Type1 Schriften verstehen – for fun and for profit

André Wobst

35 MV DANTE e.V.

14. September 2006

Rosenheim

Inhalt

Im folgenden geht es um technische Details zu Schriften.
„Wie ist festgelegt, wie ein Buchstabe aussieht?“

- Schrifttypen
- Type1 Schriften (pfb-Dateien + Metrik-Dateien)
- Outline (Umriß/Kontur) eines Glyphen
- Dekodieren versteckter Nachrichten
- Modifikation einzelner Glyphen
- Pfad-Transformationen
- Fragen

Schrifttypen

Übliche Schrifttypen in der TEX-Welt:

- METAFONT-Schriften (zusammen mit TEX von Knuth entwickelt)
- Type1 Schriften (Adobe Outline-Format von PostScript)
- TrueType Schriften (Apple/Microsoft Outline-Format)

Besonderheiten von METAFONT-Schriften:

- METAFONT ist eine Programmiersprache für Schriften
- METAFONT erzeugt gerasterte Beschreibung (pk-Dateien) für spezifische Ausgabegeräte (Auflösung und andere Geräteeigenschaften)
- keine mathematische Beschreibung der Outline der Glyphen

Schriften in TEX

- TEX weiß/versteht fast nichts von Schriften
- TEX kennt nur Metriken (einschließlich Ligaturen und Kerning) der Schriften (eigenes Format: tfm-Dateien)
- dvi-Dateien enthalten nur Schriftnamen und Positionsangaben für die einzelnen Glyphen
- ursprüngliche Idee von dvi-Treibern: gerasterte Beschreibung der kompletten Seite generieren
- Besonderheiten: kein encoding, „virtuelle Schriften“

heutzutage werden üblicherweise **keine** gerasterten Ausgaben mehr erzeugt, da Computer Modern Schriften als Type1 Schriften verfügbar sind und diese auflösungsunabhängig sind

Type1 Schriften

Eine Type1 Schrift besteht aus zwei Dateien:

afm-Datei: Metrik-Information und andere Metadaten (z.B. Ligaturen, Kerning) → Klartext

pfb-Datei: Glyph-Information → verschlüsselt (Verschlüsslung mittlerweile offengelegt)

Zur Benutzung in TEX:

- tfm-Datei notwendig (ggf. aus afm-Datei erzeugen)
- Unterstützung im Backend (also dvips, pdfTEX etc.; Zuordnung durch psfonts.map, pdftex.map)

Dokumentation von Adobe: t1_spec.pdf

Schrift zerlegen

Warum?

for fun: Wie ist die Outline kodiert?

for profit: Nur benutzte Glyphen in die Ausgabedatei einbetten!

Wo? (Bzw. worüber kann der Vortragende berichten!)

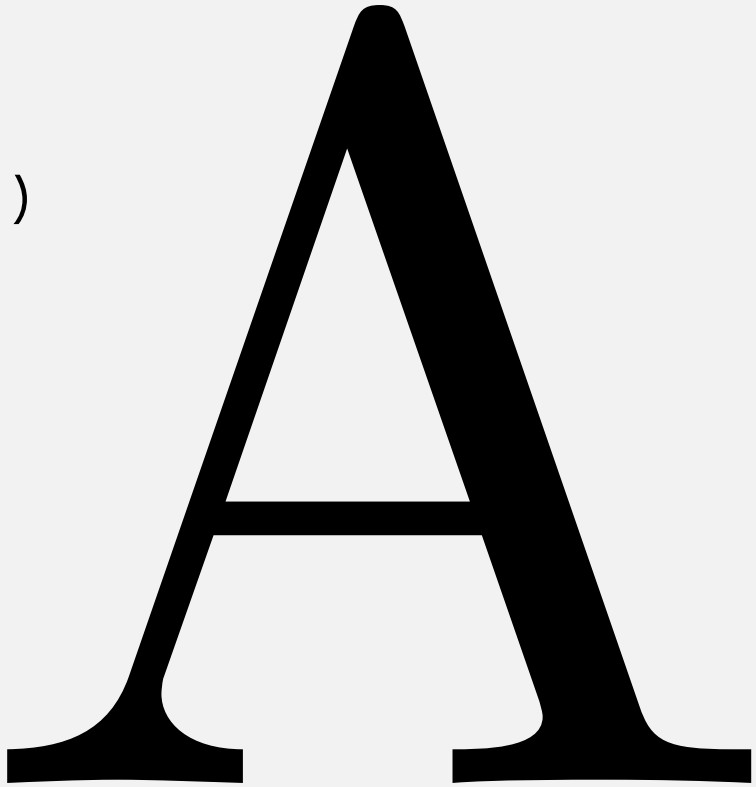
- PyX [http://pyx.sourceforge.net/]:
  Python Bibliothek zur Erzeugung von PostScript und PDF
- sehr gute TEX Integration → Verarbeitung von dvi-Daten
- Unterstützung von Type1 Schriften

