Decorating text with TikZ and lua LaTeX

Marcin Woliński

BachoTeX 2019

“Sometimes I believed in as many as six impossible things before breakfast.”

One of things, that are ‘impossible’ in TEX is underlining or striking out running text:

Kolegają twórczym i równie osiągnęła przez kinety żywicy. Druga wieka nieografimy do śmieta letnamicy na nierów, arch z tym, pobudzie ~~rodziałalnością główne eksportali się w wypadkach osłoniecznej, im łącznie wypełnione zostały do się popełnienia głównież reszczonej~~ w kryształtownego odgrodami, inację nakła. Fale posągi ~~Brak żadnych manufaktualności~~ chętnieje się między europejczyznościach, a robocze skórniczące stanowi o rezydełkamienia.

Package soul

LaTeX package soul (the name stands for ***s**trike **o**ut and **u**nder**l**ine*) works by underlining separately each letter of the text:

… p o b u d z i e ~~r o d z i a ł a l n o ś c~~

~~i ą~~ – ~~g ł ó w n e~~ – ~~e k s p o r t a l i~~ …

- Trickery used to strike out extensible glue.
- More trickery to preserve hyphenation points.
- Lines consist of myriads of short segments.
- Problematic when text contains alien elements, e.g., math formulas.

Striking out in luaTEX

Striking out gets easy in luaTEX thanks to two mechanisms:

- node attributes
- and access to the paragraph breaking callback.

The following example adapted from: *Three things you can do with LuaTEX that would be extremely painful otherwise* by Paul Isambert, TUGboat, Vol. 31(3), 2010, pp. 184–190.

Attributes in luaTEX

```
\newattribute\strikeoutattribute

\def\strikeout #1{%
  {%
  \setattribute{\strikeoutattribute}{1}%
  #1%
  }%
}

Fale posągi \strikeout{Brak żadnych manufaktualności}
chętnieje …
```

Striking out in luaTEX

- The code in the previous slide in itself typesets the text as usual.
- However, each box created within the scope of `\setattribute` carries an attribute that can be checked later on from the Lua code.
- The callback `post_linebreak_filter` allows to inspect and modify each paragraph just after line breaking.

The paragraph callback

```
strikeout_process_lines = function (head)
  for line in node.traverse_id(HLIST, head) do
    strikeout_find_spans(line)
  end
  return head
end
```

Registering the callback:

```
luatexbase.add_to_callback("post_linebreak_filter",
                           strikeout_process_lines,
                           "strikeout_process_lines")
```

Checking the attribute

```
local strikeout_attribute =
        luatexbase.attributes['strikeoutattribute']
local NT_GLUE = node.id("glue")
local NT_LEFTSKIP = node.subtype("leftskip")
local NT_RIGHTSKIP = node.subtype("rightskip")
local NT_PARFILLSKIP = node.subtype("parfillskip")

is_struck_out = function (item)
  return node.has_attribute(item, strikeout_attribute)
  and (item.id ~= NT_GLUE
  or item.subtype ~= NT_LEFTSKIP
  or item.subtype ~= NT_RIGHTSKIP
  or item.subtype ~= NT_PARFILLSKIP)
end
```

Finding spans to strike out

```
strikeout_find_spans = function (line)
  local item = line.list
  while item do
    if is_struck_out(item) then
      local end_node = item
      while end_node.next and is_struck_out(end_node.next) do
        end_node = end_node.next
      end
      local width = node.dimensions(line.glue_set,
                line.glue_sign, line.glue_order,
                item, end_node.next)
      decorate_span(line.list, end_node.next, width)
      item = end_node.next
    else
      item = item.next
    end
  end
end
```

Crossing out

```
decorate_span = function(line, item, width)
  local rule = node.new("rule")
  rule.height = tex.sp("2.4pt")
  rule.depth = tex.sp("-2pt")
  rule.width = width
  local kern = node.new("kern", "userkern")
  kern.kern = -rule.width
  node.insert_after(line, item, kern)
  node.insert_after(line, kern, rule)
end
```

Gimme more!

There are sophisticated methods to decorate text, which are available as TEX macros. For example the TikZ package provides many wonderful options:

(1) *Główną jednak atrakcją pozostał seks mimowolny, czyli koncepcje babci dotyczące naszych ukrytych stosunków.*

(2) **fwe** *Drżąc na ciele i wzbudzając ohydę w duszy, rozpocząłem wędrówkę myszką, by uruchomić bank informacji.*

(3) *Polska jest* **fpt** *etnicznie jednorodna, heteroseksualna, katolicka i wolnorynkowa.*

(4) *Potem często pokazywano ich* **fpt(pact)** *przytulających się lub całujących.*

Gimme more!

(5) Główną jednak atrakcją pozostał seks mimowolny, czyli koncepcje babci dotyczące naszych ukrytych stosunków.

Access to a box typeset by TEX from Lua

TEX:

```
\newbox\strikeoutbox

\setbox\strikeoutbox=\hbox{%
  \begin{tikzpicture}
  …
  \end{tikzpicture}%
}
```

Lua:

```
tex.box[luatexbase.registernumber("strikeoutbox")]
```

Calling TEX from Lua

- The dimensions of the picture to be generated are only known after the paragraph is broken, so we need to 'call' TEX from Lua.
- The following call allows to insert some tokens into the TEX input buffer:
  `tex.print("\setbox\strikeoutbox=…")`

Calling TEX from Lua

- The dimensions of the picture to be generated are only known after the paragraph is broken, so we need to 'call' TEX from Lua.
- The following call allows to insert some tokens into the TEX input buffer:
  `tex.print("\setbox\strikeoutbox=…")`
- Problem:
  This will not work within the `post_linebreak_filter` (or any other callback)!

Imperfect solution 1

Redefine `\par` to perform following actions:

- Call original `\par`, which will
  - break the paragraph,
  - call the post linebreak filter, which will
    - store a list of regions to be decorated and their dimensions in a Lua table
    - store the paragraph itself and return `nil`
- Now control returns to TEX, so we can call Lua to scan the list of regions and successfully emit `tex.print()`s.
- At the end of each call to TEX call back Lua to embed the box in the stored paragraph.
- Add the paragraph to the main vertical list.

Imperfect solution 1

Redefine `\par` to perform following actions:

- Call original `\par`, which will
  - break the paragraph,
  - call the post linebreak filter, which will
    - store a list of regions to be decorated and their dimensions in a Lua table
    - store the paragraph itself and return `nil`
- Now control returns to TEX, so we can call Lua to scan the list of regions and successfully emit `tex.print()`s.
- At the end of each call to TEX call back Lua to embed the box in the stored paragraph.
- Add the paragraph to the main vertical list.

Problem: While the paragraph containing decorations is being processed no other paragraphs can be formed.

Less imperfect solution 2

- No changes to `\par`.
- Register a post line break filter that will store a list of regions to be decorated and their dimensions in a Lua table (globally: continue for all paragraphs in sequence).
- Use LaTeX's `\EveryShipout` (not the Lua shipout callback) to emit `tex.print()`s and decorate all spans gathered to the point (and clear them from the table).

Conclusions

- LuaTEX allows for a robust implementation of decorations for running text.
- However, it turns out to be rather complicated due to the way Lua interacts with TEX.