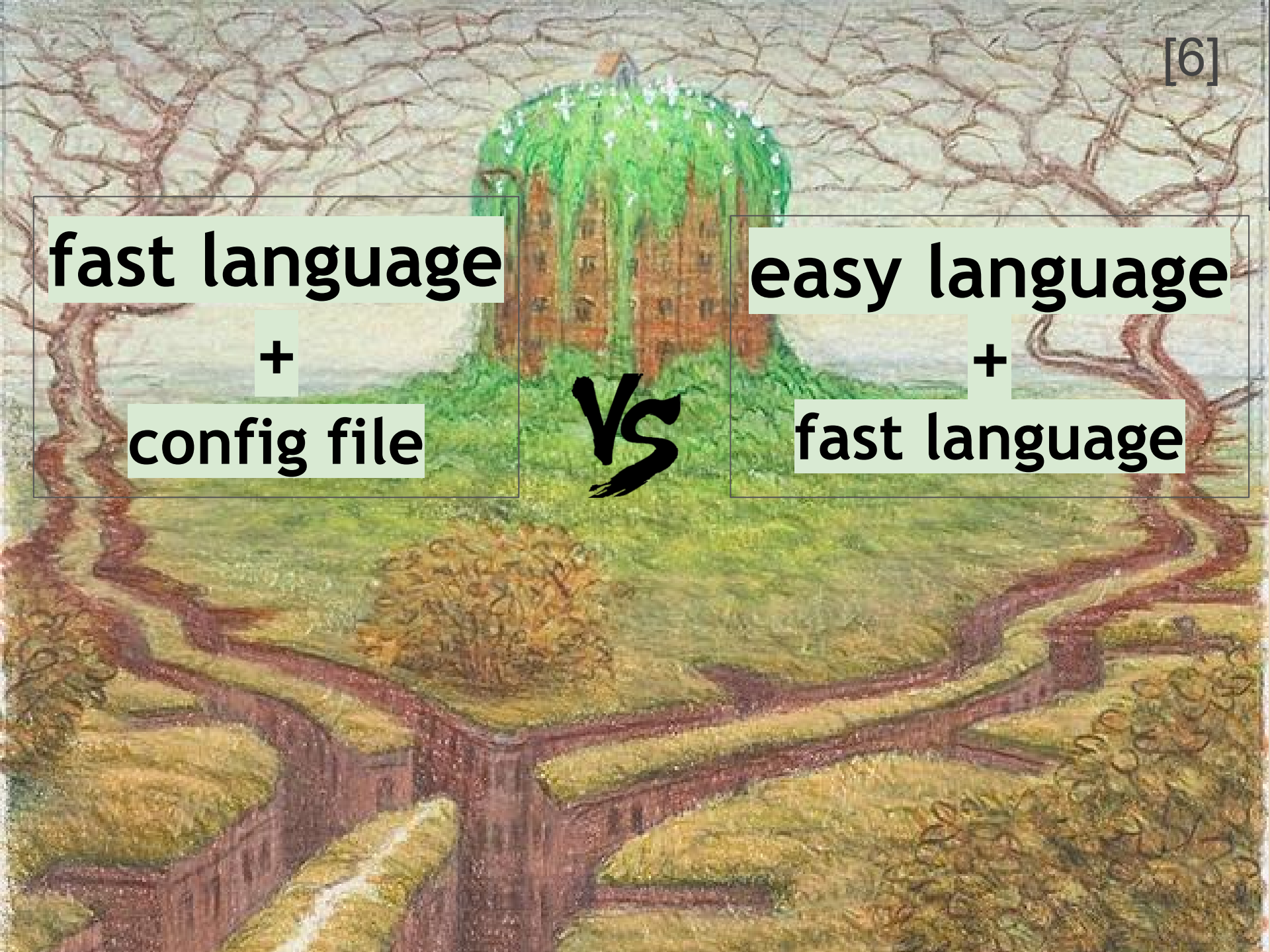


"The disadvantage of believing that all programming languages are equivalent is that it's not true. But the advantage is that it makes your life a lot simpler. And I think that's the main reason the idea is so widespread."

'Revenge of the Nerds' - Paul Graham [2]

How to choose language?

- **existing codebase**
- **project requirements/domain**
- **portability/platform support**
- experience in the team
- productivity of the language
  - complexity of language
  - tools available
  - support/stability
  - libraries
  - 'productivity' of the language


fast language
+
config file
VS
easy language
+
fast language

Plan

- Prototype in *'productive'* language
- Profile, find bottlenecks
- Rewrite parts in *'fast'* language

