Font Forum

There is no end: Omega and Zapfino

William F. Adams

Abstract

The future of type is OpenType (Adobe and Microsoft's successor to Apple's "Royal" font technology which was licensed to Microsoft as TrueType), Unicode, and other extensions of TrueType and the Type 1 font format such as ATSUI (Apple Typographic System for Unicode Information). While TEX has been extended to support other new formats and standards such as .pdf, support for the new font formats has been limited at best.

Fortunately, for Unicode in TEX, we have Omega, which coupled with the other strengths of TEX, can be sufficient to take advantage of new technologies even without explicit support, by using the proper (or improper) techniques.

This paper will be an explanation and exploration of this, looking at a specific font and format (the `.dfont` ATSUI-enabled version of Zapfino), arguably very nearly a worst-case scenario, and how it can be dissassembled into individual glyphs and seamlessly stitched back together to automatically insert ligatures and swash and variant forms using ASCII markup in an otherwise ordinary `.tex` source file which can then be used in a prepress ready workflow.

Introduction

Apple's Mac OS X derives from NeXTstep, by way of OPENSTEP, with a grafting of Apple Macintosh user interface concepts. In terms of font support, it handles Mac Resource/Suitcase fonts (both Type 1 and TrueType) and PC TrueType fonts, but loses support for Unix `.pfa` NeXT style `.font` bundles. Apple's QuickDraw/GX is added as well, now known as ATSUI (Apple Typographic System for Unicode Information) or as AAT (Apple Advanced Typography) depending on the specific context or emphasis desired.

Mac OS X provides many of its system fonts in the new `.dfont` format, which, while a straightforward storing of a Mac-style TrueType font in the file proper (the datafork in Mac parlance) instead of the resource fork as was done with Mac OS 9 and earlier, is not equivalent to a PC format TrueType font stored in a `.ttf` file. Although there are now programs which can open and parse fonts stored in a `.dfont` (Pfaedit[1] is a notable example), my interpretation of Apple's licensing agreement leads me to believe that any such parsing or conversion would not be allowed by that license.

However, having purchased Mac OS X and its $10,000 worth of fonts, one cannot help but wish to use them. Although Zapfino works well in "Cocoa" programs in Mac OS X such as TextEdit.app, its special features such as ligatures are enabled by AAT which is unfortunately not supported by the more traditional "Carbon" Macintosh applications in which class all mainstream graphic design applications are, at this writing.[2] This is unfortunately quite limiting: either one must limit oneself to Cocoa applications, or in applications such as InDesign, make use of its Glyph palette to insert alternates and ligatures by repetitive pointing-and-clicking. Since there is no TEX variant which can access system fonts on Mac OS X as of this writing,[3] one must develop a work-around which allows one to access arbitrary fonts from within TEX and to simulate the sophisticated typesetting capabilities of OpenType or Apple Advanced Typography.

The large character sets of fonts such as Apple Chancery or Zapfino make accessing characters in 8-bit blocks untenable, so Omega is an obvious choice. This serves two purposes: first, it makes the typeface, Zapfino by Prof. Hermann Zapf available for use in TEX by way of Omega; second, it provides an encoding scheme and mechanism to access arbitrary ligatures and alternates which may be of use for other projects.

History

Zapfino had its origins in Prof. Zapf's 1944 sketchbook, when he was a mapping officer during World War II. A previous attempt to render those letterforms as type, Virtuosa Script for D. Stempel, had been rather compromised by the limitations of hot metal matrices, especially the swash letters. This design was revived when David Siegel in 1993, after working on the Euler project with Prof. Zapf, and after graduating approached him about a chaotic calligraphic typeface based upon an example done for the Society of Typographic Arts in Chicago. Remembering the page from his sketchbook, Prof. Zapf saw the chance for a design without compromises

[1] Renamed to *FontForge*, this wonderful program is available from `http://pfaedit.sourceforge.net`.

[2] Since then, the open source drawing program Cenon has been released for Mac OS X as well as OPENSTEP 4.2 and GNUstep. It is available from `http://www.cenon.info`. There are also SoftMagic's "Project-M", Stone Design's Create or Purgatory Design's Intaglio, but none are widely used.

[3] Jonathan Kew has since released XeTeX, a successor to his TEX/GX program for Apple's QuickDraw/GX, which runs on Mac OS X, thus making AAT fonts accessible. It is available from `http://scripts.sil.org/xetex`.

due to the advantages afforded by digital type technology.

Digitization was done by Gino Lee, but production halted due to personal problems, and languished until Prof. Zapf showed the design to Linotype. It was then rendered as a traditional, multiple alphabet typeface family.

Installation

This then begs the question of how does one install a font into a program (system) which doesn't have direct support for that font format or its capabilities? The solution is quite obvious in retrospect: consider what the system does support (PostScript by way of dvips and the `\special` mechanism) and where that intersects with the capabilities of systems which can use the font to its fullest (Encapsulated PostScript File graphics). The solution then is to load all of the characters of a font into a file so that they may then each be output as individual `.eps` files, stitch said files together as a virtual font and then rely on dvips to put everything back together.

Zapfino, however, has so many characters (1,417 in the version bundled with Mac OS X 10.2 "Jaguar"[4]) that Omega, with its support for Unicode which provides for large character sets, is needed. Omega also affords the Omega Translation Process (OTP), which is far more efficient at enabling long ligatures than the standard TEX or PostScript mechanisms. Fortunately, odvips supports the aforementioned `special` mechanism.

Although font metric information is probably not protectable, there is no reasonable method at present to access the data stored within the Zapfino font file which wouldn't run afoul of Apple's license forbidding decompilation or other modification. Presumably, however, a program using the nsText object could access such data on a per character basis and write that out in a useful format. In lieu of such, a copy of the `.afm` files provided by Volker Schaa (he had received a copy of the original Linotype Zapfino CD-ROM from Prof. Zapf as a gift) was used as a beginning point. These files were converted into standard `.tfm` files using afm2tfm and thence to `.pl` files using tftopl. The file for the font Zapfino One served as the basis for `zapfino.ovp`, the base font file. The utility `ovp2ovf` was then used to create `ovf` and `ofm` files for Omega to use. The files are stored in (as appropriate)

```
~/Library/texmf/fonts/ovp/apple/zapfino
~/Library/texmf/fonts/ovf/apple/zapfino
~/Library/texmf/fonts/ofm/apple/zapfino
```

