Extending Peter Flynn's bookshelf package for multilanguage libraries

Boris Veytsman

Due to the COVID, TUG2020 was held online. Figure 1 shows the drawing for the conference by Jennifer Claudio. As befits a true artist, Jennifer manages to reproduce the Zeitgeist with a well-chosen detail: the stylized bookshelves. They were created with the *bookshelf* package [1], which was released during the pandemic. It was used by many of us to generate the backgrounds for remote meetings. These bookshelves remind one of the time of endless meetings, fear, loneliness, sickness and death.

Peter's package uses a clever algorithm to create interesting images, different for each TEX run. It takes a BIBTEX catalog of books (many electronic book managers, like *Calibre* [2], can export the book list in this format). For each book it performs the following steps:

1. Select a random rectangle size.
2. Select random foreground and background colors. If the contrast is too low, repeat.
3. Select a random font.
4. Typeset author and title to fit in the box.

The result for my electronic library is shown on Figure 2. (Grayscaled for print; online, you might like to zoom in to see the variety of colors and fonts used.)

Besides creation of backgrounds, this package may be used also for an amusing game, which is quite suitable for long boring remote meetings. Take a look at some spines (Figure 3). Can you guess which fonts were used to typeset them? You may add a point for each correctly guessed font, and additional points for correctly guessed author or style. To check your answers you need to know that the little numbers after the books are actually the numbers of the fonts in the main list of fonts used by the package. For example, looking at the font numbers on Figure 3, we get:

589: `AvenirLTStd-Heavy`
9541: `KyivTypeTitling-Bold2`
2784: `Concourse4Italic`, Stylistic Set 3
17266: `XITS-BoldItalic`
68: `Alegreya-ExtraBoldItalic`, Stylistic Set 4
13971: `Nunito-ExtraLight`
16390: `SourceSansPro-Black`, Small Caps
10119: `KyivTypeTitling-Bold`, Stylistic Set 4
15373: `RobotoSerif-Black`, Old Style Numbers
536: `Arsenal-Bold`, Stylistic Set 2
1490: `BradleyDJR-Micro`, Historical Ligatures

Figure 1: Jennifer Claudio's drawing for TUG2020

An astute reader might understand at this point that Figures 2 and 3 were not produced by the original version of Peter's package. The reason is that some of the books on Figure 3 have Cyrillic spines (Ukrainian and Russian, to be precise). The fonts used for these books (XITS, Alegreya, Nunito, SourceSansPro, KyivTypeTitling) contain Cyrillic glyphs. However, since the font selection is random (see item 3 in the algorithm above), we can get fonts incapable of typesetting the spines. Since the number of books with non-Latin scripts in my library is large, the probability of such events is close to 1.

At first I restricted the selection of fonts only to those that had both Latin and Cyrillic glyphs. However, there were fonts I liked to see on my shelf which did not have Cyrillic letters. Also, I wanted a solution suitable for libraries more versatile than mine, with books in Arabic, Hebrew, Malayalam, Sanskrit, etc. I wanted to be able to typeset any catalog with any number of languages, and use any suitable font.

One solution would be to use *Language* tag of the fonts: we can add this tag to each book, create separate pools of fonts for each script, and then randomly select a font from the given pool. However this would require manual tagging of each book and a rather complicated font selection algorithm, especially for the books with several scripts in the title. Therefore I decided to use the same logic Peter used for color selection: for the given book select a random font. If the book spine can be typeset with this font, use it, otherwise repeat selection.

To check whether we can typeset the given string with the given font we use the primitive `\iffontchar`.

