Writing Gregg Shorthand with METAFONT and LaTeX

Stanislav Jan Šarman
Computing Centre
Clausthal University of Technology
Erzstr. 51
38678 Clausthal
Germany
Sarman (at) rz dot tu-clausthal dot de
http://www3.rz.tu-clausthal.de/~rzsjs/steno/Gregg.php

Abstract

We present an online system, which converts English text into Gregg shorthand, a phonetic pen writing system used in the U.S. and Ireland.

Shorthand is defined [2] as a method of writing derived from the spelling system and the script of a particular language. In order to approximate the speed of speech:

- The graphic redundancy of written characters and ligatures derived from longhand letters is reduced.
- Silent letters are ignored, and complex orthographic rules are simplified. The spelling only serves as a reference system of what is largely phonetic writing.
- Abbreviations are utilized.

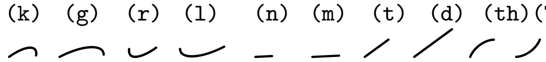
Gregg shorthand was first published in 1888. In the following three sections we want to show how its inventor J. R. Gregg has applied the above general principles in the creation of his shorthand system and how the current system version, the centennial edition [7], can be implemented in METAFONT [5].

1 Simplified writing

1.1 The alphabet of Gregg Shorthand

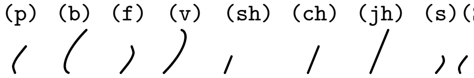
Signs of consonant phonemes[1] written forward:

`(k) (g) (r) (l) (n) (m) (t) (d) (th)(Th)`



Signs of consonant phonemes written downward:

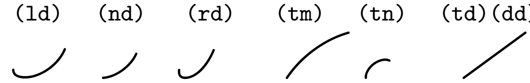
`(p) (b) (f) (v) (sh) (ch) (jh) (s)(S)`



The signs of voiced consonant phonemes are larger versions of their unvoiced opposites. As the "minimal pairs"[2] of words with phonemes `s` vs. `z`, `th` vs. `dh` and `sh` vs. `zh` are very rare, there are only the unvoiced versions of these signs. On the other hand two forms of signs exist for `s` and `th`: the right `(s)`, `(th)` and the left `(S)`, `(Th)`, respectively.

Blended consonant signs:

`(ld) (nd) (rd) (tm) (tn) (td)(dd)`



can be thought of as built-in ligatures.

Vowel signs:

		for sound(s) as in
`[a]`		at, art, ate
`[e]/[i]`		pit, pet, peat
`(o)`		pot, port , pour
`(u)`		tuck, took , tomb

Elementary diphthong (triphone) signs:

		for sounds as in
`[ai]`		high
`[aee]`		client
`[iie]`		create

When writing shorthand the curvilinear motion of a pen in the slope of longhand traces ellipse or oval-based curves. We shape these elementary shorthand signs in METAFONT as continuous curvature splines with zero curvature at the end points.

Half-consonants h, w, prefixes and suffixes are realized as markers, e.g. a dot over `[i]` such as in "he"; means preceding `h`; another dot under (along) the sign `(t)` such as in denotes the suffix -ing in "heating".

1.2 Metaform

Gregg shorthand is an alphabetic system — written words are combined together from elementary signs by joining, so that in general the Gregg shorthand glyphs have the following *metaform*:

$$\{\ (\{p,\}C\{,s\})\ |\ [\texttt{+}|\texttt{-}][\{p,\}V\{,s\}]\ \}^{+}$$

where V are circular vowels, C denotes (blended) consonants and vowels `u` and `o`, optional p and s

[1] Phonemes are denoted in typewriter type, their signs are parenthesized or bracketed.

[2] such as face vs. phase, sooth vs. soothe, mesher vs. measure, pronunciations of which differ only in one place

stand for markers of semiconsonants, prefixes or suffixes. In more detail:
V ::= `a` | `e` | `i` | `ai` | `aee` | `iie`
C ::= `b` | `d` | `dd` ... `o` | `u`
p ::= `h` | `w` | ... `over` | `under` ...
s ::= `ing` | `ly` | ...
Examples:

	metaform
he	`-[h,i,]`
heating	`-[h,i,](,t,ing)`
need	`(n)-[i](d)`
needed	`(n)-[i](dd)`
needing	`(n)-[i](,d,ing)`

The metaform corresponds directly to the METAFONT program code, e.g.:
`(n)-[i](,d,ing)` ↔ `I(,n,); V(-1,,i,); C(,d,ing);`
i.e. the shorthand glyph of this particular character is initiated with `(n)`, then the signs of right vowel `-[i]` and the sign of consonant `d` with suffix `ing` are appended.

