Punk from Metafont to MetaPost

Abstract
To make Knuth's punk font usable with ConTEXt MKIV, it had to be converted from Metafont to MetaPost input. This article highlights the most important changes that had to be made in the conversion process.

Introduction

Donald Knuth's punk font is available from CTAN and in most TEX distributions, such as TEXLive. The TEXLive description has this to say about it:

> "A response to the assertion in a lecture that 'typography tends to lag behind other stylistic changes by about 10 years'. Knuth felt it was (in 1988) time to design a replacement for his designs of the 1970s, and came up with this font! The fonts are distributed as Metafont source. The package offers LaTEX support by Rohit Grover, from an original by Sebastian Rahtz, which is slightly odd in claiming that the fonts are T1-encoded. A (possibly) more rational support package is to be found in punk-latex."

Elsewhere in this Maps 37, you can read about the rather special characteristics of the punk font, and about the stepts needed to make it usable in the latest version of mplib–enabled ConTEXt. In an effort to reduce the overall noise level on these pages, the current article will not show you what the font looks like at all. There is enough of that in the two other articles.

As said already, the original font is based on Metafont. For use with mplib, we wanted a version that could be processed repeatedly by a single mplib instance. A bit of reorganisation was needed.

Punk in Metafont

The original distribution contains about a dozen Metafont input files. The content of the Metafont files is explained below, but we were only interested in the 10 point upright font, so we will ignore files like `punksl20.mf` (that generates a 20 point slanted version of the font).

punk10.mf

This is the parameter file for the 10 point font. It contains ten parameter assignments and then inputs the `punk.mf` file.

```
% 10-point PUNK font
designsize:=10pt#; font_identifier:="PUNK";
ht#:=7pt#;      % height of characters
u#:=1/4pt#;     % unit width
s#:=1.2pt#;     % extra sidebar
px#:=.6pt#;     % horizontal thickness of pen
py#:=.5pt#;     % vertical thickness of pen
dot#:=1.3pt#;   % diameter of dots
dev#:=.3pt#;    % standard deviation of punk
                % points
slant:=0;       % obliqueness
seed:=sqrt2;    % seed for random number
                % generator
input punk
bye
```

punk.mf

This is a typical Metafont macro file. It defines a few macros and sets up various drawing parameters for the characters.

```
% Font inspired by Gerard and Marjan Unger's
% lectures, Feb 1985
mode_setup;

randomseed:=seed;

define_pixels(u,dev);
define_blacker_pixels(px,py,dot);
define_whole_pixels(s);
xoffset:=s;
pickup pencircle xscaled px yscaled py;
punk_pen:=savepen;
pickup pencircle scaled dot; def_pen_path_;
path dot_pen_path;
dot_pen_path:=currentpen_path;
currenttransform:=identity slanted slant
                  yscaled aspect_ratio;

def beginpunkchar(expr c,n,h,v) =
  % code $c$; width is $n$ units
  hdev:=h*dev; vdev:=v*dev;
  % modify horizontal and
  %   vertical amounts of deviation
  beginchar(c,n*u#,ht#,0);
  italcorr ht#*slant;
```

```
  pickup punk_pen enddef;
  extra_endchar:=extra_endchar
      & "w:=w+2s;charwd:=charwd+2s#";

def ^ = transformed currenttransform enddef;

def makebox(text rule) =
 for y=0,h:
   rule((-s,y)^,(w-s,y)^); % horizontals
 endfor
 for x=-s,0,w-2s,w-s:
   rule((x,0)^,(x,h)^);  % verticals
 endfor
 enddef;
rulepen:=pensquare;

vardef pp expr z =
  z+(hdev*normaldeviate,vdev*normaldeviate)
enddef;

def pd expr z =   % {\bf drawdot}
  addto_currentpicture contour
    dot_pen_path shifted z.t_
        withpen penspeck
enddef;

input punkl % uppercase letters
input punkae % uppercase \AE, \OE, \O
input punkg % uppercase greek
input punkp % punctuation
input punkd % digits
input punka % accents

ht#:=.6ht#; dev:=.7dev;
input punksl % special lowercase
extra_beginchar:=extra_beginchar
  & "charcode:=charcode+32;";
input punkl % lowercase letters
extra_beginchar:=extra_beginchar
  & "charcode:=charcode-35;";
input punkae % lowercase \ae, \oe, \o

font_slant:=slant;
font_quad:=18u#+2s#;
font_normal_space:=9u#+2s#;
font_normal_stretch:=6u#;
font_normal_shrink:=4u#;
font_x_height:=ht#;
font_coding_scheme:=
  "TeX text without f-ligatures";
bye
```

Note that `punkl.mf` and `punkae.mf` are loaded twice, after some redefinitions have taken place. The combined effect of

```
ht#:=.6ht#; dev:=.7dev;
```

and

```
extra_beginchar:=extra_beginchar
 & "charcode:=charcode+32;";
```

is that the drawing routines for the uppercase characters (like 'P', with character code 80) are reused for the lowercase characters (like 'p', with character code 112). The heights and widths are diminished, and this makes punk a 'Caps and Small Caps' font.

punkl.mf, punkae.mf, punkg.mf, punkp.mf, punkd.mf, punka.mf, punksl.mf
These contain the drawing routines for the characters and a few `ligtable` commands for the standard tex ligatures like `--` and `''`. There is not that much to see, just a bunch of definitions like this:

```
beginpunkchar("P",13,1,2);
z1=pp(2u,0); z2=pp(2u,1.1h);
z3=pp(2u,.5h); z4=pp(w,.6[y3,y2]);
pd z1; pd z3;
draw z1--z2--z4--z3; % stem and bowl
endchar;
```

Punk in MetaPost

In the MetaPost version, we wanted to have only one file because that makes handling the font a bit easier. The file's name is `punkfont.mp`, and even though there is only one file now, the initial setup is much the same.