Before testing could begin, it was necessary to have the letterforms themselves accessible to output. This was done by using Adobe InDesign to typeset an Adobe Tagged Text file which enumerated all of the characters in Zapfino. First, a single character was set in the font Zapfino in InDesign at 72 points size with 96 points leading and then exported (File | Export..., select "Adobe InDesign Tagged Text" in the Formats: pop-up), yielding a file with the following line needed for our purposes:

```
<cTypeface:><cSize:><cLeading:><cFont:><cHang:>
<pHyphenationLadderLimit:><pHyphenation:>
<pHyphenationZone:><pTabRuler:><ParaStyle:>
<pHyphenationLadderLimit:0><pHyphenation:0>
<pHyphenationZone:0.000000><pTabRuler:
28.000000\,Left\,.\,0\,\;56.000000\,Left\,.\,0
\,\;84.000000\,Left\,.\,0\,\;112.000000\,Left\,.
\,0\,\;140.000000\,Left\,.\,0\,\;168.000000\,
Left\,.\,0\,\;196.000000\,Left\,.\,0\,
\;216.000000\,Left\,.\,0\,\;224.000000\,Left\,.
\,0\,\;252.000000\,Left\,.\,0\,\;280.000000\,
Left\,.\,0\,\;308.000000\,Left\,.\,0\,
\;336.000000\,Left\,.\,0\,\;>
<cTypeface:Regular><cSize:72.000000>
<cLeading:96.000000><cFont:Zapfino>
<cHang:Baseline>A<0xFFFD><cTypeface:>
<cSize:><cLeading:><cFont:><cHang:>
<cSpecialGlyph:><cTypeface:Regular>
<cSize:72.000000><cLeading:96.000000>
<cFont:Zapfino><cHang:Baseline>
<cNextXChars:Page>
```

(The exported character was "A" — there is some additional text above and below said line, but it need merely be preserved in its entirety for later use.) After a little study and experimentation, it was found that the placed character could be replaced with `<cSpecialGlyph:####>` where `####` was a number ranging from 1 (the first character, "A" shown in the Unicode glyph palette in in Mac OS X) to 1417 (the last character, the open Apple symbol), so an Excel file was created with 1,417 rows and three columns. The first column was everything before the "A" endlessly repeated, with `<cSpecialGlyph:` added. The second column incremented the current row number starting from 1. The third column closed out the line, adding a `>` to close the `cSpecialGlyph` directive. This was exported as text and replaced the line shown above in the Adobe InDesign Tagged Text file which was then saved.

Next, a template file was created, as shown in Figure 1 (File | New | Document..., the Facing Pages checkbox cleared, the one for Master Text Frame

[4] Since this writing, Linotype has released Zapfino Extra OpenType which provides even more characters, most notably small caps, and Forte which provides five additional weights. The technique here should also work with this new version. More information on Zapfino Extra is available from `http://www.linotype.com/1897/linotypezapfinoextra-folder.html`

checked, the Page Size set to Letter, Orientation to Landscape, Margins and Columns left at their default). After clicking "OK" and getting a new document the "A-Master" page icon in the "Pages" palette (Window | Pages) was double-clicked to allow editing of the master text frame. The master text frame is then selected and set to the coordinates X: 43p6, Y: 31p6, W: 39p0 and H: 33p0 using the "Transform" palette (Window | Transform) as shown in Figure 2. The main document is then returned to by double-clicking on Page 1 in the Pages palette. Clicking with the Text tool in the text block, one then chooses File | Place... and navigates to the Adobe InDesign Tagged Text file created above and places it in the document so that it auto-flows to create 1,417 pages with one character per page (click on the Master Text Frame while holding down the `<Shift>` key). The file is then saved as `Zapfino-chars` in a convenient location.

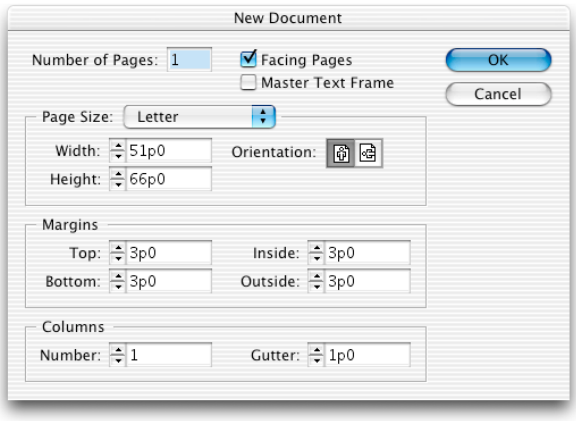


Figure 1: Adobe InDesign New Document Dialog

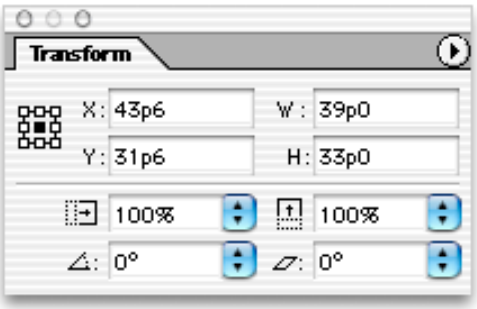


Figure 2: Adobe InDesign Transform Palette settings for master page text frame

Once we have this document with all of the desired characters, it is then a matter of exporting each

Figure 3: Z

page as a `.eps`. Fortunately, Adobe InDesign affords a menu option specifically for this, File | Export..., which includes direct support for the `.eps` format. Choosing "EPS" in the Formats pop-up menu takes one to a dialogue box where one can select various settings. For the initial font the settings used were: PostScript: Level 2, Color: Gray, Preview: None, Embed Fonts: Subset, Data Format: ASCII. InDesign is able to subset fonts in such a way that individual subsetted fonts may be recombined seamlessly within a PostScript file or `.pdf` without any of the encoding conflicts sometimes seen in fonts subsetted by Adobe Acrobat or other programs. The Data Format: must be set to ASCII, since (o)dvips cannot handle a binary encoded `.eps` file created in this fashion. The files are exported to `~/Library/texmf/fonts/eps/Apple/Zapfino` for later usage.

Then the `.pl` file for Zapfino-One was used as a basis for the initial `zapfino.ovp` Omega Virtual Font Property List. Notable settings which were necessary included setting the font's natural optical size `(DESIGNSIZE R 24)`. This technique is size-specific, and a different font must be made for each size which one wishes to typeset at. See the section *Peace* below for a work-around for this limitation.

With the character outlines now available, it is possible to place them within the virtual font using the `special` mechanism in odvips. Where each character has an entry like:

```
(MAP
   (SETCHAR 0 353)
   )
```

this is replaced with something like:

```
(MAP
   (SPECIAL PSfile=Zapfino-chars_277.eps)
   )
```

Unfortunately, when this is typeset the character is not positioned on the baseline, nor is it set to the correct size. Adjustments for the size and location of the `.eps` file placement must be worked out manually. Mostly a matter of trial and error, a test file

was created which placed a rule on the baseline, the test character and then another rule.

```
\font\zapfino=Zapfino at 24pt
\nopagenumbers
\overfullrule 0pt
\zapfino

\vrule depth 0pt \hskip-.5pt X

\vfill\eject\bye
```

Commands for controlling the position of the character and its size were then added to the `.pl` file and it was retypeset and adjustments were made until it was correct (see Figure 4). For example, the "IJ" entry looks like the following:

```
(MAP
    (PUSH)
    (MOVELEFT R 1.5734)
    (MOVEDOWN R 1.724)
    (SPECIAL PSfile=Zapfino-chars_277.eps
                  hscale=13.28 vscale=13.28)
    (POP)
    (MOVERIGHT R .543)
    )
```

First the current position is stored `(PUSH)`, then an adjustment is made for the offset of the character origin on the `.eps` file `(MOVELEFT)` and `(MOVEDOWN)`, the character is placed `(SPECIAL PSfile=Zapfino-chars_277.eps`, and scaled with `hscale` and `vscale`. Then the previous position is restored `(POP)` and the position advanced to match the `CHARWD (MOVERIGHT R .543)` (where .543 is the width of the "IJ" character).