FAST

- C
- C++

- C#
- Java

- Haskell
- Lisp
- Scala
- Pascal
- Fortran

PRODUCTIVE

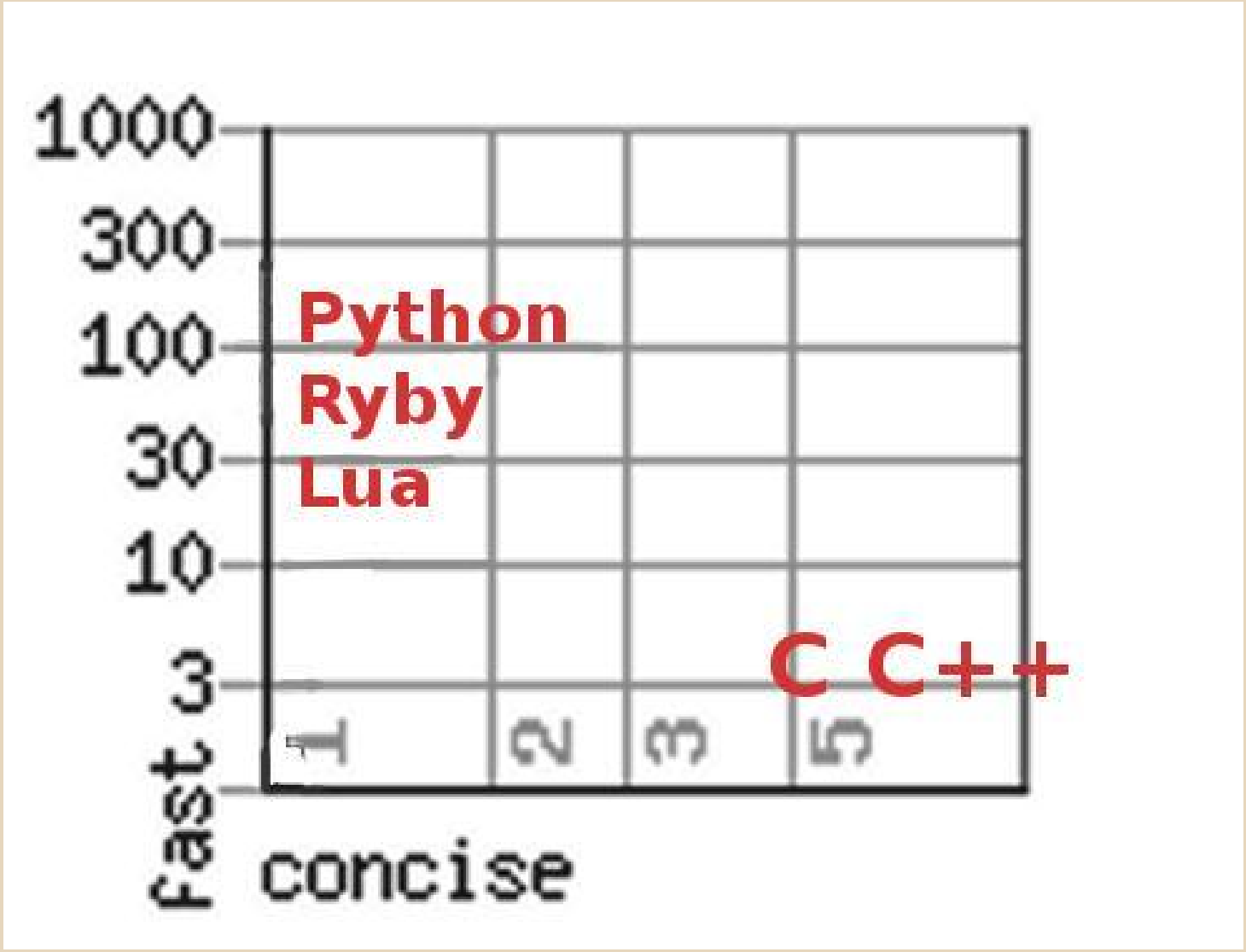
- Python
- Ruby

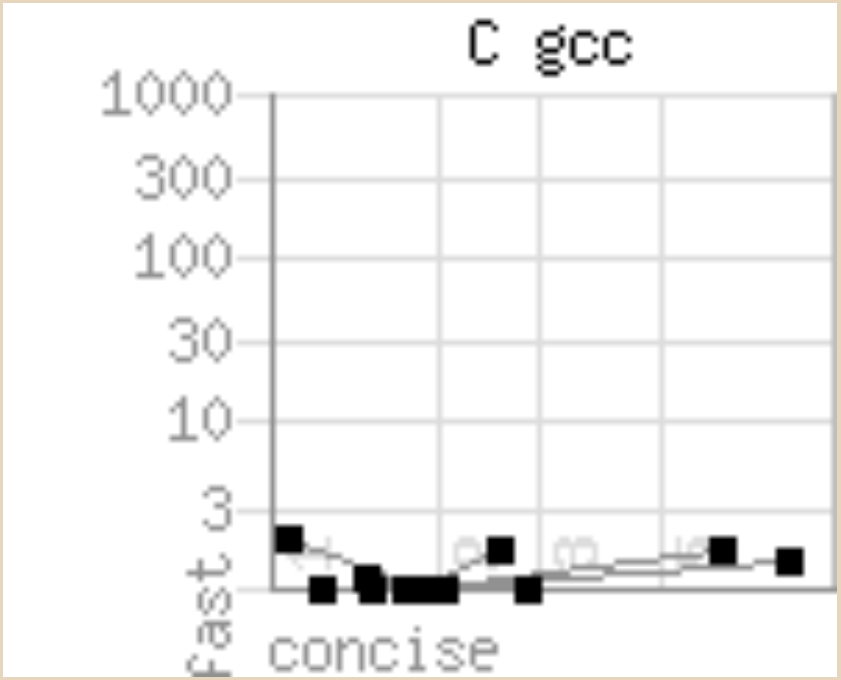
- Lua

