

ADVANCED
TI-*nspire*
LUA PROGRAMMING

CODE OPTIMIZATION & ALTERNATIVE LUA EDITORS

Adrien BERTRAND (« Adriweb »)



Table of Contents

I. CODE OPTIMIZATION

1. LUA PERFORMANCE BENCHMARKS
2. TIPS AND TRICKS
3. NSPIRE-LUA SPECIFIC THINGS

II. ALTERNATIVE LUA EDITORS

1. SIMPLE CODE EDITORS
2. AN IDE : INTELLIJ IDEA
3. WORKING WITH THE TI-NSPIRE SOFTWARE



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

Credits :

« Caspring » website's wiki (now closed)
Lua.org



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

- Localize your functions

```
for i = 1, 1000000 do  
    local x = math.sin(i)  
end
```

The following code is 30% faster :

```
local sin = math.sin  
for i = 1, 1000000 do  
    local x = sin(i)  
end
```

**Accessing global variables takes more time than accessing local ones.
Always localize your functions !**



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

- Tables optimization

```
for i = 1, 1000000 do
  local a = {}
  a[1] = 1; a[2] = 2; a[3] = 3
end
```

Help Lua know more about the tables you're going to use !

The following code is almost 3x faster:

```
for i = 1, 1000000 do
  local a = {true, true, true}
  a[1] = 1; a[2] = 2; a[3] = 3
end
```



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

- More Tables optimization

Before :

```
polyline = {  
  { x = 10, y = 20 },  
  { x = 15, y = 20 },  
  { x = 30, y = 20 },  
  ...  
}
```

Better :

```
polyline = {  
  { 10, 20 },  
  { 15, 20 },  
  { 30, 20 },  
  ...  
}
```

Best :

```
polyline = {  
  x = { 10, 15, 20... },  
  y = { 20, 20, 20... }  
}
```

**Again, help Lua know more about the tables you're going to use !
Avoid useless rehashes when possible !**



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

- Other tricks
 - ✓ Don't use `unpack()` in time-critical code
 - Unpack them yourself ;-) (get values one by one)
 - ✓ Don't use `math.max/min()` on big tables in time-critical code
 - Prefer looping through the list and using comparisons
 - ✓ Don't use `math.fmod()` for positive numbers
 - Use the `%` operator. (On negative numbers, use `math.fmod`, though)



CODE OPTIMIZATION

LUA PERFORMANCE BENCHMARKS

- Other tricks
 - ✓ Think twice before using `pairs()` or `ipairs()`
 - When you know the bounds/keys, prefer a simple `for i=1,x` loop
 - ✓ Avoid `table.insert()` when inserting at the end
 - Instead, use something like `tbl[#tbl+1] = 42`
 - ✓ When possible, use *closures*
 - (Powerful concept behind functions [returning] in another function)
More info : <http://www.lua.org/pil/6.1.html>



CODE OPTIMIZATION TIPS AND TRICKS



CODE OPTIMIZATION

TIPS AND TRICKS

- Indentation : a prerequisite

```
3572 local dirty_exit = true
3573 local tosolve
3574 local couldnotsolve = {}
3575
3576 local loops = 0
3577 while dirty_exit do
3578 loops = loops + 1
3579 if loops == 100 then error("too many loops!") end
3580 dirty_exit = false
3581
3582 for i, formula in ipairs(Formulas) do
3583
3584 local skip = false
3585 if couldnotsolve[formula] then
3586 skip = true
3587 for k, v in pairs(known) do
3588 if not couldnotsolve[formula][k] then
3589 skip = false
3590 couldnotsolve[formula] = nil
3591 break
3592 end
3593 end
3594 end
3595
3596 if ((not cid) or (cid and formula.category == cid)) and
```

```
3572 local dirty_exit = true
3573 local tosolve
3574 local couldnotsolve = {}
3575
3576 local loops = 0
3577 while dirty_exit do
3578 loops = loops + 1
3579 if loops == 100 then error("too many loops!") end
3580 dirty_exit = false
3581
3582 for i, formula in ipairs(Formulas) do
3583
3584 local skip = false
3585 if couldnotsolve[formula] then
3586 skip = true
3587 for k, v in pairs(known) do
3588 if not couldnotsolve[formula][k] then
3589 skip = false
3590 couldnotsolve[formula] = nil
3591 break
3592 end
3593 end
3594 end
3595
3596 if ((not cid) or (cid and formula.category == cid)) and
```



