COMPACT FONTS

a different approach to scaling

context 2021 meeting

How TEX uses fonts

- In a TEX engine there is a 1-to-1 mapping from characters in the input to a slot in a font.
- Combinations of characters can be mapped onto one shape: ligatures.
- Hyphenation is (also) controlled by the input encoding of a font which imposes some limitations.
- In the typeset result characters (glyphs) can be (re)positioned relatively: kerns.
- This all happens in a very interwoven way (e.g. inside the par builder).

How traditional TEX sees fonts

- A font is just a (binary) blob of data with information where characters have a width, height, depth and italic correction.
- The engine only needs the dimensions and when the same font is used with a different scale a copy with scaled dimensions is used.
- There can be recipes for ligatures and (more complex) so called math extensible shapes (like arrows, wide accents, fences and radicals)
- There are pairwise specification for kerning.
- A handful of font parameters is provided in order to control for instance spacing.
- Virtual fonts are just normal fonts but they have recipes for the backend to construct shapes: TEX uses the normal metric file, the backend also the virtual specification file.

An example of a definition and usage:

```
1 \font\cs somename [somescale]  test {\cs test} test
```

Unicode engines

- There are no fundamental differences between a traditional engine and an Unicode aware one with respect to what TEX needs from fonts: dimensions.
- There can be more than 256 characters in a font.
- Some mechanisms have to be present for applying font features (like ligaturing and kerning).
- In LuaTEX hyphenation, ligaturing and kerning are clearly separate steps.
- In LuaTEX callbacks can do lots of things with fonts and characters.
- Math fonts are handled differently and there is more info that comes from the font.
- In LuaTEX features like expansion are implemented differently, i.e. no copies of fonts are made and applied expansion travels with the glyphs.

Overhead

- In all engines scales copies are used. For traditional TEX a copy is cheap, but a font that supports more of Unicode cna be quite large.
- Different feature sets in principle make for a different font (this can be different per macro package).
- Every math scale needs three font instances: text, script and scriptscript.
- In ConTEXt lots of sharing takes place and a lot of low level optimizations happens (memory, performance). We always made sure (already in MkII) that the font system was not the bottleneck.
- In most cases font definitions are dynamic, and we delay initialization as much as possible.
- In MkIV font features can be dynamic which saves a lot of memory at the cost of some extra processing overhead.

Design sizes

- A macro package's font handling is complicated by design sizes. The low level code can look complex too.
- There are hardly any fonts that come in design sizes (basically only Computer Modern) so we need to support that.
- Design sizes make sense for math fonts where very small characters and symbols are used in scripts.
- In OpenType fonts smaller math design sizes are part of the font (that then gets loaded three times with different `ssty` settings).

2 2 2 2 2 2  2 2 2

The system

- Each bodyfont size (often a few are defined) has regular, bold, italic, bold italic, slanted, bold slanted, etc. There can be unscaled instances (design size) or scaled ones.
- Within a bodyfont size we have size variants: smaller x and xx, larger a, b, c and d.
- In MkII we inherit smallcap and oldstyle setups but in MkIV one can either do that dynamic or define an additional bodyfont instance.
- A document can use a mix of fonts mixed setup, so there is a namespace used per family.
- In OpenType symbols are more naturally part of a font, as are emoji and colored shapes.
- Lack of variants can result in artificially slanted of boldened fonts (more advanced in MkIV). Variable fonts are not special, but demand some work when loading and embedding.
- Runtime virtual fonts are part of MkIV as are fallback shapes.
- Languages and scripts can introduce an extra dimension to operate on.
- Math is greatly simplified by the fact that families are part of a font.
- Bold math fonts are provided by the boldening feature.

Practice

- Due to optimizations the actual number of loaded instances is normally small but there are exceptions.
- The LuaMetaTEX manual has 158 instances most of which are math (six per used bodyfont for math plus a regular: Lucida, Pagella, Latin Modern, Dejavu, Cambria).
- The final pdf file has 21 embedded fonts (subsets).
- The fact that different sizes each have an instance and that for math we need three per size plus one for r2l, while the only difference is scale, makes one think of other solutions.
- Using duplicates of huge cjk fonts makes it even more noticeable.

But there is a way out!

Glyph scaling

We can scale glyphs in three ways:

```
\definedfont[lmroman10-regular*default]%
test {\glyphxscale 2500 test} test
test {\glyphyscale 2500 test} test
test {\glyphscale  2500 test} test
```

test **test** test test *test* test test **test** test

We do need to honor grouping:

```
\definedfont[lmroman10-regular*default]%
e{\glyphxscale 2500 ff}icient
ef{\glyphxscale 2500 f}icient
ef{\glyphxscale 2500 fi}cient
e{\glyphxscale 2500 ffi}cient
```

e**ff**icient ef**f**icient ef**fi**cient e**ffi**cient

These scales can also be part of a font definition so we can do with less instances.

