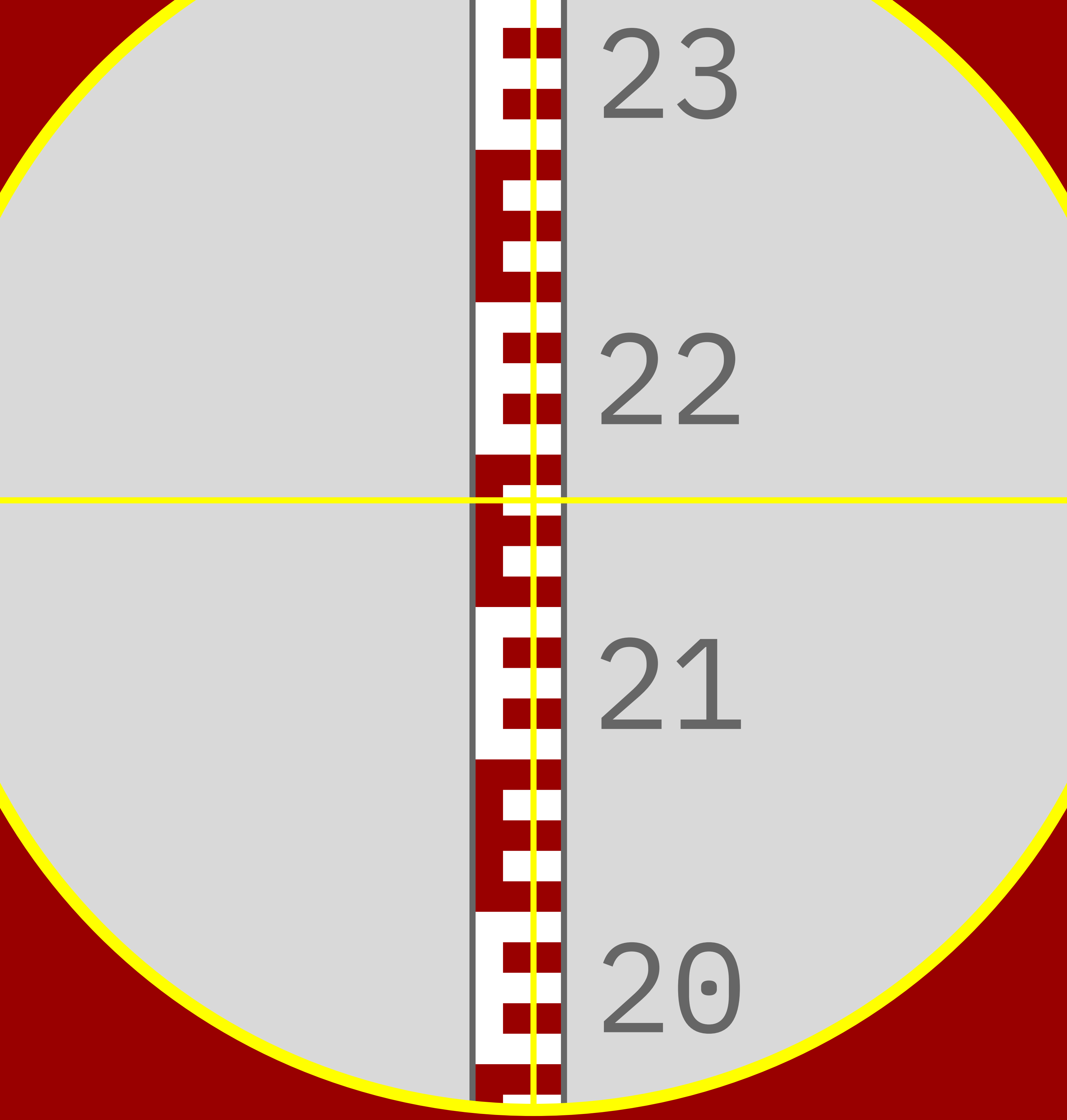

on target

luametatex & context lmtx

Table of contents

1 Introduction

This is the seventh wrapup of the LuaTEX and LuaMetaTEX development cycle. It is dedicated to all those users who kept up with developments and are always willing to test the new features. Without them a project like this would not be possible.