Glyphpfad gefüllt

```
from pyx import *
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("cmr10.pfb")

c = canvas.canvas()
c.fill(f.getglyphpath("A", 500))
c.writeEPSfile("glyphpath")
```

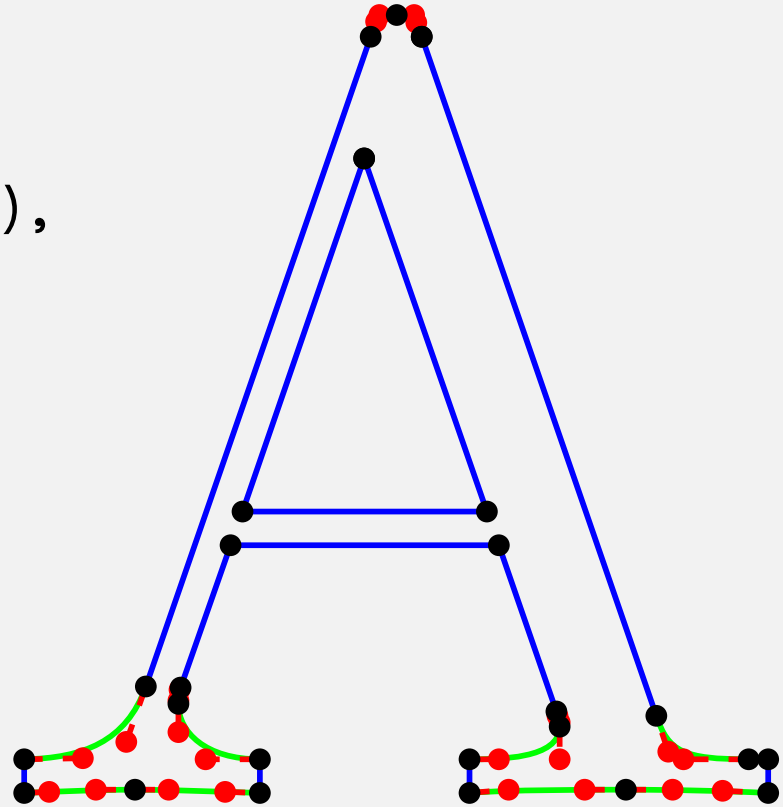


Elemente eines Glyphpfades

```
from pyx import *
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("cmr10.pfb")

c = canvas.canvas()
c.draw(f.getglyphpath("A", 500),
       [deco.shownormpath()])
c.writeEPSfile("glyphpath")
```



Was steht in cmr10.pfb?

Versteckte Nachricht (zzgl. einiger Zeilenumbrüche, weil Zeilen zu lang für diese Präsentation):

```
BKPH TeX CM
1992 Jul 21 16:14:27

Computer Modern fonts were designed by
Donald E. Knuth at Stanford,
based in part on (American) Monotype Corporation Modern 8A.
Font outlines generated by Doug Henderson
at Blue Sky Research,
using the ScanLab tool created by Ian Morrison
of Projective Solutions.
Character level hinting by Blenda Horn of Y&Y.
Adobe Type 1 encoding designed by Berthold K.P. Horn of Y&Y.
Barry Smith of Blue Sky Research coordinated the effort.
Long live TeX!
```

Dekodieren versteckter Nachrichten

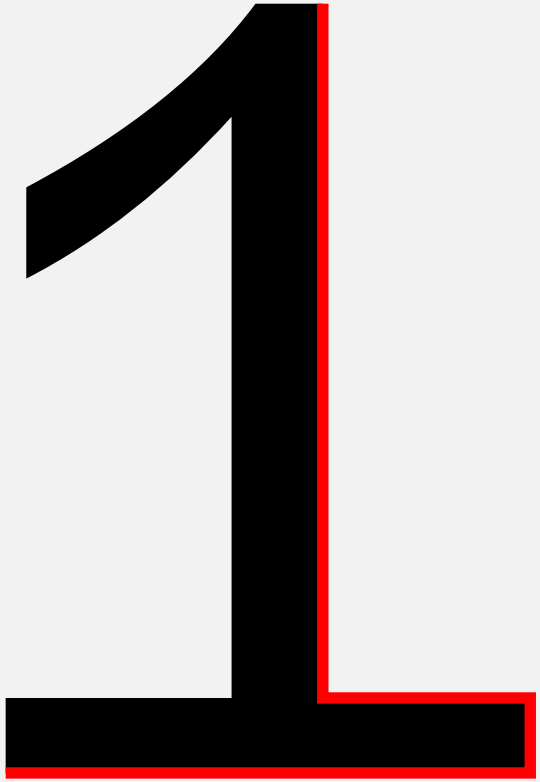
- Daten in pfb-Dateien sind verschlüsselt (ursprünglich Lizenzverkauf)
- in der Verschlüsslung stecken Zufallswerte (seeds)
- ggf. sind Nachrichten enthalten, die beim Entschlüsseln des eigentlichen Inhalts mit dekodiert werden

```
from pyx.font.t1font import T1pfbfont, decoder

f = T1pfbfont("cmr10.pfb")
f._data2decode()
data = []
for code in f.subrs:
    data.append(decoder(code, f.charstringr, 0)[:4])
for glyph in f.glyphlist[:81]:
    data.append(decoder(f.glyphs[glyph], f.charstringr, 0)[:4])
print "".join(data)
```

