TEX and Those Other Languages

Yannis Haralambous
101/11, rue Breughel, 59650 Villeneuve d'Ascq, France
33 20052880; FAX: 33 20910564
Bitnet: yannis@frcitl81

Abstract

This paper relates the author's experiences while creating a TEX package for typesetting in several languages of scholarly interest, such as Arabic, Hebrew, Syriac, Greek, Armenian, and Saxon. First, a combined use of METAFONT and a PostScript font creation program is described and commented; next, the TEXnical problems (and their solutions) with relation to each language are presented; finally, some new ideas for further development and application of TEX in non-Latin alphabet transmission through the electronic communication media are given.

Introduction

TEX's box-oriented approach to typesetting makes it the ideal tool for "exotic" languages which need two-dimensional constructions. The lack of WYSIWYG is compensated by the infinite possibilities of a programming language and the compatibility between different devices and electronic communication medias.

But let's start from the beginning. After developing the Arabic–Persian–Ottoman TEX system presented in the summer of 1990 ([2]), I was so delighted by TEX and METAFONT's possibilities in this area that I decided to continue making more alphabets for scholars. I found that this domain was underdeveloped and that many scholars were still adding non-Latin alphabet text by hand or using primitive low-resolution bitmap fonts which they had to create themselves.

For the first round (programmers call it *version*), I attacked Hebrew, Syriac, Armenian, Greek, epigraphical Greek, and Saxon. It was a beautiful experience (which took all my holidays, week-ends, free hours, and many entire nights). The result is *Scholar*TEX, a package of fonts, preprocessors, macros, documentation, and everything a *non-TEX-expert* scholar could need for his or her typesetting activities.

Since many public domain packages can be extremely interesting to scholars but sometimes difficult to find, or in need of some adaptation, I thought that *Scholar*TEX should be a platform for distribution of related important public domain software (with explicit notification of its status and origin). In this way I was kindly allowed to include EDMAC ([8]), a version of TEX–XET featuring sbTEX (for PCs), and the wsuipa IPA fonts ([1]).

A short presentation of *Scholar*TEX was made at the DANTE meeting, January 1991, in Vienna; a more general one (with an expanded part) will be made at the 6th European TEX Conference, September 1991, in Paris. In this paper I would like to describe some techniques and experiences in making the fonts and present some TEXnical problems I encountered, with their solutions.

How to Make a Font

The aim of this section is to show how METAFONT and a PostScript font creation program (in this case, Letraset Fontstudio v2.0) can be combined in a complementary way to produce an aesthetically satisfying font that would be very cumbersome to produce with either alone.

I will subdivide this method of creating a font for some "exotic" language into eight steps, each technically and emotionally different. Please note that by no means do I pretend this to be the best solution. It is more a kind of *poor man's* method, which can be used at home with the least possible equipment. Much better results could be achieved by high technology, and in far shorter time; but my method is more fun!

First step: choosing the types. This is the "outside world" part of the job. It involves looking in libraries, finding highly specialized grammars and dissertations, and trying to extract the scarce information you need about the letters, punctuation marks, symbols, etc. Sometimes you will feel like

Umberto Eco, finding out many little-known things about the past and collecting interesting information or theory[1] on the history of languages, alphabets, and typography.

But the focus of your search is to find some reliable, preferably large samples of the types you want to reproduce. You have to study them well, to see all the small typographical details, and try to find out which of these belong to tradition and which are just the result of the typographer's relative ignorance of the specific "exotic" language. Once you have found enough sources and discussed details with some specialists, you have to choose one principal type which you will "copy" and two or three others which will give you ideas for modifications.

Second step: taking photos. Now you become a photographer. You have to take pictures of small objects called letters which live on rough and not always flat paper; these pictures should be very clear and identically scaled.

The paper problem can be solved if you press a glass plate over your paper (which is not always easy, as in the case of large, old books). To have clear pictures, use very strong light sources; then you can focus more easily and use smaller lens apertures with more depth of field. As for the scaling problem, you can insert some reference object in each picture. I did all of my pictures with an old and faithful Olympus OM-2, a bellows, and an inverted f:3.5/24mm lens (only recently I bought the special f:2/20mm macro lens).

Don't hesitate to make pictures even of punctuation marks, dashes, and surrounding Latin characters — everything is important.

Third step: first paper draft. Put the developed film in the darkroom projector and copy the outlines of your characters with a black pencil on white letter format paper. At the same time you can also trace the "invisible" extensions of your character's strokes. You can see an example in Fig. 1; it's the well-known Hebrew letter aleph.[2] Since printed characters are not always as smooth as you would expect under high magnification, you may have to make some corrections while copying. I usually make these first corrections in some other colour.

