chickenize

Arno Trautmann
arno.trautmann@gmx.de

July 4, 2011

Abstract

This is the documentation of the package `chickenize`. It allows you to substitute or change the contents of a LuaLaTeX document.[1] You have e. g. the possibility to substitute every word of a document with the word "chicken", translate it into 1337 speak, make it totally colorfull or use upper/lowercase all randomly. Of course this package is *not* meant for any serious document, but only for fun and – because we can!

If you have any suggestions or comments, just drop me a mail, I'll be happy to get any response!

Contents

1 Usage

This package should be useable some time …

[1] The code is based on pure LuaTeX features, so don't try to use it with any other TeX flavour.

2 Working Principle

We make use of LuaTEXs callbacks, especially the `pre_linebreak_filter` and the `post_linebreak_filter`. Hooking a function into these, we can chanke the input (into "chicken") or add/transform the input (putting color in, changing lower/uppercase).

2.1 Package Options

There surely will be some options etc.

3 Implementation

This is the README file that should contain some important information. So far I can only tell you to run the file chickenize.dtx to produce the three files chickenize.pdf (documentation) chickenize.sty (LaTeX user interface) chickenize.lua (Lua package code)

You need an up-to-date TeX Live (2011, if possible) to use this package.

For any comments or suggestions, contact me: arno dot trautmann at gmx dot de

Hope you have fun with this!

4 Preparation

Loading of packages and defition of constants. Will change somewhat when migrating to expl3 (?)

```
\RequirePackage{
  expl3,
  luatexbase,
  xkeyval,
  xparse
}
%% So far, no keys are needed.
\ExplSyntaxOn
\keys_define:nn {chick} {
  columns.tl_gset:N = \chick_cards_colums,
  columns.default:n = 2,
  printonly.code:n = \tl_set:Nn\chick_print_only{#1}\bool_set_true:N\chick_print_only_true,
  sectionsoncards.bool_set:N = \chick_sectionsoncards_true,
  german.tl_set:N = \chick_language,
}
\NewDocumentCommand\chicksetup{m}{
  \keys_set:nn{chick}{#1}
}
\directlua{dofile("chickenize.lua")}

```

```
\NewDocumentCommand\chickenize{}{
  \directlua{luatexbase.add_to_callback("pre_linebreak_filter",chickenize,"chickenize the input
  %% We want to "chickenize" figures, too. So …
  \DeclareDocumentCommand\includegraphics{O{}m}{
     \fbox{Chicken}  %% actually, I'd love to draw a mp graph showing a chicken …
  }
}
\NewDocumentCommand\uppercasecolor{}{
  \directlua{luatexbase.add_to_callback("post_linebreak_filter",uppercasecolor,"color all uc ch
}
\NewDocumentCommand\randomuclc{}{
  \directlua{luatexbase.add_to_callback("pre_linebreak_filter",randomuclc,"randomize uc/lc char
}

\NewDocumentCommand\colorstretch{}{
  \directlua{luatexbase.add_to_callback("post_linebreak_filter",colorstretch,"show stretch and
}
\NewDocumentCommand\leetspeak{}{
  \directlua{luatexbase.add_to_callback("post_linebreak_filter",leet,"transform input to 1337",
}
```

