Experiences with Arabic font development

Sherif S. Mansour, Hossam A. H. Fahmy

Abstract

This is a report of our experiences attempting to use a new font, AlQalam, for the Arabic script within TeX. Then we want to make use of the new features introduced in LuaTeX to build our context analysis and line breaking engines to achieve a complete functional font package. We describe the challenges of producing high-quality Arabic fonts in general and what AlQalam has introduced to meet Arabic script requirements. We also describe the problems we faced trying to figure out how to use a new right-to-left font within TeX, what approaches we used to debug the font and some debugging results. This remains work in progress.

1 Arabic script and Naskh style

The Arabic alphabet is used to write many languages in many places around the world. Also, Arabic is of great importance to Muslims (about a quarter of the world's population), as it is the liturgical language of Islam.

The most distinctive features of the Arabic alphabet are that it includes 28 letters and is written from right to left in cursive style, i.e., many or all letters in a word are connected.

Arabic has six major writing styles: Kufi, Thuluth, Naskh, Riq'aa, Deewani, and Ta'liq. Naskh style is the most commonly used for printing, in both traditional texts such as the Muslim Holy Book (the Qur'an) as well as contemporary publications.

2 Challenges in producing high-quality Arabic fonts

The main challenge of producing high-quality Arabic fonts is that Arabic calligraphy is an art. The rules to follow when composing Arabic texts have great flexibility in choosing different variations of letter forms and constructing complex ligatures. These variations and ligatures add an aesthetic touch to the script and also justify the text as needed.

Every Arabic letter may have four basic forms depending on its location in the word: initial, medial, final, and isolated. Fig. 1 shows the different forms of the "Baa" letter as an example. Every letter form may have different variations to be used depending on the preceding or succeeding letter. Fig. 2 (taken from [8]) for example shows the different variants of the forms of "Baa". The first shape (starting from the right side of the first line) is the isolated "Baa" form variant. The second, third and sixth shapes

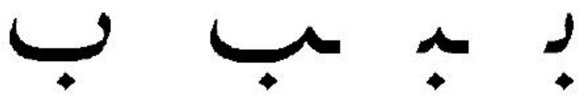
Figure 1: From right to left: initial, medial, final, and isolated forms of "Baa"

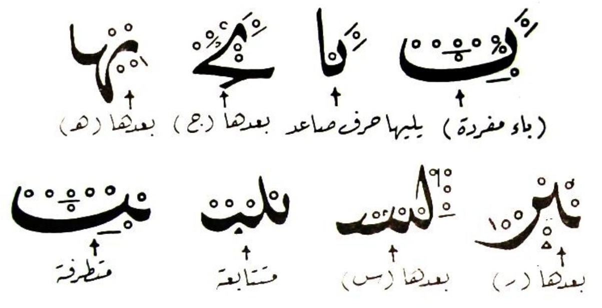

Figure 2: Letter "Baa" form variations

are initial form variants of "Baa" when rising letters like "Alef" precede it, letters like "Jeem" precede it and letters like "Seen" precede it, respectively. The fourth and fifth shapes are medial form variants of "Baa" when letters like "Haa" and letters like "Raa" precede it respectively. The seventh shape is the general medial form variant of "Baa" and the last shape is a final form variant of "Baa".

Fig. 3 shows a part of Mushaf Al-Madinah (The Holy Qur'an — Madinah print) [1] as a rich example of the usage of different combinations of variations with different elongations. This printing, like most printings of the Qur'an, was hand-written by a calligrapher and is not typeset. Computer typesetting (and to a large extent also traditional mechanical typesetting) are well behind what a calligrapher can produce for such complex texts.

The underlined shapes represent the same Arabic letter, which is "Kaf". Notice the different forms used depending on the location of the letter in the word and notice that some forms have different variations depending on the elongation requirements for line breaking, text justification and preceding or succeeding letters.

3 The font AlQalam and its features

Several trials were performed to produce high-quality Arabic fonts. One of them was a project named AlQalam ("The Pen" in Arabic), started in 2005 under the co-author's supervision [3]. AlQalam's target was to simulate an Arab calligrapher's pen for Naskh style (as used to typeset the Qur'an printing, for example). AlQalam then might be used in typesetting any traditional texts as well as any generic publi-

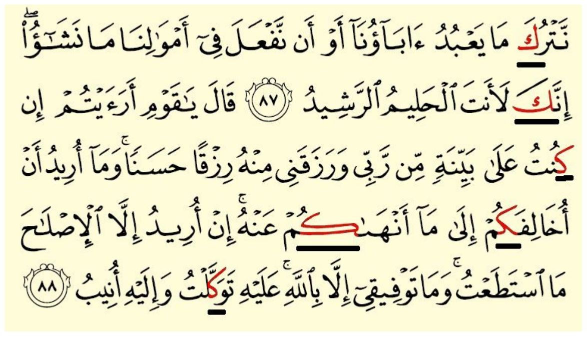

Figure 3: An example from surat Hud: forms of Kaf

cations (including scientific ones) in the languages using the Arabic script.