CODE OPTIMIZATION TIPS AND TRICKS

- Simplify your code

```
38 ccircle1=circle(2*w/30,h/10,w/30)
39 ccircle2=circle(2*w/30,h/10,w/30)
40 ccircle3=circle(2*w/30,h/10,w/30)
41 ccircle4=circle(2*w/30,h/10,w/30)
42 ccircle5=circle(2*w/30,h/10,w/30)
43
44 hcircle1=circle(2*w/30,h/2,w/50)
45 hcircle2=circle(2*w/30,h/2,w/50)
46 hcircle3=circle(2*w/30,h/2,w/50)
47 hcircle4=circle(2*w/30,h/2,w/50)
48 hcircle5=circle(2*w/30,h/2,w/50)
49 hcircle6=circle(2*w/30,h/2,w/50)
50 hcircle7=circle(2*w/30,h/2,w/50)
51 hcircle8=circle(2*w/30,h/2,w/50)
52 hcircle9=circle(2*w/30,h/2,w/50)
53 hcircle10=circle(2*w/30,h/2,w/50)
54 hcircle11=circle(2*w/30,h/2,w/50)
55 hcircle12=circle(2*w/30,h/2,w/50)
56
57 Objects={ccircle1,ccircle2,ccircle3,ccircle4,ccircle5,hcircle1,
58          hcircle2,hcircle3,hcircle4,hcircle5,hcircle6,hcircle7,
59          hcircle8,hcircle9,hcircle10,hcircle11,hcircle12}
```

```
38 Objects = {}
39 for i = 1, 5 do
40     Objects[i] = circle(2*w/30,h/10,w/30)
41 end
42
43 for i = 6, 17 do
44     Objects[i] = circle(2*w/30,h/2,w/50)
45 end
```



CODE OPTIMIZATION

TIPS AND TRICKS

Metatables

A metatable is a table which can change the behavior of the table it's attached to.

```
t = {} -- our normal table
mt = {} -- our metatable, which contains nothing right now
setmetatable(t, mt) -- sets mt to be t's metatable
getmetatable(t) -- this will return mt
```

```
(same as t = setmetatable({}, {}) )
```

```
t = setmetatable({}, {
  __index = function(t, key)
    if key == "foo" then return 0 else return t[key] end
  end
})
```



CODE OPTIMIZATION

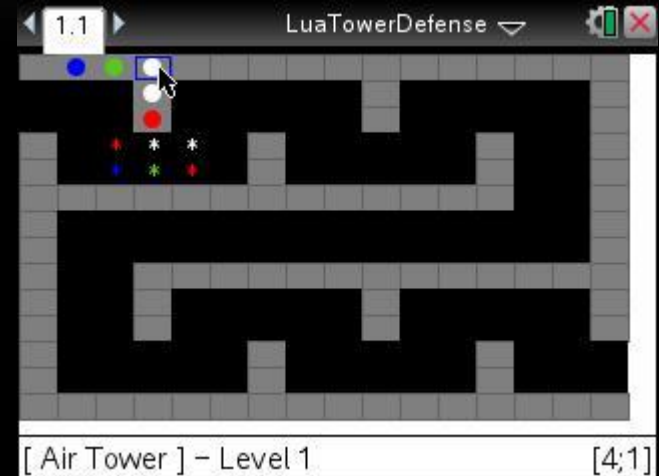
TIPS AND TRICKS

Metatable example

```
testmap = { {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},  
            {8,0,0,1,0,0,0,0,0,1,0,0,0,0,1},  
            {8,0,0,1,0,0,0,0,0,1,0,0,0,0,1},  
            [...]  
            {1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,9},  
            {1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,9},  
            {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1} }
```

```
setmetatable( testmap, { __index = {1} } )  
-- makes it so undefined areas will be walls (1).
```

```
setmetatable( self.theTypes, { __index =  
                                function(tbl, key)  
                                    return tbl[(key%#tbl)+1]  
                                end } )
```



