Igor Freiberger

Roman Numerals with OpenType

This is a code to produce automatic Roman numerals from Arabic ones within an OpenType font. It generates the Roman version to any number from 1 to 9999. To do so, users must select the correspondent Stylistic Set in OpenType options (in the example below I used set #9, but any number from 1 to 20 is suitable).

To proceed all substitutions, the font needs 30 additional glyphs which represents Roman numerals:

Units	II III IV VI VII VIII IX
Sets of Ten	XX XXX XL LX LXX LXXX XC
Hundreds	CC CCC CD DC DCC DCCC CM
Thousands	MM MMM $\overline{\text{IV}}$ $\overline{\text{V}}$ $\overline{\text{VI}}$ $\overline{\text{VII}}$ $\overline{\text{VIII}}$ $\overline{\text{IX}}$
Special	An empty glyph with no width, to replace zeros.

It's not necessary to create glyphs to I, V, X, L, C, D and M, which must be already in the font. To an easy management, I named these additional glyphs as xy.rm, where x is the unit and y indicates ten, hundred or thousand. So the glyph for DCC, for example, is sevenhundred.rm.

It's controversial if a pair of glyphs which does not touch each other must be called a ligature. In view of this, I ignored the Adobe recommendation to nam-

ing ligatures (x_y model) for these Roman numerals.

You need an OpenType class which brings together all numeric figures in the font. In this example I consider the font already have classes to each number because this is a good OpenType procedure due to many figure sets possible in these fonts - lining and old style, proportional and regular, numerator and denominator, superscript and subscript.

If you use the FreeFontPro template (which comes with FontLab Studio and it's also available to download for free[01]), classes to digits zero-nine are already present and the first step is simply to define the class NUMBERS:

```
@NUMBERS=[@ONE @TWO @THREE @FOUR @FIVE @SIX @SEVEN @EIGHT
@NINE @ZERO];
```

If your font does not have all these classes, NUMBERS could be simply the sequence of glyph zero-nine:

```
@NUMBERS=[zero-nine];
```

Then, create a Stylistic Set with four lookups of substitutions in decreasing order: thousands, hundreds, sets of ten and units. The order is essential because the OpenType interpreter will proceed the substitutions this way.

```
feature ss09 {
lookup THOUSANDS {
sub one' @NUMBERS @NUMBERS @NUMBERS by M.sc;
sub two' @NUMBERS @NUMBERS @NUMBERS by twothousand.rm;
sub three' @NUMBERS @NUMBERS @NUMBERS by threethousand.rm;
sub four' @NUMBERS @NUMBERS @NUMBERS by fourthousand.rm;
sub five' @NUMBERS @NUMBERS @NUMBERS by fivethousand.rm;
sub six' @NUMBERS @NUMBERS @NUMBERS by sixthousand.rm;
sub seven' @NUMBERS @NUMBERS @NUMBERS by seventhousand.rm;
sub eight' @NUMBERS @NUMBERS @NUMBERS by eightthousand.rm;
sub nine' @NUMBERS @NUMBERS @NUMBERS by ninethousand.rm;
} THOUSANDS;
```

Note the one thousand number (M) uses the small caps M already in font. These substitutions look for sequences of four digits and change the first of then. The same principle is applied to the other lookups:

```
lookup HUNDREDS {
ignore substitute @NUMBERS' @NUMBERS @NUMBERS @NUMBERS;
```

[01] http://www.pyrus.com/downloads/freefontpro.zip

```
sub one' @NUMBERS @NUMBERS by C.sc;
sub two' @NUMBERS @NUMBERS by twohundred.rm;
sub three' @NUMBERS @NUMBERS by threehundred.rm;
sub four' @NUMBERS @NUMBERS by fourhundred.rm;
sub five' @NUMBERS @NUMBERS by D.sc;
sub six' @NUMBERS @NUMBERS by sixhundred.rm;
sub seven' @NUMBERS @NUMBERS by sevenhundred.rm;
sub eight' @NUMBERS @NUMBERS by eighthundred.rm;
sub nine' @NUMBERS @NUMBERS by ninehundred.rm;
} HUNDREDS;
```

In HUNDREDS lookup, the first line makes interpreter ignore the first digit in sequences of four. This is used to preserve the substitution made in THOUSANDS. An ignore statement is also needed is subsequent lookups:

```
lookup TENS {
ignore substitute @NUMBERS' @NUMBERS @NUMBERS;
sub one' @NUMBERS by X.sc;
sub two' @NUMBERS by twoten.rm;
sub three' @NUMBERS by threeten.rm;
sub four' @NUMBERS by fourten.rm;
sub five' @NUMBERS by L.sc;
sub six' @NUMBERS by sixten.rm;
sub seven' @NUMBERS by seventen.rm;
sub eight' @NUMBERS by eightten.rm;
sub nine' @NUMBERS by nineten.rm;
} TENS;
```

Finally, the units:

```
lookup UNITS {
ignore substitute @NUMBERS' @NUMBERS;
sub one' by I.sc;
sub two' by two.rm;
sub three' by three.rm;
sub four' by four.rm;
sub five' by V.sc;
sub six' by six.rm;
sub seven' by seven.rm;
sub eight' by eight.rm;
sub nine' by nine.rm;
} UNITS;
```

Substitutions made up to here ignore zeros. So, a number like 1005 would be shown as M00V. This can be avoided with specific substitutions for each multiple of ten - but the code will increase very much if you do this in a straightforward procedure. The solution I found is a little trick: to hide the zeros simply changing all of them to an empty, zero-width glyph:

```
lookup ZEROS {
sub zero' by zero.rm;
} ZEROS;
} ss09;
```

It's possible to add support for more numbers using the same structure. For example, to reach 99999 numbers, you have to add nine more substitutions and nine glyphs - although high numbers are very rarely set in Roman numerals.

Please be aware this feature does not consider any separator to thousands (dot and comma, according to language in use) as Roman numerals does not used these separators. But it's also possible to add the separator in THOUSANDS lookup to convert something like 1,234 to MCCIIIIV. I also ignored separator for decimals as Romans did handle just integers.

Feel free to use and improve this code. Comments and suggestions are welcome at *contato@if.pro.br*.

IGOR FREIBERGER
Porto Alegre, 2010