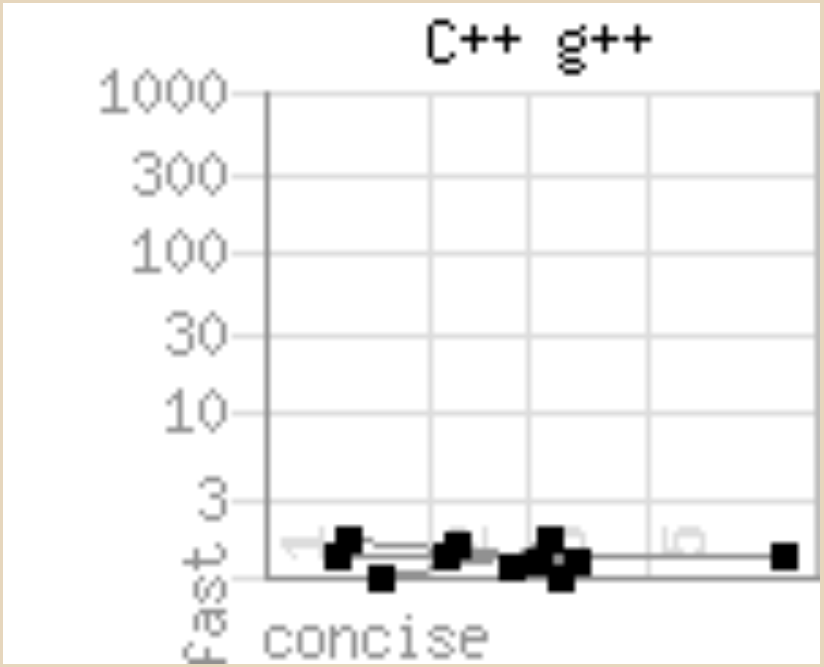
- Perl
- JavaScript (V8)
- PHP
- VB

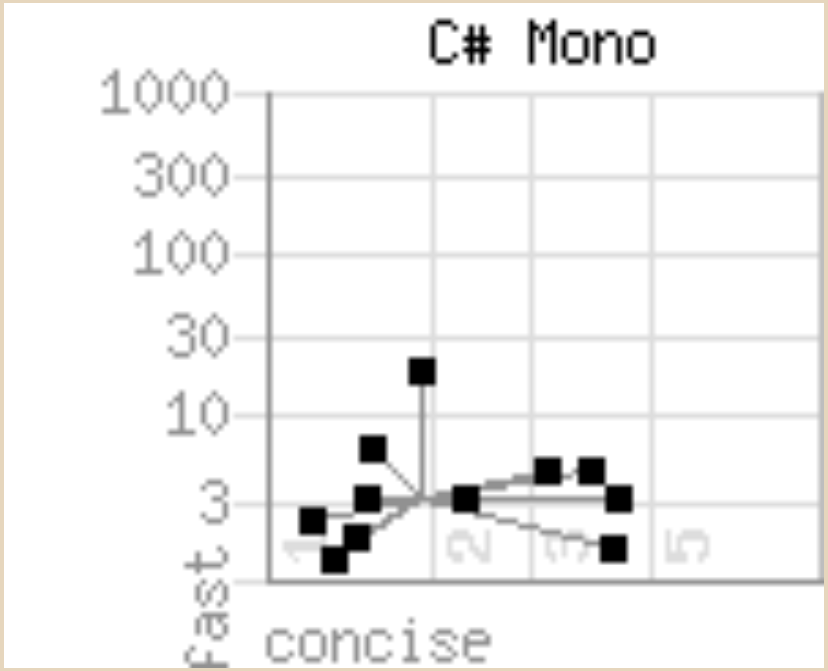
The Computer Language Benchmarks Game

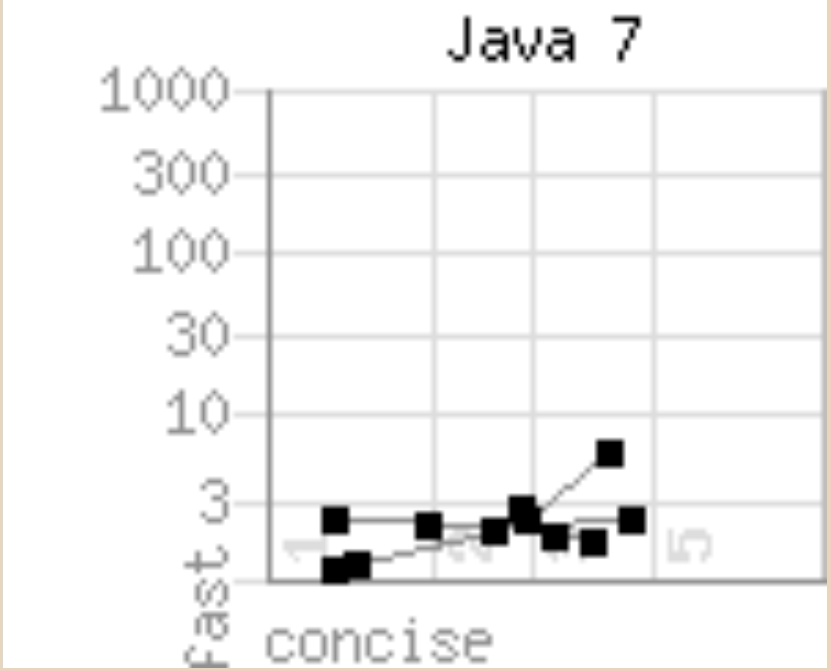
[19 March 2013]