At the time this introduction is written the LuaMetaTEX code base is rather stable and quite a bit of the MkIV code base has been adapted to new situation. But, as usual, there are always new possibilities to explore, so I expect that this document will grow over time as did the others. I'm not going to repeat all that has been done because that's what the previous episodes are about.

As the title suggest, we're still on target. When the LuaMetaTEX project started there actually was no deadline formulated so in fact we're always on target. The core components TEX, MetaPost, and Lua are all long term efforts so we're in no hurry at all. However, this is the year that a fast pace will become a slow pace with respect to the LuaMetaTEX code base. There are still some things on the agenda but in principle the goals are reached. One problem in today's code development is that useability and quality seems to relate to the amount of changes in code. No update can mean old, unusable and uninteresting. It's probably why some sources get this silly yearly copyright year update. However, the update cycle of good old TEX has an decade interval by now while it is still a pretty useable program. It would be nice to end up in such a long term cycle with LuaMetaTEX: bug fixes only.

Although ConTEXt has always adapted early to new developments (color, graphics, pdf, MetaPost, ε-TEX, pdfTEX, LuaTEX, utf, fonts) the effects on the ConTEXt code base are mostly hidden for users. There have been some changes between MkII and MkIV, simply because there has been a shift from specific eight bit encodings to utf and Type1 to OpenType fonts. Both had an impact on important subsystems: input encodings, font definitions and features, language and script support. On other subsystems the impact was hardly noticeable, like for instance backend related features (these have always been kind of abstract). That doesn't mean that these haven't changed deep down, they definitely have. Some mechanisms became better in MkIV, simply because less hackery was needed. My experience is that when users see that it gets better or easier, they are also willing to adapt the few lines in their document source that benefit from it. Of course the impact on the MetaPost integration in ConTEXt had a real large impact, especially in terms of performance.

The upgrade to LMTX, the version of ConTEXt for LuaMetaTEX, is even less visible although already some new mechanisms showed up. This time a couple of engine specific features have been improved and made more flexible. In fact, the whole code base of the engine has been overhauled. This happened stepwise because we had to make sure all things kept working. As a first step code was made independent of the compilation infrastructure and the dependencies, other than a very few small ones, have been removed. The result is a rather lean and mean setup, even when we consider what has been added at the primitive level and traditional subsystems. A benefit is that in the meantime the LuaMetaTEX LMTX combination outperforms LuaTEX with MkIV, something that was not ensured when the built-in pdf backend was removed and delegated to Lua. By binding development closely to ConTEXt we also hope that the code base stays clean of arbitrary extensions.

Because in the end, TEX is also a programming language, there have been extensions that make programming easier. There is already a stable middle layer of auxiliary macros in ConTEXt that help the user who likes to program but doesn't like real low level primitives and dirty tricks, but by extending the primitive repertoire a bit users can now stay closer to the original TEX concepts. Adding more and more layers of indirectness makes no sense if we can improve the bottom programming layer. It also makes coding a bit more natural (the TEX look), apart from offering performance benefits. This is where you can see differences between the MkIV and LMTX code base which for that reason is now nearly split completely. The

MetaPost subsystem has been extended with proper scanners so that we can enhance the interfaces in a natural way and as a result we also have an upgraded code base there. We also moved to Lua5.4 and will keep up as long as compatibility is no issue. Some Lua code is likely to remain common between MkIV and LMTX, for instance font handling and helpers but we'll see where that ends.

The LuaMetaTEX engine provides control over most internals and there are all kind of new interesting features. Decades of ConTEXt development are behind that. Also, in the days that there were discussions about extending TEX, ConTEXt was not that much of influence and on the road to and from user group meetings, Taco and I often discussed what we'd like to see added (and some was actually implemented in `eetex` but that only lived on our machines. One can consider LuaTEX to be a follow up on that, and LuaMetaTEX in turn follows up on that project, which we both liked doing a lot. In some way LuaTEX lowered the boundary for implementing some of the more intrusive extensions in LuaMetaTEX and the follow up on mplib. And once you start along that road small steps become large steps and one can as well be try to be as complete as possible. We've come a long way but eventually arrived at the destination. Personally I think we got there by not being in a hurry.

But even targets that are reached can eventually move,

Hans Hagen
Hasselt NL
August 2021^{++}