With all of this done, one can begin testing the font so as to check the metrics of the characters. A number of the characters in Zapfino were re-drawn between the original Type 1 format and the version Apple bundled with Jaguar, so this step could not be skipped. How to space and test a new typeface design is well documented in several excellent references, most notably Stephen Moye's *Fontographer: Type by Design* and Walter Tracy's *Letters of Credit*, both of which are highly recommended to the aspiring type designer (or installer). In short, one sets various "standards" and other characters between them, adjusting the most common and easily spaced characters ("n" and "o") first, working toward those which are more difficult and assigning predetermined sidebearings to similar characters (so "m" gets the same sidebearings as "n"). Often when designing a new typeface, the initial attempt to set the sidebearings will result in a determination that certain characters must be re-drawn. Naturally that was not at issue here, and with the data gleaned from the `.afm` files, the metrics quickly reached a usable state. Much hastened in this case since the left sidebearings are preserved in the consistent placing of the characters on the page in the source file, so only the right sidebearings needed to be checked or adjusted.

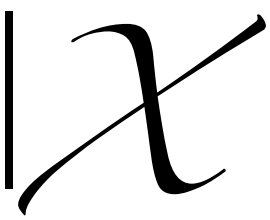


Figure 4: Final, correct `.eps` scaling and placement

Figure 5: TextEdit.app options for Zapfino in Mac OS X 10.2 were rather minimalistic

With the base character set available, next the ligatures and alternates needed to be provided for. Initially, Apple's options for Zapfino were somewhat limited as is evidenced by TextEdit.app (see Figure 5). WorldText.app, née GX/Write, exposed all of the capabilities encoded within the font, however (see Figure 6), and Apple has since expanded the nsText object to fully support Zapfino's myriad capa-

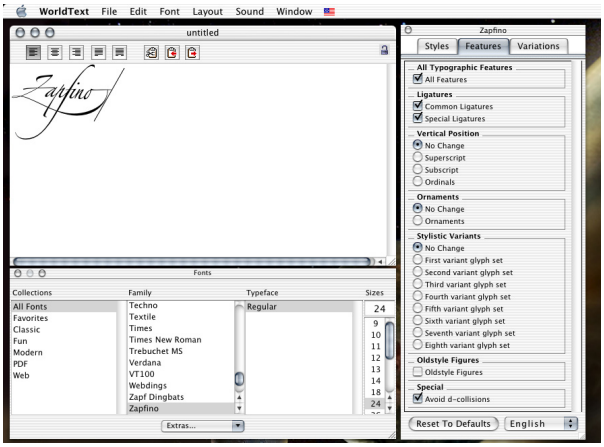


Figure 6: WorldText.app, provided in Apple's Developer Tools Samples folder, provides access to all of Zapfino's capabilities

bilities.[5] Although one may make various menu selections, or select characters from a Unicode "Character Palette" in Cocoa apps, or the Glyph palette in Adobe InDesign and other Adobe graphics applications, these options are not readily available in a text-stream composition-oriented tool like TEX.

Further, although OpenType and AAT are able to access characters by name, instead of directly with a number — and many of the more interesting characters in rich fonts such as Zapfino or Hoefler Text do *not* have Unicode code points — TEX and its variants (with the exception of the new XeTeX) require a number in an encoding vector for any given character. Toward this end, an encoding scheme was worked up to allow arbitrary ligatures of up to three characters in length, and to accommodate up to 32 variants for any given (unaccented character). While that last number may seem overkill, the OpenType specification provides for up to 20 variations of a character in its `salt` tag.[6] Having 16 bits available with Omega, the available bits were split into three sets, an initial set 6 bits long and two successive sets 5 bits long. The first set is long enough to encompass basic Latin capitals and minuscule (lowercase) letters as well as the numerals 0 through 9, with two bits left over. One of these was used as a "swash" bit, while the other remains available for use. The second and third sets encompass lowercase letters.

[5] See "Panther's Major Text Services Upgrade", `http://www.codepoetry.net/archives/2003/10/24/panthers_major_text_services_upgrade.php`

[6] See Tag: `'salt'` in `http://partners.adobe.com/asn/tech/type/opentype/appendices/features_pt.jsp`

Figure 7: Za

With all characters assigned to slots, it was then possible to create an Omega Translation Process (OTP) to replace characters with their appropriate ligatures:

```
input: 1;
output: 2;

states: VERBATIM;

expressions:

‘0’‘0’ => "{\zapfinoexpert " @"035A"}";
‘1’‘s’‘t’ => "{\zapfinoexpert " @"0653"}";
‘2’‘n’‘d’ => "{\zapfinoexpert " @"09A3"}";
‘3’‘r’‘d’ => "{\zapfinoexpert " @"0E23"}";
...
‘C’‘i’‘e’ => "{\zapfinoexpert " @"3504"}";
‘C’‘o’‘.’ => "{\zapfinoexpert " @"35DA"}";
‘D’‘r’‘.’ => "{\zapfinoexpert " @"3A3A"}";
...
‘T’‘h’ => "{\zapfinoexpert " @"78FA"}";
...
‘f’‘i’ => "{\zapfinoexpert " @"A91A"}";
%‘f’‘i’ => "{\zapfinoexpert " @"10DD"}";%fi-alt
%‘f’‘i’ => "{\zapfinoexpert " @"10DE"}";%fi-swash
...
‘g’‘g’ => "{\zapfinoexpert " @"ACDA"}";
%‘g’‘g’ => "{\zapfinoexpert " @"ACDB"}";%gg-alt
...
‘u’‘z’ => "{\zapfinoexpert " @"E73A"}";
‘w’‘n’ => "{\zapfinoexpert " @"EDBA"}";

@"F000 => <push: VERBATIM> ;

<VERBATIM>@"0021-@"007F => #(\1 + @"F000) ;

<VERBATIM>@"F001 => <pop:> ;

. => \1;
```

The Omega documentation[7] explains OTPs in detail. Each line in an OTP to handle a particular

[7] *Draft documentation for the Omega system* (John Plaice and Yannis Haralambous, 7 March 1998, available from `http://www.loria.fr/services/tex/moteurs/omega7mar1998.pdf`

case is fairly straightforward. First the characters to be replaced are identified (the `=>` indicates a replacement is being made), then a verbatim sequence of replacement commands is given within quotes, with a character's encoding being provided in an "escaped" fashion outside of the quotes (hence, `^^^^035a` is `"035A`). This was saved in a file `lat2zapf.otp` and processed with the command

```
otp2ocp lat2zapf.otp lat2zapf
```

and the files moved to `~/Library/texmf/omega/otp` and `~/Library/texmf/omega/ocp` respectively.