CODE OPTIMIZATION

TIPS AND TRICKS

More fun with metatables

Operator overloading

`__add`: Addition (+)
`__sub`: Subtraction (-)
`__mul`: Multiplication (*)
`__div`: Division (/)
`__mod`: Modulos (%)
`__unm`: Unary - (negation)
`__concat`: Concatenation (..)
`__eq`: Equality (==)
`__lt`: Less than (<)
`__le`: Less than or equal to (<=)

A table that supports the multiplication operator (*):

```
t = setmetatable({ 1, 2, 3 }, {  
    __mul = function(t, nbr)  
        local res = {}  
        for k, v in pairs(t) do  
            res[k] = t[k] * nbr  
        end  
        return res  
    end })
```

`t = t * 2` -- gives : { 2, 4, 6 }



CODE OPTIMIZATION

TIPS AND TRICKS

Memoization :

Storing the result of some computation for a given input so that, when the same input is given again, the script simply reuses that previous result.

```
function memoize(f)
  local mem = {}           -- memoizing table
  setmetatable(mem, {__mode = "v"}) -- weak table
  return function (x) -- new memoizing version of 'f'
    local r = mem[x]
    if r == nil then      -- any previous result ?
      r = f(x)           -- calls original function
      mem[x] = r         -- store result for reuse
    end
    return r
  end
end

loadstring = memoize(loadstring)
```



CODE OPTIMIZATION

NSPIRE-LUA SPECIFIC THINGS



CODE OPTIMIZATION

NSPIRE-LUA SPECIFIC THINGS

- Do **not** do anything else than drawing in `on.paint()`
- Particularly, here's what you should **avoid** in `on.paint()`
 - `image.new()`, `image.copy()`, `image.rotate()` ← Way too slow
 - Events definition (like `on.enterKey()`, etc.) ← Not appropriate
 - `platform.window:invalidate()` ← Useless here

Reminder : except if you're dealing with animations, try not to refresh the screen a lot, but only when needed, it will save CPU and memory !



CODE OPTIMIZATION

NSPIRE-LUA SPECIFIC THINGS

- Use Classes
 - No need to state the obvious on the advantages
- Use a screen manager / GUI Toolkit



WZGUILib



TiMasterBox



ETK

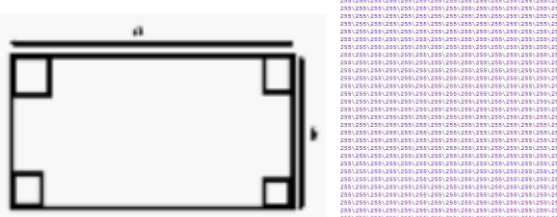


CODE OPTIMIZATION

INSPIRE-LUA SPECIFIC THINGS

- Avoid images when possible
- Images

```
img1=Image.new("S\001\000\000\027\001\000\000\000\000")
```



6 KB
6 images = 26KB

Polygons

```
polys = {  
  {  
    {-1.8, -1, 1.8, -1, 1.8, 1, -1.8, 1, -1.8, -1},  
    {-1.8, -1, -1.6, -1, -1.6, -.8, -1.8, -.8, -1.8, -1},  
    {1.6, -1, 1.8, -1, 1.8, -.8, 1.6, -.8, 1.6, -1},  
    {-1.8, 1, -1.6, 1, -1.6, .8, -1.8, .8, -1.8, 1},  
    {1.6, 1, 1.8, 1, 1.8, .8, 1.6, .8, 1.6, 1},  
  },  
}
```



1 KB
6 polygons = 2KB

CODE OPTIMIZATION

NSPIRE-LUA SPECIFIC THINGS

- Static Width or Height

```
if ww == 793 then
    gc:setFont("sansserif","bi",20)
else
    gc:setFont("sansserif","bi",11)
end
```

```
if ww > 320 then
    gc:setFont("sansserif","bi",20)
else
    gc:setFont("sansserif","bi",11)
end
```

Other examples :

```
gc:setFont("sansserif", "bi", math.min(255, math.max(6, ww/25)))
```

Re-use later :

```
f_medium = math.min(255, math.max(6, ww/25))
...
gc:setFont("sansserif", "bi", f_medium)
```