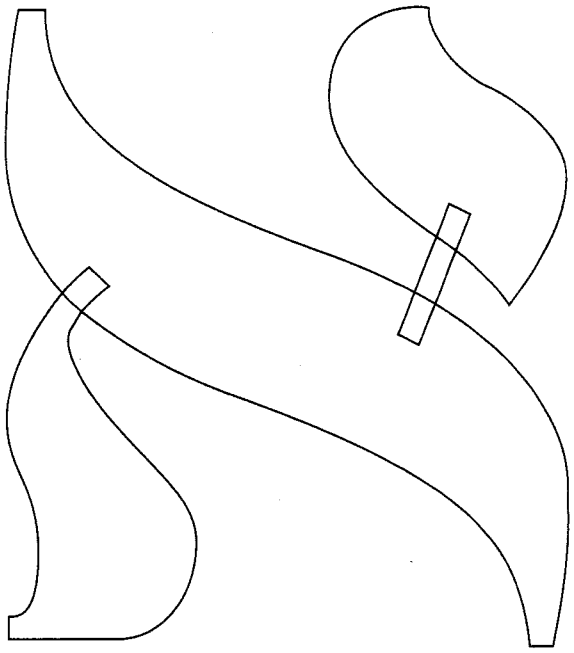


Figure 1: A first paper draft.

Fourth step: setting guidepoints and pen positions. This is the most important and most artistic part of the job. Now you are a designer. You have pens of every possible shape and can make pen strokes with them; you can also fill outlines; the only restriction is that all curves must be Bézier curves. You are entirely free, *but* there are two fundamental rules:

1. use as few Bézier curves as possible; and
2. don't forget METAness.

The first rule is to remember that your curves are so beautiful, not — or *not only* — because you are a great designer, but because they are Bézier curves, obeying very strict mathematical rules (see the METAFONTbook, p. 131, or Yanai and Berry's paper [13]; and if you haven't done it yet, I sincerely advise you to read "Mathematical Typography" [6]). You must be extremely careful when joining such curves — the result may disappoint you.

The second rule is more METAFONT-specific: all pen positions or guidepoints must keep track of the character's ability to transform according to the parameter values you are going to impose on it. For example, in Fig. 2 — where numbered line segments denote penpositions and the arrows indicate orientation — you see that pen positions 3 and 4 (left stem)

[1] For example, did you knew that the Greek letter alpha A, the Arabic alif ا, and the Hebrew aleph א are derived from the same Phœnician letter ∀ called aleph which means "calf" because it looks like a calf's head? (See [5].)

[2] I don't guarantee this to be the exact shape of the aleph א in font `yhbr`; many changes occurred later.

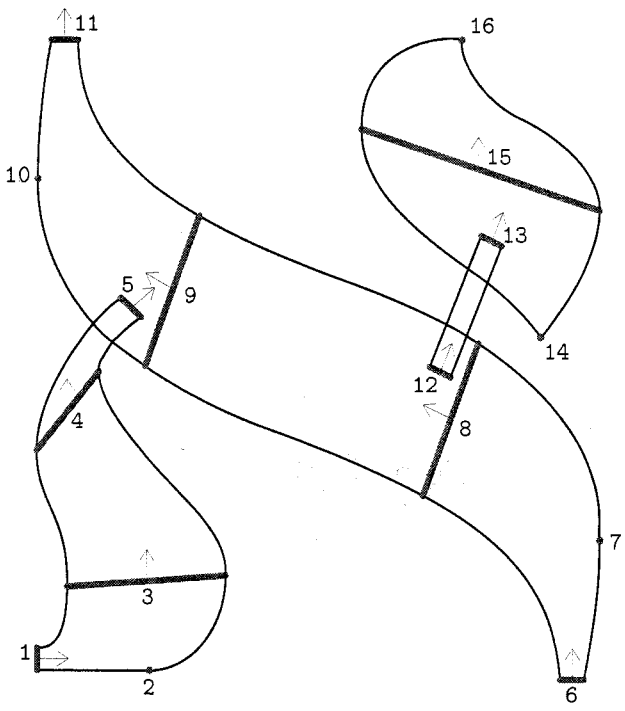


Figure 2: Guidepoints and penpositions.

have been chosen so that the tangents at their left edges are vertical, but the pen angles are different. So if you increase the pen widths, the upper part of the stem will change to a lesser extent than the lower part. It is a good habit to put pen positions and guidepoints at places where tangents to curves are vertical or horizontal. But sometimes this is not enough, as in the case of pen positions 8 and 9 (diagonal strokes).

This step requires the most attention and, at the same time, the most intuitive handling (*if I were a pen stroke leaving here in this direction, where would I go?*). I usually use a second colour for pen positions and guidepoints.

