The bookshelf document class*

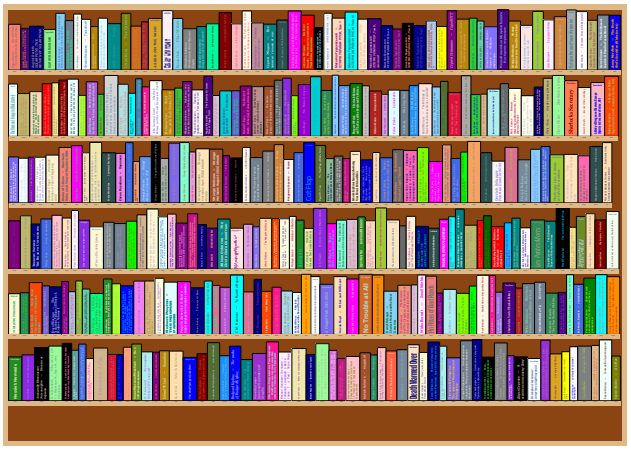
Turn your bibliography into a bookshelf image

Peter Flynn
Silmaril Consultants
Instant Textual Gratification Division
(peter@silmaril.ie)
Boris Veytsman
(borisv@lk.net)

9th October 2024

Summary

The bookshelf package uses your BIBTEX bibliography file into a randomly-coloured, randomly-sized shelf of books, with the title and author in a randomly-chosen typeface. The image (converted to JPEG from PDF) can then be used as a background in *Zoom*, *Teams*, *WhatsApp* etc video calls. It requires a little preliminary work with the supplied scripts to set up a list of your fonts and their stylistic variants, but otherwise should work on any modern TEX distribution.



*This document corresponds to bookshelf v1.2, dated 2024/10/09.

Contents

Note on required and optional features

In this document, the keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL have a specific meaning when shown in THIS TYPESTYLE, and MUST be interpreted as described in RFC 2119 [1].

When shown in normal type, these words keep their conventional contextual degree of meaning.

Foreword by BV

I have spent many hours admiring colorful bookcases produced by Peter Flynn's *bookshelf* package. They helped me to survive many boring remote meetings during the COVID pandemic and its aftermath. However, since my library had books in different languages, I wanted to showcase them as well. I started to write patches for the package, and at some point Peter kindly decided to transfer the maintenance to me.

I described the project at TUG 2024 [3]. At present the following changes has been implemented:

1. The package now can typeset book spines in any language. It automatically selects a random font capable to typeset a given spine.
2. The new bookshelf-listallfonts script lists all system and TEX fonts with "interesting" variants, while the new bookshelf-mkfontsel script populates the fontsel directory.
3. The package now understands fonts in OTF, TTF
4. The switch from Biber to BibTEX made the processing much faster, and eliminated the need for the separate entries.tex file: now the .bbl file has the right format.
5. The switch from *fontspec* to primitive font loading also made the processing faster—and increased the number of fonts we can display.

There is, however, a price for these improvements: now the package is LuaLATEX-only.

Boris Veytsman, October 2024

Latest changes

v.1.0(2025/10/02)

New maintainer. Several rewrites.

v.0.5 (2020-05-24)

Finished initial testing

- Replaced hyperref with hypdoc to avoid *makeindex* bug

v.0.1 (2020-05-7)

First packaged draft

- Done manually from .tex file

v.0.4 (2020-05-19)

Completed documentation

- Updated note on bug in *biber* when processing sgml.bib
- Removed sgml.bib as example until problems are resolved
- Backtracked on attempt to use the monographic title for articles, chapters, etc
- Revised notes on production

v.0.3 (2020-05-14)

Finished first pass on documentation

- Done preliminary testing
- Script adapted for Mac OS X

Acknowledgments

Thanks to many people for the original suggestion; and to Isabel Yorke, Bethan Tovey-Walsh, Nelson Beebe, The LaTeX Ninja, Stephan Lukasczyk, and others for their thesis bibliographies and testing comments.

1 Documentation

During the era of the COVID-19 lockdown, the popularity of group video messaging grew rapidly, both for business and domestic use. As people sought for what they believed to be more representative backgrounds than a messy kitchen, an untidy workroom, or a sterile blank wall, a well-populated bookshelf was a frequent choice.

This package is for those who cannot use (or don't have, or don't want to use) such a bookshelf, but can still lay their hands on a bibliography or reference list in BIBTEX format — perhaps from a recent or long-forgotten thesis, book, article, or other document.

You also may want to showcase your electronic library. Many programs like *Calibre* [2] can export the list of your electronic books in BibTEX format. This is how the sample library spines.pdf was created.

Another important use of this package is to provide a diversion during long boring remote meetings. Try to guess the font the given spine was typeset with—and use these tiny numbers under the books to check your knowledge!

> **LuaLATEX**
>
> To avoid problems with accented characters, and to make it easier to maintain, this document class uses only LuaLATEX.
>
> It will not work with the *pdflatex* or XETEX.

