Introducing oldstyle figures in existing virtual fonts*

Abstract
This paper describes a *Ruby* script *osf* that can be used to make a copy of a virtual font with its figures replaced with old style figures.

Keywords
ruby script txfonts pxfonts oldstyle figures

Introduction

When I was typesetting a biography I chose palatino for its font because I like palatino, especially for texts about history. I could have used the `palatino` package, but I don't like its math, nor its typewriter face, so I normally use the `pxfonts` package instead. However, both packages use table figures while, especially in palatino, old style would fit a lot better.

Reading the `pxfonts` documentation, I was happy to see that it contains tables with Text Companion Fonts, with old style figures. However, although the tables are there, the documentation doesn't say anything about how to get such characters in your document. Inspection of the style file `pxfonts.sty`, also, shows that it ignores these fonts.

So I asked the gurus: how do I get old style figures with `pxfonts`? And I was showered with font jargon, most of which I had heard about, but never really understood. But what I distilled from it was: have a look at virtual fonts–you can export them from the binary `.vf` files with `vftovp`, edit the resulting `.vpl` file and then convert that back to `.vf` with `vptovf`.

Virtual font files

So, for a start, I had a look at the virtual font files for the *Roman Upright* fonts, `pxr.vf`, with table figures, and the *Text Companion Roman Upright* fonts, `pcxr.vf`, with old style figures. I did so by running:

```
vftovp pxr pxr pxr
vftovp pcxr pcxr pcxr
```

*Without the help of Siep Kroonenberg on font matters this article would never have seen daylight

As you can read in the `vftovp` documentation, this converts the information in pxr.vf and pxr.tfm into the readable and editable *Virtual Property List*: `pxr.vpl` in your current directory; similarly for `pcxr`. Here are aligned excerpts of the two files:

```
 1  (VTITLE PXR)                  (VTITLE PCXR)
 2  (FAMILY PXR)                  (FAMILY UNSPECIFIED)
 3  (FACE F MRR)                  (FACE F MRR)
 4  (CODINGSCHEME TEX TEXT)      (CODINGSCHEME TEX TEXT CO...
 5  (DESIGNSIZE R 10.0)           (DESIGNSIZE R 10.0)
 6  (COMMENT DESIGNSIZE IS IN ... (COMMENT DESIGNSIZE IS IN...
 7  (COMMENT OTHER SIZES ARE M... (COMMENT OTHER SIZES ARE
 8  (CHECKSUM O 22721014137)      (CHECKSUM O 32012256317)
 9                                (SEVENBITSAFEFLAG TRUE)
10  (FONTDIMEN                    (FONTDIMEN
11     (SLANT R 0.0)                 (SLANT R 0.0)
12     (SPACE R 0.25)                (SPACE R 0.25)
13     (STRETCH R 0.2)               (STRETCH R 0.2)
14     (SHRINK R 0.1)                (SHRINK R 0.1)
15     (XHEIGHT R 0.469)             (XHEIGHT R 0.469)
16     (QUAD R 1.0)                  (QUAD R 1.0)
17     (EXTRASPACE R 0.1)            (EXTRASPACE R 0.1)
18     )                             )
19  (MAPFONT D 0                  (MAPFONT D 0
20     (FONTNAME RPXPPLR)            (FONTNAME RPCXR)
21    (FONTCHECKSUM O 3657114...    (FONTCHECKSUM O 450707...
22     (FONTAT R 1.0)                (FONTAT R 1.0)
23     (FONTDSIZE R 10.0)            (FONTDSIZE R 10.0)
24     )                             )
25  (MAPFONT D 1                  (MAPFONT D 1
26     (FONTNAME RPXR)               (FONTNAME RPXPPLR)
27    (FONTCHECKSUM O 2703202...    (FONTCHECKSUM O 365711...
28     (FONTAT R 1.0)                (FONTAT R 1.0)
29     (FONTDSIZE R 10.0)            (FONTDSIZE R 10.0)
30     )                             )
31  (LIGTABLE                     [... no ligtable ...]
32
33  [... lots of lines ...]       [... lots of lines ...]
34
35     )
36  (CHARACTER O 0                (CHARACTER O 0
37     (CHARWD R 0.556)              (CHARWD R 0.332996)
38     (CHARHT R 0.686247)           (CHARHT R 0.685245)
39     (CHARDP R 0.00475)
40     (MAP                          (MAP
41        (SELECTFONT D 1)              (SELECTFONT D 1)
42        (SETCHAR O 0)                 (SETCHAR O 36)
43        )                             )
44     )                             )
45
46  [... octal 1..57 ...]         [... octal 1..57 ...]
47
48  (CHARACTER C 0                (CHARACTER C 0
```

```
49     (CHARWD R 0.5)                (CHARWD R 0.5)
50     (CHARHT R 0.686247)           (CHARHT R 0.476)
51     (CHARDP R 0.017999)           (CHARDP R 0.011)
52     (MAP                          (MAP
53        (SETCHAR C 0)                 (SETCHAR C 0)
54        )                             )
55     )                             )
56   (CHARACTER C 1                (CHARACTER C 1
57     (CHARWD R 0.5)                (CHARWD R 0.5)
58     (CHARHT R 0.702999)           (CHARHT R 0.476)
59     (CHARDP R 0.00475)            (CHARDP R 0.011)
60     (MAP                          (MAP
61        (SETCHAR C 1)                 (SETCHAR C 1)
62        )                             )
63     )                             )
64
65   [... digits 2..8 ...]         [... digits 2..8 ...]
66
67   (CHARACTER C 9                (CHARACTER C 9
68     (CHARWD R 0.5)                (CHARWD R 0.5)
69     (CHARHT R 0.686247)           (CHARHT R 0.476)
70     (CHARDP R 0.017999)           (CHARDP R 0.237247)
71     (MAP                          (MAP
72        (SETCHAR C 9)                 (SETCHAR C 9)
73        )                             )
74     )                             )
75
76   [... more characters ...]     [... more characters ...]
```

