Arabic typesetting using a METAFONT-based dynamic font

Amine Anane
ananeamine@gmail.com

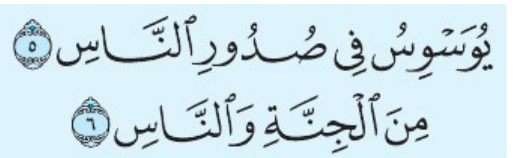
TUG 2019 - Palo Alto, California, USA
August 9-11, 2019

How the project started ?

I cannot find a searchable pdf document of the Quran.

PDF documents are raster images of hand-written book :

- cannot be searchable.
- big size.
- blurry image due to pixelation



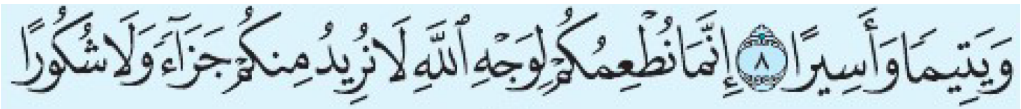
Other formats (Office Word, HTML [http://tanzil.net]) :

- based on Unicode text file + OpenType Font.

OpenType Font vs Handwritten text

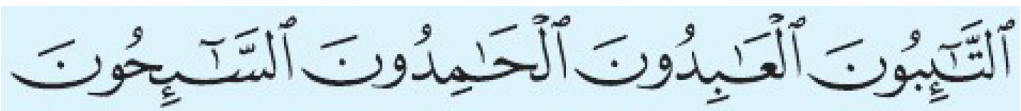
Much smaller font size to accommodate the widest line

وَيَتِيمًا وَأَسِيرًا ﴿٨﴾ إِنَّمَا نُطْعِمُكُمْ لِوَجْهِ ٱللَّهِ لَا نُرِيدُ مِنكُمْ جَزَآءً وَلَا شُكُورًا



Increase in interword space for other lines

ٱلتَّـٰٓئِبُونَ ٱلْعَـٰبِدُونَ ٱلْحَـٰمِدُونَ ٱلسَّـٰٓئِحُونَ



Objective

Typeset an electronic copy of the Medina Mus'haf (book) respecting calligraphic rules and having the same Font size, page number and line

Why Medina book ?

- Quranic book is a reference in Arabic calligraphy
- Written by one of the most renowned Arabic calligrapher
- beautiful, clear, easy-to-read style
- each page finishes by a verse and each section (30 sections) in 20 pages (15 lines each)
  - $\rightarrow$ Justification is applied extensively

If objective is reached, we should reach ultimate goal : High-quality Arabic typesetter fulfilling Arabic calligraphic rules

Where to start?

Existing systems and technology for typesetting Arabic text

- Font standards such as OpenType and Apple Advanced Typography
  - Advantage : Availability of tools
  - Disadvantage : justification does not conform to Arabic calligraphy such as horizontal stretching using Kashida (Tatweel)

- ditroff/ffortid system
  - Advantage : based on dynamic PostScript fonts
  - Disadvantage : does not model the way stretching is done in hand-written Arabic text
- Alqalam : An extension of ArabTEX
  - Advantage : based on METAFONT to design a dynamic font
  - Disadvantage : Very difficult to define the different context-sensitive shaping rules

Where to start? (Cont'd)

- Oriental project which leads to the development of LuaTEX : successor of Omega and pdfTEX
  - Advantage : Can be customized and extended using Lua and have native support for OpenType fonts,
  - Disadvantage : does not consider dynamic fonts

Selected Framework :

- LuaTEX and OpenType (i.e ConTEXt Lua modules)
  - OpenType engine will be extended to support dynamic font
- METAFONT to design the dynamic Font
  - High level structure to describe curve (direction, tension, curl, ...)
  - Linear equations
  - Macros to extend the language with new syntax and functions
  - Integrates well with LuaTEX (i.e mplib)

First attempt

- Have a beautiful font identical to the hand-written one : FontForge to trace the bitmaps to vector image and convert them to METAFONT
- LuaTEX internal OpenType tables to define substitution and positioning rules

Issues

- Takes a lot of time
- Difficult to debug
- Cannot take full advantage of METAFONT : Have to start from cubic curves with constant values