Fifth step: writing and running the METAFONT source. Once you have the outline, you become a METAFONTer. The goal is not only to write all this in METAFONT language, but also to define the necessary parameters to make it possible for your picture to provide other styles (at least **boldface**, since *slanted* is easy, while `typewriter` and sans serif often need a completely different code). You should start with the minimum number of parameters (often `hair` and `stem` are enough for lowercase). If there is a need for other, more special parameters, you can always add them later.[3] If the fourth step has been well done, this one should be straightforward.

Try not to use the `...tension xxx...` command too often. Every time I tried to use it there was a more natural way which gave a far better result. (Bézier curves are like humans; their best state is the natural one—too much tension spoils them.)

Some points will have to be defined by coordinates. Be careful when defining pen positions; the pen width may change later on, and you must take this into consideration now. In our example, for pen position 3, if you fix the coordinates of point `z3` then by increasing the width, the path `z3l{up}..z4l{up}` would become more and more flat. In this case, you should fix the coordinates of `z3l`. In the same way, pen position `15` was defined by fixing the coordinates of `z15r`, etc.

While you are running METAFONT and visualizing your character on screen, you will already discover many weak points in your draft; you can then go back to step 4 and make the necessary changes. On the other hand, I am always amazed to see how easily one can obtain *exactly* the same curve as in a good sample, which means that the old masters of the past may have used Bézier curves, without knowing it.

Sixth step: going to Fontstudio. By using the following `mode_def` (which changes the definition of `endchar`, and instead of a grid, inserts cropmarks), enter:

```
mode_def fontstudio =
def nothing(text r) = enddef;
pixels_per_inch :=2200; blacker :=0;
o_correction :=.4; fillin :=0;
proofing :=1; fontmaking :=0;
tracingtitles:=1; mag:=2.4;
screen_rows:=1200; screen_cols:=2000;
let makebox=nothing;
enddef;

def endchar = scantokens extra_endchar;
if (proofing>0) and not (mode=fontstudio):
makebox(proofrule); fi
if (mode=fontstudio):
pickup pencircle scaled 1;      %really 1 pixel!
draw (l-10,0)--(l+10,0);draw (r-10,0)--(r+10,0);
draw (l,-10)--(l,10);   draw (l,h-10)--(l,h+10);
draw (r,-10)--(r,10);   draw (r,h-10)--(r,h+10);
draw (l-10,h)--(l+10,h);draw (r-10,h)--(r+10,h);
```

[3] For example, in Arabic, besides `hair` and `stem`, a parameter was needed for the width of the base-stroke, which can be completely independent from the widths of other strokes.