Implications

- The text processing part of the engine has to scale on the fly (e.g. when dimensions are needed in paragraph building and (box) packing.
- Font processing (features) already distinguishes per instance but now also has to differentiate by (upto three) scales (we use the same font id for scaled instances).
- The math machinery has to check for smaller sizes and also scale dimensions on the fly.
- For math that means two times scaling: the main scale (like glyph scale) plus the relative scaling of script and scriptscript.
- When they are used font dimensions and math parameters are to be scaled as are rules in some math extensibles.
- This all has to be done efficiently so that there is no significant drop in performance.
- But quite a bit less memory is used and loading time has become less too.

Some day this will become default (which means a sentimental process of dropping ancient code):

```
\enableexperiments[fonts.compact]
```

Details

The tricks shown on this page and following pages have been in place and tested for a while now. Because TEX uses integers a scale value of 1000 means 1.000, as in other mechanisms:

```
\definedfont[lmroman10-regular*default]%
test {\glyphscale 2500 test} test
```

test test test

The font definition mechanism supports horizontal and vertical scaling:

```
\definefont[FooA][Serif*default @ 12pt 1800 500]
\definefont[FooB][Serif*default @ 12pt 0.85 0.4]
\definefont[FooC][Serif*default @ 12pt]

There is also a new command:

\definetweakedfont[runwider] [xscale=1.5]
\definetweakedfont[runtaller][yscale=2.5,xscale=.8,yoffset=-.2ex]

{\FooA test test \runwider test test \runtaller test test}\par
{\FooB test test \runwider test test \runtaller test test}\par
{\FooC test test \runwider test test \runtaller test test}\par
```

There is also a new command:

test test **test test** test test

test test **test test** test test

test test **test test** test test

Tweaking also works in math:

```
\definetweakedfont[squeezed][xscale=0.9]
\definetweakedfont[squoozed][xscale=0.7]

\startlines
$a = b^2 + \sqrt{c}$
{\squeezed $a = b^2 + \sqrt{c}$}
{\squoozed $a = b^2 + \sqrt{c}$}
{$\squoozed a = b^2 + \sqrt{c}$}
{$\scriptstyle a = b^2 + \sqrt{c}$}
\stoplines
```

$a = b^2 + \sqrt{c}$
$a = b^2 + \sqrt{c}$
$a = b^2 + \sqrt{c}$
$a = b^2 + \sqrt{c}$
$a=b^2+\sqrt{c}$

```
\startcombination[3*1]
    {\bTABLE
        \bTR \bTD foo \eTD \bTD[style=\squeezed] $x = 1$ \eTD \eTR
        \bTR \bTD oof \eTD \bTD[style=\squeezed] $x = 2$ \eTD \eTR
     \eTABLE} {local}
    {\bTABLE[style=\squeezed]
        \bTR \bTD $x = 1$ \eTD \bTD  $x = 3$ \eTD \eTR
        \bTR \bTD $x = 2$ \eTD \bTD  $x = 4$ \eTD \eTR
     \eTABLE} {global}
    {\bTABLE[style=\squeezed\squeezed\squeezed\squeezed]
        \bTR \bTD $x = 1$ \eTD \bTD  $x = 3$ \eTD \eTR
        \bTR \bTD $x = 2$ \eTD \bTD  $x = 4$ \eTD \eTR
     \eTABLE} {multiple}
\stopcombination
```

foo	$x = 1$
oof	$x = 2$

local

$x = 1$	$x = 3$
$x = 2$	$x = 4$

global

$x = 1$	$x = 3$
$x = 2$	$x = 4$

multiple

Tweaking can be combined with styles:

```
\definetweakedfont[MyLargerFontA][scale=2000,style=bold]
test {\MyLargerFontA test} test
```

test **test** test

We can use negative scale values:

```
\bTABLE[align=middle]
  \bTR
    \bTD a{\glyphxscale  1000 \glyphyscale  1000 bc}d \eTD
    \bTD a{\glyphxscale  1000 \glyphyscale -1000 bc}d \eTD
    \bTD a{\glyphxscale -1000 \glyphyscale -1000 bc}d \eTD
    \bTD a{\glyphxscale -1000 \glyphyscale  1000 bc}d \eTD
  \eTR
  \bTR
    \bTD \tttf +1000 +1000 \eTD
    \bTD \tttf +1000 -1000 \eTD
    \bTD \tttf -1000 -1000 \eTD
    \bTD \tttf -1000 +1000 \eTD
  \eTR
\eTABLE
```

abcd	a pc d	da sq	dd
+1000 +1000	+1000 -1000	-1000 -1000	-1000 +1000

There are more glyph properties:

```
\ruledhbox{
    \ruledhbox{\glyph yoffset 1ex options 0 123} % left curly brace
    \ruledhbox{\glyph xoffset .5em yoffset 1ex options "C0 123}
    \ruledhbox{oeps{\glyphyoffset 1ex \glyphxscale 800
                    \glyphyscale\glyphxscale oeps}oeps}
}
```