Solution

- Develop a graphical tool to easy the task : **Visual METAFONT**

Visual METAFONT

- Based on QT5 (C++)
- mplib : project supported by LuaTEX team in order to turn MetaPost into system library that can be used by many different applications.
- Harfbuzz : Text shaping engine supporting OpenType and AAT used in Android, Chrome, ChromeOS, Firefox, GNOME, GTK+, KDE, LibreOffice, OpenJDK, PlayStation, Qt, XeTeX, etc

Visual METAFONT - Main View

Visual METAFONT - Glyph View

Examples of expandable glyphs

- Medial meem : [-0.3,6] dots from the left, [-0.3,6] dots from the right

- Expandable final Kaf

- Expandable final feh

OpenType Rules

OpenType rule = list of sequences to match
If a sequence of the input matches a sequence of the rule
→ Apply **substitution** or **positioning** to the matched input sequence

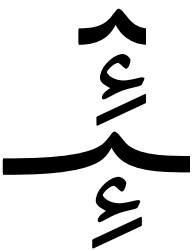
substitution : replace one or more glyph by one or more glyph : composition, decomposition, ligature, alternates

positioning : kerning, cursive attachment, mark over base, mark over mark

organized into **lookups** which are organized into **features**

OpenType positioning extension

- Positions are not fixed anymore : depend on parameter values applied during shaping
- Should be a function which calculates positions at runtime



OpenType feature file
Anchor function example

```
lookup hamzaabovebelow {

    feature mkmk;

    lookup hamzaabovebelow.l1 {
        pos base  /^behshape.medi/ <anchor function defaultbaseanchorforlow>
            markClass hamzaabove <anchor function defaullowmarkanchor> @hamzaabove ;
    } hamzaabovebelow.l1;

    lookup hamzaabovebelow.l2 {
        pos mark  hamzaabove <anchor function defaultbaseanchorforlow>
            markClass [kasra kasratan kasratanidgham] <anchor function defaullowmarkanchor> @markclass3 ;
    } hamzaabovebelow.l2;

    pos /^behshape.medi/ hamzaabove'lookup hamzaabovebelow.l1 [kasra kasratan kasratanidgham]'lookup hamzaabovebelow.l2;

} hamzaabovebelow;
```

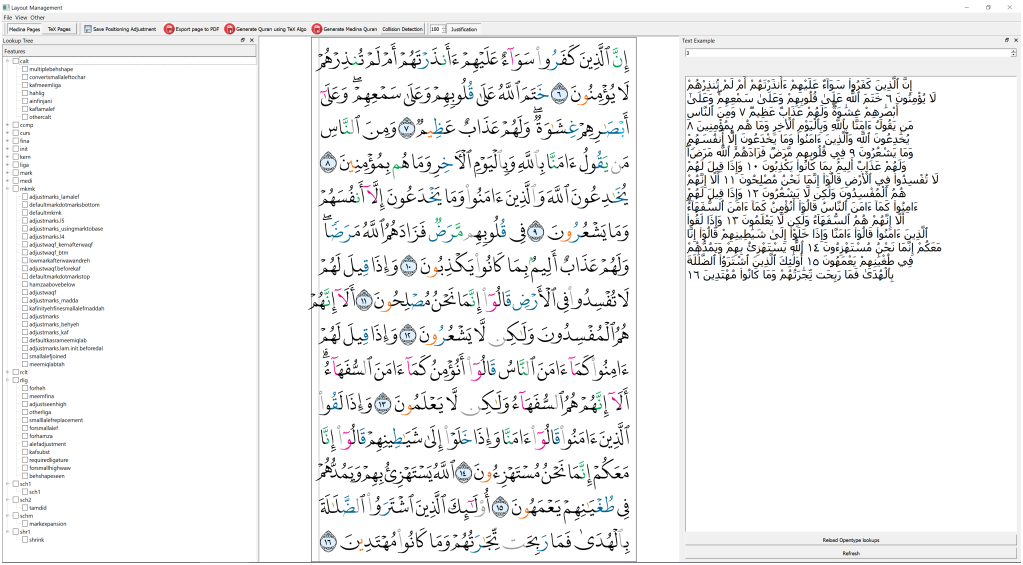
- Anchor functions which take as parameters class name, glyph name and applied expansion parameters