A number of font transformations normally handled with ligatures are transferred to the OTP. Although not shown, em and en dashes are created in the OTP, as are "smart" quotes. Similarly, the standard ligatures *ff*, *fi*, *fl* and *ffi* are handled in the OTP (there is no *ffl*). There are several reasons for this: conformance with other Omega fonts, to preclude ligature formation interfering with contextual processing of alternates, efficiency (OTPs can process a single string of arbitrary length, but TEX's ligature mechanism requires sequential processing of all iterations leading up to the final form; see below) and the small matter of Omega crashing when forming a ligature past the 8-bit point.

```
partial listing of cmr10.pl
(LIGTABLE
...
   (LABEL C f)
...
   (LIG C f O 13)
...
   (STOP)
   (LABEL O 13)
   (LIG C i O 16)
...
```

Note that there were several ligatures which were commented out with `%`, alternate or swash forms.

It was then possible to test the font for the first time with a (Plain) TEX file like:

```
\font\zapfino=Zapfino at 24pt
\font\zapfinoexpert=Zapfino-expert at 24pt
\ocp\zapfOCP=lat2zapf
\ocplist\zapfOCPlist=%
\addbeforeocplist 1 \zapfOCP
\nullocplist
\nopagenumbers
\overfullrule 0pt
\zapfino\pushocplist \zapfOCPlist

00 1st 2nd 3rd 4th 5th
   6th 7th 8th 9th 10th

Cie. Co. Dr. Esq. Ht. Jr. Ltd. McSullivan
Mlle. Mme. Mr. Mrs. Ms. No. 9 Sig. Sr.
Sra. Srta. St.

Thorn dicot draw presage fez type. affianced
effect fit fluent aft egg isthmus out
Zapf phone happy Arial art mesh spell
mass stay the matte spritz uzo own
\vfill\eject\bye
```

which shows all of the ligatures, as well as pointing out that perhaps using old-style figures as the default wasn't such a great idea, at least not if one intends to use the "0th" ordinal ligature.

Once the basic sidebearings of the font were set and the ligatures "wired up", the interplay of specific letterpairs needs to be addressed — the graphical interface which Mac OS X provides for Zapfino provides some hints on this, most notably, "Avoid *d* collisions." As one can see in figure 9, the standard, forward slanting lowercase *d* in Zapfino collides with the preceding character quite often, especially when preceded by a capital letter. Figure 8 also hints at awkward character interactions such as the near-intersection of *ppy*.

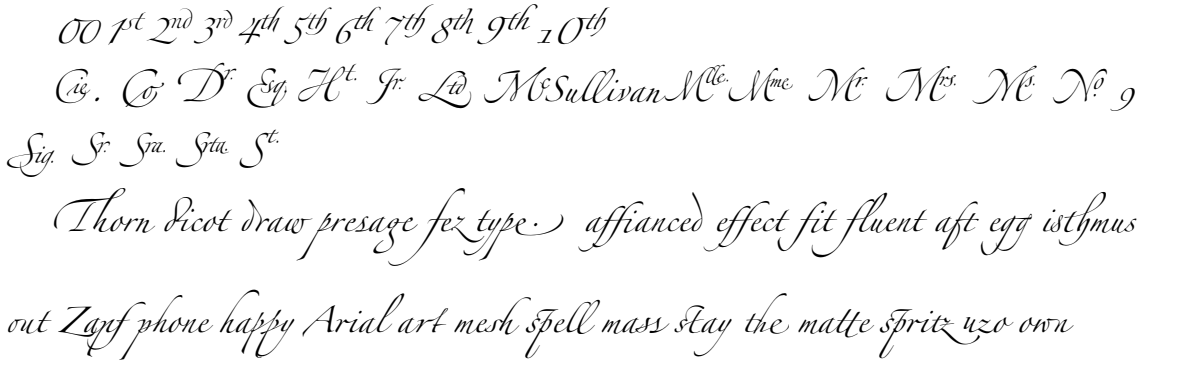


Figure 8: Initial test output

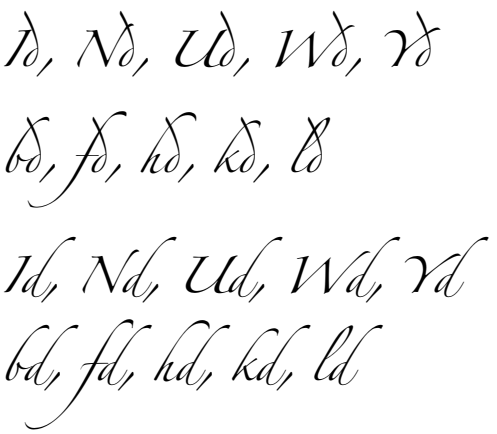


Figure 9: In Zapfino, *d*'s collide

While one is tempted to merely set the second style of *d* described in the font file, or Linotype's Adobe Type 1 version of Zapfino as `d.2`,[8] this brings to light an excellent chance to add some "chaos" to the typeface and pick different character versions.

The overall intent is to create a system which will make use of all possible characters in some circumstance, and a collection of test files which will test all such circumstances. While one is tempted to just "fix" all such occurrences, this leads to unnecessary processing effort. Instead, a listing of all possible digraphs (letterpairs) in the English language was created (drawing from *Webster's Dictionary*, *The Complete Works of Shakespeare* and *The King James Bible* — NeXT users will recognize the choice of references as those bundled with NeXTstep or readily available as texts for NeXT's Digital Librarian program), so that it was possible to determine that only *Id*, *Nd*, *Ud*, *Wd*, and *Yd* needed to be considered for the capitals (thus saving by not creating entries for *Fd*, *Hd*, *Jd*, *Kd*, *Td*, *Vd*, *Xd*, and *Zd* which do not occur in the English language sample set used). The following lines in the OTP prevent such collisions by replacing the normal *d* with the `d.2` alternate.

```
‘I’‘d’ => "I{\zapfinoexpert " @"FC62"}";
‘N’‘d’ => "N{\zapfinoexpert " @"FC62"}";
‘U’‘d’ => "U{\zapfinoexpert " @"FC62"}";
‘W’‘d’ => "W{\zapfinoexpert " @"FC62"}";
‘Y’‘d’ => "Y{\zapfinoexpert " @"FC62"}";
```

With the obvious change taken care of, more subtle changes were then considered. Working from the most frequently occurring letters to the rarer ones, a file for each letter, containing a word for each letterpair extant in the sample texts was typeset in the font Zapfino and then examined carefully for awkward interactions. When one was found, a test file was typeset with a basic macro to set the word with all possible variations for a given letter:

Figure 10: Zap

```
\def\testa#1#2#3{{#1}{#2}{#3}\par%   a
{#1}{\zapfinoexpert ^^^^fc02}{#3}\par%
{#1}{\zapfinoexpert ^^^^fc03}{#3}\par%
{#1}{\zapfinoexpert ^^^^fc04}{#3}\par%
{#1}{\zapfinoexpert ^^^^fc05}{#3}\par%
{#1}{\zapfinoexpert ^^^^fc06}{#3}\par%
}
```

allowing the selection of the best choice.

