MATH

complete control

context 2021 meeting

The benchmark

- Traditional TEX has set the benchmark for typesetting math.
- It uses three categories of fonts: alphabet, symbol, extensible.
- Limitations in fonts have lead to some curious handling of dimensions.
- The fact that there is `\over` makes for multipass scanning and processing.
- The last family counts (when entering the second pass) so one has to use very controlled font switching.
- Some features (limits, integrals, primes) rely on special macros and parsing.
- One can end up in tricky font switching. Also, quite some fonts are loaded in order to set up the machinery.

Todays reality

- We now use OpenType math fonts. In fact, in MkIV we always used only one font instance (unless we mix font families).
- We operate in the Unicode domain, so in the end there was no need to bump the number of families in LuaTEX.
- In ConTEXt we use only a few font families: regular, regular r2l, bold, bold r2l.
- All the tricky stuff is done with the help of Lua. This already started very early in the LuaTEX project and has not changed.
- But we still run into issues because the available fonts are inconsistent, incompatible, have issues and that is unlikely to change.
- We have a virtual font system in place that was used during the transition (when no fonts were available).
- In ConTEXt there are various ways to deal with shortcomings in or extensions to fonts.
- At some point we need to make up our minds, accept the issues with fonts, and just fix things runtime.

The engine

- In LuaTEX we started with a hybrid approach but eventually ended up with more split code paths.
- In LuaMetaTEX nearly all aspects of the engine had made configurable and are under user control.
- This permits experiments where we can apply old methods onto new fonts.

The tricky things

- Spacing in traditional math is a combination of widths and italic correction: we need to add them and sometimes afterwards subtract the italic correction. In OpenType math we don't lie about dimensions and apply italic correction selectively (as we have staircase kerns).
- Special symbols like primes are quite inconsistent wrt dimensions and positioning and we need to catch that. We also need robust ways to collapse them to the proper Unicode symbol.
- Larger variants (these `\bigg` things) needs some attention too.
- We need to map characters onto the right shaped (alphabets) because hardly anyone will enter the Unicode math characters directly.
- We might want to fix scripts, italics, kern pairs etc.

Implementation

Among the new features (introduced over a period of time) are:

- All kind of inter-class spacing parameters (in addition to what LuaTEX already provides).
- Dozens of OpenType font related spacing parameters (more that traditional TEX).
- Opened up additional font (and taste) related parameters (hard coded in traditional TEX).
- Opened up style related parameters (hard coded in traditional TEX).
- Control codes that make the engine follow a different (the traditional) code path (which can come in handy when testing or writing manuals).
- The ability to simplify the result a bit wrt characters (the engine loves to box a lot).
- Efficient font, style and parameter scaling by reusing fonts and thereby limiting the number of instances.
- More options to special (constructed) symbols.
- Let math related nodes carry around more control properties and states.
- Support of local changes to math parameters.