OpenType feature file
Other extensions example

```
pos [ fathatanidgham kasratanidgham dammatanidgham /^noon/]'lookup lgray
    ([/^alef.fina/ /^alef.isol/ /^yehshape/] | NULL)'
    space'
    (/^behshape.init/ twodotsdown | [ /^meem.init/ /^waw.isol/ ])' lookup green
    [@marks]'lookup green
    (@marks | NULL)'lookup green
;
```

- Regular expression : All initial behshape glyphs : /^behshape[.]init/
- | operator : 1 rule instead of 6 rules in the example above

Visual METAFONT - OpenType View



Justification example

- Space justification

ٱلتَّٰٓئِبُونَ ٱلۡعَٰبِدُونَ ٱلۡحَٰمِدُونَ ٱلسَّٰٓئِحُونَ

ٱلرَّٰكِعُونَ ٱلسَّٰجِدُونَ ٱلۡأٓمِرُونَ بِٱلۡمَعۡرُوفِ

- Justification using glyph expansion.

ٱلتَّٰٓئِبُونَ ٱلۡعَٰبِدُونَ ٱلۡحَٰمِدُونَ ٱلسَّٰٓئِحُونَ

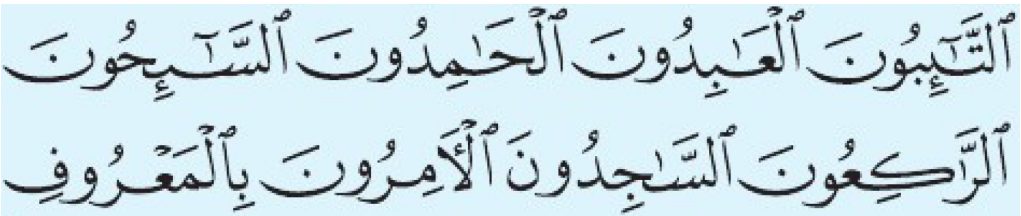
ٱلرَّٰكِعُونَ ٱلسَّٰجِدُونَ ٱلۡأٓمِرُونَ بِٱلۡمَعۡرُوفِ

All the final letters of each word are expanded by the same amount

Comparison with handwritten text

ٱلتَّٰٓئِبُونَ ٱلْعَٰبِدُونَ ٱلْحَٰمِدُونَ ٱلسَّٰٓئِحُونَ

ٱلرَّٰكِعُونَ ٱلسَّٰجِدُونَ ٱلْأٰمِرُونَ بِٱلْمَعْرُوفِ



- ligature decomposition (first line, third word) : ٱلْحَٰ → ٱلْحَٰ
- alternate glyph initial Kaf (second line, first word) : ك → ك

PDF generation - Font

Generate Type3 Font on the fly

Advantage : Can use arbitrary PDF graphics operators.

→ no limit in the use of METAFONT (as draw with pen)

Drawback : no hinting mechanism for improving output at low resolutions

Should not be anymore an issue since nowadays most devices have high resolution.

Type 3 character (verse 6) with color using other graphic objects :

PDF generation - Searching and Copy&Paste

ToUnicode which supported by most PDF Reader cannot be used :

- Cannot encode many glyphs to one as in $beh \rightarrow behshape + dot\_below$
- position order $\neq$ logical order $\rightarrow$ reversed glyph especially with diacritic marks

Use ActualText construct :

- Group base glyph with following diacritics into the same ActualText
- In special case (glyph over another) group two base glyphs together otherwise they will be reversed
- Supported only in Acrobat Reader (PC version)

Other features

- overlapping glyphs detection
- adjust mark/cursive/kern positioning graphically

Future Works

Short and medium term

- Justification Algorithm
  - Add other justification techniques such as ligature decomposition
  - During expansion, stretch the two adjusting letters
  - Add the necessary shrinking rules
- Automatic mark adjustment algorithm to eliminate overlapping : especially during line shrinking when glyphs are very tight
- Clean the code and publish it as open source
- Implement this work in LuaTEX

Other interesting works to consider in the long term

- Extend TEX breaking paragraph to take into account glyph stretching and shrinking
- Extend METAFONT to allow to use a rotating pen with changing breadth

Thank you