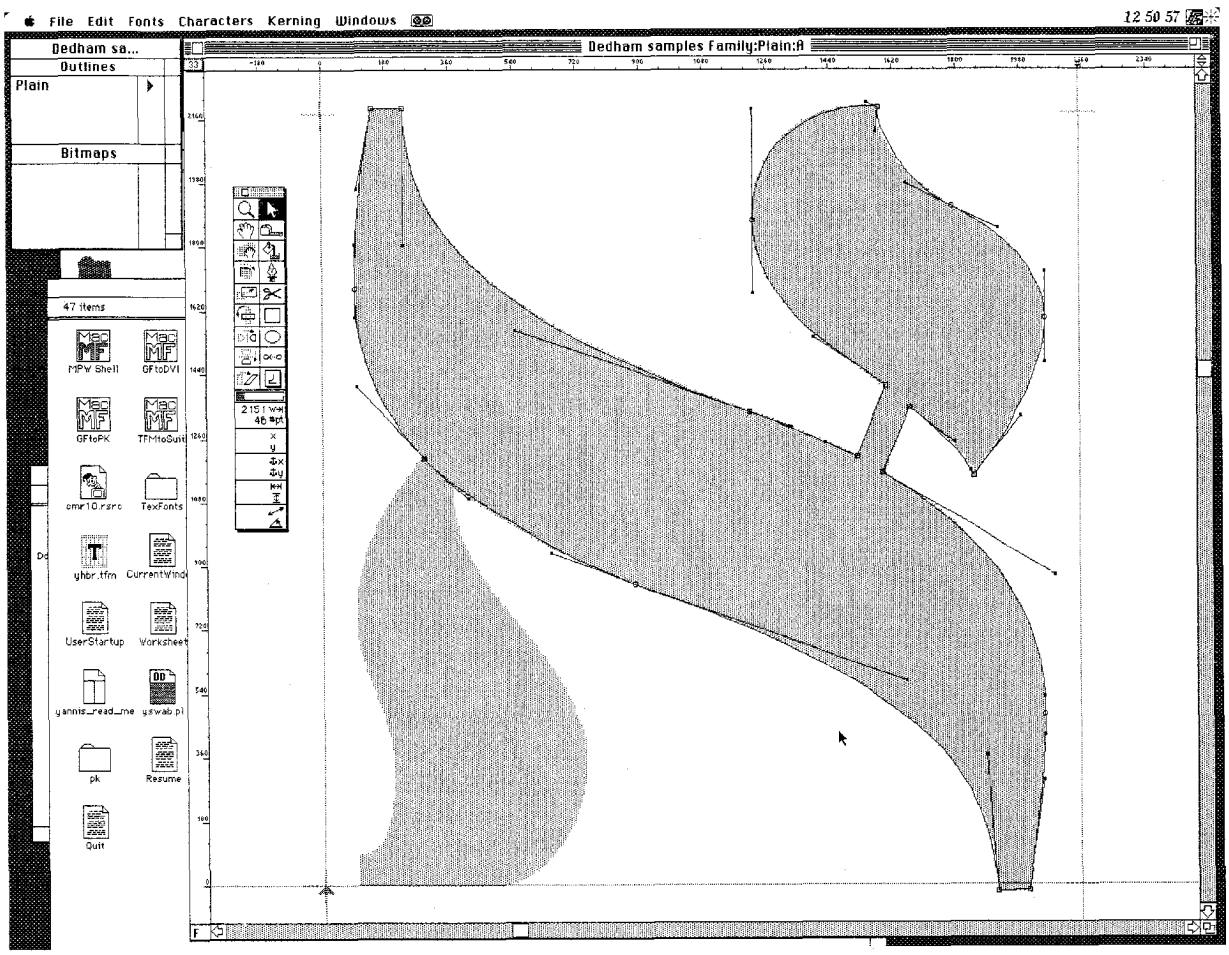


Figure 3: Outline drawing in Fontstudio.

```
fi
chardx:=w; shipit;
if displaying>0: makebox(screenrule); showit; fi
endgroup enddef;
```

I make screen copies of all characters (at the same scale, approximately 10 to 12 inches high for uppercase characters). These screencopies are visualized in Fontstudio's workbench as grey masks. First, you have to place the mask in the right position by identifying the left lower cropmark as point (0,0) and setting the right cropmark to be the character's width (finer corrections can be done later). Then, using Illustrator-like techniques you copy the outline of the character, as in Fig. 3. Don't use the "Autotrace" feature! The point is that by drawing the character yourself, you can follow the guidepoint scheme you have established in step 4. You might argue that this way of doing it is unprofessional and that there are METAFONT→PostScript translation programs (for example, see Yanai and Berry [13]). This may be the viewpoint of a programmer, but not that of a designer. The best work is still done by hand. Seeing the character in front of you, and at this size, enables you to learn it better. Playing with Bézier curves on the screen will give you ideas, and you may even go back to step 5 and change your METAFONT source (this happened to me several times).

Both programs contribute in a complementary way: METAFONT keeps the uniformity (all thin lines will be exactly of `hair` width, all stems will be of `stem` width, and so on), while Fontstudio brings new ideas and a deeper understanding of the character's shape because of its WYSIWYG features. This is true for the new fonts you are just creating. If you just want to make outlines from existing fonts, you can achieve an acceptable degree of precision by this method, but a translation program will do it more quickly.

Figure 4: Kerning in Fontstudio.

Seventh step: kerning and going back to METAFONT. A big advantage of Fontstudio is visual kerning. As you see in Fig. 4, you can display any text on your screen, select a character, and move it to the correct position. All kerning pairs are written in alphabetical order, and you just have to copy their list to make a METAFONT `ligtable`. If you see a character occurring in pairs with more than half of the others, you should change its margins; in Fontstudio this is done visually and in METAFONT just by changing the value of `adjust_fit`.

Eighth step: making `tfm` files. Fontstudio delivers an AFM metric file which you can easily convert to a PL one. I advise you to compare this PL file with the one originally created by METAFONT; this will prevent many errors. Of course, the PL file coming from Fontstudio is useless for TEX; take, for example, characters such as the Computer Modern "large math operators" from file `bigop.mf` (see Knuth 1986, p. 103–121]); they all have considerable depth, but their boxes are of depth zero, and this is not accepted by Fontstudio. You will of course use the `tfm` and PL files created by METAFONT. Some systems (such as *Textures*) may need small changes in the PL file; for example:

```
(FAMILY YARBA)
(CODINGSCHEME PostScript YarbaNaskhi)
```

where `YARBA` is the font name for QuickDraw and `YarbaNaskhi` for the printer. Other systems, e.g., OzTEX, keep a list of all PostScript font names in a special file (for OzTEX version 1.3, any file in the `Configs` folder).