Ändern eines Glyphen

org.pfb:



mod.pfb:

Ändern eines Glyphen

```
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("org.pfb")
cmds = f.getglyphcmds("one")
for i, cmd in enumerate(cmds):
    print "Zeile %2i: %s" % (i, cmd)

print "----"
cmds[5] = 250
cmds[7] = 84
cmds[9] = 708
cmds[11:15] = []
cmds[29:33] = []

f.setglyphcmds("one", cmds)
for i, cmd in enumerate(cmds):
    print "Zeile %2i: %s" % (i, cmd)
f.outputPFB(open("mod.pfb", "wb"))
```

Teil der Ausgabe:

```
Zeile  5: 2
Zeile  6: hmoveto
Zeile  7: 483
Zeile  8: hlineto
Zeile  9: 69
Zeile 10: vlineto
Zeile 11: -191
Zeile 12: hlineto
Zeile 13: 639
Zeile 14: vlineto
----
Zeile  5: 250
Zeile  6: hmoveto
Zeile  7: 84
Zeile  8: hlineto
Zeile  9: 708
Zeile 10: vlineto
```

einfache Kapitälchen

Nachgeahmte Kapitälchen:

IowanOldStyleBT

einfache Kapitälchen

```
from pyx import *
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("biwr8a.pfb")
c = canvas.canvas()
x1_pt = x2_pt = 0
for s in "IowanOldStyleBT":
    if s.islower():
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s.upper(), 50)



    else:
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s, 100)


    c.fill(p, [trafo.translate_pt(x1_pt, 100)])
    x1_pt += wx_pt
c.writeEPSfile("caps")
```

bessere Kapitälchen

Nachgeahmte Kapitälchen ohne und mit Korrektur der Strichdicke:

IowanOldStyleBT

IowanOldStyleBT

...durch PyX-Pfad-Transformationen

```
from pyx import *
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("biwr8a.pfb")
c = canvas.canvas()
x1_pt = x2_pt = 0
for s in "IowanOldStyleBT":
    if s.islower():
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s.upper(), 50)
        c.fill(p, [trafo.translate_pt(x2_pt, unit.topt(0.04)),
                   deformer.parallel(-0.04, sharpoutercorners=1)])
        x2_pt += wx_pt + 2*unit.topt(0.04)
    else:
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s, 100)
        c.fill(p, [trafo.translate_pt(x2_pt, 0)])
        x2_pt += wx_pt
    c.fill(p, [trafo.translate_pt(x1_pt, 100)])
    x1_pt += wx_pt
c.writeEPSfile("caps")
```

prima Kapitälchen

Nachgeahmte Kapitälchen ohne und mit Korrektur der Strichdicke mit sinnvollem Größenverhältnis:

IowanOldStyleBT

IowanOldStyleBT

. . . durch sinnvolle Parameter

```
from pyx import *
from pyx.font.t1font import T1pfbfont

f = T1pfbfont("biwr8a.pfb")
c = canvas.canvas()
x1_pt = x2_pt = 0
for s in "IowanOldStyleBT":
    if s.islower():
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s.upper(), 75)
        c.fill(p, [trafo.translate_pt(x2_pt, unit.topt(0.02)),
                   deformer.parallel(-0.02, sharpoutercorners=1)])
        x2_pt += wx_pt + 2*unit.topt(0.02)
    else:
        p, wx_pt, wy_pt = f.getglyphpathwxwy_pt(s, 100)
        c.fill(p, [trafo.translate_pt(x2_pt, 0)])
        x2_pt += wx_pt
    c.fill(p, [trafo.translate_pt(x1_pt, 100)])
    x1_pt += wx_pt
c.writeEPSfile("caps")
```

Fragen?

contact@wobsta.de
http://www.wobsta.de/