This is a work-in-progress: there are bugs (see section 1.4 on page 9).

1.1 What the package does

The bookshelf package generates what looks like shelves of book spines from your list of references, using random dimensions (within specified limits) in random but contrasting colors, with a randomly-selected typeface.

It does this by creating a box (rectangle) for each entry in your list, assigning colors to the background and foreground, deciding on the layout and font, and then stacking the boxes side-by-side as if they were letters on a line.

1.2 Preparation

To get things ready for this, you need to install this document class, and provide the following data:

1. The list of book as a BIBTEX file.
2. A list of all your usable text fonts and the total number of them.
3. A list of all the colors to choose from.

These are explained in more detail in the subsections below.

1.2.1 Your BIBTEX file

Your BIBTEX (.bib) file, suitable for use with *biber* rather than *bibtex*.

You may need to replace all the old-style symbolic notation accented characters like {\"a} for 'ä' and {\l} for 'ł'.

If you have a bibliography in *EndNote*, *Mendeley*, *Zotero*, *ProCite*, *Reference Manager*, etc, you should be able to export it in either BIBTEX or RIS format. A RIS file can easily be converted to BIBTEX by opening it in *JabRef* and saving it as BIBTEX.

1.2.2 Font file list

A set of 2–line files in a subdirectory called fontsel representing of all the usable text fonts on your system.

Each file MUST be numbered sequentially in its name (eg 1.tex, 2.tex, 3.tex, etc) and MUST contain a \font command and define the name of the font, for example

```
\font\SILmfont={[/usr/local/texlive/2024/texmf-dist/fonts/opentype/public/junicode/Junicode-SmCondMediumItalic.otf]:+clig;+liga;+tlig;+swsh}\SILmfont
\def\SILmfontname{Junicode SmCond Medium Italic; Swash}%
```

This list can be created using the scripts bookshelf-listallfonts and bookshelf-mkfontsel, see section 1.3.1 on the following page.

1.2.3 Maximum number of fonts

A file called pickfont.tex containing a \setcounter{SIL@maxfont}{...} command to set the total number of the fonts which are represented in the fontsel subdirectory above

This file can be created using the scripts bookshelf-listallfonts and bookshelf-mkfontsel, see section 1.3.1 on the next page.

1.2.4 List of colors

A list of all the colors represented by the SVG palette used by the xcolor package: the file is called called bookshelf-svgnam.tex. This contains three definitions:

1. the command \SIL@svgcolname which uses an \ifcase command to return the name of the *n*th color in alphabetical order;
2. the command \SIL@svgcolval which does the same to return the brightness value of that color, computed by the formula on on page 15 (see script for details);
3. the counter SIL@maxcolno which holds the number of colors available.

This file is included in the distribution.

In the previous versions of the ditribution it was created with the from svgnam.def database. The script svgnam.sh is retained in the distribution and can be used to recreate the file.

1.3 Producing your bookshelf

1.3.1 Producing the list of fonts

Each system has its own fonts installed, so you need to create the list of fonts installed on *your* system. It is done in three steps.

1. Update the database of fonts known to LuaTeX:

   ```
   luaotfload-tool --update --force
   ```

   The key --force forces the update even if LuaTeX thinks it is not necessary: sometimes it is mistaken.

2. Create the list of the usable fonts and font variants:

   ```
   bookshelf-listallfonts [options] > allfonts
   ```

   The script lists all fonts in OTF, TTF or TTC format, known to LuaLaTeX. For scripts in OTF and TTF format it looks at the “interesting” Open Type features like swash and stylistic variant and lists them too. This means that the bookshelf might have spines typeset in the different variants of the same font—they are considered different fonts! You can change the list of feaures using the key -f FEATURES_FILE; see the doc directory for an example. The script tries to exclude broken font files; you can change the list with the key -x EXCLUDED_PATTERNS. Again, there is an example in the doc directory.

3. Create the fontsel directory and the file pickfont.tex using the command

   bookshelf-mkfontsel allfonts

1.3.2 Driver file

The distribution contains the file spines.tex

```
% !TEX TS-program = lualatex
% !TEX encoding = UTF-8 Unicode
% !BIB TS-program = bibtex
\documentclass[landscape]{bookshelf}
\begin{document}
\nocite{*}
\bibliography{sample}
\bibliographystyle{bookshelf}
\end{document}
```

Change sample to the name of your bibliography, and typeset the file using lualatex and bibtex.

You should see your books nicely placed on the shelves. The small numbers under each book are meaningful: they are the line numbers in the allfonts file. You may use them if you play “Guess the font” game.

1.3.3 Options

The class comes set for making an a0paper page (1189 mm × 841 mm or 4′ 11″ × 2′ 10″) in landscape mode, suitable for large bibliographies. If you have a smaller .bib file, or if you want fewer volumes per page, you can change the paper size option in the \documentclass command to a smaller one: all the ‘A’ sizes from 5 to 0 are supported, plus the common US office sizes including Ledger (Tabloid).