These virtual property listings look like lisp code. Their structure is described in the source of Donald Knuth's `vptovf` program; you can generate the documentation with:

```
weave vptovf.web
pdftex vptovf
```

You can find `vptovf.web` on the TEXLive Collection's DVD in the directory `ctan/systems/knuth/etc`.

The virtual property listings start with some general comments and dimensions (lines 1–18), followed by one or more font mapping sections (lines 19–30), a ligature table (lines 31–35, not in `pcxr.vpl`) and finally one section for each defined character, starting with octal 0 (line 36–44) and potentially ending with octal 377. Characters are identified with their octal number (like `CHARACTER O 0` on line 36), except for the digits and letters, which are identified by themselves (like `CHARACTER C 0` for the digit 0 on line 48).

A character section defines width, height and depth for the corresponding glyph (lines 37–39 for octal 0 for example) and it tells from which font it is taken (line 41) and from which position in that font (line 42). Thus the glyph for character octal 0 in `pxr` is taken from position 0 of font 1, which points to the font `rpxr` on line 26 in the second font mapping section. Similarly, the glyph for octal 0 of `pcxr` is taken from font `rpxpplr`, position 0.

Now what we are interested in is the glyphs for the digits. In both virtual fonts, `pxr` and `pcxr`, these are taken from the default font, because there is no `SELECTFONT` statement in their character sections. This default is the first font, which means `rpxpplr` for `pxr` and `rpcxr` for `pcxr`. This made me think that what I probably had to do was to take the first `MAPFONT` section (pointing to `rpcxr`) from the old style virtual font `pcxr`, and add it to the `pxr` virtual font to give it a third `MAPFONT` section. I simply appended it to the end of the `.vpl` file. Of course, it needed a new unique identifier, so I changed `MAPFONT D 0` into `MAPFONT D 2`. Furthermore, I removed the character sections of the digits, and appended the (old style) digit character sections taken from `pcxr.vpl`. These had of course to point to the new `MAPFONT` section, so I added `SELECTFONT D 2` to each of them.

Finally, I converted the new `pxr.vpl` back into `pxr.vf` and `pxr.tfm`:

```
vptovf pxr pxr pxr
```

and I made a little test file:

```
\documentclass{article}
\usepackage{pxfonts}
\begin{document}
   Hello number 0123456789!
\end{document}
```

And it worked!

Editing `.vf` files with a script

Editing a `.vf` file as described takes quite some time. Moreover, we need to edit more than a single file if we want old style figures to appear not only in roman upright text, but also in bold, sans serif, slanted, italic and so on. In the case of `pxfonts`, this means editing 16 files, in the case of `txfonts` even 42 files!