The normal usage was to set the text to be tested without any special treatment, and then to insert the macro into the text so as to call out alternates.

```
...
halfcock

\testf{hal}{f}{cock}
...
```

Twenty-six such macros were created, and a second set with only two arguments was also made for the capitals. Often it was necessary to test the characters before or after as well.

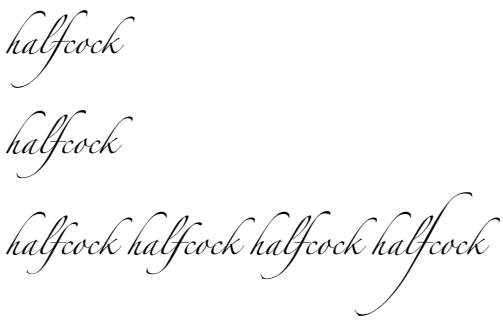


Figure 11: Test macro output

[8] Apparently, this route was taken by Apple and Linotype in the implementation in Mac OS X 10.3 "Panther".

Figure 12: Zapf without the *pf* ligature

Figure 14: Ligatures required examination of trigraphs and tetragraphs

The pair *nf* and a number of other pairs ending in *f* did not work well because the *f* in Zapfino One does not connect to the character before it, so a number of lines similar to the following were added to the OTP to handle such cases:

```
‘n’‘f’ => "n{\zapfinoexpert " @"FCA4"}";
‘o’‘f’ => "{\zapfinoexpert " @"FDC2 @"FCA4"}";
‘i’‘f’ => "i{\zapfinoexpert " @"FCA4"}";
‘i’‘j’ => "i{\zapfinoexpert " @"FD24"}";
```

Examining that file will show the reader which character pairs were felt to require adjustment. For the most part, replacing one character or the other (occasionally both) with an alternate provided an æsthetically pleasing combination, though a few uncommon pairs (*gj, Kz*) did not yield such.

Interestingly, the *gj* pair actually convinced me to revisit an initial decision to set the normal *j* (from the font Zapfino One) only at the end of characters, instead choosing it as the only appropriate form to use after a *g*.

Zapfino's many lowercase ligatures required that the list of digraphs be extended to encompass the trigraphs and tetragraphs which encompass these lettersets in addition to the digraphs mentioned above. For each such letter or ligature there is a file (e.g., `di-.txt` which is shown next), which encompasses the extant lettersets including said letter or ligature's letters with the hyphen indicating the location of the letters being checked for, so that file ranges from “Diagram” through “dizzy.” Since the *di* and *dr* ligatures do not reach as far to the left, most such letterpairs were acceptable. Notable exceptions were “ldr” and “ldi” which were set to make use of the alternate character `1.4` as shown in Figure 14.

```
Diagram diagram
Dibs dibs
Dice dice
Did did
Died died
Different different
Dig dig
Dihedron dihedron
Diiodemethane diiodemethane
Dijon Jehudijah
Dikdik dikdik
Dill dill
Dim dim
Din din
Diode diode
Dip dip
Diquat diquat
Dire dire
Distill distill
Ditto ditto
Diurnal diurnal
Divine divine
Diwali diwan
Dixie dixie
Diyarbakir
Dizzy dizzy
```

One hundred and fourteen such files were created. Usage with a leading capital is shown first if possible, followed by lower-case usage, again, if possible (*diy* apparently is only possible when capitalized as in *Diyarbakir*). If only lowercase usage is possible (not shown above) that line is indented by a space.

Zapfino is so narrow, however, with such a low x-height, and such large ascenders, that it is actually possible for the ascender on a *d* to collide with a character two characters before it as in, “led.” Thus a set of test files encompassing all characters with ascenders followed by any letter, followed by a “d” was created. Unfortunately, it is also possible for a

Figure 13: Alternate treatments

"d" at the beginning of a word to clash with the previous word if it ends with an ascender. Suggestions for dealing with this situation would be welcome.[9]

Doubled characters required special attention if there is no ligature for them, requiring that one check to determine which letter versions look attractive together.

After all such files were typeset and examined twice (once to establish the initial replacements, a second time to check for interactions between the replacements and characters which follow or precede them) it was possible to begin using Zapfino in Omega. The first such usage was for a holiday card as discussed in the section *Peace*.

Swash markup

That initial usage did not however, allow for user-level control of individual characters. One could of course use direct access with constructs such as `{\zapfinoexpert\char^^^^2804}`, but that's not exactly user-friendly (or even mnemonic). Although the use of swash alternates has been considered for Omega before,[10] and Alan Hoenig has considered macro-based schemes,[11] there exists no generally accepted system for indicating "swashness" in running text. The following version of a markup scheme for accessing arbitrary alternates and swashes is an experimental interface to determine how well this could be done using just an OTP.

The "swash" OTP uses the following options:

`+----a`	(first level alternate)
`+-----a`	(second)
`+------a`	(third)
`+-------a`	(fourth)
...	
`a----+`	(first level alternate at the end)
`a-----+`	(second level alternate)
&c.	

As always, the best laid plans ran into problems of implementation. The original intent was to add additional `+`s instead of continuing to add `-`s, but this became infeasible when a limit in the number of instructions (10,000) in a given state in an OTP was reached.[12]

[9] This is probably why Apple and Linotype chose to change the default "d" as noted in footnote number 8.

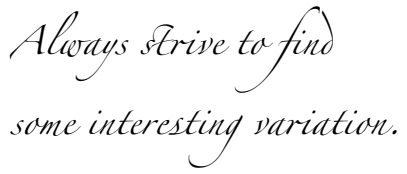
[10] Yannis Haralambous and John Plaice, "First applications of Ω: Adobe Poetica, Arabic, Greek, Khmer". *TUGboat* **15**(3), 1994.

[11] Alan Hoenig, "The Poetica family: Fancy fonts with TEX and LATEX". *TUGboat* **16**(3), 1995.

[12] My thanks to Giuseppe Bilotta for his patience and technical acumen, not merely determining the nature of the difficulty, but also providing a sample file showing how to work around it.

Figure 15: Zapf with the *pf* ligature

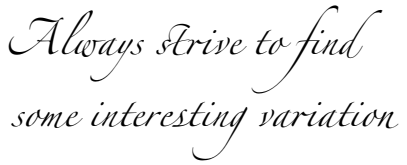
Before adding swash variations, here is a text using a modest set of alternates and most ligatures:



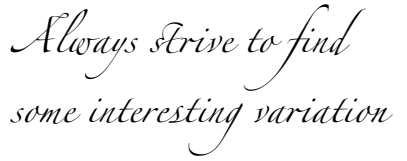
At the first level of ornamentation, which would be indicated as follows:

```
+-----Always strive to find-----+
+-----some interesting variation-----+
```

it appears thus:

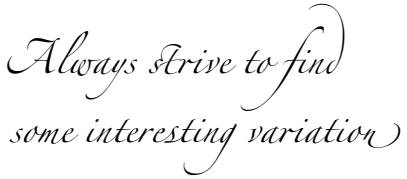


Continuing, the process yields:



or even:

Adding and subtracting hyphens, one might finally arrive at:



Left unaddressed is the matter of arbitrary swash variants in the middle of a text. Either of the above schemes can be used within a word, but interferes with spell-checking.