There is also a portrait option to produce the page in that format instead of landscape.

1.4 Bugs

Some things don’t yet work as they should, and there are some features that may or may not make the final cut. Please report bugs at https://github.com/borisveytsman/bookshelf

2 Implementation

2.1 Auto-initialisation

This section is added automatically by *ClassPack* as a preamble to all classes and style packages. The fixltx2e package, which used to be included automatically, is no longer preloaded, as its features are now a part of the latest LaTeX 2_ε kernel.

The code starts with identity and requirements which are generated automatically as needed by the DocTeX system. For details see the ltxdoc package documentation.

```
\NeedsTeXFormat{LaTeX2e}[2017/04/15]
\ProvidesClass{bookshelf}[2020/05/24 v0.5
  Turn your bibliography into a bookshelf image]
```

fix-cm Preloaded functions to override the default LaTeX step-size font sizes (which can still be used, but are no longer restrictions).

```
\RequirePackage{fix-cm}
```

svgnames Pass the svgnames option to the xcolor package if that gets loaded later. This avoids a conflict with any other packages (eg hyperref) which use their own default is when they load xcolor.

```
  \PassOptionsToPackage{svgnames}{xcolor}
```

2.2 Options

The paper size and orientation are the only two valid options, both of which are the same as the standard documentclass options, and will be passed to the underlying class automatically, but they need recording so that they can be used by the geometry package. The default is for A0 paper, landscape.

```
\def\SIL@paper{a0paper}%
\DeclareOption{a0paper}{%
  \def\SIL@paper{a0paper}%
```

```
    \setlength\paperheight {1189mm}%
    \setlength\paperwidth  {841mm}}
\DeclareOption{a1paper}{%
    \def\SIL@paper{a1paper}%
    \setlength\paperheight {841mm}%
    \setlength\paperwidth  {594mm}}
\DeclareOption{a2paper}{%
    \def\SIL@paper{a2paper}%
    \setlength\paperheight {594mm}%
    \setlength\paperwidth  {420mm}}
\DeclareOption{a3paper}{%
    \def\SIL@paper{a3paper}%
    \setlength\paperheight {420mm}%
    \setlength\paperwidth  {297mm}}
\DeclareOption{a4paper}{%
    \def\SIL@paper{a4paper}%
    \setlength\paperheight {297mm}%
    \setlength\paperwidth  {210mm}}
\DeclareOption{a5paper}{%
    \def\SIL@paper{a5paper}%
    \setlength\paperheight {210mm}%
    \setlength\paperwidth  {148mm}}
\DeclareOption{b5paper}{%
    \def\SIL@paper{b5paper}%
    \setlength\paperheight {250mm}%
    \setlength\paperwidth  {176mm}}
\DeclareOption{letterpaper}{%
    \def\SIL@paper{letterpaper}%
    \setlength\paperheight {11in}%
    \setlength\paperwidth  {8.5in}}
\DeclareOption{legalpaper}{%
    \def\SIL@paper{legalpaper}%
    \setlength\paperheight {14in}%
    \setlength\paperwidth  {8.5in}}
\DeclareOption{executivepaper}{%
    \def\SIL@paper{executivepaper}%
    \setlength\paperheight {10.5in}%
    \setlength\paperwidth  {7.25in}}
\DeclareOption{ledgerpaper}{%
    \def\SIL@paper{ledgerpaper}%
    \setlength\paperheight {17in}%
    \setlength\paperwidth  {11in}}
\DeclareOption{tabloidpaper}{%
```

```
    \def\SIL@paper{tabloidpaper}%
    \setlength\paperheight {17in}%
    \setlength\paperwidth  {11in}}
\def\SIL@orient{landscape}%
\DeclareOption{landscape}{%
    \def\SIL@orient{landscape}%
    \setlength\@tempdima   {\paperheight}%
    \setlength\paperheight {\paperwidth}%
    \setlength\paperwidth  {\@tempdima}}
\DeclareOption{portrait}{%
    \def\SIL@orient{}}
```

2.3 Load the document base class

report This class is based on the standard LaTeX report class, with no special options except the extra sizes defined above. The default is A0 paper, landscape.

```
\DeclareOption*{\ClassWarning{bookshelf}{%
   Unknown option '\CurrentOption', please RTFM}}
\ProcessOptions\relax
\LoadClass{report}
```

2.4 Packages required for the class or package

fontspec Font specification setup for use with XeLaTeX.

```
\RequirePackage{fontspec}%
```

calc Required for calculations involving lengths or counters, such as changes to widths for margin adjustment.

```
\RequirePackage{calc}%
```

fp Used for fixed-point calculations;

```
\RequirePackage{fp}%
```

graphicx Provide for graphics (PNG, JPG, or PDF format (only) for pdflatex; EPS format (only) for standard LaTeX).

```
\RequirePackage{graphicx}%
```

xcolor Provide color.

```
\RequirePackage{xcolor}%
 \@ifundefined{T}{%
   \newcommand{\T}[2]{{\fontencoding{T1}%
     \selectfont#2}}}{}
```

