Dice3D OpenType

(quick font hack two)

Abstract
The previous article by Yuri Robbers shows simulated 3D dice. That font existed only as a MetaFont source file, so for the ConTEXt-format Maps article, I had to quickly create an OpenType version.

Using MetaPost instead of MetaFont

The font comes as a single file called `dice3d.mf`, which contains all needed definitions. It is intended for the `plain.mf` macros with no extra macro packages needed. This makes it possible to run the file directly in MetaPost, which a slightly complicated command line:

```
mpost --mem=mfplain
  -s outputtemplate='"%j-%c.eps"'
  '\\mag=36; mode=lowres; input dice3d.mf'
```

the bits and pieces:

`--mem=mfplain`
: This preloads the `mfplain.mp` file, which mimics the MetaFont `plain` macros.

`-s outputtemplate='"%j-%c.eps"'`
: This make MetaPost produce eps files with nice names like `dice3d-99.eps` instead of `dice3d.99`.

`'\mag=36; mode=lowres; input dice3d.mf'`
: For font-making, it is necessary to set up the MetaFont 'mode' and magnification. This magnification setting ensures that the output images are such that they fit the typical 1000-units-per-em for a PostScript-type font. The mode is a predefined mode in `mfplain` that distorts the image as little as possible while still being a font-making mode (as opposed to 'proof' mode).

  This part of the command line is enclosed in quotes and starts with a backslash so that we can set these parameters without having to alter the actual font source. The backslash prevents the assumed `input` command at the start of the line. The internal `jobname` is then set by the next actual `input` command.

After running the command, there are 30 eps files.

Here is one as an example:

A bit of FontForge

Those images are then imported into FontForge in their respective slots. Once all 30 are imported, they are tweaked a little as usual:

- ☐ All the points are rounded to integers
- ☐ All overlap is removed
- ☐ All right side-bearings are set to be equal to the left side-bearings.

All those actions are single commands after selecting all font glyphs, which makes this process really fast.

Then, as this font as some ligatures, a `liga` table needs to be created. I did this manually for this font because it is so simple (the ligature table is listed at the end of `dice3d.mf`):

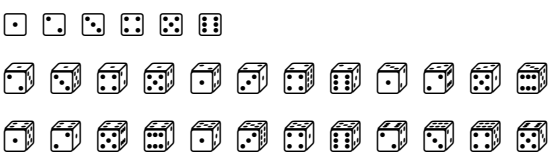
```
%   ligature tables for 3D dice:
%   #a, #b, #c, #d, where # is the value on the top face,
%                   and the letter indicates the value on
%                   the front face: "a" -> smallest,
%                                   "d" -> largest
ligtable "1": "a" =: "a", "b" =: "b", "c" =: "c", "d" =: "d";
ligtable "2": "a" =: "e", "b" =: "f", "c" =: "g", "d" =: "h";
ligtable "3": "a" =: "i", "b" =: "j", "c" =: "k", "d" =: "l";
ligtable "4": "a" =: "m", "b" =: "n", "c" =: "o", "d" =: "p";
ligtable "5": "a" =: "q", "b" =: "r", "c" =: "s", "d" =: "t";
ligtable "6": "a" =: "u", "b" =: "v", "c" =: "w", "d" =: "x";
```

Finally, I adjusted the font name and generated an OpenType version.

Usage

Using the OpenType version is quite straightforward:

```
\font\contextdice=dice3d*default
{\contextdice 1 2 3 4 5 6 \crlf
1a 1b 1c 1d 2a 2b 2c 2d 3a 3b 3c 3d \crlf
4a 4b 4c 4d 5a 5b 5c 5d 6a 6b 6c 6d}
```



Taco Hoekwater