Plain TEX and OpenType

Hans Hagen

Abstract

This article demonstrates how you can use OpenType fonts in plain LuaTEX, using a few generic modules that ship with the ConTEXt distribution.

1 Running

For testing basic LuaTEX functionality it makes sense to have a minimal system, and traditionally plain TEX has been the most natural candidate. It is for this reason that it had been on the agenda for a while to provide basic OpenType font support for plain TEX as well. Although the MkIV node mode subsystem is not yet perfect, the time was right to start experimenting with a subset of the MkIV code.

Using plain roughly comes down to the following. First you need to generate a format:

```
luatex --ini --fmt=luatex.fmt luatex-plain.tex
```

This format has to be moved to a place where it can be found by the `kpse` library. Since this can differ per distribution there is no clear recipe for it, but for TEX Live some path ending in `web2c/luatex` is probably the right spot. After that you can run

```
luatex luatex-test.tex
```

This file lives under `generic/context`. When it is run it is quite likely that you will get an error message because the font name database cannot be found. You can generate one with the following command (which assumes that you have ConTEXt installed):

```
mtxrun --usekpse --script fonts --names
```

The resulting file `luatex-fonts-names.lua` has to be placed somewhere in your TEX tree so that it can be found anytime. Beware: the `--usekpse` flag is only used outside ConTEXt and provides very limited functionality, just enough for this task. Again this is a distribution specific issue so we will not dwell upon it here.

The way fonts are defined is modelled after XETEX, as it makes no sense to support the somewhat more fancy ConTEXt way of doing things. Keep in mind that although ConTEXt MkIV does support the XETEX syntax too, the preferred way there is to use a more symbolic feature definition approach.

As this is an experimental setup, it might not always work out as expected. Around LuaTEX version 0.50 we expect the code to be more or less okay.

2 Implementation

The `luatex-fonts.lua` file is the first in a series of basic functionality enhancements for LuaTEX derived from the ConTEXt MkIV code base. Please don't pollute the `luatex-*` namespace with code not coming from the ConTEXt development team as we may add more files.

This file implements a basic font system for a bare LuaTEX system. By default LuaTEX only knows about the classic TFM fonts but it can read other font formats and pass them to Lua. With some glue code one can then construct a suitable TFM representation that LuaTEX can work with. For more advanced font support a bit more code is needed that needs to be hooked into the callback mechanism.

This file is currently rather simple: it just loads the Lua file with the same name. An example of a `luatex.tex` file that is just the plain TEX format:

```
\catcode`\{=1 % { is begin-group character
\catcode`\}=2 % } is end-group character
\input plain
\everyjob\expandafter{\the\everyjob
    \input luatex-fonts\relax}
\dump
```

We could load the Lua file in `\everyjob` but maybe some day we will need more here.

When defining a font, in addition to the XETEX way, you can use two prefixes. A `file:` prefix forces a file search, while a `name:` prefix will result in consulting the names database. The font definitions shown in figure 1 are all valid.

You can load maths fonts but as Plain TEX is set up for Computer Modern (and as we don't adapt Plain TEX) loading Cambria does not give you support for its math features automatically.

If you want access by name you need to generate a font database, using:

```
mtxrun --script font --names
```

and put the resulting file in a spot where LuaTEX can find it.

3 Remarks

The code loaded in `luatex-fonts.lua` does not come out of thin air, but is mostly shared with ConTEXt; however, in that macro package we go beyond what is provided in the plain variant. When using this code you need to keep a few things in mind:

- This subsystem will be extended, improved etc. at about the same pace as ConTEXt MkIV. However, because ConTEXt provides a rather high level of integration not all features will be supported in the same quality. Use ConTEXt if you want more goodies.
- There is no official API yet, which means that using functions implemented here is at your own risk, in the sense that names and namespaces

```
\font\testa=file:lmroman10-regular at 12pt
\font\testb=file:lmroman12-regular:+liga; at 24pt
\font\testc=file:lmroman12-regular:mode=node;+liga; at 24pt
\font\testd=name:lmroman10bold at 12pt
\font\testh=cmr10
\font\testi=ptmr8t
\font\teste=[lmroman12-regular]:+liga at 30pt
\font\testf=[lmroman12-regular] at 40pt
\font\testj=adobesongstd-light % cid font
\font\testk=cambria(math) {\mathtest 123}
\font\testl=file:IranNastaliq.ttf:mode=node;script=arab;\
    language=dflt;+calt;+ccmp;+init;+isol;+medi;+fina;+liga;\
    +rlig;+kern;+mark;+mkmk at 14pt
```

Figure 1: Font definition examples in LuaTEX

might change. There will be a minimal API defined once LuaTEX version 1.0 is out. Instead of patching the files it's better to overload functions if needed.

- The modules are not stripped too much, which makes it possible to benefit from improvements in the code that take place in the perspective of ConTEXt development. They might be split a bit more in due time so the baseline might become smaller.
- The code is maintained and tested by the ConTEXt development team. As such it might be better suited for this macro package and integration in other systems might demand some additional wrapping. The plain version discussed here is the benchmark and should be treated as a kind of black box.
- Problems can be reported to the team but as we use ConTEXt MkIV as our baseline, you'd better check if the problem is a general ConTEXt problem too.
- The more high level support for features that is provided in ConTEXt is not part of the code loaded here as it makes no sense elsewhere. Some experimental features are not part of this code either but some might show up later.
- Math font support will be added but only in its basic form once the Latin Modern and TEX Gyre math fonts are available. Currently traditional and OpenType math fonts can be loaded.
- At this moment the more nifty speedups are not enabled because they work in tandem with the alternative file handling that ConTEXt uses. Maybe around LuaTEX 1.0 we will bring some speedup into this code too (if it pays off at all).
- The code defines a few global tables. If this code is used in a larger perspective then you can best make sure that no conflicts occur. The ConTEXt package expects users to work in their own namespace (`userdata`, `thirddata`, `moduledata` or `document`). We give ourselves the freedom to use any table at the global level but will not use tables that are named after macro packages. Later, ConTEXt might operate in a more controlled namespace but it has a low priority.
- There is some tracing code present but this is not enabled and not supported as it integrates quite tightly into ConTEXt. In case of problems you can use ConTEXt for tracking down problems.
- Patching the original code in distributions is dangerous as it might fix your problem but introduce new ones for ConTEXt. So, best keep the original code as it is and overload functions and callbacks when needed. This is trivial in Lua.
- Attributes are (automatically) taken from the range 127–255 so you'd best not use these yourself. Don't count on an attribute number staying the same and don't mess with these attributes.

If this all sounds a bit strict, keep in mind that it makes no sense for us to maintain multiple code bases and we happen to use ConTEXt.

◇ Hans Hagen
Pragma ADE
The Netherlands
http://pragma-ade.org