Glyph scales are internal counter values, like any other:

```
\samplefile{jojomayer}
{\glyphyoffset .8ex
 \glyphxscale 700 \glyphyscale\glyphxscale
 \samplefile{jojomayer}}
{\glyphyscale\numexpr3*\glyphxscale/2\relax
 \samplefile{jojomayer}}
{\glyphyoffset -.2ex
 \glyphxscale 500 \glyphyscale\glyphxscale
 \samplefile{jojomayer}}
\samplefile{jojomayer}
```

If we surrender the thing that separates us from machines, we will be replaced by machines. The more advanced machines will be, the more human we will have to become. If we surrender the thing that separates us from machines, we will be replaced by machines. The more advanced machines will be, the more human we will have to become.

If we surrender the thing that separates us from machines, we will be replaced by machines. The more advanced machines will be, the more human we will have to become. If we surrender the thing that separates us from machines, we will be replaced by machines. The more advanced machines will be, the more human we will have to become. If we surrender the thing that separates us from machines, we will be replaced by machines. The more advanced machines will be, the more human we will have to become.

There is a helper for real (float) scales:

```
1200 : \the\numericscale1200
1.20 : \the\numericscale1.200
```

These lines produce the same integer:

```
1200 : 1200
1.20 : 1200
```

You can do this but it's not always predictable (depends on outer scales too):

```
\def\UnKernedTeX
  {T%
   {\glyph xoffset -.2ex yoffset -.4ex `E}%
   {\glyph xoffset -.4ex options "60 `X}}

We use \UnKernedTeX\ and {\bf \UnKernedTeX} and {\bs \UnKernedTeX}:
the slanted version could use some more left shifting of the E.
```

We use TEX and **TEX** and *TEX*: the slanted version could use some more left shifting of the E.

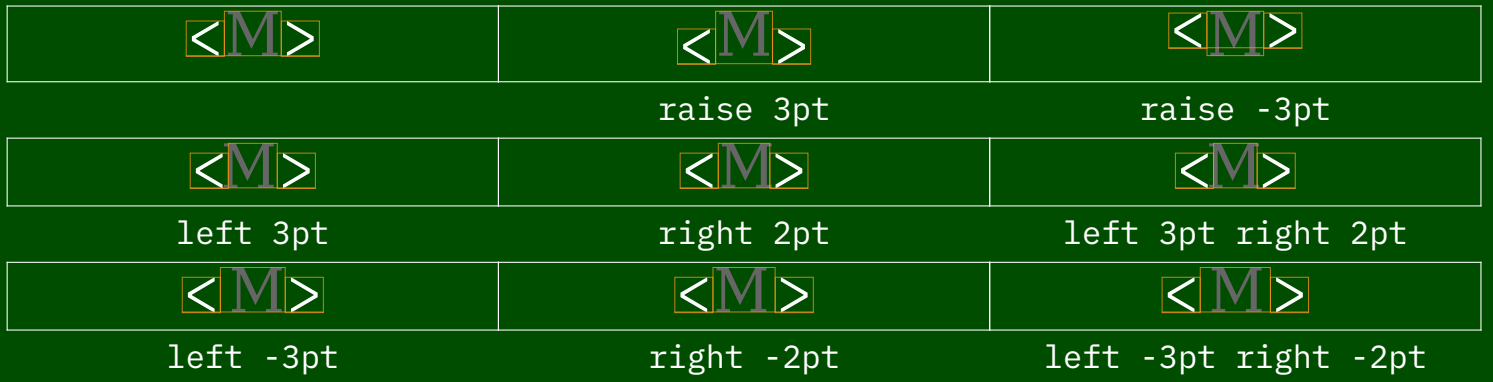
Marks and cursive features can interfere, so this is an alternative:

```
\def\UnKernedTeX
  {T\glyph left .2ex raise -.4ex `E\glyph left .2ex `X\relax}
```

We use TEX and **TEX** and *TEX*: the slanted version could use some more left shifting of the E.

Yet another glyph option:

```
\multiply\glyphscale by 2 <{\darkgray \glyph                       `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph raise 3pt             `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph raise -3pt            `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph left  3pt             `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph           right  2pt `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph left  3pt right  2pt `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph left -3pt             `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph           right -2pt `M}>
\multiply\glyphscale by 2 <{\darkgray \glyph left -3pt right -2pt `M}>
```



A larger example:

```
\glyphscale 4000
\vl\glyph                          `M\vl\quad
\vl\glyph raise  .2em              `M\vl\quad
\vl\glyph left   .3em              `M\vl\quad
\vl\glyph               right  .2em`M\vl\quad
\vl\glyph left  -.2em right -.2em`M\vl\quad
\vl\glyph raise -.2em right  .4em`M\vl
```