It begins with parameter settings, like this:

```
if unknown punk_font_loaded :

    if unknown scale_factor :
        scale_factor := 1 ;
    fi ;

    boolean punk_font_loaded ;

    punk_font_loaded := true ;
    warningcheck     := 0 ;
    designsize       := 10pt#;
    font_identifier  := "Punk Nova" ;

    ht#  :=  7pt# ; % height of characters
    u#   :=1/4pt# ; % unit width
    s#   := 0     ; % extra sidebar
    px#  := .6pt# ; % horizontal pen thickness
    py#  := .5pt# ; % vertical pen thickness
```

```
    dot# :=1.3pt# ; % diameter of dots
    dev# := .3pt# ; % standard deviation of
                    % punk points

  % seed          := sqrt2 ;
```

Most if the changes above should be self–explanatory. The only things worth noting are the test and setting of the `punk_font_loaded` boolean (this prevents errors when the file is being read multiple times) and the commented out definition of `seed`. That latter change is because we wanted the font to be truly random. Knuth's original only appears to be random. In fact it always has the exact same 'randomness'.

The next bit contains the assignments and macro definitions, much like `punk.mf`:

```
proofing     := 0 ;
pt           := .1pt ;
mag          := scale_factor * 10 ;
bp_per_pixel := bpppix_ * mag ;
```

MetaPost's `mfplain` doesn't have the `mode_setup` macro, so the important settings from that are given explicitly. The trickery with `scale_factor` and `pt` is just so the resulting figures will have a usable range (in PostScript big points).

Going on:

```
define_pixels(u,dev) ;
define_blacker_pixels(px,py,dot) ;
define_whole_pixels(s) ;
xoffset := s ;

pickup pencircle xscaled px yscaled py ;
punk_pen := savepen ;
pickup pencircle scaled dot ;
path dot_pen_path ;
dot_pen_path :=tensepath makepath currentpen;

defaultcolormodel := 1 ;

def beginpunkchar(expr c,n,h,v) =
  % code $c$; width is $n$ units
  hdev := h * dev ;
  % modify horizontal amounts of deviation
  vdev := v * dev ;
  % modify vertical amounts of deviation
  beginchar(c,n*u#,ht#,0) ;
  italcorr 0 ;
  pickup punk_pen
enddef ;
```

```
extra_endchar := extra_endchar
  & "w := w+2s ; charwd := charwd+2s# ;";

extra_endchar := extra_endchar
  & "setbounds currentpicture to (0,-d)"
  & "--(w*1.2,-d)--(w*1.2,ht#)--(0,ht#)"
  & "--cycle;";

def ^ = transformed currenttransform enddef ;

def makebox(text rule) =
    for y=0, h : % horizontals
        rule((-s,y)^,(w-s,y)^) ;
    endfor
    for x=-s, 0, w-2s, w-s : % verticals
        rule((x,0)^,(x,h)^) ;
    endfor
enddef ;

rulepen := pensquare ;

vardef pp expr z =
    z + (hdev * normaldeviate,
         vdev * normaldeviate)
enddef;

def pd expr z = % {\bf drawdot}
    addto currentpicture
       contour dot_pen_path
       shifted z.t_ withpen penspeck
enddef;
```

This is all pretty much the same as in Metafont. The trick with the multiple loading doesn't work because there is only the one file, but we did not want to manually adjust the drawing macros within de `beginpunkchar` commands. That is why the following definitions were added:

```
   def initialize_punk_upper =
       ht# := 7pt# ; dev# := .3pt# ;
   enddef ;
   def initialize_punk_lower =
       sht# := ht#; sdev := dev;
       ht# := .6ht# ; dev := .7dev ;
   enddef ;
   def revert_punk_lower =
       ht# := sht#; dev := sdev;
   enddef ;

fi ;
```

The `fi` ends the boolean test that was started at the top of the file, everything below this point can be safely re-interpreted.

The rest of the file consists of a few calls to these three macros and a whole bunch of character definitions. It starts like this:

```
initialize_punk_upper ;

beginpunkchar("A",13,1,2);
  z1=pp(1.5u,0); z2=(.5w,1.1h);
  z3=pp(w-1.5u,0);
  pd z1; pd z3;
  draw z1--z2--z3; % left and right diagonals
  z4=pp .3[z1,z2];
  z5=pp .3[z3,z2];
  pd z4; pd z5;
  draw z4--z5; % crossbar
endchar;
```

The MetaPost version of the font does not have any `ligtable` commands; the ligatures are automatically generated by ConTEXt. This is possible because the loaded font uses an (incomplete) Unicode encoding.

For example, we have:

```
beginpunkchar(8221,9,.3,.5);
  % '' quotedblright
  z1=pp(.5w-.5u,h); z2=pp(u,.6h);
  z3=pp(w-u,.95h); pd z1; pd z3;
  draw z1--z2--z3;  % stroke
endchar;
```

Incidentally, this is why the `warningcheck:=0;` above was needed. Without it, MetaPost would have complained about `Number too large`.

The last thing worth mentioning is that the original font was using 7-bit TEX roman encoding, which doesn't have a full ASCII set. We have added the missing definitions: underscore, caret, left brace, right brace, backslash, and the straight quote and double quote.

Taco Hoekwater & Hans Hagen