doi.org/10.47397/tb/45-2/tb140veytsman-bookshelf

Figure 2: The author's electronic library

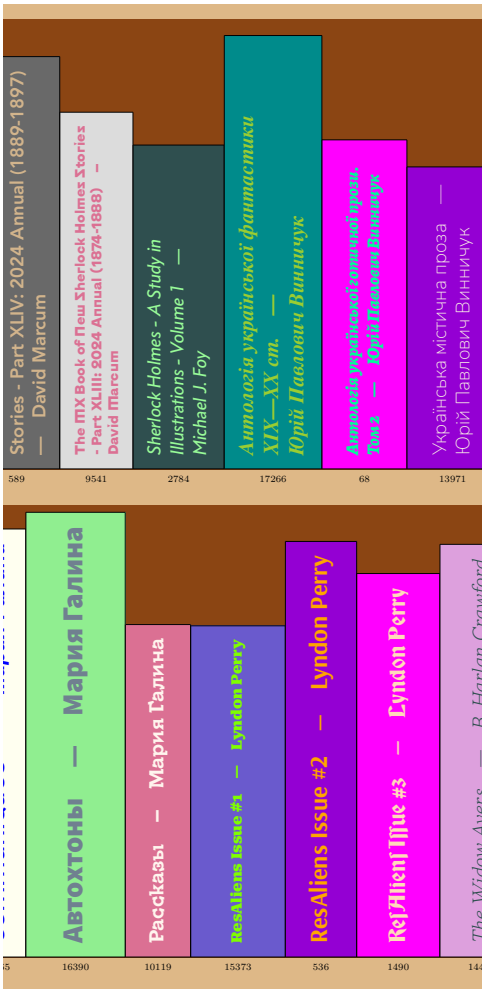

Figure 3: Several books from the author's library

The logic of the algorithm is straightforward: we map the primitive over the string, and bail out early if we find a character that cannot be typeset. The implementation is easier in *expl3* language; see Figure 4. This code defines a macro `\CanTypesetTF{`⟨*string*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`. It calls either `{`⟨*true*⟩`}` branch or `{`⟨*false*⟩`}` branch depending on the results of the typesetting test.

Since I wanted to demonstrate the possibilities of my fonts, I decided to change the source of them in the package. Both XeTeX and LuaTeX can use system fonts (those in the locations known to all applications on your machine), and TeX fonts (those known to your TeX installation). Peter's package can use any source, but the scripts provided with it get the list of fonts in the system directories. TeX Live has a very large collection of interesting fonts, to which I have added some that I've purchased or downloaded. Thus I decided to switch to the TeX fonts. I also wanted to demonstrate stylistic variants, swashes, old-style figures, so I wrote a script that lists these variants for the given font, as shown on Figure 5.

These changes lead to another problem. The number of fonts together with their variants turned out to be huge (19 183 on my machine). The trial-and-error algorithm for choosing a random font may open several fonts per book. A decent library (Figure 2 has 1584 books) probes many fonts from this list. Thus the package may want to open thousands of fonts for a single run. The number of fonts that a modern engine can open is much larger than in the old days, and can be further extended by changing the config file (I am grateful to Frank Mittelbach

```
\prg_new_conditional:Nnn \__SIL_primitive_font_glyph_if_exists:n  {TF,F}
  {
    \tex_iffontchar:D \l_fontspec_font `#1 \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
\prg_new_conditional:Nnn \__SIL_can_typeset:n {TF}
  {
   \typeout{Trying ~ to ~ typeset ~ #1}
    \bool_set_true:N \l_tmpa_bool
    \str_map_inline:nn {#1} {
      \__SIL_primitive_font_glyph_if_exists:nTF {##1} {}{
        \bool_set_false:N \l_tmpa_bool
        \typeout{Cannot ~ typeset ~ ##1}
        \str_map_break:
      }
    }
    \bool_if:nTF \l_tmpa_bool {\prg_return_true:} {\prg_return_false:}
}
\cs_generate_variant:Nn \__SIL_can_typeset:nTF {x}
\NewDocumentCommand\CanTypesetTF { m m m}{
  \__SIL_can_typeset:xTF{#1}{#2}{#3}
}
```

Figure 4: Checking whether a given string can be typeset with a given font

```
...
Arimo-Bold.ttf
Arimo-BoldItalic.ttf
Arimo-Italic.ttf
Arimo-Regular.ttf
Arsenal-Bold.otf
Arsenal-Bold.otf hist
Arsenal-Bold.otf smcp
Arsenal-Bold.otf ss01
Arsenal-Bold.otf ss02
Arsenal-Bold.otf swsh
...
```

Figure 5: Fragment of the font list

for this remark). Still, I found out that the engines choke when the number of fonts in the document exceeds 5500. I did not want to recompile the engines, so I employed several mitigation strategies:

1. The package does not load different sizes of a font to fit a spine. Instead, it changes the sizes of the rectangle that represents the spine, and then uses `\resizebox`. Generally, such resizing of fonts is a bad typographic practice; this is one of the rare cases when it seems to be appropriate.
2. The actual algorithm for choosing a random font has two stages. On the first stage we randomly select a font from the general list *and* save its number in the stack of opened fonts. When the size of this stack exceeds the limit, we no longer use the general list, but randomly select the font from the stack.

With these changes the package was able to typeset Figure 2.

The code is now available at the Github repository `github.com/borisveytsman/bookshelf`. Peter kindly allowed me to take over the maintenance of the package on CTAN, so the new version with all these changes will be released after some code cleaning. There are some features I'd like to add, including colorblind palettes, streamlining the typesetting, making the package aware of the size of the actual book (so large books have larger spines).

It is difficult to find a "practical" application for this package. Still, it brought much fun to me. I am grateful to Peter for inventing it, and hope my extensions are welcomed by other users.

References

[1] P. Flynn. *The bookshelf package*, 2020. `ctan.org/pkg/bookshelf`

[2] K. Goyal. *calibre User Manual*, 2024. `manual.calibre-ebook.com`

◇ Boris Veytsman
TEX Users Group
`borisv (at) lk dot net`
`https://borisv.lk.net`