5 Definition of Macros

```
%
```

6 Lua Module

This file contains all the necessary functions.

```
local HLIST = node.id("hlist")
local RULE = node.id("rule")
local GLUE = node.id("glue")
local WHAT = node.id("whatsit")
local COL = node.subtype("pdf_colorstack")
local GLYPH = node.id("glyph")

local color_push = node.new(WHAT,COL)
local color_pop = node.new(WHAT,COL)
color_push.stack = 0
color_pop.stack = 0
color_push.cmd = 1
color_pop.cmd = 2

uppercasecolor = function (head)
  for line in node.traverse_id(HLIST,head) do
    for upper in node.traverse_id(GLYPH,line.list) do
      if (((upper.char > 64) and (upper.char < 91)) or
          ((upper.char > 57424) and (upper.char < 57451)))  then  -- for small caps! nice
        color_push.data = math.random()..math.random()..math.random().." rg"
```

```
        line.head = node.insert_before(line.list,upper,node.copy(color_push))
        node.insert_after(line.list,upper,node.copy(color_pop))
      end
    end
  end
  return head
end

randomuclc = function(head)
  for i in node.traverse_id(37,head) do
    if math.random() < 0.5 then
      i.char = tex.uccode[i.char]
    else
      i.char = tex.lccode[i.char]
      i.yoffset = "15 pt"
end
  end
  return head
end

function chickenize(head)
  for i in node.traverse_id(37,head) do  --find start of a word
    while ((i.next.id == 37) or (i.next.id == 11) or (i.next.id == 7) or (i.next.id == 0)) do
      i.next = i.next.next
    end

    chicken = {}
    chicken[0] = node.new(37,1)
    for i = 1,7 do
      chicken[i] = node.new(37,1)
      chicken[i].font = font.current()
    end
    node.insert_before(head,i,chicken[1])

  -- randomize upper/lower case to get a more natural output.
  -- however, this may make break points inconsistent!
if (math.random() > 0.8) then
    chicken[7].char = 67  else
    chicken[7].char = 99
end

    chicken[6].char = 104
    chicken[5].char = 105
    chicken[4].char = 99
    chicken[3].char = 107
    chicken[2].char = 101
    chicken[1].char = 110
lang.hyphenate(chicken[1])
    for k = 1,6 do
      node.insert_before(head,chicken[k],chicken[k+1])
```

```
    end
    chicken[1].next = i.next
  end

  return head
end

leettable = {
  [101] = 51, -- e
  [105] = 49, -- i
  [108] = 49, -- l
  [111] = 48, -- o
  [115] = 53, -- s
  [116] = 55, -- t

  [101-32] = 51, -- e
  [105-32] = 49, -- i
  [108-32] = 49, -- l
  [111-32] = 48, -- o
  [115-32] = 53, -- s
  [116-32] = 55, -- t
}

function leet(head)
  for line in node.traverse_id(HLIST,head) do
    for i in node.traverse_id(GLYPH,line.list) do
      if leettable[i.char] then
        i.char = leettable[i.char]
      end
    end
  end
  return head
end


-- The good parts of the following function are written by Paul Isambert.
-- I merely copied it and changed a few parameters. Thanks for the code
-- and support, Paul!

colorstretch = function (head)
  -- name convention: "expansion" means stretching of spaces
  --                  "stretch/shrink" means microtypographic expansion of glyphs

  local f = font.getfont(font.current()).characters
  for line in node.traverse_id(HLIST,head) do
    local rule_bad = node.new(RULE)

if colorexpansion then  -- if also the stretch/shrink of letters should be shown
      rule_bad.width = 0.5*line.width

```

```
      local g = line.head
        while not(g.id == 37) do
         g = g.next
        end
      exp_factor = g.width / f[g.char].width
      exp_color = .5 + (1-exp_factor)*10 .. " g"

    else
      rule_bad.width = line.width  -- only the space expansion should be shown
    end

    local glue_ratio = 0
    if line.glue_order == 0 then
      if line.glue_sign == 1 then
        glue_ratio = .5 * math.min(line.glue_set,1)
      else
        glue_ratio = -.5 * math.min(line.glue_set,1)
      end
    end
    color_push.data = .5 + glue_ratio .. " g"

-- set up output
    local p = line.list
-- first, a rule with the badness color
    line.list = node.copy(color_push)
    node.flush_list(p)
    node.insert_after(line.list,line.list,rule_bad)
    node.insert_after(line.list,rule_bad,node.copy(color_pop))

-- then a rule with the expansion color
if colorexpansion then  -- if also the stretch/shrink of letters should be shown
      color_push.data = exp_color
      node.insert_before(line.list,node.tail(line.list),node.copy(color_push))
      node.insert_before(line.list,node.tail(line.list),node.copy(rule_bad))
      node.insert_before(line.list,node.tail(line.list),node.copy(color_pop))
    end
  end
  return head
end
```

7 Known Bugs

There are surely some bugs …

???

8 To Dos

Some things that should be implemented but aren't so far or are very poor at the moment:

?