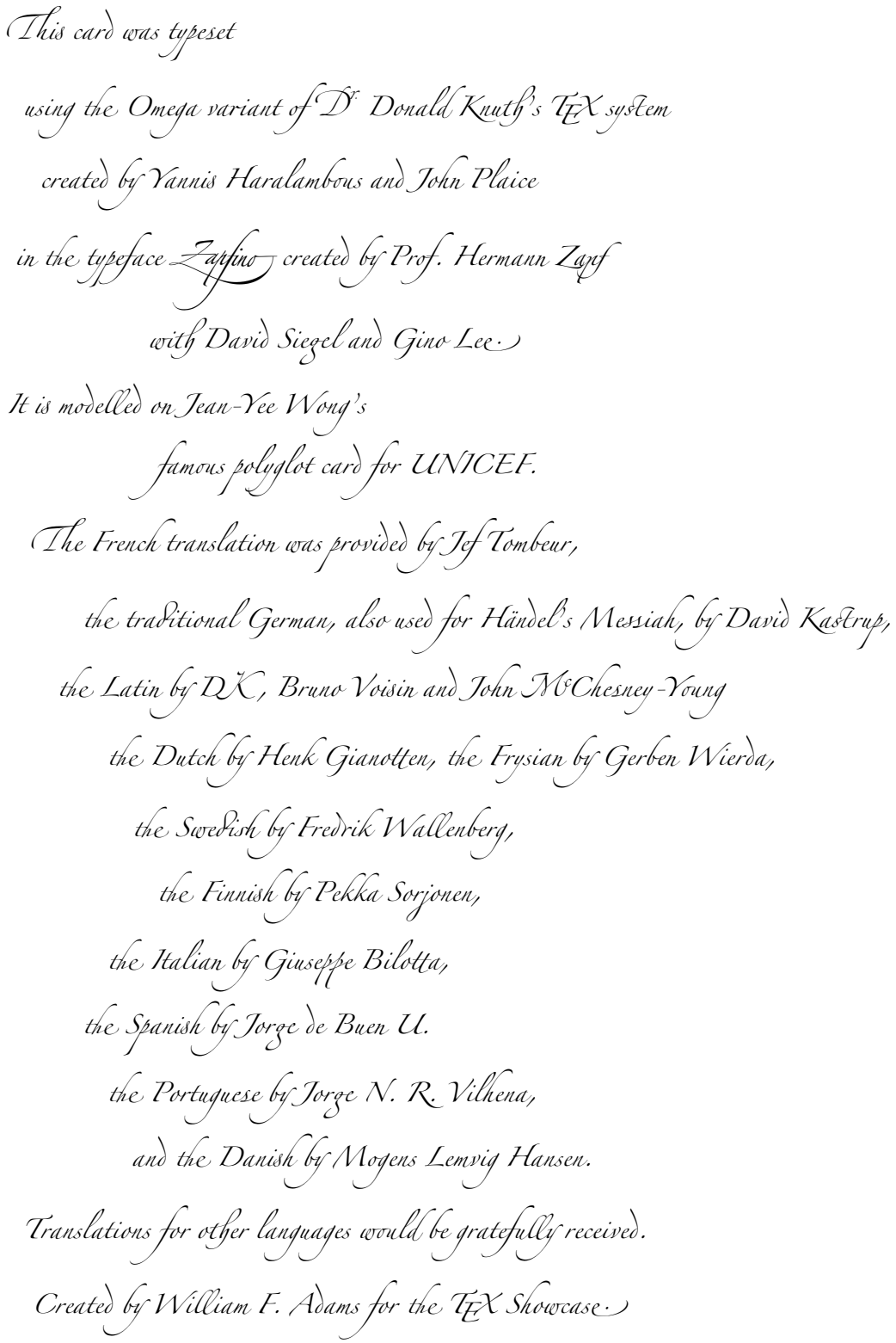


Figure 16: Peace on Earth card back page

Peace on Earth
Good Will Toward Men

Figure 17: Peace on Earth interior

Zapfi

Figure 18: Zapfi

Peace

The first use of this Zapfino install in Omega was to set a holiday card modelled on Jeanyee Wong's famous polyglot card for UNICEF.[13] After this was initially set, it was announced on Usenet and various mailing lists related to TEX or typography in the hope of readers providing additional translations. Jef Tombeur and Apostolos Syropoulos were kind enough to provide translations for French and Greek respectively (though since Zapfino doesn't contain a full Greek alphabet the latter was provided typeset in the Kerkis font), and a revised version was made available as a part of the TEX Showcase.[14]

In the course of preparing this paper, a second, more successful, announcement and request for translations was made, with many people providing translations and commentary (I had neglected to mention the text, "Peace on earth, good will toward men" as being the latter part of the verse Luke 2:14 from *The King James Bible*, so in addition to straightforward transcriptions from various Bibles, I also received a number of personal interpretations — this verse is also quite controversial due to varying transcriptions of the original Greek text and subsequent emendations by various scribes) resulting in an expansion of the card to three pages, almost a dozen languages (Greek has been temporarily omitted) and twenty names.

Once typeset as a three page `.pdf`, the first and last pages (which were not filled completely) were cropped, then each page was exported as a `.eps` and imposed in Macromedia FreeHand where each was scaled appropriately to fit nicely on a tabloid or A3 sheet which could then be folded twice to make a gatefold card. This affords the illusion of having three different sizes of Zapfino available, despite only one font at a fixed size (24 pt.) having been made.

Paix sur terre
Bonne entente entre toutes et tous

Friede auf Erden
und den Menschen ein Wohlgefallen

In terra pax
hominibus bonæ voluntatis

Vrede op aarde
aan de mensen van goede wil

Frede op ierde
ûnder minsken fen it wolbehagen

Frid på jorden,
till människorna ett gott behag

Rauha Maassa
Ja Ihmisillä Hyvä Tahto

Pace in terra
agli uomini di buona volontà

Paz en la tierra
a los hombres de buena voluntad

Paz na terra
entre os homens de boa vontade

Fred til mennesker
med Guds velbehag!

Figure 19: Peace on Earth card front page text

[13] Exhibit 214. UNICEF Christmas Card. Jeanyee Wong. 1962. *Two Thousand Years of Calligraphy: A Three-part exhibition organized by the Baltimore Museum of Art, the Peabody Institute Library and the Walters Art Gallery, June 6–July 18, 1965, A Comprehensive Catalog*: Baltimore, Maryland, 1965.