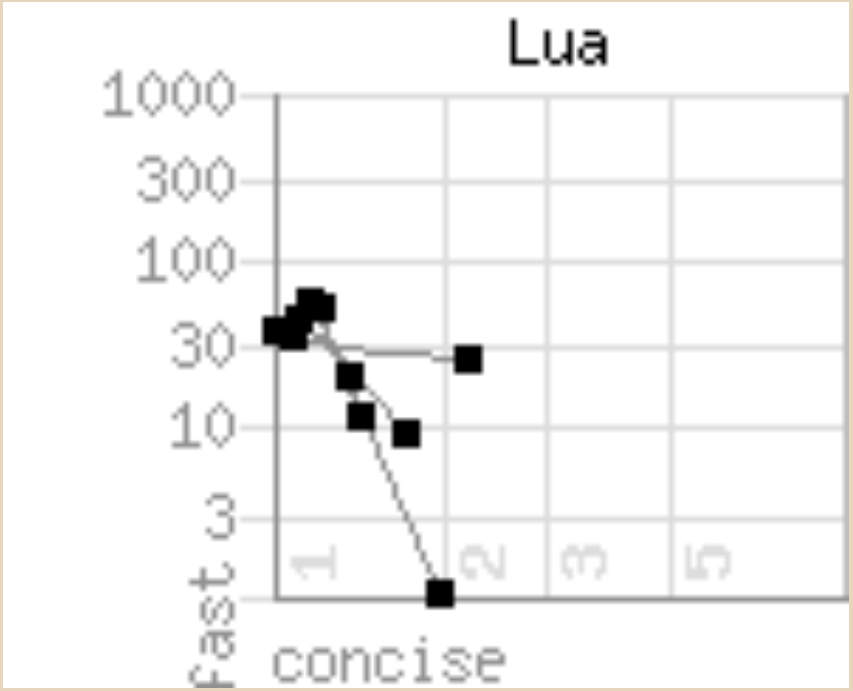


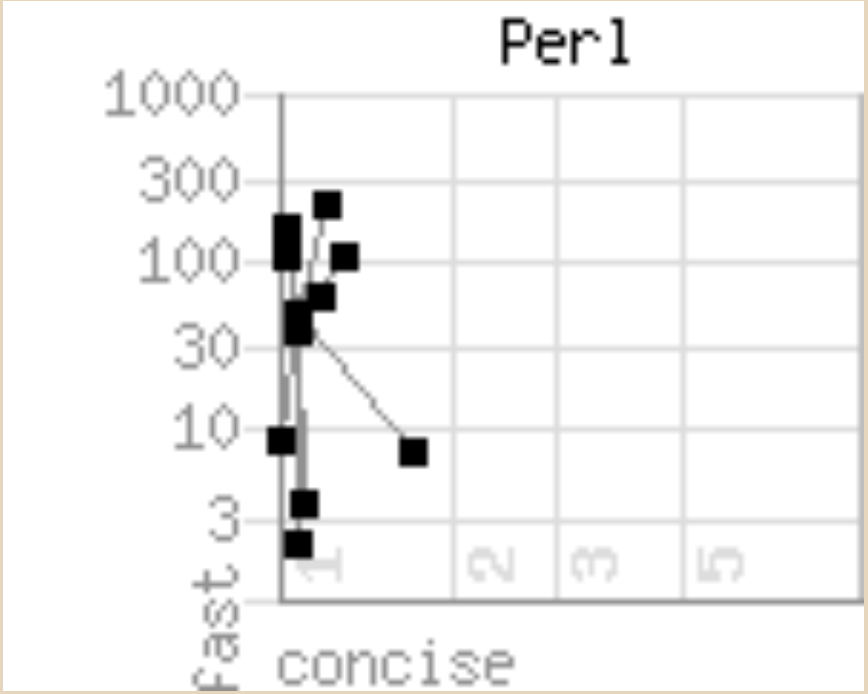

C gcc
1000
300
100
30
10
3
Fast
concise

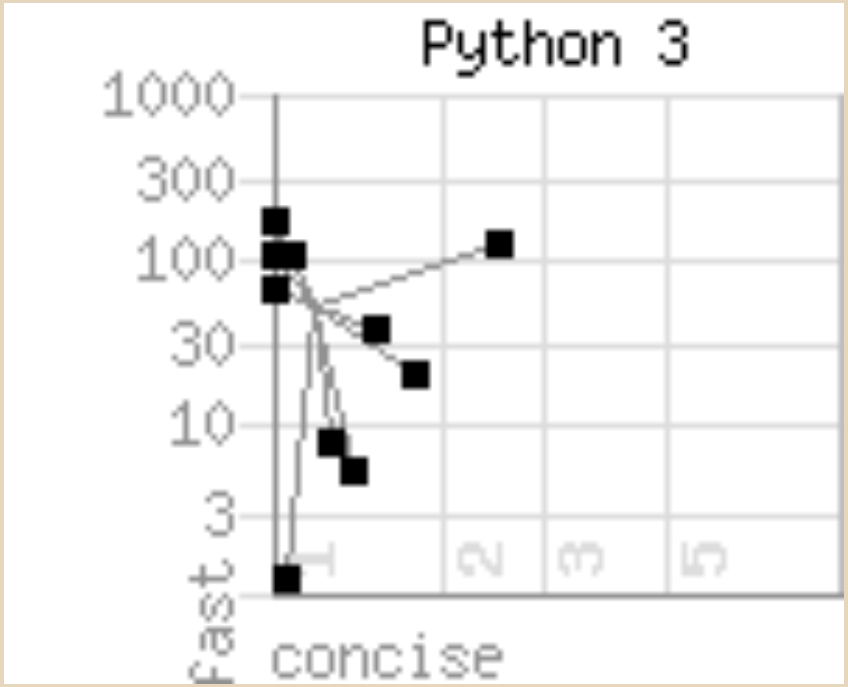

C++ g++
1000
300
100
30
10
3
Fast
concise

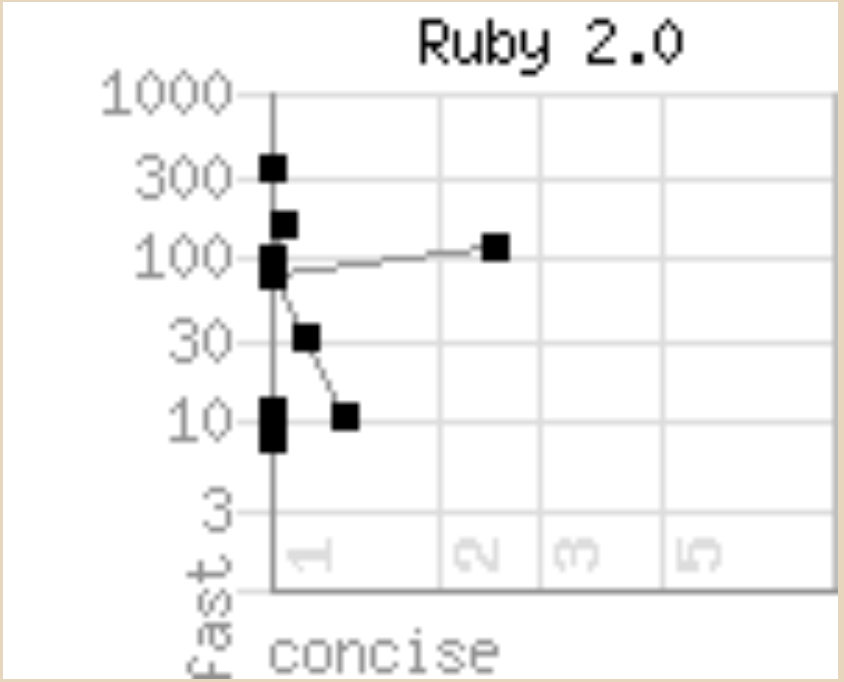

C# Mono
1000
300
100
30
10
3
Fast
concise


Java 7
1000
300
100
30
10
3
Fast
concise


Lua
1000
300
100
30
10
3
1
2
3
5
fast
concise


Perl
1000
300
100
30
10
3
1
2
3
5
fast
concise


Python 3
1000
300
100
30
10
3
1
2
3
5
fast
concise


Ruby 2.0
1000
300
100
30
10
3
1
2
3
5
fast
concise

FAST

- C
- C++

- C#
- Java

- Haskell
- Lisp
- Scala
- Pascal
- Fortran

PRODUCTIVE

- Lua
- Python
- Ruby

- Perl
- JavaScript (V8)
- PHP
- VB

Mixing languages

- Communication layer
  - SOAP
  - ICE
  - D-Bus
  - CORBA
- Multi-lingual runtime
  - .Net's CLR
- Foreign Function Interface

C&C++

```
extern "C" {
  foo(int i, char c)
  #include "my-C-code.
h"
}

int main()
{
  foo(7, 'x');
  ...
}
```

```
#ifdef __cplusplus
extern "C" {
#endif

< your C header file>

#ifdef __cplusplus
}
#endif
```

Lua *(do latim* ***Luna****) é o único satélite natural da Terra, situando-se a uma distância de cerca de 384.405 km do nosso planeta. Seu perigeu máximo é de 356.577 km, e seu apogeu máximo é de 406.655 km. [3]*

[4]

Why Lua?