Therefore I made a *Ruby* script to take over the work. The script is listed at the end of this article.[1] It's heart is a subroutine *convert_font*, which takes two arguments: the name of the font to be converted (say `pxr`) and the name of the font containing old style digits (say `pcxr`). In essence, it does te following:

- convert the table font to `.vpl`, saving it in a new `.vpl` file, `pxr.vpl` removing the digits on the fly and remembering the highest mapfont identifier in it.
- convert the old style font to `.vpl`, isolating it's map font and digit sections.
- append these to the new `pxr.vpl` file after fixing the map font identifier.
- convert the new `pxr.vpl` file to `pxr.vf` and `pxr.tfm` with `vptovf`.

The rest of the script handles the command line argument to give you some options:

with two arguments, the first should be a font to be converted, the second a font from which the old style digits should come from. So you can say:

```
osf pxr pcxr
```

which would create two files in the current directory: `pxr.vf` and `pxr.tfm`.

with one argument, it should be a font to be converted. The old style font from which the digits are to be taken will be searched in the directory where the converted font occurs. You will be presented a list of those fonts from which you can make your choice. There will be a default which the script thinks is most likely on the basis of its name. So the following would be a possible dialog:

```
osf pxr
I found 10 fonts with old style digits:
 1  3 pcxb
 2  4 pcxbi
 3  5 pcxbsl
 4  3 pcxi
 5  1 pcxr
 6  4 pcxsl
 7  4 pxbmi
 8  5 pxbmi1
 9  3 pxmi
10  4 pxmi1
My guess is 5 (pcxr)
Your guess [5]:
```

with no arguments you can make use of predefined combinations of fonts and their old style companions. Currently, two combinations are defined in the DATA section of the script: `pxfonts` and `txfonts`. The following would be a possible dialog:

```
osf
2 font conversions have been predefined
1 pxfonts
2 txfonts
Please make your choice [1]:
Converting pxfonts
p1xb       pcxb
p1xbi      pcxbi
p1xbsc     pcxb
p1xbsl     pcxbsl
p1xi       pcxi
p1xr       pcxr
p1xsc      pcxr
p1xsl      pcxsl
pxb        pcxb
pxbi       pcxbi
pxbsc      pcxb
pxbsl      pcxbsl
pxi        pcxi
pxr        pcxr
pxsc       pcxr
pxsl       pcxsl
```

As a result, you would find `.vf` and `.tfm` files for all faces of `pxfonts` that may need conversion to old style.

Of course, storing all these files together with your LaTeX document is not very elegant, although you might go for this option occasionally. Another option would be to store these files in your user tree, or even in the local tree. But that would mean that you would get old style figures in *all* your documents, which is not necessarily what you want. A better alternative is to give your font a new name.

Renaming the font

This section will be dedicated to `pxfonts` and will describe how to create a new font, `osf-pxfonts`, from it, including a style file, `osf-pxfonts.sty` which gives you access to the same font families, series and shapes as does `pxfonts`, but with oldstyle figures for the roman and sans serif families. The encoding will be T1. It's easy to translate what you read here to other font collections.

You can give your font another name by adding a prefix (say `osf-`) to all names, and moving them to a new directory. However, you now have a problem: since you have changed the name, you cannot use the style file (`pxfonts.sty`) anymore. But fortunately, there is a solution for this: you can make a new style file (say `osf-pxfonts.sty`) which imports the original style file and then tweak it a little, like this:

```
\RequirePackage{pxfonts,t1enc}
\AtBeginDocument{%
  \usefont{T1}{osf-pxr}{m}{n}
  \renewcommand{\sfdefault}{osf-pxss}
}
```

This defines a style file which uses T1 encoding and oldstyle figures for the roman (`osf-pxr`) and sans serif (`osf-pxss`) fonts. However, this is not enough: LaTeX uses font definition (`.fd`) files, one for each font family, to map fonts to various series (weights and widths) and shapes (normal, italic, slanted, small caps). These font definition files are named after, and refer to, the fonts we just renamed, so we need renamed copies of these files and we also need to rename the font references inside them.

This is not really complicated. We need new font definition files for only two families: roman and sans serif. Since we used the T1 encoding for our style file, we can confine ourselves to `t1pxr.fd` for the roman

fonts and t1pxss.fd for the sans serif fonts. We rename these to t1osf-pxr.fd and t1osf-pxss.fd[2].

If you have a look inside t1pxr.fd, you easily recognize the font names that have been listed when you ran the script to convert the pxfonts and for which new font files were produced with osf-prefixed names. So all there is to be done is to add osf- before all these names; and since all names start with pxr or p1x we can simply replace pxr with osf-pxr and p1x with osf-p1x everywhere.