At first, AlQalam grew out of modifications to ArabTEX [4]. Modifications were mainly intended to respond to the specific needs of typesetting the Qur'an such as adding pause signs, some additional diacritics (marks used as phonetic guides) and the abilities to stack them on top of each other, scale them and correctly position them on the word. Also, some modifications to the pen used were made to improve the shape of some letters and symbols.

In 2008, a new font project for AlQalam was started [7]. That font is meta-designed such that each character is described by a number of parameters to allow the creation of many variants that connect with the surrounding characters correctly. Those variants may be different in shape and in their amount of elongation. Starting from this period many modifications were made and new features added.

AlQalam's font features up till now:

1. All font shapes are meta-designed using METAFONT to enable greater flexibility while joining glyphs together and provide smoother letter extensions.
2. It contains the generic four different forms of Arabic letters (initial, medial, final, and isolated).
3. It also contains different parameterized shapes for letter forms (the main source of the form variants is Mushaf Al-Madina).
4. It is based on the concept of primitives (reusable glyphs). For example, Fig. 4 shows the Waw head primitive (the small circle). This Waw head is reused in combination with the body (skeleton) of letter "Baa" to produce the letter "Faa". Also, the "Waw" head can be reused in combination with the body of the letter "Noon" to produce the letter "Qaf".
5. The font supports vertical placement of glyphs: Various ligatures have been added to the font to support complex vertical placement combinations as shown in Fig. 5.
6. Kerning: Borders of letter boxes have been adjusted to support kerning as shown in Fig. 6.
7. Joining glyphs with kashidas: the kashida is the most widely used glyph to join letters. AlQalam implements the kashida as a dynamic smooth glyph, as shown in Fig. 7 [2]. This is preferable to the current standard fonts that implement the kashida as a static fixed length glyph, as shown in Fig. 8.
8. Parameterized diacritics: A complete set of parameterized diacritics that can be elongated according to the width of the associated letter is available, as shown in Fig. 9.
9. Mathematical symbols: AlQalam is one of three fonts that have a complete set of Arabic math symbols at the time of writing. (The other

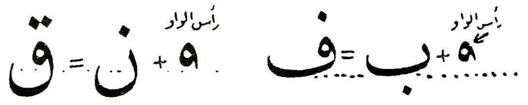

Figure 4: The "Waw" head primitive

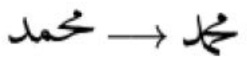

Figure 5: Vertical placement of glyphs

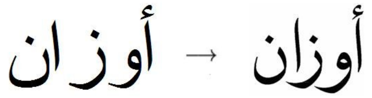

Figure 6: Kerning

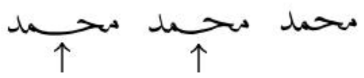

Figure 7: Joining glyphs with smooth, dynamic kashidas

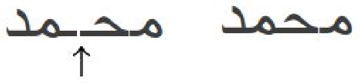

Figure 8: Static fixed length kashidas

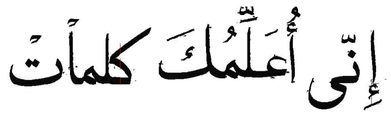

Figure 9: Parameterized diacritics

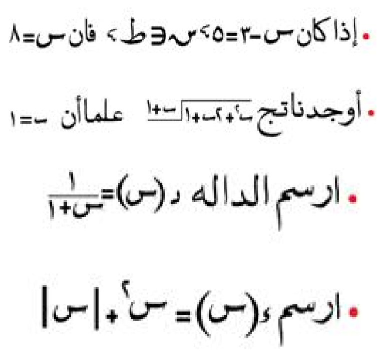
Figure 10: Set of Arabic mathematical equations typeset with AlQalam

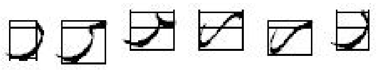
Figure 11: Character boxes approach

two fonts are RyDArab [5] and Mathfont [6].) Examples for Arabic equations generated using AlQalam are shown in Fig. 10.

4 Calling the different shapes of a letter

Testing of individual shapes was done along with the development of the font letter forms, form variants, mathematical symbols, etc. Once the basic development was finished, testing of the letters' positions against the baseline and against each other when included in the same word was needed. In addition, there was a need to perform checks for missing forms, to assure smooth joins between letters and correct kerning.

This drove us to call the METAFONT letters under TeX to debug the font. To do that, two approaches were possible:

1. Define the character's borders through boxes, similar to the Latin script, and define a character for each shape and with fixed elongation as shown in Fig. 11. This approach, by definition, must have a finite number of characters in the font at the end. Such a finite, i.e. limited, number of characters means that we must use only a limited set of values for the parameters of each character. So, while debugging we used for example only the "Noon" form shown in Fig. 12 despite the font's capability to generate elongated shapes of this form as shown in Fig. 13.

2. Using LuaTeX's embedded Lua and METAPOST engines to call the correct shape with the suitable elongation value. The main advantage of this approach is that we will be able to benefit from all the font features. But lacking experience in this approach made us think that we should postpone using it to the next phases, as more time will be needed to debug it.