- Fast
- Proven
  - WoW, Adobes Photoshop Lightroom, embeded
- Small size
  - (<1MB)
- Simple, yet powerful
  - meta-mechanisms
- Portability
  - ANSI C compliler
- Free
  - MIT licence

Lua - Tiobe Index

Mar 2013	Mar 2012	Delta in Position	Programming Language	Ratings Mar 2013	Delta Mar 2012	Status
19	21		Lua	0.697%	+0.17%	B

March 2013

Lua
The basics

```
--[[  A simple chunk of code
      Types in Lua              ]]--

print(type (a))           -- nil
a = "Hello Lua!"
print(type (a))           -- string
local b = 2
print(type (b))           -- number/nil
print (type (b == a))       -- boolean
print(type(print))        -- function
print (type ({A, B, C})) -- table
--[[   --userdata  --thread  ]]--
```

```
--[[   Strings  ]]--

--[[example of Lua code: ]]--
some_string = [[ very...
   ...long...
   text]]
str1 = 'Here we can use: " without escape.'
str2 = "Here we can use: ' without escape."

print(#some_string)   -- 29
```

```
--[[   Strings  ]]--

giant_string = [=[
--[[example of Lua code: ]]--
some_string = [[ very...
   ...long...
   text]]
str1 = 'Here we can use: " without escape.'
str2 = "Here we can use: ' without escape."
]=]
print(#giant_string)     -- 171
```

```
--[[  Tables: "The data structure" ]]--

lua_details = {"Roberto Ierusalimschy",
"Waldemar Celes",
"Luiz Henrique de Figueiredo";
["v_1.0"] = 1993,
["v_5.2.1"] = 2012}

lua_details["graphics"] = "love"
lua_details.graphics2D = "love"

lua_details.graphics2D = nil

print (lua_details[1])
                  --Roberto Ierusalimschy
```

```
--[[   Expressions, Statements   ]]--


print(4~=5)             -- true
print(not nil)          -- true
print(not false)        -- true
print(not 0)            -- false


-- multiple assignment
x, y, z = 1, 2
x, y = y, x
print(x, y, z)          -- 2, 1, nil
```

```
--[[   Control Structures   ]]--

for i = 1, 4 do print(i) end        --1 2 3 4
for i = 3, 0, -1 do print(i) end    --3 2 1 0
-- i is local to the for
for k,v in pairs(t) do print(k,v) end

--[[ And also:
        if-then-else
        while
        repeat-until
        break
        return
        goto
        do-end
   ]]--
```

```lua
--[[   Functions   ]]--

function sum_mult(...) do
   local s, m = 0, 1
   for k, v in ipairs{...} do
      s = s + v
      m = m * v
   end
   return s, m
end

print(sum_mult(3, 3, 0)) -- 6  0
```

```
--[[      Closures          ]]--

function newCounter()
   i = 0
   return function ()
               i = i + 1
               return i
            end
end

local c1 = newCounter()
print(c1)         -- function: 0x1227c90
print(c1())       -- 1
print(c1())       -- 2
local c2 = newCounter()
print(c2())       -- ?
```

```
--[[      Closures          ]]--

function newCounter()
   local i = 0
   return function ()
               i = i + 1
               return i
            end
end

local c1 = newCounter()
print(c1())                   -- 1
print(c1())                   -- 2
local c2 = newCounter()
print(c2())                   -- 1
print(c1())                   -- 3
```

```
--[[      Sandboxes      ]]--

local oldSin = math.sin
local k = math.pi/180
math.sin = function (x)
   return oldSin(x*k)
end



local oldOpen = io.open
```

Lua - other topics

- Coroutines
- Metatables
- Metamethods
- Environment
- OO programming
- Weak Tables, Finalizers

- Standard library
  - Math
  - Bitwise
  - Table
  - String
  - IO
  - OS
  - Debug

Call Lua from C

```
--[[   Lua as config file:
       graph_config.lua        ]]--


  width = 200
  height = 400
```

```
$luac -o p.lc p.lua
$lua p.lc
```

bin

lua luac

include

lua.h luaconf.h lualib.h lauxlib.h lua.hpp

lib

liblua.a

man/man1

lua.1 luac.1

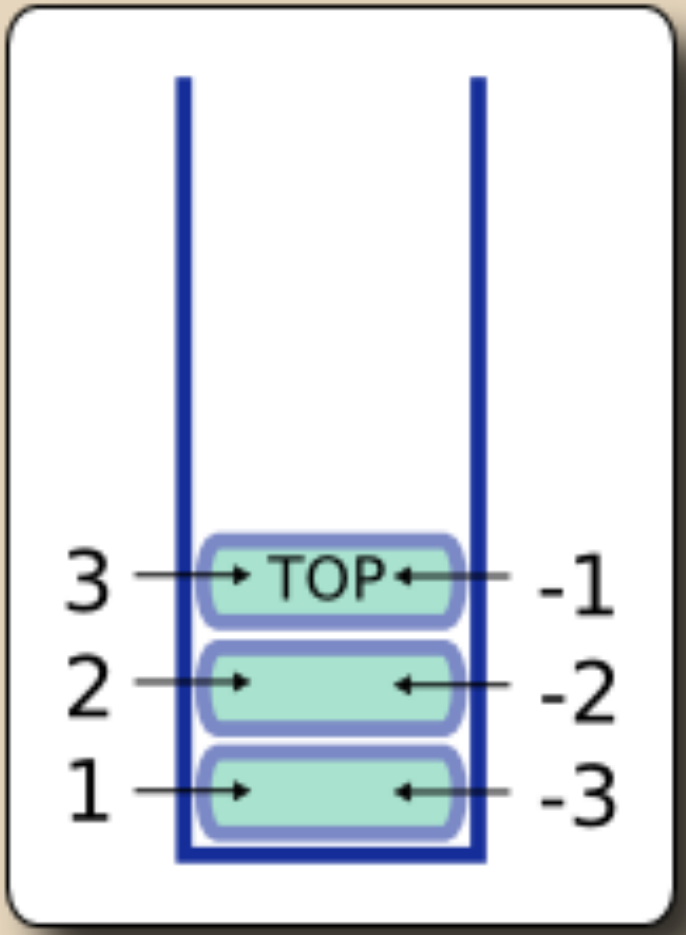
```
gcc graph.c -o graph -I/usr/local/include -
L/usr/local/lib -llua -lm -ldl
```

```c
#include "lua.h"
#include "lauxlib.h"
#include "lualib.h"


int main()
{
    lua_State *L = luaL_newstate();
    luaL_openlibs(L);
    int w, h;
    load(L, "graph_config.lua", &w, &h);
    printf("w: %d, h: %d\n", w, h);
    lua_close(L);
}
```

```
void load (lua_State *L, const char *fname, int *w,
int *h)
{
  luaL_loadfile(L,fname);
  lua_pcall(L,0,0,0);

  lua_getglobal(L, "width");
  lua_getglobal(L, "height");

  *w = lua_tointeger(L, -2); //(L, 1)
  *h = lua_tointeger(L, -1); //(L, 2)
}


// w: 200, h: 400
```

