Hershey Fonts

Larry Bartholdi & his gang

June 9, 1993

Contents

1 Introduction

This document defines procedures to display Hershey fonts using the BGI[1] interface. A replacement for `(out-text ...)` is provided. The format is intended to be an imitation of Donald Knuth's TEX typesetting system:

- New fonts are selected with \⟨*name*⟩ or \⟨*name*⟩@⟨*scale*⟩. Remember to type double backslashes in a Scheme string!
- `\small` selects a font 2 levels smaller.
- `\large` selects a font 2 levels larger.
- `\horiz` and `\vert` set the text direction.
- `{}` delimit a group. The previous orientation, size, and font are restored after the closing acculade.
- `^` and `_` cause superscripting and subscripting. As in TEX, they apply to the next character or group, so sugar would be entered '`C_6H_{12}O_6`', for instance. `Out-hershey` attempts to use a smaller size when displaying a subscript (actually two units smaller). If this is not possible, it will try to use a smaller stroke set in the same family. By now we're getting real desperate, so `Out-hershey` just makes the current font as small as possible.
- All usual symbols, and many others, are available as "\⟨*mnemonic*⟩".

[1] Borland's Graphical Interface

If this sounded too theoretical, see the last chapter, containing a few examples.

A few differences exist between `out-hershey` and `out-text`: `out-hershey` works only with one text justification setting, `'left 'bottom`. No special care is given to enforce these settings, as they are by far the most useful and common ones. Characters such as '^' and '_' must be prefixed by a '\', as they have a TeX meaning.

2 Code

This is a huge environment that contains all TeX definitions. It is implemented as a separate environment so

- Users can add their definitions as they wish
- This big piece of code is separated from the rest, which is a good thing from a hacker's point of view...
- Global variables (like "shall we just measure, or really draw?") can be hidden from the global environment.

A few special variables are included in this environment. They all contain an underscore (_) so they cannot be accessed from `Out-hershey` by the user.

`draw_it` A boolean telling us whether we're actually drawing, or just measuring.

`font_sibling` An association list describing the "cut-down" versions of fonts: `RM2B` can be reduced to `RM10`, for instance.

`raw_handler` A procedure displaying its first argument as "raw". This is used by the main routine to display text without it having a special TeX meaning.

`font_0` The default font.

`size_0` The default size for the default font.

`direction_0` The default direction for the default font.