There seems to be a bug in the T1 encoding of some package (unidentified, but possibly xcolor) which uses the command \T1, which is an impossibility (no digits allowed in command names). So we fake it here to stop LaTeX complaining, by dropping the first argument on the floor.

eso-pic Add picture commands (or backgrounds) to every page.

```
\RequirePackage{eso-pic}%
```

geometry Package for establishing margins and text area.

```
\RequirePackage[\SIL@paper,\SIL@orient,nohead,
 nofoot,margin=1cm]{geometry}%
```

2.5 Non-package resources

random.tex There is one resource not available in packaged form, the module that lets LaTeX create random values. This is in random.tex, which on the author's system is hiding in a directory texmf/tex/generic/genmisc/, in the texmf-dist tree, and indexed by an ls-R database, so it should therefore be findable by any TeX system.

```
\input{random.tex}
```

2.6 The code

This is beta software: the code is messy and covered in tracing output.

2.6.1 Font selection

maxfont This is set in the \input file pickfont.tex, which is created by the preparatory data script prepdata.sh. It is the number of working text fonts found on the system.

```
79 \newcounter{SIL@maxfont}
```

SIL@fontsel This is set to a random number between one and SIL@maxfont, and used as the name of the file containing the font name.

```
80 \newcounter{SIL@fontsel}
```

pickfont.tex This file is created by the preparatory data script prepdata.sh after it sets up the subdirectory list of valid text fonts. It sets SIL@maxfont the value of SIL@maxfont.

```
81 \input{pickfont.tex}
```

2.6.2 Color selection

SIL@maxcolno This value is set at the end of the file bookshelf-svgnam.tex. This is the number of color names found by the routine in prepdata.sh which extracts the color names.

```
82 \newcounter{SIL@maxcolno}
```

bookshelf-svgnam.tex The preparatory data script prepdata.sh retrieves the colors named in the svgnames option to the xcolor package and instantiates them as a LaTeX \ifcase list in the file bookshelf-svgnam.tex as the command \SIL@svgcolname.

```
83 \input{bookshelf-svgnam.tex}
```

SIL@loopcount The random font selection is done in a loop because of the need to test the values. This counter counts the iterations...

```
84 \newcounter{SIL@loopcount}
```

SIL@maxloop ...and this one the limit.

```
85 \newcounter{SIL@maxloop}
```

SIL@bgcolno The colors are selected numerically. This value is the background color of the spine of a book.

```
86 \newcounter{SIL@bgcolno}
```

SIL@fgcolno And this is the foreground color, used to typeset the title and author on the spine of a book.

```
87 \newcounter{SIL@fgcolno}
```

\SIL@splitpoint SIL@bgcolno SIL@fgcolno To make sure that SIL@bgcolno and SIL@fgcolno are distinct, we will need to pick one 'dark' and one 'light', crudely distinguished by examining their 'brightness' (monochrome intensity value) using the formula $b = \sqrt{(.241r^2 + .691g^2 + .068b^2)}$ due to Nir Dobovizki. From inspection, the modal point of the SVG values occurs around 0.6, so use use this to determine if the randomly-selected color is 'dark' or 'light'. Because it's a decimal fraction, we express it as a dimension and strip off the 'pt' later.

```
88 \newlength{\SIL@splitpoint}
89 \setlength{\SIL@splitpoint}{0.6pt}
```

\SIL@bgcol We establish defaults for the background color...

```
90 \def\SIL@bgcol{White}
```

\SIL@fgcol ...and the foreground color.

```
91 \def\SIL@fgcol{Black}
```

\SIL@bgval The values computed by the prepdata.sh script and stored in bookshelf-svgnam.tex are decimal fractions, to they need to be retrieved as lengths. This is the background value...

```
92 \newlength{\SIL@bgval}
```

\SIL@fgval ...and the foreground value.

```
93 \newlength{\SIL@fgval}
```

\SIL@bgfgdiff The 'dark' or 'light' test discussed above also needs to test if the values are too close to the splitpoint. By examination, if the values have an absolute difference of 0.2 they should be visually distinct enough. The difference is calculated and stored in this length variable, as it's a decimal fraction.

```
\newlength{\SIL@bgfgdiff}
```

\SIL@notyetcols In the testing for colors, the nested conditionals set this switch true or false, so that it can be used to control the iteration through successive attempts to find suitable random values.

```
\newif\ifSIL@notyetcols
```

2.6.3 Page border setup

\AddToShipoutPictureBG The page background color is set to a pale brown roughly matching the pine veneer of IKEA bookcases, with the inner page (behind the books) in a dark shadow brown. The technique for imposing a colored margin is due to Ulrike Fischer and uses the commands from the eso-pic package.

```
\pagecolor{BurlyWood}
\AddToShipoutPictureBG{%
 \AtTextLowerLeft{\color{SaddleBrown}%
  \rule[-\footskip]{\textwidth}{%
   \dimexpr\textheight+\footskip}}}
```