We have seen how the combined use of METAFONT and Fontstudio allows easy and efficient font creation and solves many problems such as kerning or better understanding of Bézier curves in the case of METAFONT, and drawing pen strokes and keeping homogeneity of all characters in the case

of Fontstudio. I would conclude by calling METAFONT's approach a rationalistic one, while Fontstudio's approach is empirical; and you know that both are important.

Problems with Languages

In the previous section we saw how to make a font for an exotic alphabet. I specify that this approach is valid only if one or two styles are required (plain and **boldface**, for example). If you are planning to use a complete library of styles (such as CM), you will have to spend infinitely more time in organizing and checking METAness parameters.

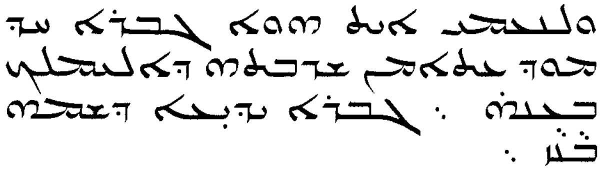
Now we have to use these fonts. Several problems arise, and I propose to examine each alphabet separately.

Arabic

صحيح اننا نستفيذ من هذه الوضعية بتفتحنا
على العالم الخارجي ولكن الاستعمال
المكثف للغة الفرنسية ينتج عنه حيف في
المجتمع المغربي لايسمح بتنمية متوازنة ☼

Most of the details about the Arabic alphabet have already been set forth in Haralambous, 1990. Since the first YARB version, I have added new characters to cover also Pashto, old and modern Urdu and Malay; I have also entirely changed the preprocessor (now called ysemtex): the characters are taken from 3 *real* fonts and about 14 *virtual* ones (the virtual fonts are used for the precise placement of diacritical marks). The input encoding, as well as all escape characters, are now user-definable; this data is stored in a text file which is loaded while running. You have the choice between plain TeX and TeX-XeT output, for the same input.[4]

Syriac



From the typographical point of view, Syriac is structured like Arabic, so I just had to define a new escape character and tell ysemtex which input and output data to load.

The problem with Syriac is the lack of typographical evolution in the last few centuries. There are at least two kinds of Syriac alphabet: *Estrangello* and *Serto*. I began with Estrangello (Serto will follow). The Estrangello type I encountered in most books is just an imitation of handwriting. I tried to make some aesthetic improvements, which I had to withdraw *immediately* when I showed the font to specialists.

Hebrew

שִׁמְעוּ הַדָּבָר הַזֶּה פָּרוֹת הַבָּשָׁן אֲשֶׁר
בְּהַר שֹׁמְרוֹן הָעֹשְׁקוֹת בַּלִּים הָרֹצְצוֹת
אֶבְיוֹנִם הָאֹמְרוֹת לַאֲדֹנֵיהֶם הָבִיאָה
וְנִשְׁתֶּה׃

Since it was strictly forbidden to change the Holy Texts and the Jewish people saw the oral tradition disappearing (because of changes in the pronunciation of the language), they decided to add diacritical marks to the Text, starting with vowels and going to more and more specialized and rare symbols. Today, one can find up to four of these symbols which, for the sake of briefness, we will call *accents*, on each letter (plus eventually the *dagesh* point inside the character). TeX can handle this situation very well by using box constructions. The output provided by ysemtex contains the information on accents in the following way: for each letter which contains at least one accent, a macro

$$"n_0o_0\text{-}n_1o_1\text{-}n_2o_2\text{-}n_3o_3\text{-}n_4o_4!$$

is used, where o_0 is the octal code of the character which is in font n_0, and o_1 to o_4 the octal codes of the four possible accents (in fonts n_1 to n_4), starting from upper left and finishing with lower right. Of course, some combinations of accents and characters deserve special accent positioning and are contained as separate characters in the font (e.g. ךּ, ךְ, ןְ, הּ, לׂ, etc.).

Contrary to Syriac, Hebrew has a very rich typographic tradition (Tamari, 1989). I chose a rather simple type which better suits a 10-point text than big titles (a real calligraphic type for head titles is planned). The accents were taken from the TABULA ACCENTUUM of the BIBLIA HEBRAICA STUTTGARTENSIA.

[4] Even if you use TeX-XeT, the preprocessor is unavoidable because of the Arabic character forms.

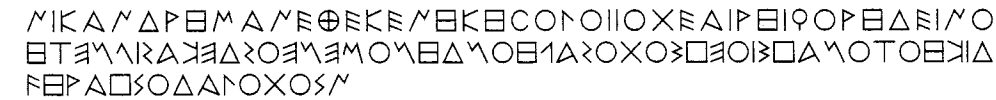


Figure 5: Sample of Greek epigraphical text.