Lua stack

lua_push*
lua_checkstack
lua_pop
lua_gettop
lua_settop
lua_pushvalue
lua_remove
lua_insert
lua_replace
lua_copy



```
void load (lua_State *L, const char *fname,
                                  int *w, int *h)
{
  if(luaL_loadfile(L,fname) || lua_pcall(L,0,0,0))
    error(L, "No file: %s\n", lua_tostring(L,-1));
  lua_getglobal(L, "width");
  lua_getglobal(L, "height");
  if(!lua_isnumber(L, -2))
    error(L, "width should be a number\n");
  if(!lua_isnumber(L, -1))
    error(L, "height should be a number\n");
  *w = lua_tointeger(L, -2);
  *h = lua_tointeger(L, -1);
  lua_pop(L, 2);
}
```

```
-- define windows size

if os.getenv("DISPLAY") == ":0.0" then
    width = 300; height = 300
else
    width = 100; height = 100
```

```
-- define windows size

if os.getenv("DISPLAY") == ":0.0" then
    width = 300; height = 300
else
    width = 100; height = 100
```

```
$graph
cannot run config file. File: graph_config.lua:
8: 'end' expected (to close 'if' at line 2)
near <eof>
```

```
-- define windows size

if os.getenv("DISPLAY") == ":0" then
    width = 300; height = 300
else
    width = 100; height = 100
end




-- w: 300, h: 300
```

- Table

```
lua_getglobal(L, "my_table");
lua_pushstring(L, key);
lua_gettable(L, -2);
```

```
function f(x, y)
   return
  (x^2*math.sin(y))
end
```

- Function

```
lua_getglobal(L, "f");
lua_pushnumber(L, x);
lua_pushnumber(L, y);
lua_pcall(L, 2, 1, 0);
lua_tonumber(L, -1);
```

call C from Lua

```
static int average(lua_State *L)
{
  int n = lua_gettop(L); // # of arguments
  double sum = 0;
  int i;
  for(i = 1; i <= n; i++)
    sum += lua_tonumber(L, i);
  lua_pushnumber(L, sum / n);
  lua_pushnumber(L, sum);
  return 2;
}
```

```
int luaopen_myClib(lua_State *L)
{
    lua_register(L, "c_average", average);
    return 0;
}
```

```
gcc myClib.c -fPIC -shared -o myClib.so
```

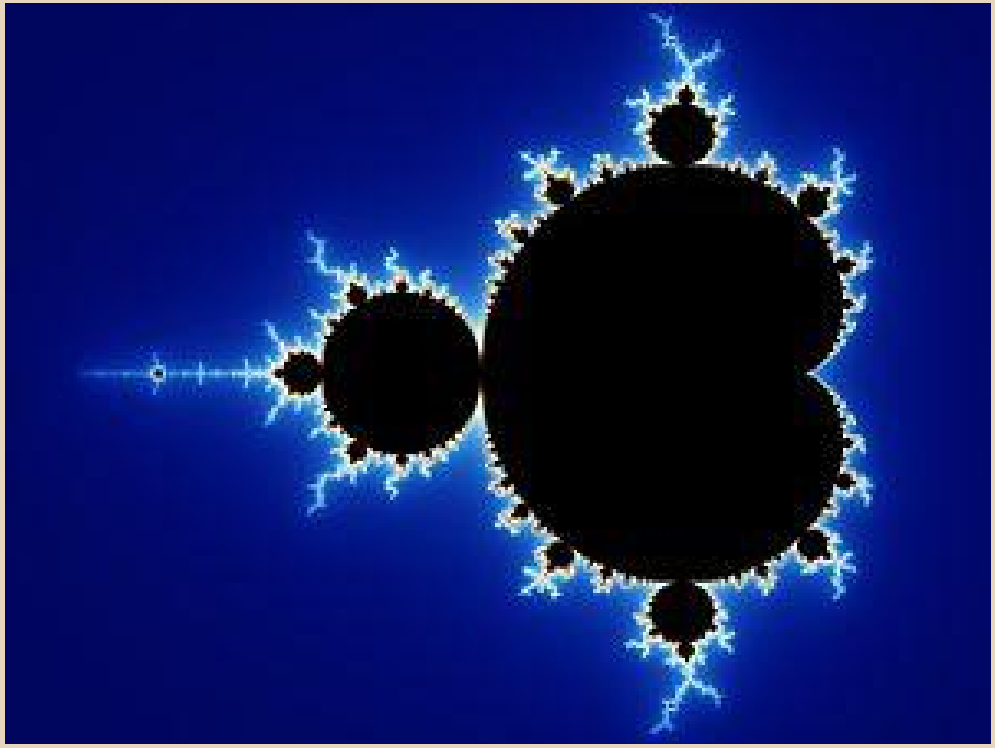
```
$ lua
Lua 5.2.1  Copyright (C) 1994-2012 Lua.org, PUC-
Rio
> require "myClib"
> c_average(4, 1)
> print(c_average(4, 1))
2.5    5
```