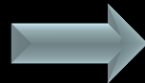


CODE OPTIMIZATION

NSPIRE-LUA SPECIFIC THINGS

Use `on.varChange()` instead
of recalling variables in `on.timer()`

```
function on.timer()  
    ch = var.recall("ch")  
    platform.window:invalidate()  
end  
  
timer.start(0.1)
```



```
function on.construction()  
    local v = {"quadrilateral"}  
    vars = {}  
    for i, k in ipairs(v) do  
        vars[k] = var.recall(k) or 1  
        var.monitor(k)  
    end  
end  
  
function on.varChange(list)  
    for _, k in pairs(list) do  
        if vars[k] then  
            vars[k] = var.recall(k) or 1  
        end  
    end  
    platform.window:invalidate()  
end
```



ALTERNATIVE LUA EDITORS

SIMPLE CODE EDITORS

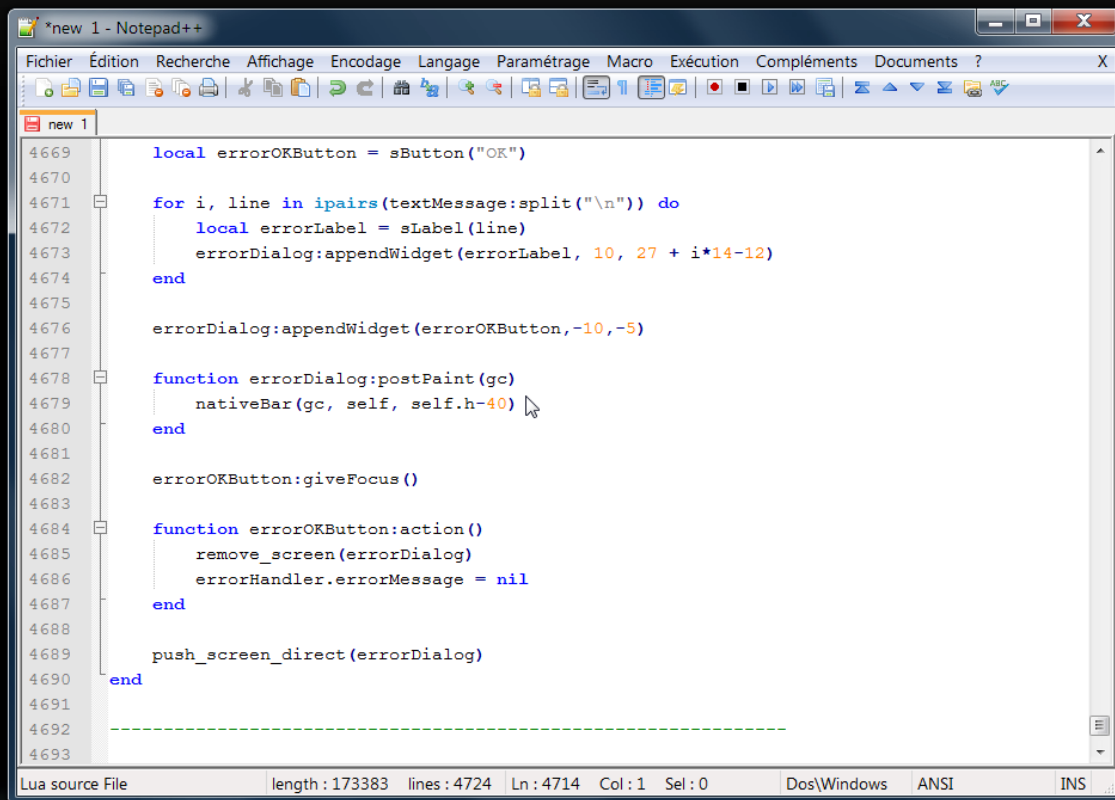


ALTERNATIVE LUA EDITORS

SIMPLE CODE EDITORS

Notepad++

- Windows only
- Lots of languages
- Nice set of features
- Plugins



The screenshot shows the Notepad++ application window with a menu bar (Fichier, Édition, Recherche, Affichage, Encodage, Langage, Paramétrage, Macro, Exécution, Compléments, Documents, ?) and a toolbar. The main editing area contains Lua code for an error dialog. The code is as follows:

```
4669     local errorOKButton = sButton("OK")
4670
4671     for i, line in ipairs(textMessage:split("\n")) do
4672         local errorLabel = sLabel(line)