2.6.4 Size and shape

Each book is assigned a random height and width, within the bounds set by the maxima and minima. The final dimensions may then be modified by the choice of layout and font.

```
\newlength{\SIL@bookheight}
\newlength{\SIL@bookwidth}
\newlength{\SIL@minbookwidth}
\newlength{\SIL@maxbookwidth}
\newlength{\SIL@minbookheight}
\newlength{\SIL@maxbookheight}
```

2.6.5 Title and author dimensions

The title and author need to be measured, and decisions are made about what size they need to be. The two layouts (author separately at the top, and author inline to title) are distinguished with the \SIL@topauthor conditional. If the title (with or without the author can fit on one line (rather than multiple lines) this is signalled with the \SIL@titleoneline conditional.

```
\newlength{\SIL@titlewidth}
\newlength{\SIL@authorwidth}
\newlength{\SIL@titleheight}
\newlength{\SIL@authorheight}
\newlength{\SIL@scaledtitle}
\newlength{\SIL@heightfortitle}
\newbox\SIL@titlebox
\newif\ifSIL@topauthor
\newif\ifSIL@titleoneline
```

2.6.6 Handling the math

SIL@scale To extract the integer part of a fixed-point value, we define a simple strip which uses the integer and throws away the rest. The integer ends up in this counter.

```
\newcounter{SIL@scale}
```

\SIL@scaleint The integer macro returns the counter above.

```
\def\SIL@scaleint#1.#2\sentinel{%
  \setcounter{SIL@scale}{#1}}
```

2.6.7 Settings

We set the space around a box and the thickness of the rule, and remove the page numbers.

```
\fboxsep1em\fboxrule.1pt
\pagestyle{empty}
```

2.6.8 Auxillary macro: fitting text in a box

\SIL@fittext For typesetting title we use an auxillary macro \SIL@fittext. It has four parameters: the text to be typeset, the width (W), the height H, and the box $W \times H$ to put the text into. We

want to get the maximal font size that still fits in the box. Unfortunately there is a limitation on the maximal number of fonts TEX can handle (currently 9000 by default). Since each size change counts as a new font, things can quickly go out of hand. Therefore instead of scaling the font we scale the box.

So our aim is to find the maximal scaling factor S such as (1) the text is typeset in a $w \times h$ box, (2) the text box scales to the given width, $W = Sw$, (3) the text box does not overflow the height, $H \leq Sh$.

The algorithm is the following:

1. If the text fits in one line, we expand the box for the line to occupy W, setting $S = W/w$.
2. Otherwise, we try to typeset the text in the box of horizontal size W ($S = 1$) and calculate box height h. We determine how much we can expand or shrink the box, setting $S = H/h$.
3. We typeset the text in the box of width $w = W/S$.
4. Due to changed line breaks its height h might be higher than H/S. In this case we start to decrease S by 5% on each step and repeat typesetting until $w \leq W/S$

Now, the implementation.

First, we calculate W and H by stripping the pt dimension

```
\def\SIL@fittext#1#2#3#4{%
  \@tempdima=#2\relax
  \edef\SIL@W{\strip@pt\@tempdima}%
  \@tempdima=#3\relax
  \edef\SIL@H{\strip@pt\@tempdima}%
```

Try to set up the text in one line

```
  \setbox#4=\hbox{\raggedright\noindent#1}%
  \@tempdima=\wd#4\relax
  \edef\SIL@w{\strip@pt\@tempdima}%
  \ifdim#2>\@tempdima\relax
    \FPeval\SIL@S{\SIL@W/\SIL@w}%
   \typeout{Text fits in one line: have H=\SIL@w pt, want \SIL@W pt}%
```

```
    \typeout{Trying S=\SIL@S}%
  \else
```

We start with the scale factor $S = 1$. We add \vskip0pt to the text to set the box depth to zero.

```
    \typeout{Text does not fit in one line}%
    \def\SIL@S{1}%
    \FPeval\SIL@w{\SIL@W/\SIL@S}%
    \setbox#4=\vbox{\hsize=\SIL@w pt\relax
      \raggedright\noindent#1\vskip0pt}%
    \@tempdima=\ht#4\relax
    \edef\SIL@h{\strip@pt\@tempdima}%
    \@tempdima = \SIL@S \@tempdima\relax
   \typeout{Trying S=\SIL@S. Got H=\the\@tempdima. Want \SIL@H pt}%
    \FPeval\SIL@S{\SIL@H/\SIL@h}%
  \fi
```

Rescaling the box for the first time. If S on the previous step is below 1, start again with 1.

```
  \FPmax\SIL@S\SIL@S{1}%
  \FPeval\SIL@w{\SIL@W/\SIL@S}%
  \setbox#4=\vbox{\hsize=\SIL@w pt\relax
    \raggedright\noindent#1\vskip0pt}%
  \@tempdima=\ht#4\relax
  \edef\SIL@h{\strip@pt\@tempdima}%
  \@tempdima = \SIL@S \@tempdima\relax
 \typeout{Trying S=\SIL@S. Got H=\the\@tempdima. Want \SIL@H pt}%
```