1.3 Joining

We distinguish between joinings where circular signs are involved and joinings without them.

1.3.1 *CC* Joinings

Examples of CC joinings ordered by their connectivity grade $G^{(0)}$–$G^{(2)}$ follow:

continuous: `(d)(n)`, `(u)(n)`, `(t)(S)` (with turning point!)

tangent continuous: + = `(f)(l)`,

+ = `(p)(r)`, + = `(u)(p)`,

+ = `(g)(v)`

curvature continuous: + = `(r)(k)`

The first two types of joinings are handled by METAFONT means; $G^{(2)}$ continuous connecting can be done only in special cases using Hermite interpolation for Bézier splines, as follows.

If two endpoints z_i, unit tangents d_i, $d_i {d_i}^t = 1$ and curvatures κ_i, $i = 1, 2$ are given, the control points $z_0^+ = z_0 + \delta_0 d_0$, $z_1^- = z_1 - \delta_1 d_1$ have to be determined from the following system of nonlinear equations [3]:

$$(d_0 \times d_1)\delta_0 = (a \times d_1) - {}^3\!/\!{}_2\kappa_1\delta_1^2 \qquad (1)$$
$$(d_0 \times d_1)\delta_1 = (d_0 \times a) - {}^3\!/\!{}_2\kappa_0\delta_0^2 \qquad (2)$$

$(a = z_1 - z_0)$.

If the tangents are parallel, i.e. $(d_0 \times d_1) = 0$ and the curvatures κ_i have opposite signs, the equations (1)–(2) are trivially solvable, so that $G^{(2)}$ joining is possible (here after some kerning):

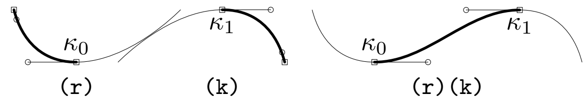
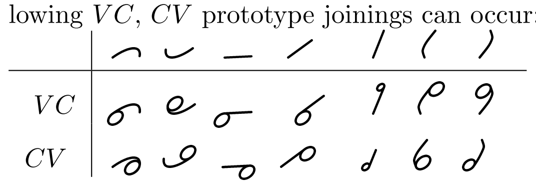


Figure 1: An example of a $G^{(2)}$ continuous CC joining

1.3.2 Joinings with circular signs

The signs of consonants can be classified into seven categories. Looped vowels before or after a consonant are written within the curves or combined with straight consonants as right curves, so that the following VC, CV prototype joinings can occur:



Observe the realization of diphthongs as VC or CV joinings, too:

		for sounds as in
`-[a](u)`		how
`(o)+[i]`		toy
`-[i](u)`		few

All VC or CV joinings are done in such a manner that both components have identical tangents and curvature equal to zero at the connection point. As an example of $G^{(2)}$ connecting of a consonant sign with a vowel sign consider prepending the sign of `-[a]` before `(k)` as in :

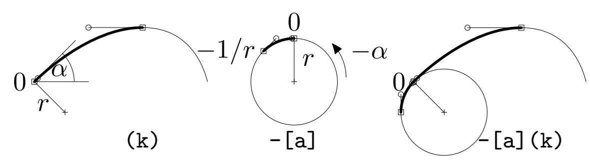


Figure 2: An example of a $G^{(2)}$ continuous VC joining

Here the spline approximation of `(k)` is such that the curvature equals zero at the end points; this is the case with other consonants, too. The spline approximation of right circle with radius r for the sign `-[a]` consists of 8 segments such that the curvature $\kappa = 0$ at the top point and $\kappa \approx -1/r$ at the other points. Let α be the angle of the tangent in the zeroth point of `(k)`. Placing the `-[a]` circle at the point with distance r from the zeroth point of `(k)` and rotating it by the angle of $-\alpha$ a curvature continuous joining is obtained.

For a $G^{(2)}$ continuous spline approximation of the unit circle with curvature $\kappa = 0$ in one point we use the traditional Bézier spline circle segment approximation for the segments 1–7 (see the 7th segment on the left) $z(1/2) = 1$, so that $\delta_0 = \delta_1 = 4/3 \tan(\theta/4)$ and $\kappa_i = -(1 - \sin^4(\theta/4))$. Using this value as left point curvature of the 8th segment (see on the right) and demanding $\kappa_1 = 0$, the Hermite interpolation with the equations (1)–(2) for the unknowns δ_0 und δ_1 can be solved for segments with $\theta < 60°$.

