The follow up

LuaMeta\TeX

Bacho\TeX May 2019

Hans & Alan
• This talk is not about how we can use Lua\TeX{} to control domotica applications. We already discussed that.

• This talk is not about how we can use Con\TeX{}t in advanced rendering, for instance as part of web-based workflows. That’s old news.

• This talk is not about how much fun it would be to have a Microsoft HoloLens and see what Con\TeX{}t and Lua\TeX{} could do with it. We just can’t afford it.

• This talk is not about more complexity, but it is about keeping things simple. It’s about turning a burden into a pleasure.

• To quote the Riverside\footnote{A Polish progrrock band I recently saw live in the Netherlands. A band related to Lunatic Soul.} frontman: I hope you all leave here a bit younger than you felt when you came here. This talk is about turning lead into gold.
• We have the weight and experience of about 40 years of \TeX{} and its usage on our shoulders.

• Good old \TeX{} got extended: \( \varepsilon \)-\TeX{}, Omega (Aleph), pdf\TeX{}, X\TeX{}, \([e][u][p]\)\TeX{} and \LaTeX{} (\& Luajit\TeX{}) showed up.

• The dvi output got complemented by pdf.

• Bitmap fonts were replaced by Type1 that itself got replaced by the container formats OpenType and TrueType. Variable fonts were introduced.

• Math got upgraded to OpenType math, thanks to Microsoft.

• Unicode got accepted and utf is nowadays the preferred input encoding.

• The community supported the development of many fonts that found their place in distributions.

• Alongside plain \TeX{} the macro packages \LaTeX{} and Con\TeX{}t both evolved into large collections of resources.
• There is no doubt that \TeX{} is a success. We can find nice examples but also some horrible looking documents.

• A large distribution is no guarantee for quality and continuous success, nor is the number of incidental (forced) users.

• A \texttt{Con\TeX}t user doesn’t need that much: just the \texttt{Lua\TeX} binary will do, plus a bunch of MkIV macros, completed by a reasonable set of fonts.

• Currently all that is embedded in a large ecosystem, although we always use only a small, reasonable subset.

• Getting the whole machinery up and running from scratch (source code) is not trivial.

• The source code base is rather large and compilation is complex: it builds on decades of being nice for all platforms and fulfilling all demands.

• What we consider gold could also be seen as lead in disguise. Some alchemy might be needed to go back to where we came from.
• At some point you need to stabilize and for Lua\TeX{}, version 1.10 is that moment.

• But some ideas and experiments have been delayed because the engine was already in use, also outside of Con\TeX{}t.

• Compatibility is a \textbf{big} issue in the Te\TeX{} community (which is good) so we’re in a sort of a deadlock (which is bad).

• And we wanted to take a next step in Con\TeX{}t development. It’s not strictly necessary to make drastic changes, but we need to adapt.

• The question is how we can guarantee users a long-term stability of the both macro package as well as the engine it runs on.
• For ConTExt we want just one binary. We don’t really need LuajitTEX for LuaJIT is not following Lua anyway.

• We don’t want (for windows) a special stub binary. After all we already have the context job manager and mtxrun script manager. All platforms should be treated alike.

• Performance should be stable and not influenced by code added to the binary. In fact, performance should constantly improve!

• The engine should not depend on libraries that are floating, get updated frequently, and can come from places out of our control (versions).

• The memory footprint should be acceptable for running in containers (or small virtual machines). Energy consumption matters too.

• The binary should be kept small because it also serves as the Lua interpreter.
• Around the ConTeXt meeting I took LuaTeX 1.0.0 experimental as starting point and began stripping.

• Before that, I already had written some test code to see what could be replaced.

• Stepwise redundant components were removed. This took time because each (small) step was tested on real documents, the test suite, etc.

• ... maybe some examples & /install-lmtx/* ...

• I played with some ideas that were put on hold, some were accepted and some were rejected and more and more got in the mood.

• Also LuaJIT was dropped, but its removal was compensated by large performance boosts in other areas.

• The build was simplified (it took some time to find what was irrelevant) and compilation now is about half a minute, or less!
• We have an experimental new installer for ConTExt lmtx (the new name). It uses http: and just the engine for fetching data. Updating goes fast.

• The lmtx distribution is MkIV only and much smaller than the full installation.

• Eventually (soon) the source code of the used engine will be in the distribution so that we have a self contained package. Users on new or unique systems can compile.

• The development of the engine is under control of the ConTExt developers: that way there is no danger that things break. We like to have a playground.

• Extensions can make it into LuaTEX once found useful and stable as long as they don’t break LuaTEX upward compatibility (unlikely on the short term).
• There is no backend code. We generate the pdf output in Lua (this was already the case for much of it.)

• There is no bitmap image inclusion code present. All is done in Lua.

• There is no font loading code present. This already happened mostly in Lua anyway.

• Some libraries have been removed and some have been simplified. A few experimental helper libraries were added (like math). The dependencies are minimal.

• The code is undergoing some restructuring but it might take some years before I’ve reached the (informal) goals.

• Alan and I are exploring new possibilities that this setup gives (especially in combining \TeX, MetaPost and Lua. Stay tuned.
• As of April 1, 2019, users can test the experimental distribution. A few were already in the loop.

• It looks like there are no big issues, and speed gains can be impressive.

• As a consequence we can start dropping in replacement code in regular MkIV some day soon too.

• Around the next ConTEXt meeting the source code will become part of the regular distribution (given that I’m satisfied with it).

• Before that we hope to have the compile farm up and running for LuaMetaTEX.

• From that moment on, the ConTEXt users will have a self contained, archival, independent, lean and mean installation available, which will become the default.

• Because LuaMetaTEX is a subset of LuaTEX, there are no plans right now for supporting plain TEX. We’ll see. (I might come up with generic backend code some day.)