4673         errorDialog:appendWidget(errorLabel, 10, 27 + i*14-12)
4674     end
4675
4676     errorDialog:appendWidget(errorOKButton,-10,-5)
4677
4678     function errorDialog:postPaint(gc)
4679         nativeBar(gc, self, self.h-40)
4680     end
4681
4682     errorOKButton:giveFocus()
4683
4684     function errorOKButton:action()
4685         remove_screen(errorDialog)
4686         errorHandler.errorMessage = nil
4687     end
4688
4689     push_screen_direct(errorDialog)
4690 end
```

The status bar at the bottom indicates: Lua source File | length : 173383 | lines : 4724 | Ln : 4714 | Col : 1 | Sel : 0 | Dos\Windows | ANSI | INS



ALTERNATIVE LUA EDITORS

SIMPLE CODE EDITORS

TextWrangler / BBEdit

- Mac only
- Lots of languages
- Nice set of features
- Plugins (for BBEdit)



```
122 function fillRoundRect(x,y,wd,ht,radius) -- wd = width
123 if radius > ht/2 then radius = ht/2 end -- avoid d
124 pla
125 k platform.gc():finish
126 k platform.gc():getStringHeight
127 k platform.gc():getStringWidth
128 k platform.gc():setAlpha
129 k platform.gc():setColorRGB
130 k platform.gc():setFont
131 k platform.gc():setPen
132 k platform.isColorDisplay
133 k platform.isDeviceModeRendering
134
135
136
```

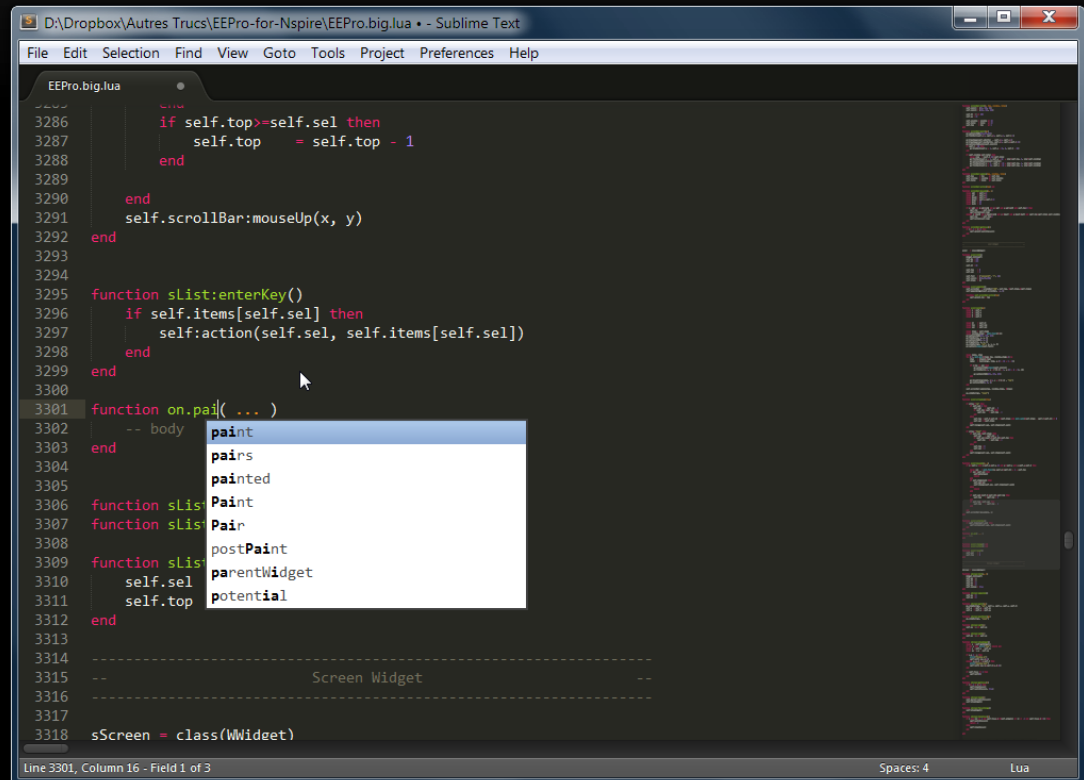


ALTERNATIVE LUA EDITORS

SIMPLE CODE EDITORS

SublimeText

- Windows/Mac/Linux
- Lots of languages
- Nice set of features
- Customizable
- Plugins