Greek

> Ὁ Βασιλεὺς Δουριοδάνας, ἰδὼν παρατεταγμένον τὸν στρατὸν τῶν Πανδοϊδῶν προσελθὼν τῷ διδασκάλῳ Δρόνα ἔλεξε τάδε· «Σκόπει διδάσκαλε τόν μέγα τοῦτον στρατὸν τῶν υἱῶν τοῦ Πανδοῦ, τὸν παρατεταγμένον ὑπὸ τοῦ εἰδήμονος μαθητοῦ σου Δρυσταδεούμνα, υἱοῦ τοῦ Δρουπάδα».

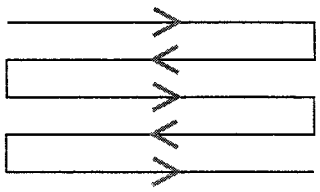
Some Greek fonts have already been designed by Silvio Levy ([9]). For several reasons — one is that I found a really beautiful type in a Greek book — I decided to make some new fonts from scratch, except of course for uppercase letters, which remain those of the CM family. The circumflex accent is encoded as = and *all* accented characters are included as ligatures; thus, to start typesetting in Greek, you just have to select the Greek font.

I included the symbols |, ‖, ⟦, ⟧, {, }, ⟨, ανδ ⟩ , used in epigraphical texts by Oxford editions (see [11]). All possible ⟨*character* + *accent*⟩ combinations are separate characters in the font and can be reached either by 8-bit input, or by 7-bit ligatures. You will find a discussion on encoding and transliteration problems concerning both ancient and modern Greek in [4].

Epigraphical Greek. See Fig. 5 for a sample of the fonts. Designing this font was straightforward; it is meant to be simple and most of the lines are straight. The problems which arose were more of a TEXnical nature:

1. there are no blank spaces between words;
2. lines are imposed; and
3. all lines (except the last) should be of equal length.

And adding the fact that most of the inscriptions are written βουστροφηδὸν



(which means that text direction *as well as character shapes* alternates at each line), this is enough to cause a typographer's headache.

Another — more TEX-related — problem is encoding: there are 14 different alphas, 10 different betas, etc. The user would like a readable text and not a sequence of `\char'xxx\char'yyy\char'zzz`. Even a sequence such as `\A12\B07\G03\A04` would not be very readable.

Here are the solutions I propose:

1. The uppercase ASCII characters `A..Z` are locally set to be active. They receive definitions of the form `\def␣A{\char'xxx\␣}` where *xxx* is set by the user as the octal font position of the required alpha, depending on the epoque and idiom in which the inscription is written (one can always use ordinary macros for exceptions, as long as their names do not contain uppercase characters).
2. You may be wondering what the `\␣` stands for. Well, the second idea is to set the blank space of length 0, expandable to `10u#`:

```
font_normal_space 0;
font_normal_stretch 10u#;
font_normal_shrink 0;
```

   (where, at 10pt size, `u#` is as usual `20/36pt#`).

   You proceed in the following way: choose the potentially longest line (I should write an algorithm to make that automatic) and write it first inside a macro

```
\longestline{...}
```

   Then the contents of all lines will be placed in centered boxes with this length, and by the expanding feature of ␣, letters will be equally spread inside each box.[5]
3. The βουστροφηδὸν problem is solved by having a second font which is the mirror image of the first. You can choose between writing your text from left to right, or from right to left (here the well-known `\reflect` macro is applied).

[5] Actually, the definitions of the active uppercase characters are slightly more complicated because of the last character of the line, which should not be followed by a `\␣`.

Armenian

ԳԱՅԼԸ ՊԱՅՏԱՐ

Գայլը երբ մի օր շրջում էր լեռներում, արօտների մէջ կապած մի էշ տեսաւ։ Էշը հասկացաւ, որ իր վերջն եկել է, ուստի դիմեց գայլին և ասաց. «Գոհութի՛ւն Աստուծոյ, որ քեզ ինձ մօտ բերաւ, ո՛վ գայլ։ Ուրախ եմ, որ ուտես ինձ և ազատես այս սուտ կեանքից.»

When I started working on Armenian, I thought it would be a straightforward job. Most of the lowercase characters are made of straight lines; uppercase characters exist in two forms: plain and calligraphic. Since slanted as well as upright characters are used (as a matter of fact, their rôles have been exchanged), this makes four fonts. Text is written from left to right, hyphenation is allowed — Dikran Karagueuzian offered me his hyphenation table for Armenian, which I adapted to my 8-bit and/or 7-bit ligature-based font encodings — so there should not have been any particular problem.