If the text does not fit, keep reducing it by 5% at a type

```
  \ifdim\@tempdima>#3\relax
    \loop
      \FPeval\SIL@S{0.95*\SIL@S}%
      \FPeval\SIL@w{\SIL@W/\SIL@S}%
      \setbox#4=\vbox{\hsize=\SIL@w pt\relax
        \raggedright\noindent#1\vskip0pt}%
      \@tempdima=\ht#4\relax
      \edef\SIL@h{\strip@pt\@tempdima}%
      \@tempdima = \SIL@S \@tempdima\relax
    \typeout{Trying S=\SIL@S. Got H=\the\@tempdima. Want \SIL@H pt}%
    \ifdim\@tempdima>#3\repeat
  \fi
```

And the final typesetting

```
  \setbox#4=\vbox to #3{\hsize=#2\relax
    \vfill
    \noindent
    \scalebox{\SIL@S}{\vbox{\hsize=\SIL@w pt\relax
       \raggedright\noindent#1\vskip0pt}}%
    \vfill}%
}
```

2.6.9 Selecting the font for the book

In a multilingual library some books can be typeset only in specific fonts. Here we randomly select a font to typeset the given book.

We define a macro that checks whether the given string can be typeset in the font just defined by fontspec.

We write the program in expl3 syntax because it has nice mapping subroutines and becasue fontspec internal variables are in expl3.

```
\ExplSyntaxOn
```

An auxillary routine checking whether the character can be typeset with the current font. Copied from fontspec internals

```
\prg_new_conditional:Nnn \___SIL_primitive_font_glyph_if_exists:n {TF,F}
  {
    \tex_iffontchar:D \SILmfont `#1 \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

And the document command

```
\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}
}
\ExplSyntaxOff
```

We define some counters and flags for the font selection

```
\def\SIL@maxfonttries{100}
\newif\ifSIL@fontfound
```

Another twist is that we cannot have too many fonts used. Therefore we add all fontsel files to a stack, and after having 2100 of them, we reuse opened files. Again, it is easy to do in expl3.

```
\ExplSyntaxOn
\seq_new:N \l___SIL_fontstack
\NewDocumentCommand\AddFontToStack {m} {%
  \seq_gput_right:Ne \l___SIL_fontstack {#1}
}
\NewDocumentCommand\ReuseFont {} {
  \seq_rand_item:N \l___SIL_fontstack
}
\ExplSyntaxOff
\newcount\SIL@num@fontsel@files
\SIL@num@fontsel@files=0
\def\SIL@max@fontsel@files{5500}
```

2.6.10 Making the book

The \makebook macro is huge, and handles all the detail of making a book spine. It takes two mandatory arguments: the author and the title of the book.

\makebook Start by announcing the entry label and setting the values that need to be reset every time.

```
\newcommand{\makebook}[2]{%
  \typeout{^^JTypesetting #1---#2}%
  \setcounter{SIL@maxloop}{10}%
  \setcounter{SIL@loopcount}{0}%
% observed
  \setlength{\SIL@minbookwidth}{5mm}%
  \setlength{\SIL@maxbookwidth}{20mm}%
% A5 to A4 height
  \setlength{\SIL@minbookheight}{70mm}%
  \setlength{\SIL@maxbookheight}{110mm}%
  \setlength{\SIL@bookwidth}{0pt}%
  \setlength{\SIL@bookheight}{0pt}%
  \setlength{\SIL@heightfortitle}{0pt}%
  \SIL@topauthorfalse
```

\loop Start a loop which will pick two random integers, one for background and one for foreground colors. Look these up in the \SIL@svgcolval (in bookshelf-svgnam.tex) to get the brightness values, and calculate the absolute distance between them.

```
  \loop
    \addtocounter{SIL@loopcount}{1}%
    \typeout{Try \theSIL@loopcount}%
    \setrannum{\c@SIL@bgcolno}{1}{%
      \c@SIL@maxcolno}%
    \typeout{BG=\theSIL@bgcolno}%
    \setrannum{\c@SIL@fgcolno}{1}{%
      \c@SIL@maxcolno}%
    \typeout{FG=\theSIL@fgcolno}%
    \setlength{\SIL@bgval}{%
      \SIL@svgcolval{\theSIL@bgcolno}pt}%
    \typeout{BGval=\the\SIL@bgval}%
    \setlength{\SIL@fgval}{%
      \SIL@svgcolval{\theSIL@fgcolno}pt}%
    \typeout{FGval=\the\SIL@fgval}%
    \setlength{\SIL@bgfgdiff}{%
      \SIL@bgval - \SIL@fgval}%
    \typeout{Split gap is \the\SIL@bgfgdiff}%
    \ifdim\SIL@bgfgdiff<0pt
```