The screenshot shows the Sublime Text editor window titled "D:\Dropbox\Autres Trucs\EEPro-for-Nspire\EEPro.big.lua • - Sublime Text". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The main editor area displays Lua code with a completion menu open over the word "paint" on line 3303. The menu lists the following options: paint, pairs, painted, Paint, Pair, postPaint, parentWidget, and potential. The code includes functions like `self.scrollBar:mouseUp(x, y)`, `sList:enterKey()`, and `on.paint(...)`. The status bar at the bottom indicates "Line 3301, Column 16 - Field 1 of 3", "Spaces: 4", and "Lua".



ALTERNATIVE LUA EDITORS

AN IDE : INTELIJ IDEA



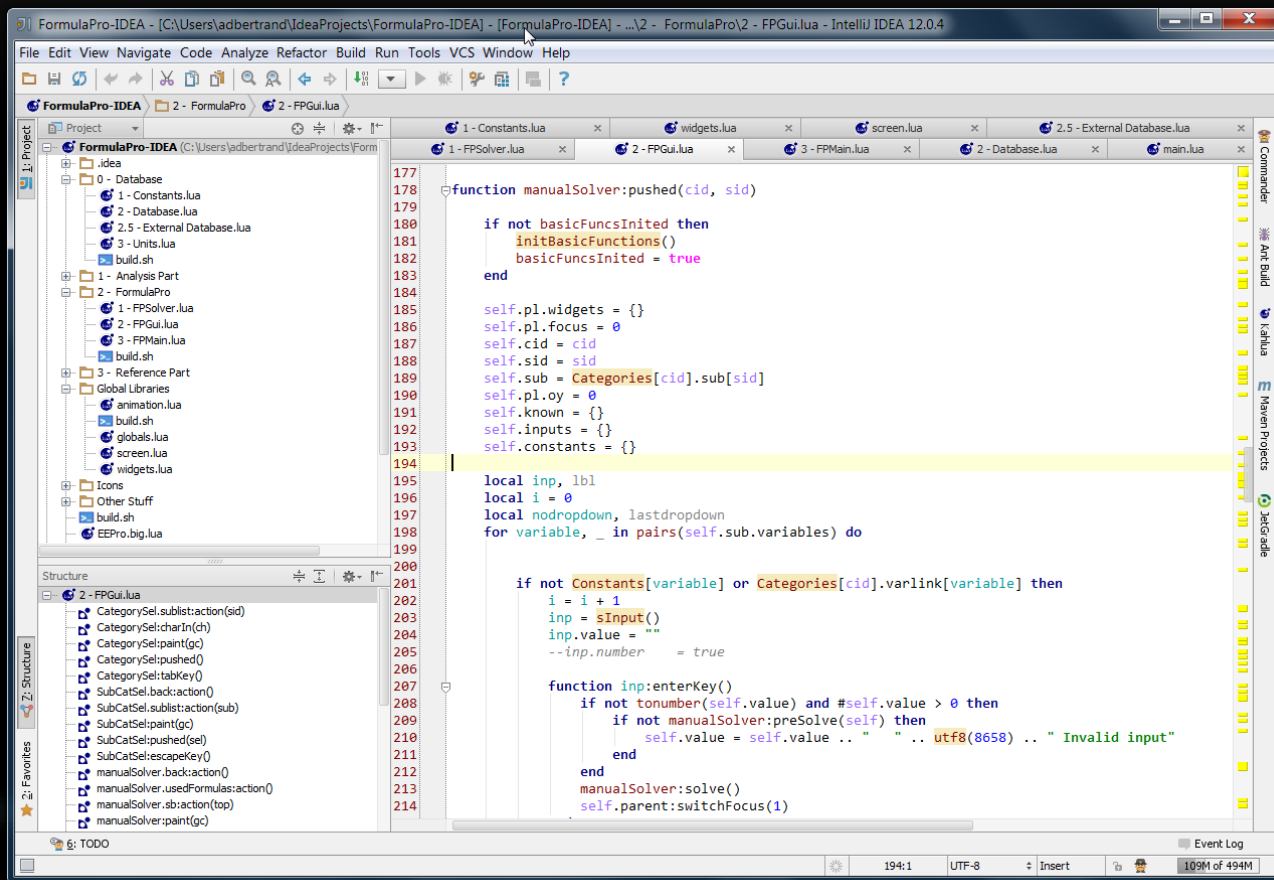
ALTERNATIVE LUA EDITORS

AN IDE : INTELLIJ IDEA

Integrated
Development
Environment

Windows/Mac/Linux
Free !

A "million" of features
Many plugins



ALTERNATIVE LUA EDITORS

AN IDE : INTELLIJ IDEA

Setting up the beast

1. Download and install IntelliJ
 1. <http://www.jetbrains.com/idea/>
 2. Select the Free Edition
2. Install its Lua plugin
 1. Go to Settings > Install Plugin
 2. Add the "Lua" one in Browse
3. Setup the Nspire-Lua addon