[14] `http://www.tug.org/texshowcase`

The Duck & the Mouse

by Jessica Adams

*One day in June a *DUCK* and a *MOUSE* were walking in a field when the *DUCK* saw a *TULIP*. The *DUCK* said, "Look at the pretty little red flower."*

*"Yes it's very pretty, but look..." said the *MOUSE*.*

*"What?" asked the *DUCK*.*

*"What is this funny long round thing?" asked the *MOUSE* digging in the leaves beneath the *TULIP*. "It's very pretty too."*

*"It's a people thing, a *PEN*." answered the *DUCK*.*

*"I wonder why it's here?" asked the *MOUSE* as it pulled the *PEN* out onto the path.*

*"Shhh! Listen!" whispered the *DUCK*.*

*"Humans! Hide!" hissed the *MOUSE*, and the two hid behind the *TULIP*.*

*"Don't worry, they're children." said the *DUCK*.*

*The big girl human asked, "Why'd you take my good *PEN*, Charlie?"*

"But Lizzie, I didn't mean to lose it!"

*"Look my *PEN*!" shouted the girl as she picked up her *PEN* from the ground where the *MOUSE* had dropped it.*

*The *DUCK* and the *MOUSE* breathed a sigh of relief and together they walked into the sunset.*

The End.

Figure 20: A child's story

```
\font\zapfino=Zapfino at 24pt
\font\zapfinoexpert=Zapfino-expert at 24pt
\ocp\zapfOCP=lat2zapf
\ocp\swashOCP=sws4zapf
%uncomment the rebusocp lines for pictures
%rebus\ocp\rebusOCP=rbs4zapf
\ocplist\zapfOCPlist=%
\addbeforeocplist 1 \zapfOCP
\addbeforeocplist 1 \swashOCP
%rebus\addbeforeocplist 1 \rebusOCP
\nullocplist
\nopagenumbers\overfullrule 0pt
\zapfino\pushocplist \zapfOCPlist
The +------D\kern2ptuck (-------ET{}
the +------Mouse \smallskip
by +------Jessica +-----Adams
\smallskip
One day in June a *DUCK* and a *MOUSE*
were walking in a field when the *DUCK*
saw a *TULIP*. The *DUCK* said,
``Look at the pretty little red flower.''

``Yes it's very pretty, but look...''
said the *MOUSE*.

``What?'' asked the *DUCK*.

``What is this funny long round thing?''
asked the *MOUSE*, digging in the leaves
beneath the *TULIP*. ``It's very pretty too.''

``It's a people thing, a *PEN*.''
answered the *DUCK*.

``I wonder why it's here?'' asked the *MOUSE*
as it pulled the *PEN* out onto the path.

``Shhh! Listen!'' whispered the *DUCK*.

``Humans! Hide!'' hissed the *MOUSE*,
and the two hid behind the *TULIP*.

``Don't worry, they're children.''
said the *DUCK*.

The big girl human asked,
''Why'd you take my good *PEN*, Charlie?"

``But Lizzie, I didn't mean to lose it!''

``Look, my *PEN*!'' shouted the girl
as she picked up her *PEN* from the ground
where the *MOUSE* had dropped it.

The *DUCK* and the *MOUSE*
breathed a sigh of relief and together they
walked into the sunset.

The End.\vfill\eject\bye
```

Zapfin

Figure 21: Zapfin

And pictures too

In addition to alternates and ligatures, Zapfino also has a number of ornaments available. Typically, access to such ornaments in TEX has been rather prosaically done using a macro which calls a particular ornament by its number. Although serviceable, this requires the user to have access to a chart which matches things up, and is not well-suited to usage in running text, or to easy typing or proofing. Oddly, although Helvetica seems to have a "Rebus" option in Mac OS X 10.3 "Panther", this does not seem to have been implemented for Zapfino, reducing one to point-and-clicking to access them from a glyph palette of some sort. Since that's not an option for normal TEX usage, an alternative scheme needed to be worked up.

Drawing upon the "Rebus" option for inspiration, a descriptive text was matched up for each ornament, and a new OTP was created to allow said text, set in all caps and surrounded by asterisks (e.g., `*PEN*`) to be replaced by the appropriate ornament when the OTP is loaded.

I prevailed upon my daughter to write a story making use of animals and things represented as ornaments in the Zapfino font. A typical setting of the story for proofreading purposes is shown on the facing page. The source text is shown to the left.

Some of the ornaments available with Zapfino include:

Figure 22: Ornaments

Lastly, the story with appropriate words replaced with ornaments from Zapfino (a rebus text) is shown on the following page in Figure 23.

Unfortunately, as noted above, the ornaments are numbered, not given descriptive labels. While some of them are obvious, others are less so. Suggestions on tags for unnamed ornaments would be welcome.

The Duck & the Mouse

by Jessica Adams

One day in June a 🦆 and a 🐭 were walking in a field when the 🦆 saw a 🌷. The 🦆 said, "Look at the pretty little red flower."

"Yes it's very pretty, but look..." said the 🐭.

"What?" asked the 🦆.

"What is this funny long round thing?" asked the 🐭 digging in the leaves beneath the 🌷. "It's very pretty too."

"It's a people thing, a 🍎." answered the 🦆.

"I wonder why it's here?" asked the 🐭 as it pulled the 🍎 out onto the path.

"Shhh! Listen!" whispered the 🦆.

"Humans! Hide!" hissed the 🐭, and the two hid behind the 🌷.

"Don't worry, they're children." said the 🦆.

The big girl human asked, "Why'd you take my good 🍎, Charlie?"

"But Lizzie, I didn't mean to lose it!"

"Look my 🍎!" shouted the girl as she picked up her 🍎 from the ground where the 🐭 had dropped it.

The 🦆 and the 🐭 breathed a sigh of relief and together they walked into the sunset.

The End.

Figure 23: A child's picture story

Figure 24: Zapfino without full ligation

Figure 25: Zapfino

Overview

Thus is it shown that with a complete disregard for efficiency, storage space and processing time, one can access *any* font from within TEX so long as it is supported on a platform whose capabilities somewhere intersect those utilities and variations which have grown up around TEX. This technique is completely platform agnostic after the initial stage, and if one foregoes the best font handling, can be done with bundled and free software, without recourse to commercial applications such as Adobe Acrobat.

Take an application which can access all of the characters in the OpenType or AAT/ATSUI font to typeset every character one wants to get at as a document, convert each page into `.eps` files, then create a virtual font which places the appropriate `.eps` file for a given letter, which is then used to typeset a PostScript file which places each character as necessary. When the resulting file is distilled with Adobe Acrobat Distiller the subsetted fonts in the `.eps` files are transparently stitched back together in a fashion which any reasonably capable PostScript interpreter or PDF viewing program can handle.

Advantages The good things about this technique include:

- No font conversion necessary
- No issues with licensing (Apple's software license forbids derivative products, so one may not convert or decompile typefaces covered by Apple's licensing agreements — Apple has even been stripping out tables to cripple their fonts so they won't work on other platforms if conversion is attempted)
- One can get at all 1,400+ characters in Zapfino without re-encoding or using awkward macros or active characters.
- Version invariant — already there have been *four* distinct versions of Zapfino, and little provision for backwards compatibility has been made.
- One is not limited to the normal TEX color models, but may use anything which InDesign has access to, including spot colors.

Disadvantages As alluded to above, this technique does have a number of drawbacks:

- Must make and store the individual `.eps` files (hundreds of megabytes, requires commercial software for the best results).
- Large file sizes (tens of megabytes for a single lengthy sentence — an attempt to create a complete sample of all digraphs and trigraphs ran into the Windows 9x file size limit of 512MB).
- Processing time (distill times in the tens of seconds for a couple of sentences, generating the `.ps` file with odvips takes even longer).