```
248       \setlength{\SIL@bgfgdiff}{%
249         \SIL@fgval - \SIL@bgval}%
250       \typeout{Using absolute value
251         \the\SIL@bgfgdiff}%
252     \fi
```

The colours need to be separated either side of the 0.6 splitpoint value of the calculated brightness, so make this the outer test, and make the inner test for the separation difference. This will return true if the colors are separated enough, and come from opposite sides of the split point. If the loop makes SIL@maxloop iterations without finding a pair of values, use the default white on black.

SIL@maxloop

```
253     \ifdim\SIL@bgval<\SIL@splitpoint
254       \ifdim\SIL@fgval>\SIL@splitpoint
255         \ifdim\SIL@bgfgdiff>0.2pt
256           \SIL@notyetcolsfalse
257         \else
258           \SIL@notyetcolstrue
259         \fi
260       \else
261         \SIL@notyetcolstrue
262       \fi
263     \else
264       \ifdim\SIL@fgval<\SIL@splitpoint
265         \ifdim\SIL@bgfgdiff>0.2pt
266           \SIL@notyetcolsfalse
267         \else
268           \SIL@notyetcolstrue
269         \fi
270       \else
271         \SIL@notyetcolstrue
272       \fi
273     \fi
274     \typeout{BG=\theSIL@bgcolno,
275           FG=\theSIL@fgcolno}%
276     \ifnum\c@SIL@loopcount>\c@SIL@maxloop
277       \SIL@notyetcolsfalse
278     \fi
279   \ifSIL@notyetcols\repeat
280   \def\SIL@bgcol{\SIL@svgcolname{%
281     \theSIL@bgcolno}}%
```

```
    \def\SIL@fgcol{\SIL@svgcolname{%
      \theSIL@fgcolno}}%
    \ifnum\c@SIL@loopcount>\c@SIL@maxloop
     \typeout{Using default colors after \the\c@SIL@maxloop\space attempts}%
      \def\SIL@bgcol{Black}%
      \def\SIL@fgcol{White}%
    \fi
    \typeout{BG=\SIL@bgcol, FG=\SIL@fgcol}%
```

Now pick a random font: the files generated by prepdata.sh are named as integers with a .tex extension in the fontsel directory. These files load the font as \SILmfont (no @ sign, because this is occurring in user mode), and define \SILmfontname as the name (for the same reason).

```
    \c@SIL@loopcount=1\relax
    \loop
      \ifnum\SIL@num@fontsel@files<\SIL@max@fontsel@files
        \advance\SIL@num@fontsel@files by 1\relax
      \typeout{Opening new fontsel file, counter=\the\SIL@num@fontsel@files}%
        \setrannum{\c@SIL@fontsel}{1}{\c@SIL@maxfont}%
        \AddFontToStack{\the\c@SIL@fontsel}%
      \else
        \typeout{Reusing fontsel file}%
        \expandafter\c@SIL@fontsel\ReuseFont\relax
      \fi
      \input{fontsel/\theSIL@fontsel.tex}\unskip%
     \typeout{Trying \SILmfontname, attempt \the\c@SIL@loopcount}%
      \CanTypesetTF{#2---#1}{\global
        \SIL@fontfoundtrue}{\global
        \SIL@fontfoundfalse}%
      \ifSIL@fontfound
        \c@SIL@loopcount=\SIL@maxfonttries\relax
      \else
        \addtocounter{SIL@loopcount}{1}%
      \fi
    \ifnum\c@SIL@loopcount<\SIL@maxfonttries\repeat
    \ifSIL@fontfound
    \typeout{Set in \SILmfontname}%
```

Measure the author width and height at the default size (10pt). If the author fits in 90% of the maximum width of the book, we put it at the top of the spine and shrink the book

width to 1.1 times the set width, provided that is not less than the defined minimum width. The book width is therefore fixed at this point and won't change later.

```
\settowidth{\SIL@authorwidth}{%
          \SILmfont#1}%
\typeout{Author width: \the\SIL@authorwidth}%
\settoheight{\SIL@authorheight}{%
           \SILmfont#1}%
\typeout{Author height: \the\SIL@authorheight}%
\ifdim\SIL@authorwidth<.9\SIL@maxbookwidth
  \typeout{Author width is less than 90\%
        of \the\SIL@maxbookwidth}%
  \setlength{\SIL@bookwidth}{%
           1.1\SIL@authorwidth}%
  \typeout{Book width set to \the\SIL@bookwidth}%
  \ifdim\SIL@bookwidth<\SIL@minbookwidth
    \setlength{\SIL@bookwidth}{%
             \SIL@minbookwidth}%
    \typeout{Book width reset to min
           \the\SIL@minbookheight}%
  \fi
  \SIL@topauthortrue
\else
  \typeout{Author won't fit in .9 of
        \the\SIL@maxbookwidth}%
\fi
```

We now have enough data to make a shot at the dimensions. Pick a random book height and set the height available for the title (set sideways) to 90% of that, so that it fits comfortably. Then if the author was earlier assigned to the top of the spine, reduce this height available for the title by 1.2 times the height occupied by the author (again, to leave a little space). In this case, the width has already been set; otherwise, generate a random width now.