The same can be done for t1pxss.fd; here however, we see that the sans serif fonts are actually borrowed from the txfonts. This means that for the pxfonts we need to convert the txfonts as well if we want to have oldstyle figures in sans serif. And, of course, we need to substitute pxss with osf-pxss and t1x (not p1x) with osf-t1x in the .fd file.

Now the good news is that for pxfonts and txfonts, predictable as it is, you don't have to do this editing yourself: the script does it for you and creates the necessary .fd files.

Switching between oldstyle and table figures

Although you may like oldstyle figures, there are occasions, even within one document, where you might like to switch back to table figures. For example, a table listing many numbers does not look very good with oldstyle figures.

As the style file presented above already suggests, it is easy to make switching between oldstyle and table figures possible, either by defining options to the package or, for switching on the fly, by defining switching commands. The following example does both. In addition, instead of boldly switching fonts with \usefont, it first checks which family is currently in effect and changes the font accordingly, so that one can say, for example, \bfseries\itshape\osfigures without the \osfigures resetting series and shape to the defaults:

```
\RequirePackage{pxfonts,t1enc}

\def\tb@rm{pxr}
\def\tb@sf{pxss}
\def\osf@rm{osf-pxr}
\def\osf@sf{osf-pxss}

\newcommand{\osfigures}{%
  \renewcommand{\rmdefault}{\osf@rm}%
  \renewcommand{\sfdefault}{\osf@sf}%
  \ifx\f@family \tb@rm \rmfamily\fi
  \ifx\f@family \tb@sf \sffamily\fi}

\newcommand{\tbfigures}{%
  \renewcommand{\rmdefault}{\tb@rm}%
  \renewcommand{\sfdefault}{\tb@sf}%
  \ifx\f@family \osf@rm \rmfamily\fi
  \ifx\f@family \osf@sf \sffamily\fi}

\DeclareOption{osfigures}{
   \osfigures
   \let\@Fam\osfigures
}
\DeclareOption{tbfigures}{
   \tbfigures
   \let\@Fam\tbfigures
}
\ExecuteOptions{osfigures}
\ProcessOptions
\AtBeginDocument{\@Fam}
```

For the predefined fonts, the script creates this style file also for you.

Configuration

Once having created a full-blown set of files for oldstyle pxfonts and txfonts, I thought I could as well let the script move the files to directories in the TEX-tree where LATEX expects them. So when converting predefined fonts, that is: when run with no arguments, the script will store the files in the usual subdirectories in the user tree ($HOME/texmf). And before finishing, the script will run mktexlsr, so that the files will be found.

Testing

You will probably want to test your newly created font and verify that the series- and shape-switching commands work. Here is a test source that does so for you– just change the second line if you want to test other fonts:

```
\documentclass{article}
\usepackage{osf-txfonts}
\parindent0pt
\def\text{Hello Wörld! 0123456789 }
\newcommand{\test}[2]{%
  \def\F{Sans}\def\Arg{#2}
  \ifx\Arg\F\let\F\sf\else\let\F\relax\fi
  \begin{tabular}{ll}
    \multicolumn{2}{l}{#1 #2}\\\hline
    normal:       &\F\text\\
    slanted:      &\F\textsl{\text}\\
    italic:       &\F\textit{\text}\\
    small caps:   &\F\textsc{\text}\\
    bold normal:  &\F\textbf{\text}\\
    bold italic:  &\F\textbf{\textit{\text}}\\
    bold slanted: &\F\textbf{\textsl{\text}}\\
```

```
    bold small caps:&\F\textbf{\textsc{\text}}\\
  \end{tabular}\\[2ex]
}
\pagestyle{empty}

\begin{document}
\test{Oldstyle}{Roman}
\test{Oldstyle}{Sans}
\tbfigures
\test{Table}   {Roman}
\test{Table}   {Sans}
\end{document}
```

Here is its output:

Oldstyle Roman	
normal:	Hello Wörld! 0123456789
slanted:	*Hello Wörld! 0123456789*
italic:	*Hello Wörld! 0123456789*
small caps:	Hello Wörld! 0123456789
bold normal:	**Hello Wörld! 0123456789**
bold italic:	***Hello Wörld! 0123456789***
bold slanted:	***Hello Wörld! 0123456789***
bold small caps:	**Hello Wörld! 0123456789**