There was: *the kerning!* Armenian has many combinations such as *լ* + *ո* = *լո*, *ա* + *յ* = *այ* where kerning is unavoidable. Armenian printers have solved the worst cases by creating the following beautiful ligatures:

ե + ւ	մ + է	մ + ի	մ + խ
և	մէ	ﬕ	ﬗ
մ + կ	մ + ե	մ + ն	վ + ն
մկ	ﬔ	ﬓ	ﬖ

After a night of Fontstudio kerning — Armenian has 38 characters in uppercase (U) and lowercase (L) form, the number of UU, UL and LL combinations is 4332... — I had a minimum of 450 kerning and ligature pairs. In the METAFONTbook Prof. Knuth asserts that "Novices often go overboard on kerning. Things usually work out best if you kern by at most the half of what looks right to you at first, since kerning should not be noticeable by its presence (only by its absence)." But surely he was not thinking of Armenian.

Saxon

Æfter
ure Drihtnes Hælandes Cristes ȝebẏrtide an þusend ƿintra · ⁊ seofan ⁊ hundeahtatiȝ ƿintra · on þam an ⁊ tƿentiȝann ȝeare þær þe Ƿillelm ƿeolde ⁊ stihte Enȝleland · sƿa him God uðe · ȝeƿearð sƿiðe hefelic ⁊ sƿiðe ƿoldberendlic ȝear on þissum lande. Sþẏlc coðe com on mannum · ꝥ fullneah æfre þe oðer man ƿearð on þam ƿẏrrestan ẏfele · þet is on þam ðrife · ⁊ þet sƿa stranȝlice ꝥ mæniȝe menn sƿulton on þam ẏfele.

In the absence of an Old English font, scholars often use characters from the International Phonetic Alphabet to represent ʒ, þ, ð, etc. As a matter of fact, a cmr-like font (with IPA characters taken from wsuipa) for a "modern" output of the same input text is provided below:

> Æfter ure Drihtnes Hælandes Cristes ʒebẏrtide an þusend vintra; and seofan and hundeahtatiʒ vintra; on þam an and tventiʒann ʒeare þær þe Pillelm veolde and stihte Enʒleland; sva him God uðe; ʒevearð sviðe hefelic and sviðe voldberendlic ʒear on þissum lande. Sþẏlc coðe com on mannum; þæt fullneah æfre þe oðer man vearð on þam vẏrrestan ẏfele; þet is on þam drife; and þet sva stranʒlice þæt mæniʒe menn svulton on þam ẏfele.

The symbol ꝥ is an abbreviation for þæt, and ⁊ is a runic symbol for "and". There were no problems with this font: the j J encoding is used for the thorn character þ Þ because of the TEX input transliteration of the Greek θ which has the same sound (in modern Greek!). The pointed ẏ Ẏ are separate characters.

Old German

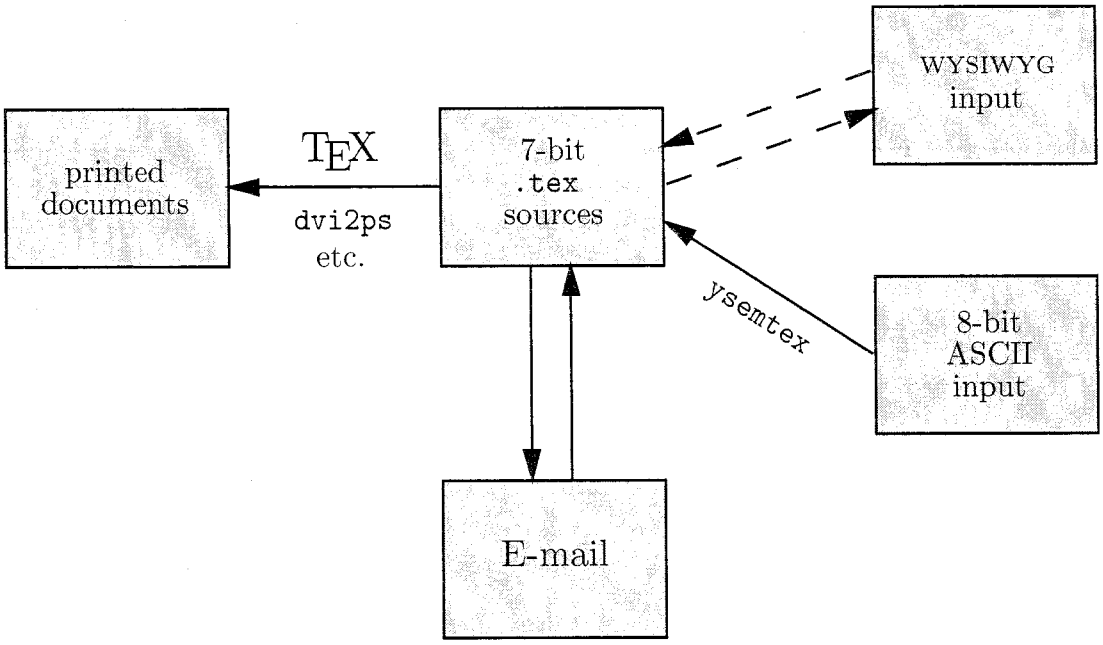
Het verdriet van België

Toen kwam Zuster Adam achter de doornhaag. Louis was zeker dat hij net vóór zij verſcheen het geritſel van haar kleed gehoord had, toen het langs de doornen ſtreek. Zij bleef ſtaan, niet lang, met gevouwen armen, zodat de wijde mouwen voor haar middenrif een zwart altaartje vormden. Vondeyne zag haar ook.