Next comes the real drawing recursive routine. It will display string `str` with the specified font, direction and size, while `mode` describes how much information it must handle before return. possible values are `'char` (get only a character), `'raw` (no TeX interpretation), `'single` (only one pass) or a combination of these values. When no mode argument is provided, `TeX-out` will call itself tail-recursively until it has displayed the whole string.

```
(define TeX-environment
  (let ((out-char (lambda (cnum cfont args)
                    (let* ((str (car args))
                           (font (cadr args))
                           (direction (caddr args))
                           (size (cadddr args)))
```

```
                    ((access TeX-out TeX-environment)
                     (list->string (list (integer->char cnum)))
                     cfont direction size)
                    str)))
    )
(make-environment
  (define draw_it #t)                          ; shall we draw, or measure ?
  (define font_siblings '((gr2b . gr10) (gr2l . gr10)
                          (rm2b . rm10) (rm2l . rm10)
                          (sl2b . sl10) (sl2l . sl10)
                          (ss2b . ss10) (ss2l . ss10)
                          (sy20 . sy10)))
  (define font_0 'RM10)
  (define size_0 2)
  (define direction_0 'HORIZ)
  (define (o_sym n)
    (lambda (s . a)
      (out-char
        n
        (if (member (cadr a) '(SY10 SS10 RM10 GR10 SL10)) 'SY10 'SY20)
        a)))
  (define (o_svar fif felse)
    (lambda (s . a)
      (out-char
        (if (member (cadr a) (cdr fif)) (car fif) felse)
        (if (member (cadr a) '(SY10 SS10 RM10 GR10 SL10)) 'SY10 'SY20)
        a)))
  (define (o_s10 n)
    (lambda (s . a) (out-char n 'SY10 a)))
  (define (o_s20 n)
    (lambda (s . a) (out-char n 'SY20 a)))
  (define (o_grk n)
    (lambda (s . a)
      (out-char n
                 (case (cadr a)
                   ((SY10 SS10 RM10 GR10 SL10) 'GR10)
                   ((SS2L RM2L GR2L SL2L) 'GR2L)
                   (else 'GR2B))
                 a)))
  (define (o_lig n)
    (lambda (s . a)
      (out-char n (cadr a) a)))

  (define Font-Composite? pair?)
  (define Font-X car)
  (define Font-Y cadr)
  (define Font-Make cons)
  (define (Font- size)
    (if (Font-Composite? size)
        (Font-Make (max 1 (- (Font-X size) 2))
                   (max 1 (- (Font-Y size) 2)))
        (max 1 (- size 2))))
```

```
(define (Font+ size)
  (if (Font-Composite? size)
      (Font-Make (+ (Font-X size) 2)
                 (+ (Font-Y size) 2))
      (+ size 2)))
(define (Font* size)
  (if (Font-Composite? size)
      (Font-Make (+ (Font-X size) 1)
                 (+ (Font-Y size) 1))
      (+ size 1)))
(define (Font-Big? size)
  (if (Font-Composite? size)
      (> (max (Font-X size) (Font-Y size)) 2)
      (> size 2)))
(define |small| (lambda (s str font direction size)
                  (TeX-out str font direction (Font- size))))
(define |large| (lambda (s str font direction size)
                  (TeX-out str font direction (Font+ size))))
(define |horiz| (lambda (s str font direction size)
                  (TeX-out str font 'horiz size)))
(define |vert| (lambda (s str font direction size)
                 (TeX-out str font 'vert size)))
(define |raw_handler|
  (lambda (s str font direction size)
    (cond ((equal? s "_") (|underscore| str font direction size))
          (else (TeX-out s font direction size 'single 'raw)
                str))))
(define TeX-codes
  '((|langle| . (sym 1))
    (|rangle| . (sym 2))
    (|(| . (sym 3))
    (|)| . (sym 4))
    (|[| . (sym 5))
    (|]| . (sym 6))
    (|{| . (sym 7))
    (|}| . (sym 8))
    (|rw| . (sym 9))
    (|wr| . (sym 10))
    (|parallel| . (sym 11))
    (|pm| . (sym 12))
    (|mp| . (sym 13))
    (|times| . (svar '(14 SS2L GR2L EN2L SS10) 15))
    (|cdot| . (svar '(16 SS2L GR2L EN2L SS10) 17))
    (|div| . (sym 18))
    (|neq| . (sym 19))
    (|equiv| . (sym 20))
    (|leq| . (sym 21))
    (|geq| . (sym 22))
    (|propto| . (sym 23))
    (|subset| . (sym 24))
    (|cup| . (sym 25))
    (|supset| . (sym 26))