Future developments The following things remain to be done for this particular technique for accessing Zapfino:

- The base font needs to be filled in beyond just ISO Latin 1 so as to provide the balance of the Unicode-encoded characters available in Zapfino. This would be much easier if utilities such as afm2tfm and fontinst would completely support character sets larger than 8 bits.
- Similarly, most accented letters have swash variants, but the encoding scheme worked up for handling swashes and ligatures doesn't encompass such. Suggestions for a solution for this would be welcome.
- The text test set should be enlarged to encompass all possible trigraphs or even tetragraphs, and the OTP adjusted to at once increase randomness, and also reduce the incidence of certain characters which have a stronger presence than might be desireable.
- A LATEX 2_ε package should be put together to move usage out of the realm of hackery and into more mainstream production.
- Examine usage with Mac OS X Panther.

The *X* Factor

Since this presentation was made, Jonathan Kew has released XeTeX, the successor to TEX/GX, a TEX-variant which made use of QuickDraw/GX, from which Apple Advanced Typography is derived. Rapidly improving, it is well suited to personal use, or production use in an up-to-date production environment which is not hobbled by a need for backwards compatibility with older PostScript RIPs or other software.

It is especially well suited for use with languages which use character sets other than those based on Latin (which is its main focus).

By way of comparison, here is the source code for the Apple acknowledgement/"Thank You" which appeared in the TUG 2003 proceedings:

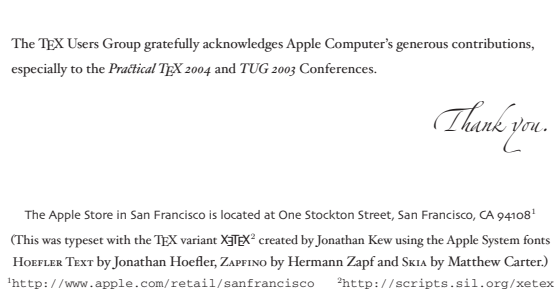


Figure 26: Thanking Apple

```
%&personaltex
%
%Copied from the XeTeX example file ‘‘XeTeX-notes.tex’’
\font\tx="Hoefler Text:Number Case=Upper Case Numbers" at 12pt
\font\txsmall="Hoefler Text:Number Case=Upper Case Numbers" at 11pt
\font\sc="Hoefler Text:Letter Case=Small Caps" at 11pt
\font\it="Hoefler Text Italic" at 12pt
\font\mt="Hoefler Text Black" at 16pt
\font\sh="Hoefler Text Black" at 12pt

\font\tt="Courier" at 11pt

\font\lg="Lucida Grande" at 11pt
\font\lgb="Lucida Grande Bold" at 16pt
\def\LaTeX{\leavevmode L\kern-.25em\raise.5ex\hbox{\sc a}\kern-.1em\TeX}
\def\XeTeX{\leavevmode
  \setbox0=\hbox{\lg X\lower.5ex\hbox{\kern-.1667em }\kern-.15em \TeX}%
  \dp0=0pt\ht0=0pt\box0 }
\def\mtXeTeX{\leavevmode
  \setbox0=\hbox{\lgb X\lower.5ex\hbox{\kern-.1667em }\kern-.15em \TeX}%
  \dp0=0pt\ht0=0pt\box0 }
\def\TeXgX{\TeX\lower.5ex\hbox{\kern-.15em G}\kern-.25em X}

% don't like how the hyphen in Hoefler appears, so we hack it:
\catcode‘\-=\active \def-{\lower.2ex\hbox{\char‘\-}\penalty0 }

%Copied from the XeTeX example file ‘‘story-zapfino.tex’’
\font\zapfinoreg="Zapfino" at 14pt
\font\zapfinofirst="Zapfino:Stylistic Variants=First variant glyph set" at 14pt
\font\zapfinosecond="Zapfino:Stylistic Variants=Second variant glyph set" at 14pt

%Copied from the XeTeX example file ‘‘FontSamples.tex’’
\frenchspacing % often works better with XeTeX
\baselineskip16pt
\parindent0pt
\parskip\medskipamount
\nopagenumbers

\font\1="Hoefler Text Italic:Letter Case=Small Caps" at 12pt
\font\1="Hoefler Text Italic" at 12pt
```

A note about Figures 3, 7, 10, 12 15, 18, 21, 24, 25 and 27

After my presentation in Hawai'i, Barbara Beeton specifically asked that I include an animation of the sequence of typing Zapfino, the contextual replacement of letters with the appropriate ligatures in the on-line version. Due to time constraints, I did not manage to do so. However, creating animated `.gif`s has always reminded me of study hall in my younger school days and doodling little figures in the margins of notebooks and pads, and of a Warner-Brothers book I had as a child which featured Wile E. Coyote and the Roadrunner of cartoon fame with such a sequence in its margin. I believe this is the first *TUGboat* article to include a flipbook (the figures in the upper right-hand corner of the recto pages). If present, a matching figure on the upper left of a page shows one what the text looks like without ligatures.

Figure 27: Zapfino ornament

◇ William F. Adams
ATLIS Graphics
75 Utley Drive, Suite 110
Camp Hill, PA 17011
USA
willadams@aol.com

Of the making of books, there is no end.

— The Prophet Muhammad

```
\font\italt="Hoefler Text Italic:Ligatures=Rare Ligatures,Diphthongs;
               Smart Swashes=Archaic Long s Swash;
               Character Alternatives=Swash Caps"
               at 12pt
\font\sr="Skia Regular" at 11pt
\font\1="Skia Regular:width=1.25" at 12pt
\font\1="Skia Regular:width=0.8" at 12pt
\font\1="Skia Regular:width=0.8;weight=1.5" at 12pt
\font\1="Skia Regular:width=0.8;weight=2.5;Number Case=Upper Case Numbers" at 12pt
\font\1="Skia Regular:width=0.8;weight=0.6;Ligatures=Rare Ligatures" at 12pt

\font\1="Skia Regular" at 18pt

\tx%
The \TeX\ Users Group gratefully acknowledges Apple Computer's generous contributions,

especially to the {\italt Practical \TeX\ 2004}
and {\italt TUG 2003} Conferences.
\bigskip
\hfill{\zapfinosecond Thank you.}
\bigskip \strut \bigskip
\centerline{\sr The Apple Store in San Francisco is located at One Stockton Street,
San Francisco, CA 94108$^1$}

\medskip
\centerline{\txsmall (This was typeset with the \TeX\ variant \XeTeX$^2$ created by Jonathan Kew
using the Apple System fonts}
\smallskip
\centerline{{\sc Hoefler Text} by Jonathan Hoefler, {\sc Zapfino} by %Prof.
Hermann Zapf and {\sc Skia} by Matthew Carter.)}
\smallskip
\centerline{$^1$\tt http://www.apple.com/retail/sanfrancisco ~ $^2$http://scripts.sil.org/xetex}

\vfill\eject\bye
```