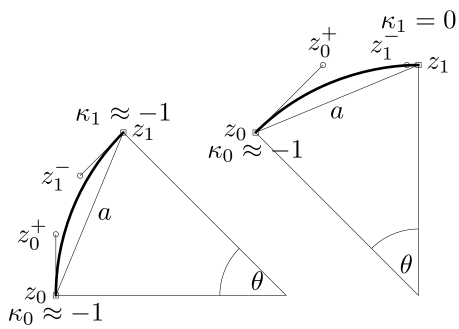
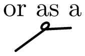
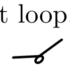
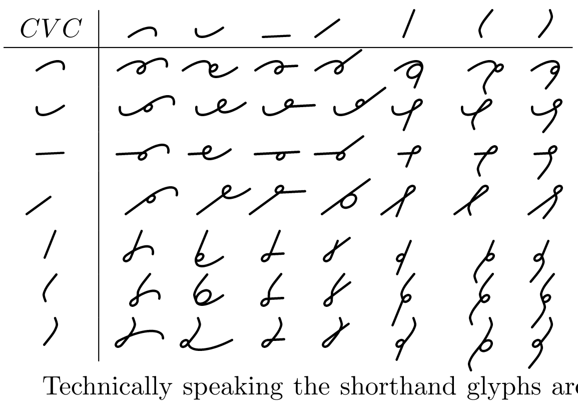


Figure 3: 7th and 8th segment ($\theta = 45°$) of our unit right circle spline approximation

For *CVC* joinings 7×7 cases are possible in all — many of them can be transformed by reflection and rotation into one another. All these joinings can be made $G^{(2)}$ continuous. It must be decided before writing a right consonant, whether the loop is to be written as a left or as a right loop, compare:

team `(t)+[i](m)`  vs.  `(m)-[i](t)` meet



Technically speaking the shorthand glyphs are realized as

- an array of $G^{(0)}$–$G^{(2)}$ continuous METAFONT-paths and
- an array of discontinous marker paths.

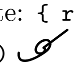
Slanting (default 22.5°) and tilting[3] of characters is done at the time of their shipping.

[3] $G^{(2)}$ continuity is invariant under affine transformations.

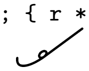
2 Phonetic writing

Gregg shorthand uses its own orthography, which must be acquired by the shorthand learner and in a system such as ours, the pronunciation of a word has to be known. We use the Unisyn multi-accent lexicon[4] with about 118,000 entries [4] comprising British and American spelling variants. The fields of lexicon entries are: spelling, optional identifier, part-of-speech, pronunciation, enriched orthography, and frequency. Examples are:

```
owed;;VBD/VBN; { * ouw }> d >;{owe}>ed>;2588
live;1;VB/VBP; { l * i v } ;{live};72417
live;2;JJ; { l * ae v } ;{live};72417}
```

Homonyma cases such as "live" above, in which the pronunciation helps to identify word meaning are much more rarer than the cases in which the use of pronunciation yields homophones resulting in shorthand homographs. Consider the very frequent right, rite, wright, write: `{ r * ai t }` with shorthand glyph `(r)+[ai](t)`  or the heterophones

```
read;1;VB/NN/NNP/VBP; { r * ii d };{read};94567
read;2;VBN/VBD; { r * e d };{read};94567
```

both written as  `(r)+[e|i](d)`. Thus phonetic writing may speed up the shorthand recording, but the newly-created homographs complicate the deciphering of written notes.

The pronunciation of a word has to be transformed to the above defined metaform, e.g.:

`needed: { n * ii d }.> I7 d >`$\Rightarrow$ `(n)-[i](dd)`
`needing: { n * ii d }.> i ng >`$\Rightarrow$ `(n)-[i](,d,ing)`

This major task consists of a number of transformations done in this order:

- Elimination of minor (redundant) vowels such as the schwa `@r` in
  `fewer: { f y * iu }> @r r >.`
  There is a general problem with the schwas (`@`). Which are to be ruled out and which of them are to be backtransformed[5] to the spelling equivalent? Consider
  `data: { d * ee . t == @ }` $\Rightarrow$ `(d)-[a](t]-[a]`
  `upon: { @ . p * o n }` $\Rightarrow$ `(u)(p)(n)`
  Both `@` are backtransformed, but stressed `o` is cancelled.
- Finding of prefixes (un-, re-, ...), suffixes (-ing, -ly, ...), handling of semiconsonants h, w.
- Creation of ligatures such as `dd`, parenthesizing consonants and bracketing circular vowels.
- Finding the proper orientation of loops.