Oldstyle Sans	
normal:	Hello Wörld! 0123456789
slanted:	*Hello Wörld! 0123456789*
italic:	*Hello Wörld! 0123456789*
small caps:	Hello Wörld! 0123456789
bold normal:	**Hello Wörld! 0123456789**
bold italic:	***Hello Wörld! 0123456789***
bold slanted:	***Hello Wörld! 0123456789***
bold small caps:	**Hello Wörld! 0123456789**

Table Roman	
normal:	Hello Wörld! 0123456789
slanted:	*Hello Wörld! 0123456789*
italic:	*Hello Wörld! 0123456789*
small caps:	Hello Wörld! 0123456789
bold normal:	**Hello Wörld! 0123456789**
bold italic:	***Hello Wörld! 0123456789***
bold slanted:	***Hello Wörld! 0123456789***
bold small caps:	**Hello Wörld! 0123456789**

Table Sans	
normal:	Hello Wörld! 0123456789
slanted:	*Hello Wörld! 0123456789*
italic:	*Hello Wörld! 0123456789*
small caps:	Hello Wörld! 0123456789
bold normal:	**Hello Wörld! 0123456789**
bold italic:	***Hello Wörld! 0123456789***
bold slanted:	***Hello Wörld! 0123456789***
bold small caps:	**Hello Wörld! 0123456789**

The script

This section presents a listing of the Ruby script. The first 100 lines are comment lines in `rdoc` format. You can convert those into `html` by running:

```
rdoc osf
```

This creates a subdirectory `doc`. Point to the file `index.html` inside it in your browser and you will see a nicely formatted page, which will also contain separate sections for all methods defined in the script. Clicking in the headers of those section shows their sources in a popup window.

```
#!/usr/bin/ruby

=begin rdoc

=osf - convert digits in virtual font files to oldstyle

==Synopsis

osf [virtual_font_name [replacing_font_name]]

==Description

*osf* converts one or more virtual font (<tt>.vf</tt> and
<tt>.tfm</tt>) files, replacing the digits with old style
variants.

There are two ways to run the script: with one or two arguments
or with no arguments at all.

===Running with argument(s)

The first argument, if any, is the name of the virtual font file
to be converted. If a second argument is present, it is assumed
to be the name of the virtual font file containing old style
digits. If no second argument is present, the directory where the
first argument's virtual font file lives is searched for other
virtual font files containing old style digits and you are
presented a list of those from which you can make a choice.

The converted virtual font (<tt>.vf</tt>) files are stored in
your working directory, together with the corresponding
<tt>.tfm</tt> files. As a result, TeX documents compiled in that
directory using the converting fonts (for example by using
\usepackage{pxfonts}) will produce output with old style digits.

===Running without any arguments

If the script is run without arguments, a list is presented of
predefined virtual font sets from which you can make your choice.
Currently these are either the txfonts or the pxfonts.

In this case, the new font collection is renamed by prefixing
file names with <tt>osf-</tt> and a new style file is created,
together with the necesssary font definition (<tt>.fd</tt>)
files; these files, too, are named after the original files by
prefixing them with <tt>osf-</tt>.

The new style file has two options to switch to oldstyle or
table figures:

osfigures::  start with oldstyle figures (this is the default)
tbfigures::  start with table figures

The style file also creates two commands with the same goal:
\osfigures:: switch to oldstyle figures
\tbfigures:: switch to table figures