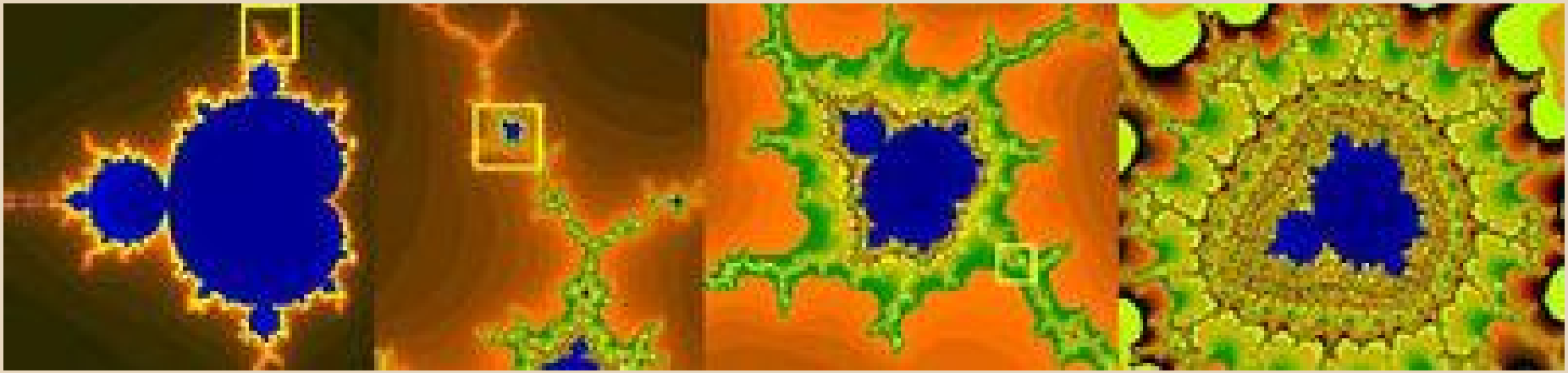
```
int main ( int argc, char *argv[] )
{
   lua_State* L = luaL_newstate();
   luaL_openlibs(L);
   lua_register(L, "average", average);
   luaL_dofile(L, "avg.lua");
   lua_close(L);
   return 0;
}
```

```
#define lua_register(L,n,f) \
 (lua_pushcfunction(L, f),lua_setglobal(L, n))
```