Figure 12: Isolated form of the letter "Noon" without elongation

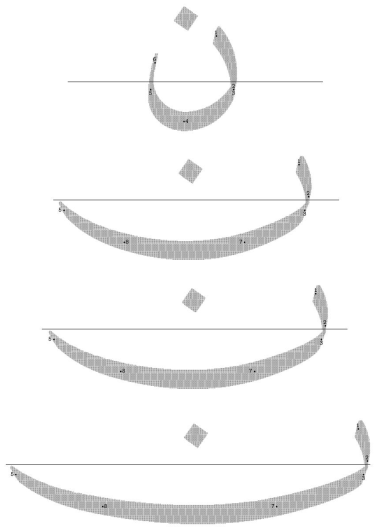
Figure 13: Isolated form of the letter "Noon" with different elongation values

Hence we started with the first approach to complete the current debugging phase. Meanwhile we will learn about the possibilities of calling METAPOST from within LuaTeX in order to generate dynamic fonts in the future.

5 Problems found during font debugging

As expected, some bugs appeared when using the font under TeX. If we take a look at Fig. 14 we can easily notice the bugs, from right to left as follows. The first and fifth cross signs mark a kerning bug with the letter "Waw" as it appears too far from the succeeding letter. The second and fourth signs mark bad joins of letter "Ain". Also the letter shape itself needs some modifications as marked by the third sign. More bad joins appear between letters "Qaf" and "Sad" and letters "Lam" and "Dal" as marked by the sixth and ninth signs. The seventh sign marks a missing letter that should be added to the font package. The eighth sign marks a mistake in specifying the border of letter "Alef" that caused it to be too close to the succeeding letter.

We have worked on fixing these bugs and many

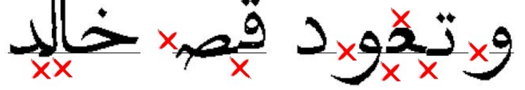

Figure 14: Font state at an early debugging phase

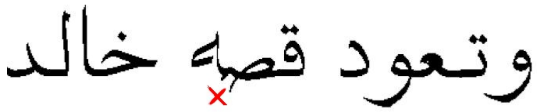

Figure 15: Font state after fixing bugs

others that appeared in turn, to get to a state with the font that made us ready to move to the next step (despite having very few joining bugs as shown in Fig. 15). This is implementing the multi-layer context analysis algorithm to make using the font more user friendly, by automatically choosing the suitable letter forms and form variants instead of the user having to do so manually.

6 Work in progress and future work

First, adding context analysis via LuaTEX's embedded Lua and METAPOST engines: This will enable calling the suitable METAFONT form with the suitable elongation parameters according to context. Multi-layer context analysis is expected, as a basic layer would be needed to choose the suitable form of the letter according to its preceding or succeeding letter, and a second layer would construct ligatures, correctly update the positions of the diacritics and associated symbols and also control the elongation parameters according to the text justification requirements. We are currently working on implementing the first layer.

Second, implementing a new line breaking algorithm with a new penalty system that supports usage of elongation, ligatures, form variants in addition to spacings when taking the breaking decisions [2]. A dialogue between METAPOST and TEX about word widths and line width requirements is expected. This will be performed through the usage of LuaTEX's callbacks feature to override the main algorithm.

References

[1] *The Holy Qur'an.* King Fahd Complex for Printing the Holy Qur'an, Madinah, KSA, 1986.

[2] Mohamed Jamal Eddine Benatia, Mohamed Elyaakoubi, and Azzeddine Lazrek. Arabic text justification. *TUGboat*, 27(2):137–146, January 2007.

[3] Hossam A. H. Fahmy. AlQalam for typesetting traditional Arabic texts. *TUGboat*, 27(2):159–166, January 2007.

[4] Klaus Lagally. ArabTEX — Typesetting Arabic with vowels and ligatures. In Jiří Zlatuška, editor, *EuroTEX '92: Proceedings of the 7th European TEX Conference, Prague, Czechoslovakia, September 14–18, 1992*, Proceedings of the European TEX Conference, pages 153–172, Brno, Czechoslovakia, September 1992. Masarykova Universita.

[5] Azzeddine Lazrek. RyDArab — Typesetting Arabic mathematical expressions. *TUGboat*, 25(2):141–149, 2004.

[6] Mathfont. `http://mhafiz.deyaa.org/mathfont.html`.

[7] Ameer M. Sherif and Hossam A. H. Fahmy. Meta-designing parameterized Arabic fonts for AlQalam. *TUGboat*, 29(3):435–443, January 2008.

[8] Ahmad Sabry Zayed. *Ahdath Al-turuq Leta'leem Al-khotot Al-'arabiya [New methods for learning Arabic calligraphy].* Maktabat ibn-Sina, Cairo, Egypt, 1990.

◇ Sherif S. Mansour, Hossam A. H. Fahmy
Electronics and Communications Dept.
Faculty of Engineering, Cairo University
Egypt
`sherif.s.mansour (at) gmail dot com`
`hfahmy (at) alumni dot stanford dot edu`