 1. Download it on TI-Planet
 2. Extract it somewhere
 3. Set it as the project's SDK



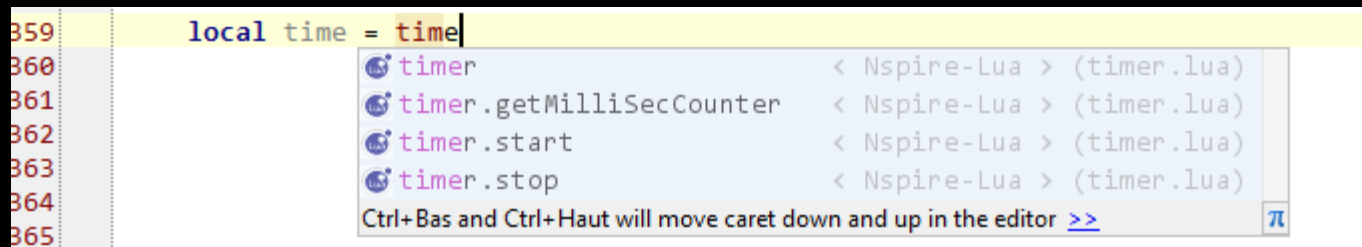
ALTERNATIVE LUA EDITORS

AN IDE : INTELIJ IDEA

Once all that is done, here's what you'll have :

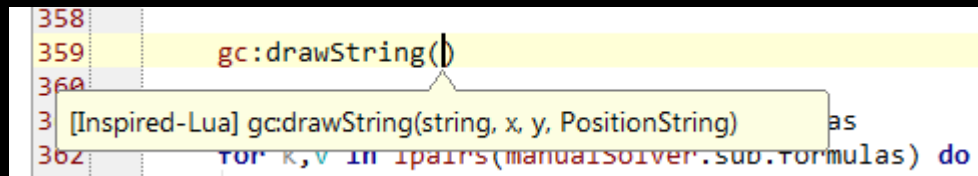
- ✓ Nspire-Lua specific auto-completion

```
359 local time = time|
360
361
362
363
364
365
```



- ✓ Inline syntax help for common methods

```
358
359 gc:drawString()
360
3 [Inspired-Lua] gc:drawString(string, x, y, PositionString) as
362 for k,v in ipairs(manualSolver.sub.formulas) do
```



- ✓ Dynamic API help frame with info from Inspired-Lua



ALTERNATIVE LUA EDITORS

WORKING WITH THE TI-NSPIRE SOFTWARE



ALTERNATIVE LUA EDITORS

WORKING WITH THE TI-NSPIRE SOFTWARE

Using the built-in SDK : no worries.

Using an external editor :

- Copy/Paste
- Use the « preliminary » official tool
- Use Luna (better choice)
 - Usage : **luna[.exe] input.lua ouput.tns**
 - Video tutorial :
<http://www.youtube.com/watch?v=h18GV8luizg>



CREDITS

Jérémy Anselme (“Levak”)
Jim Bauwens

Steve Arnold
John Powers

Inspired-Lua.org
TI-Planet.org



ANY QUESTIONS ?



(original images by TI)