```

```
== Testing your font
Here is a LaTeX source that can be used to test your changes to
the txfonts and the pxfonts:

  \documentclass{article}
  \usepackage{osf-pxfonts}
  \parindent0pt
  \def\text{Hello World! 0123456789 äëïöéèê}
  \newcommand{\test}[2]{
     \def\Fam{Sans}\def\Arg{#2}
     \ifx\Arg\Fam\let\Fam\sf\else\let\Fam\relax\fi
     \begin{tabular}{@{}p{4em}ll@{}}
        #1 & normal:          & \Fam\text\\
        #2 & slanted:         & \Fam\textsl{\text}\\
           & italic:          & \Fam\textit{\text}\\
           & small caps:      & \Fam\textsc{\text}\\
           & bold normal:     & \Fam\textbf{\text}\\
           & bold italic:     & \Fam\textbf{\textit{\text}}\\
           & bold slanted:    & \Fam\textbf{\textsl{\text}}\\
           & bold small caps: & \Fam\textbf{\textsc{\text}}\\
     \end{tabular}\vfill
  }
  \pagestyle{empty}

  \begin{document}
  \test{Oldstyle}{Roman}
  \test{Oldstyle}{Sans}
  \tbfigures
  \test{Table}   {Roman}
  \test{Table}   {Sans}
  \end{document}

  $$ y = x^{123} \mathrm{text}$$

  \end{document}

==Version
$Id: osf,v 1.5 2004/05/06 20:35:20 wybo Exp $

==Author
Wybo Dekker (<tt>wybo@servalys.nl</tt>)

=end

require 'ftools'

class Hash
   # return the sum of squares of the values of a hash
   def sum_of_squares
      s = 0
      self.each { |x,y| s += y*y }
      return s
   end
end

# Check if a file has oldstyle digits
# Typically, table digits have equal heights and zero depth
# Oldstyle digits 0, 1, 2, 6, and 8 have zero depth,
# while 3, 4, 5, 7, and 9 have significant depths

def has_oldstyle_digits(file)
   indigit = false
   ht = dp = 0
   dif = Array.new
   open("|vftovp #{file}").readlines.each { |line|
      case line
      when /CHARACTER C (\d)$/   then indigit = $1.to_i
      when /CHARHT.* ([\d\.]+)/  then ht = $1.to_f
      when /CHARDP.* ([\d\.]+)/  then dp = $1.to_f
      when /MAP/ then
         if indigit
            dif[indigit] = ht-dp
            indigit = false
         end
      end
   }
   if dif.size > 0
      if (dif[3]+dif[4]+dif[5]+dif[7]+dif[9])/
         (dif[0]+dif[1]+dif[2]+dif[6]+dif[8]) > 0.9
         return false
      else
         return true
      end
   else
      return false
   end
end

# print a message, then exit with a fatal error

def die(message)
   puts 'fatal: ' + message
   exit 1
end

# find basename and directory of a virtual font file

def findfont(name)
   font = `kpsewhich #{name}.vf`.chomp
   die "Could not find #{name}.vf" if font == ''
   return File.basename(font,'.vf'), File.dirname(font)
end

# from a virtual font, isolate the digit sections and the mapfont
# section defining them. Renumber the mapfont to fontnr and
# change the SELECTION commands in the digits to point to it

def find_mapfonts_and_digits(font,fontnr)
   vpl = open("|vftovp #{font}.vf")
   mapfonts = Array.new           # one of these is returned
   digits = Array.new             # all returned
   sel = -1                       # font used for digits
   while line = vpl.gets
      case line
      when /MAPFONT D (\d+)/ then    # mapfont section?
         i = $1.to_i
         mapfonts[i] = line
         while l = vpl.gets
            mapfonts[i] += l
            break if l =~ /^   \)/
         end
      when /CHARACTER C (\d)$/ then  # digit?
         i = $1.to_i
         digits[i] = line
         while l = vpl.gets
            if l =~ /SELECTFONT D (\d)/
               sel = $1.to_i
               l.sub!(/\d+/,fontnr.to_s)
            end
            digits[i] += l
            break if l =~ /^   \)/
         end
      end
   end
   if sel == -1 then # no SELECTONT found?
      sel = 0        # use the default
      # insert SELECTFONT command in the digits
      digits.each { |d|
         d.sub!(/MAP$/,"MAP\n      (SELECTFONT D #{fontnr})")
      }
   end
   # renumber the mapfont
   mf = mapfonts[sel].sub(/MAPFONT D.*/,"MAPFONT D #{fontnr}")
   return mf,digits
end

# convert digits in virtual font file to old style
# collection:   name of the font collection (like `pxfonts')
# font:         font with table figures to be converted
# oldstylefont: font containing oldstyle figures
```

```
208
209 def convert_font(collection,font,oldstylefont)
210    # collection undefined: save in current dir
211    prefix = ''