```
\typeout{Limits: width=\the\SIL@minbookwidth
          –\the\SIL@maxbookwidth;
              height=\the\SIL@minbookheight
          –\the\SIL@maxbookheight}%
\setrandim{\SIL@bookheight}%
        {\SIL@minbookheight}%
```

```
          {\SIL@maxbookheight}%
\typeout{Height generated as
        \the\SIL@bookheight}%
\setlength{\SIL@heightfortitle}%
          {.9\SIL@bookheight}%
\typeout{Height available for title (90\%):
        \the\SIL@heightfortitle}%
\ifSIL@topauthor
  \typeout{Width set because author fits:
          \the\SIL@bookwidth}%
  \addtolength{\SIL@heightfortitle}%
              {-1.2\SIL@authorheight}%
  \typeout{Height available for title reset to
          \the\SIL@heightfortitle}%
\else
\setrandim{\SIL@bookwidth}%
          {\SIL@minbookwidth}%
          {\SIL@maxbookwidth}%
\typeout{Width generated as
        \the\SIL@bookwidth}%
\fi
```

Finally, set a \vbox to the defined width *less* the space occupied by the \fcolorbox border and rule; then set the \fcolorbox with the chosen colors, with the author at the top if that's what was selected earlier, and the title below, either scaled using \scalebox if it was a single-line title, or with the amended font size if it was a multiline title.

For a setting with the author inline to the title, just do the scaling of the title.

```
\leavevmode\vbox{\hsize\SIL@bookwidth
  \advance\hsize by2\fboxsep
  \advance\hsize by2\fboxrule
\fcolorbox{black}{\SIL@bgcol}{%
  \ifSIL@topauthor
    \typeout{Setting with top author}%
    \vbox to\SIL@bookheight{\hsize\SIL@bookwidth
      \typeout{Spine is a vbox to
        \the\SIL@bookheight,
        hsize=\the\SIL@bookwidth}%
      \centering
      \SILmfont\color{\SIL@fgcol}%
```

```
376         #1%
377         \par\vfill
378         \SIL@fittext{\color{\SIL@fgcol}\SILmfont
379           #2}{\SIL@heightfortitle}{\SIL@bookwidth}%
380         {\SIL@titlebox}%
381         \rotatebox{90}{\box\SIL@titlebox}%
382         }%
383       \else
384         \typeout{Setting author inline to title}%
385         \vbox to\SIL@bookheight{\hsize\SIL@bookwidth
386           \SIL@fittext{\color{\SIL@fgcol}\SILmfont
387             #1\quad
388             ---\quad#2}{\SIL@bookheight}{\SIL@bookwidth}%
389           {\SIL@titlebox}%
390           \rotatebox{90}{\box\SIL@titlebox}%
391         }%
392       \fi
393     }%
```

At the bottom, add a colored bar to fake up the shelf the books stand on. The number is the number of the font that was selected, and is there for error-tracing purposes only.

```
394     \\\fboxsep0pt\fboxrule0pt%
395     \colorbox{BurlyWood}{\hbox to\hsize{%
396         \hfil\vrule height3mm depth6mm width0pt
397         \normalfont\scriptsize\theSIL@fontsel\hfil}}%
398      }%
399      \penalty0
400   \else % font not found
401       \typeout{Did not find font for #1--#2}%
402
403   \fi}

404   \raggedright
```

References

[1] Scott O. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. RFC 2119. Mar. 1997. doi: 10.17487/RFC2119. url: https://www.rfc-editor.org/info/rfc2119.

[2] Kovid Goyal. calibre *User Manual*. 2024. url: https://manual.calibre-ebook.com/.

[3] Boris Veytsman. “Extending Peter Flynn’s bookshelf package for multilanguage libraries”. In: *TUGboat* 45.2 (2024), pp. 193–195. doi: 10.47397/tb/45-2/tb140veytsman-bookshelf. url: https://tug.org/TUGboat/tb45-2/tb140veytsman-bookshelf.pdf.

Change History

v0.1
General: First packaged draft: Done manually from .tex file. 1
v0.2
General: Started documentation: Code doc done, user doc still missing. 1
v0.3
General: Finished first pass on documentation: 1) Done preliminary testing; 2) Script adapted for Mac OS X. 1
v0.4
General: Completed documentation: 1) Updated note on bug in biber when processing sgml.bib; 2) Removed sgml.bib as example until problems are resolved; 3) Backtracked on attempt to use the monographic title for articles, chapters, etc; 4) Revised notes on production. 1
v0.5
General: Finished initial testing: Replaced hyperref with hypdoc to avoid makeindex bug. 1
v1.0
General: New maintainer. Starting rewriting 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Q

R

S

T

V

W