These tranformations are done through a series of cascaded context dependent rewrite rules, realized

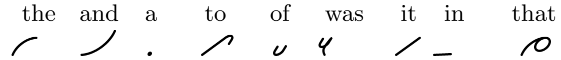
[4] which itself is a part of a text-to-speech system
[5] by a lex program

by finite state transducers (FSTs). An example of such a rule is
`" I7" -> 0 || [ t | d ] " }.>" _ " d >"`
which zeroes the string `" I7"` in the left context of consonant `t` or `d` and the right context of `"d >"`; i.e.:
`{ n * ii d }.> I7 d >` ⇒ `{ n * ii d }.> dd >`
Effectively this rule creates the ligature `(dd)` and can be thought of as the reverse of the phonetic rule
`" e" -> " I7" || [ t | d ] " }.>" _ " d >"`
which determines the pronunciation of perfect tense suffix `> e d >` in the left context of `t` or `d`:
`{ n * ii d }.> I7 d >` ⇐ `{ n * ii d }.> e d >`

3 Abbreviations

The Gregg shorthand, like every other shorthand system since Tironian notes (43 B.C.), takes advantage of the statistics of speaking[6] and defines about 380 so called *brief forms*. Here are the most common English words:

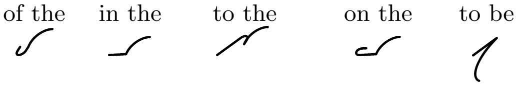
the and a to of was it in that



`(th) (nd) (a) (t)(o) (o) (u)(S) (t) (n) (th)-[a]`

Writing the most common bigrams together, e.g.:

of the in the to the on the to be



`(o)(th) (n)(th) (t)(o)(th) (O)(n)(th) (t)(b)`

spares lifting the pen between words thus increasing shorthand speed.

These bigrams are entries in the dictionary of about 420 *phrases* as well as the following examples:

as soon as possible: `-[a](s)(n)(S)(p)`: 

if you cannot: `+[i](f)(u)(k)(n)`: 

if you can be: `+[i](f)(u)(k)(b)`: 

thank you for your order: `(th)(U)(f)(u)(d)`: 

you might have: `(u)(m)-[ai](t)(v)`: 

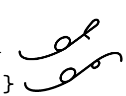
The abbreviation dictionary is coded in `lexc` [1].

4 text2Gregg

Our text2Gregg software is an online system,[7] which records input text as Gregg shorthand. LaTeX input notation is accepted.

Homonyma variants such as

`latex;1,rubber; { l * ee . t e k s }`
`latex;2,computing; { l * ee . t e k }`



are entered in the form `latex` (i.e. `latex#1`) and `latex#2`, respectively.

The task of the online conversion of given text to shorthand notes is done in the following four steps:

1. The input stream, stripped of LaTeX commands, is tokenized.[8] There are two kinds of tokens:
   - Punctuation marks, numbers and common words for which a metaform entry in the abbreviation dictionary of *brief forms* and *phrases* exists and
   - the other words.
2. At first for a word of the latter category its pronunciation has to be found in Unisyn lexicon [4]. From this its metaform is built by a program coded as the tokenizer above in the XEROX-FST tool `xfst` [1].
3. In a `mf` run for each of the tokens using its metaform a shorthand glyph (i.e. a METAFONT character) is generated on-the-fly.
4. Then the text is set with LaTeX, rendered in the pipeline → `dvips` → `gs` → `ppmtogif` and sent from the server to the browser.

PDF output with better resolution can be generated. Also a djvu-backend exists, which produces an annotated and searchable shorthand record. PostScript Type 3 vector fonts can be obtained, too.

References

[1] Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Publications, Stanford, 2003.

[2] Florian Coulmas. *Blackwell Encyclopedia of Writing Systems*. Blackwell Publishers, 1999.

[3] C. deBoor, K. Höllig, and M. Sabin. High accuracy geometric hermite interpolation. *Computer Aided Geometric Design*, 4:269–278, 1987.

[4] Susan Fitt. Unisyn multi-accent lexicon, 2006. `http://www.cstr.ed.ac.uk/projects/unisyn/`.

[5] Donald E. Knuth. *The METAFONTbook*, volume C of *Computers and Typesetting*. Addison-Wesley Publishing Company, Reading, Mass., 5th edition, 1990.

[6] Stanislav Jan Šarman. DEK-Verkehrsschrift mit METAFONT und LaTeX schreiben. *Die TeXnische Komödie*, to be published.

[7] Charles E. Zoubek. *Gregg Shorthand Dictionary, Centennial Edition, Abridged Version*. Glencoe Divison of Macmillan/McGraw-Hill, 1990.

[6] The 15 most frequent words in any text make up 25% of it; the first 100, 60%, ...

[7] see project URL and also `DEK.php` for the German shorthand DEK counterpart [6]

[8] Care has to be taken for proper nouns, which are underlined in Gregg shorthand.