212    vfdir = tfmdir = '.'
213    # if collection is defined, save files in user tree
214    if collection
215       prefix = 'osf-'
216       d = "#{$textree}/fonts/@/osf-#{collection}"
217       vfdir = d.sub(/@/,'vf')
218       tfmdir = d.sub(/@/,'tfm')
219       File.mkpath(vfdir) or
220          die "Could not create directory #{dir}"
221       File.mkpath(tfmdir)
222    end
223    # the new vpl file:
224    newvpl = open("/tmp/#{$$}.vpl","w")
225    maxfont = 0
226    # read the vpl file to be corrected:
227    vpl = open("|vftovp #{font}")
228    while line = vpl.gets
229       case line
230       when /MAPFONT D (\d+)/ then
231          # remember the maximum font ident
232          maxfont = [maxfont, $1.to_i].max
233          newvpl.print line
234       when /CHARACTER C \d/ then           # digit?
235          while l = vpl.gets                # skip to...
236             break if l =~ /^   \)/         # ... end of digit
237          end
238          next
239       else
240          # print everything else to the new vpl:
241          newvpl.print line
242       end
243    end
244    # append map font and digits from the old style virtual font
245    newvpl.print find_mapfonts_and_digits(oldstylefont,maxfont+1)
246
247    vpl.close
248    newvpl.close
249    name = prefix + font
250    system <<-EOF
251       vptovf /tmp/#{$$} \
252       #{vfdir}/#{name}  \
253       #{tfmdir}/#{name} >/dev/null
254    EOF
255 end
256
257 # define the style file in terms of the names of:
258 # arg 1: the font collection (pxfonts, txfonts, ...)
259 # arg 2: the roman font (pxr, txr, ...)
260 # arg 3: the sans font (pxss, txss, ...)
261
262 def style(collection,roman,sans)
263    return <<-EOF.gsub(/^      /,'')
264       \\RequirePackage{#{collection},t1enc}
265
266       \\def\\tb@rm{#{roman}}
267       \\def\\tb@sf{#{sans}}
268       \\def\\osf@rm{osf-#{roman}}
269       \\def\\osf@sf{osf-#{sans}}
270
271       \\newcommand{\\osfigures}{%
272         \\renewcommand{\\rmdefault}{\\osf@rm}%
273         \\renewcommand{\\sfdefault}{\\osf@sf}%
274         \\ifx\\f@family \\tb@rm \\rmfamily\\fi
275         \\ifx\\f@family \\tb@sf \\sffamily\\fi}
276
277       \\newcommand{\\tbfigures}{%
278         \\renewcommand{\\rmdefault}{\\tb@rm}%
279         \\renewcommand{\\sfdefault}{\\tb@sf}%
280         \\ifx\\f@family \\osf@rm \\rmfamily\\fi
281         \\ifx\\f@family \\osf@sf \\sffamily\\fi}
282
283       \\DeclareOption{osfigures}{
284          \\osfigures
285          \\let\\@Fam\\osfigures
286       }
287       \\DeclareOption{tbfigures}{
288          \\tbfigures
289          \\let\\@Fam\\tbfigures
290       }
291       \\ExecuteOptions{osfigures}
292       \\ProcessOptions
293       \\AtBeginDocument{\\@Fam}
294    EOF
295 end
296
297 # list the fonts for which the DATA section contains ready input
298 # this is used if no fonts are given on the command line
299
300 predef = %w{pxfonts txfonts}
301 $textree = "#{ENV['HOME']}/texmf"
302
303 if ARGV.size > 0   # one or two arguments, say pxr (and pcxr):
304    font,dir = findfont(ARGV[0])
305    # pxr, .../vf/public/pxfonts
306    has_oldstyle_digits("#{dir}/#{font}.vf") and
307       die "#{font} has oldstyle digits already"
308
309    if ARGV[1]
310       osfont,osdir = findfont(ARGV[1])
311       # pcxr, .../vf/public/pxfonts
312       has_oldstyle_digits("#{osdir}/#{osfont}.vf") or
313          die "#{osfont} has no oldstyle digits"
314    else
315       # no oldstyle font given: propose one
316       osdir = dir
317       osf = Array.new
318       Dir["#{dir}/*.vf"].each { |fontfile|
319          f = File.basename(fontfile,'.vf')
320          next if f == font
321          osf.push(f) if has_oldstyle_digits(fontfile)
322       }
323       # osf array contains all oldstyle fonts found
324       case osf.size
325        when 0 then puts "I found no accompanying fonts with " +
326                         " old style digits in #{dir}"
327          when 1 then osfont = osf[0]
328          else
329             # more than 1 found: find best matching name:
330             nearest = distance = 1000
331           puts "I found #{osf.size} accompanying fonts with " +
332                  "old style digits:"
333             for i in 1..osf.size do
334                f = osf[i-1]