Old German fonts have been described in [3]. I included two of them (Fraktur and Schwabacher) in *Scholar*TEX, for scholars who want to distinguish old German text, or want to keep track of the original orthography (concerning long and short s, ligatures, etc.) in study editions (see also [12]). Now there is an end-of-word ligature for the "short" s, but the ligature `s:` must still be used inside words such as `Aus:gang` for Ausgang.

Further Ideas

The alphabets which are next on my schedule are Glagolitic, Old Church Cyrillic, Byzantine Greek (the uppercase letters used today by the Greek Orthodox Church), Coptic, Irish (calligraphic), Uiguric Mongolian (written from top to bottom), and a second Syriac font (Serto).



Another project is to combine the preprocessors with elementary WYSIWYG text editors providing screen fonts for all *Scholar*TEX fonts. The ordinary 7-bit TEX sources written by `ysemtex` would then still be read as source files, but each language would be displayed in the proper font.

For example, the `.tex` source file

```
My dear
\ins\arbon
\arbword{\yarb{}{\char'170}{\char'327}%
{\char'160}{\char'024}}
\arboff,
how are you?
```

which, if printed would produce

My dear أحمد, how are you?

would be visualized (here, in 9pt Monaco) as

```
My dear
\ins
أحمد,
how are you?
```

The text editor would only need to suppress all brackets, backslashes, and unnecessary macros, and display the characters in an Arabic screen font.

This would solve input encoding problems as well as problems concerning communication by electronic media (which allow only 7-bit ASCII text). The TEX sources produced by `ysemtex` could serve as an intermediary between e-mail, screen visualization and input, and printed output (see Fig. 6). By automating these procedures, one would have real e-mail in any possible alphabet.

Conclusion

TEX can easily and efficiently handle "those other languages", reaching the same quality level as with the more usual languages. METAFONT is essential for the creation of fonts of professional quality; the tools it provides are so powerful that you can make such fonts even at home, during your free hours, provided you invest the necessary care and feeling. But I think they deserve it, don't they?

References

[1] Guenther, Dean. "TEXT1 Goes Public Domain." *TUGboat* 11(1), pages 54–56, 1990.

[2] Haralambous, Yannis. "Arabic, Persian and Ottoman TEX for Mac and PC." *TUGboat* 11(4), pages 520–524, 1990.

[3] Haralambous, Yannis. "Typesetting Old German with TEX: Fraktur, Schwabacher, Gotisch and Initials." *TUGboat* 12(1), pages 129–138, 1991.

[4] Haralambous, Yannis. "On TEX and Greek..." *TUGboat* 12(2), pages 224–226, 1991.

[5] Jensen, H. *Die Schrift in Vergangenheit und Gegenwart*. Glückstadt/Hamburg, 1935.

[6] Knuth, Donald E. "Mathematical Typography." *Bulletin of the American Mathematical Society*, 1979.

[7] Knuth, Donald E. *Computers & Typesetting: Vol. E, Computer Modern Typefaces.* Reading, Mass.: Addison-Wesley, 1986.

[8] Lavagnino, John and Dominik Wujastyk. "An Overview of EDMAC: A plain TEX Format for Critical Editions." *TUGboat* 11(4), pages 623–643, 1990.

[9] Levy, Silvio. "Using Greek Fonts with TEX." *TUGboat* 9(1), pages 20–24, 1988.

[10] Tamari, Ittai Joseph. "Digitization of Hebrew fonts." In *Raster Imaging and Digital Typography*, Jacques André and Roger Hersch, eds. The Cambridge series on Electronic Publishing. Cambridge: Cambridge University Press, 1989.

[11] Tod, Marcus N., ed. *A Selection of Greek Historical Inscriptions.* Oxford: Clarendon Press, 1948.

[12] Wonneberger, Richard. "'Verheißung und Versprechen'. A Third Generation Approach to Theological Yypesetting." Pages 180–198 in *TEX for Scientific Documentation*, Jacques Désarménien, ed. *Lecture Notes in Computer Science* 236. Heidelberg: Springer, 1986.

[13] Yanai, Shimon and Daniel M. Berry. "Environment for Translating METAFONT to PostScript." *TUGboat* 11(4), pages 525–541, 1990.