$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + p \end{cases}$$

$$\forall_{n \in \mathbb{N}} |z_n| < 2$$



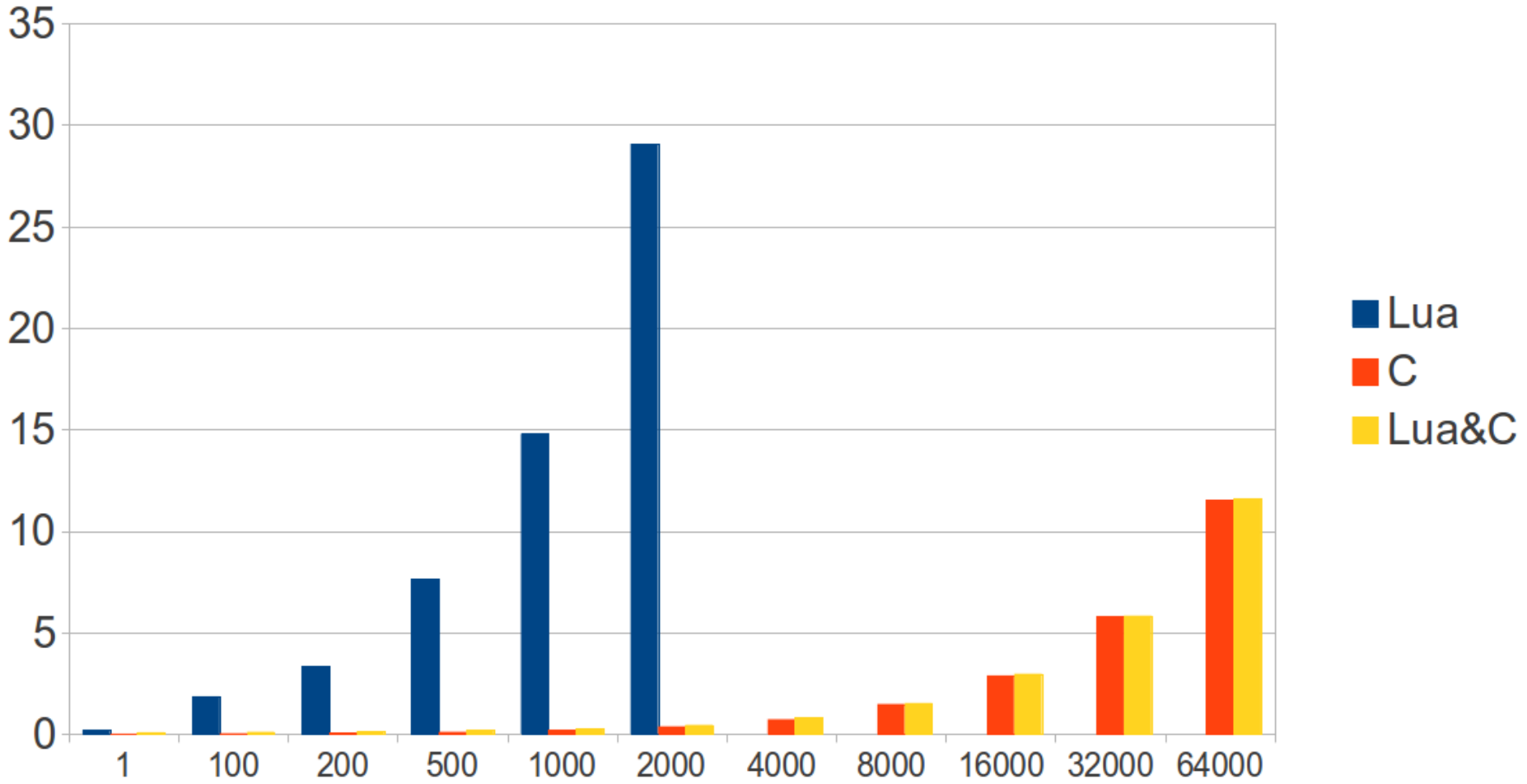


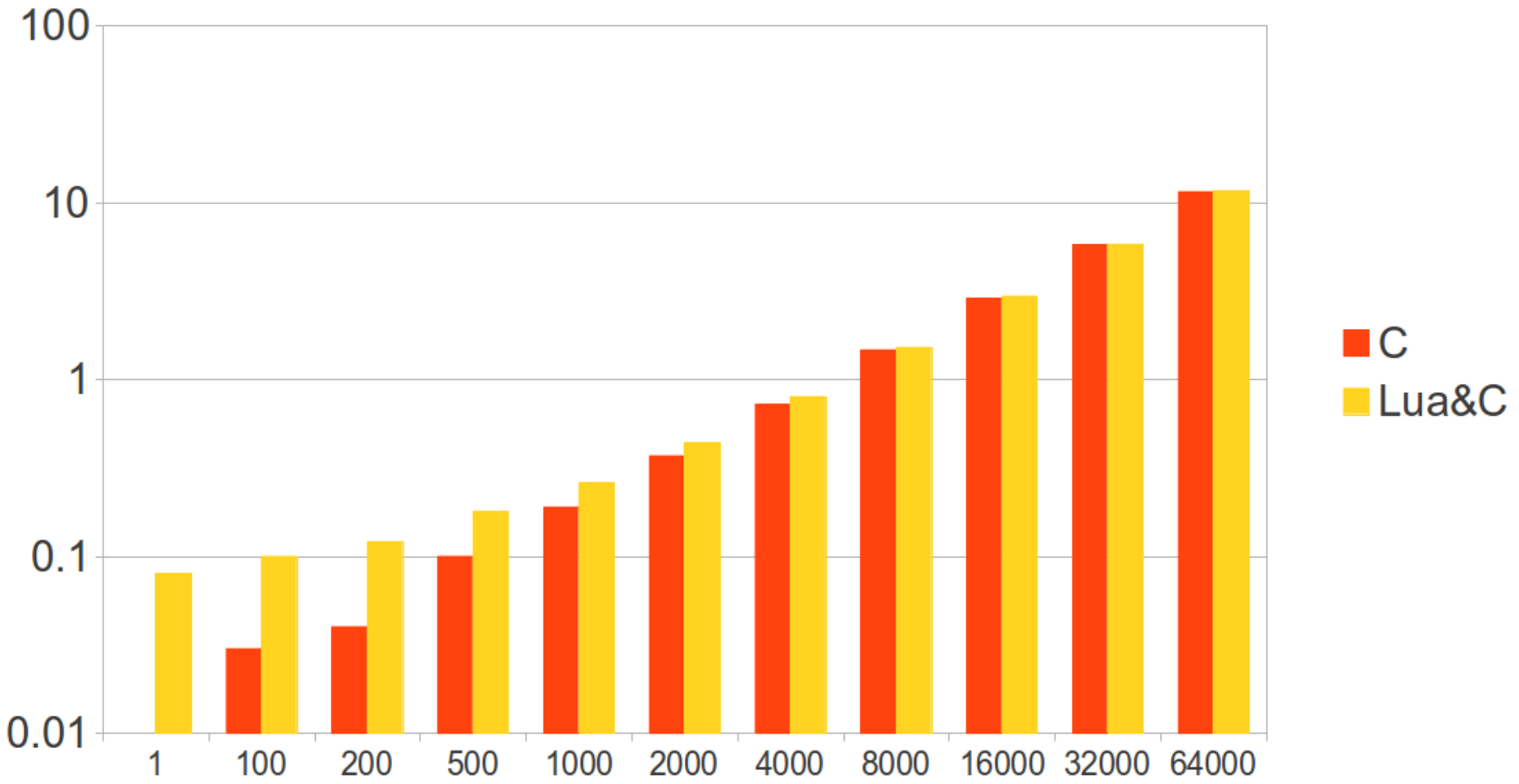
Pseudocode

```
for(x: 0 -> screen_width)
   for(y: 0 -> screen_height)
      if(isMandelbrot(noIterations))
         draw_pixel(x, y)


bool isMandelbrot(noIterations)
   for(i: 0 -> noIterations)
      z_next = z_prev^2 + p
      z_prev = z_next
   return ( |Z_next| < 2 )
```

Plan

- Write code in Lua
- Write code in C
- Compare performance
- Rewrite the *isMandelbrot* function in C, call from Lua
- Compare performance


35
30
25
20
15
10
5
0
1
100
200
500
1000
2000
4000
8000
16000
32000
64000
Lua
C
Lua&C


100
10
1
0.1
0.01
1
100
200
500
1000
2000
4000
8000
16000
32000
64000
C
Lua&C

Project - conclusions

- Lua is great for prototyping
  - LOVE
- Easy Lua-C communication
- Lua+C: very good performance

	Lua	Python	Ruby
3rd party libs	-lua.bind - tolua, tolua++ - SWIG	- boost.Python - SWIG ---------- - Pyrex - Cython	- RICE - FFI - SWIG
Value passed	Lua stack	Objects	Objects
Memory	Lua's GC - copy values before removing from stack	Need to manually decrease reference count	Mark-and-Sweep process (need own 'free' function for user defined structures

Bibliography

[1] "Programming in Lua" Roberto Ierusalimschy (third edition)

[2] The Computer Language Benchmarks Game http://benchmarksgame.alioth.debian.org/u64q/code-used-time-used-shapes.php

[3] www.lua.org

[4] Reflective Techniques in Extensible Languages - Jonathan Riehl

[5] docs.python.org

[6] www.paulgraham.com/icad.html

Bibliography - images

[7] fractal image - http://foodofjeff.blogspot.co.uk/2011/02/edible-fractals.html

[8] moon image - www.ccvalg.pt

[9] two ways image: Jacek Yerka, ROZDROŻA www.agraart.pl/nowe/auction.php?off=36&id_aukcji=270

[10] image of lua stack - www.pplux.com/2008/04/16/lua-api-introduccion/