335                h = Hash.new
336                for j in f.split('') do
337                   h[j] = (h[j] || 0) + 1
338                end
339                for j in font.split('') do
340                   h[j] = (h[j] || 0) - 1
341                   h.delete(j) if h[j] == 0
342                end
343                printf("%2d %2d %s\n",i,h.sum_of_squares,f)
344                if h.sum_of_squares < distance
345                   nearest = i
346                   distance = h.sum_of_squares
347                end
348             end
349             puts "My guess is #{nearest} (#{osf[nearest-1]})"
350             print "Your guess [#{nearest}]: "
351             i = STDIN.gets.chomp
352             i = i == '' ? nearest : i.to_i
353             osfont = osf[i-1]
354       end
355    end
356    convert_font(nil,font,osfont)
357 else
```

```
    # no arguments: use DATA section
    puts "#{predef.size} font conversions have been predefined"
    (1..predef.size).each { |i|
        puts "#{i} #{predef[i-1]}"
    }
    choice = 0
    until choice > 0 && choice <= predef.size
       print "Please make your choice [1]: "
       choice = STDIN.gets.to_i
       choice = 1 if choice == 0
    end
    collection = predef[choice-1]
    puts "Converting #{collection}"
    latexdir = "#{$textree}/tex/latex/osf-#{collection}"
    File.mkpath(latexdir) or
       die "Could not create directory #{latexdir}"
    DATA.each { |line|
       break if line.chomp == collection
    }
   roman_sans = Array.new # will names of roman and sans families
                          # used for the style file
    DATA.each { |line|
       line.chomp!
       case line
          when '' then break
          when /^fd\s/ then
             dummy,fd,*pat = line.split
             # fd commands: roman must come first, sans second
           # first pattern must be the name of roman/sans family
             roman_sans.push(pat[0])
             fd =~ /^t1/ or
                die ".fd filename must start with 't1'"
             infd = `kpsewhich #{fd}.fd`.chomp
             die "Could not find #{fd}.fd" if fd == ''
           out = open("#{latexdir}/#{fd.sub(/^t1/,'t1osf-')}.fd",
                      'w')
             open(infd).each { |l|
                # put prefix before all patterns:
                for p in pat do
                   l.gsub!(/(#{p})/,'osf-\1')
                end
                out.print(l)
             }
             out.close
          else
             puts line
             font,osfont = line.split
             # if the .vf exists from a previous run delete it,
             # and its .tfm companion:
             if FileTest.exist?("#{font}.vf")
                File.delete("#{font}.vf")
                File.delete("#{font}.tfm")
             end
             convert_font(collection,font,osfont)
          end
    }
    out = open("#{latexdir}/osf-#{collection}.sty",'w')
    out.print style(collection,*roman_sans)
    system('mktexlsr')
end

__END__
pxfonts
fd         t1pxr pxr p1x
fd         t1pxss pxss t1x
p1xb       pcxb
p1xbi      pcxbi
p1xbsc     pcxb
p1xbsl     pcxbsl
p1xi       pcxi
p1xr       pcxr
p1xsc      pcxr
p1xsl      pcxsl
pxb        pcxb
pxbi       pcxbi
pxbsc      pcxb
pxbsl      pcxbsl
pxi        pcxi
pxr        pcxr
pxsc       pcxr
pxsl       pcxsl

txfonts
fd         t1txr txr t1x
fd         t1txss txss t1x
t1xb       tcxb
t1xbi      tcxbi
t1xbsc     tcxb
t1xbsl     tcxbsl
t1xbss     tcxbss
t1xbsssc   tcxbss
t1xbsssl   tcxbsssl
t1xi       tcxi
t1xr       tcxr
t1xsc      tcxr
t1xsl      tcxsl
t1xss      tcxss
t1xsssc    tcxss
t1xsssl    tcxsssl
txb        tcxb
txbi       tcxbi
txbsc      tcxb
txbsl      tcxbsl
txbss      tcxbss
txbsssc    tcxbss
txbsssl    tcxbsssl
txi        tcxi
txr        tcxr
txsc       tcxr
txsl       tcxsl
txss       tcxss
txsssc     tcxss
txsssl     tcxsssl
tyxb       tcxb
tyxbi      tcxbi
tyxbsc     tcxb
tyxbsl     tcxbsl
tyxbss     tcxbss
tyxbsssc   tcxbss
tyxbsssl   tcxbsssl
tyxi       tcxi
tyxr       tcxr
tyxsc      tcxr
tyxsl      tcxsl
tyxss      tcxss
tyxsssc    tcxss
tyxsssl    tcxsssl
```

Notes

1. The *osf* script can be downloaded from `www.servalys.nl/tex/`
2. the T1 must stay in front, because LaTeX expects it there

Wybo Dekker
`wybo@servalys.nl`