```

```
(|cap| . (sym 27))
(|in| . (sym 28))
(|nabla| . (svar '(29 SS2L GR2L EN2L) 30))
(|varsurd| . (sym 31))
(|surd| . (sym 32))
(|varint| . (sym 33))
(|int| . (sym 34))
(|oint| . (sym 35))
(|sum| . (sym 36))
(|prod| . (sym 37))
(|infty| . (sym 38))
(|exists| . (sym 39))
(|otimes| . (s20 40))
(|perp| . (s20 42))
(|angle| . (s20 43))
(|thatis| . (s20 44))
(|angstrom| . (s20 46))
(|hbar| . (s20 47))
(|'| . (sym 64))
(|`| . (sym 65))
(|u| . (sym 66))
(|''| . (sym 67))
(|``| . (sym 68))
(|`\"| . (sym 69))
(|'\"| . (sym 70))
(|rightarrow| . (svar '(71 SS2L GR2L RM2L SS10) 72))
(|uparrow| . (sym 73))
(|leftarrow| . (sym 74))
(|downarrow| . (sym 75))
(|S| . (sym 76))
(|dagger| . (sym 77))
(|ddagger| . (sym 78))
(|box| . (sym 79))
(|odot| . (sym 80))
(|sun| . (sym 80))
(|mercury| . (sym 81))
(|venus| . (sym 82))
(|oplus| . (sym 83))
(|earth| . (sym 83))
(|mars| . (sym 84))
(|jupiter| . (sym 85))
(|saturn| . (sym 86))
(|uranus| . (sym 87))
(|neptune| . (sym 88))
(|pluto| . (sym 89))
(|moon| . (sym 90))
(|comet| . (sym 91))
(|asteroid| . (sym 92))
(|ver| . (sym 93))
(|autumnis| . (sym 94))
(|bullet| . (s10 95))
(|spadesuit| . (s20 95))
```

```
(|heartsuit| . (s20 96))
(|dash| . (s10 97))
(|diamondsuit| . (s20 97))
(|sqcap| . (s10 98))
(|clubsuit| . (s20 98))
(|wedge| . (s10 99))
(|varclub| . (s20 99))
(|underscore| . (sym 48))
(|wp| . (s20 50))
(|scout| . (s20 100))
(|bigtriangledown| . (s10 101))
(|circle| . (s10 160))
(|square| . (s10 161))
(|triangle| . (s10 162))
(|diamond| . (s10 163))
(|star| . (s10 164))
(|smash| . (s10 167))
(|CIRCLE| . (s10 168))
(|SQUARE| . (s10 169))
(|UTRIANGLE| . (s10 170))
(|LTRIANGLE| . (s10 171))
(|DTRIANGLE| . (s10 172))
(|RTRIANGLE| . (s10 173))
(|STAR| . (s10 174))
(|FLAG| . (s10 175))
(|anchor| . (s10 176))
(|plane| . (s10 177))
(|work| . (s10 178))
(|oil| . (s10 179))
(|boat| . (s10 180))
(|skew| . (s10 181))
(|christ| . (s10 182))
(|muslim| . (s10 183))
(|jew| . (s10 184))
(|bell| . (s10 185))
(|palmtree| . (s10 186))
(|firtree| . (s10 187))
(|oaktree| . (s10 188))
(|tree| . (s10 189))
(|sun| . (s10 190))
(|county| . (s20 147))
(|district| . (s20 148))
(|aries| . (s20 149))
(|taurus| . (s20 150))
(|gemini| . (s20 151))
(|cancer| . (s20 152))
(|leo| . (s20 153))
(|virgo| . (s20 154))
(|libra| . (s20 155))
(|scorpio| . (s20 156))
(|sagittarius| . (s20 157))
(|capricorn| . (s20 158))
```

```
(|aquarius| . (s20 159))
(|pisces| . (s20 160))
(|steer| . (s20 161))
(|cent| . (s20 162))
(|verb*| . (s20 163))
(|mdot| . (s20 192))
(|m`| . (s20 193))
(|m'| . (s20 194))
(|full| . (s20 195))
(|half| . (s20 196))
(|quarter| . (s20 197))
(|sharp| . (s20 198))
(|natural| . (s20 199))
(|flat| . (s20 200))
(|rest| . (s20 201))
(|hrest| . (s20 202))
(|qrest| . (s20 203))
(|erest| . (s20 204))
(|Gclef| . (s20 205))
(|Fclef| . (s20 206))
(|tenorclef| . (s20 207))
(|aleph| . (s20 49))
(|Alpha| . (grk 65))
(|Beta| . (grk 66))
(|Gamma| . (grk 67))
(|Delta| . (grk 68))
(|Epsilon| . (grk 69))
(|Zeta| . (grk 70))
(|Eta| . (grk 71))
(|Theta| . (grk 72))
(|Iota| . (grk 73))
(|Kappa| . (grk 74))
(|Lambda| . (grk 75))
(|Mu| . (grk 76))
(|Nu| . (grk 77))
(|Xi| . (grk 78))
(|Omicron| . (grk 79))
(|Pi| . (grk 80))
(|Rho| . (grk 81))
(|Sigma| . (grk 82))
(|Tau| . (grk 83))
(|Upsilon| . (grk 84))
(|Phi| . (grk 85))
(|Khi| . (grk 86))
(|Psi| . (grk 87))
(|Omega| . (grk 88))
(|alpha| . (grk 97))
(|beta| . (grk 98))
(|gamma| . (grk 99))
(|delta| . (grk 100))
(|epsilon| . (grk 101))
(|zeta| . (grk 102))
```

```
    (|eta| . (grk 103))
    (|theta| . (grk 104))
    (|iota| . (grk 105))
    (|kappa| . (grk 106))
    (|lambda| . (grk 107))
    (|mu| . (grk 108))
    (|nu| . (grk 109))
    (|xi| . (grk 110))
    (|omicron| . (grk 111))
    (|pi| . (grk 112))
    (|rho| . (grk 113))
    (|sigma| . (grk 114))
    (|tau| . (grk 115))
    (|upsilon| . (grk 116))
    (|phi| . (grk 117))
    (|khi| . (grk 118))
    (|psi| . (grk 119))
    (|omega| . (grk 120))
    (|vardelta| . (grk 1))
    (|varepsilon| . (grk 2))
    (|vartheta| . (grk 3))
    (|varphi| . (grk 4))
    (|varsigma| . (grk 5))
    (|ff| . (lig 1))
    (|fi| . (lig 2))
    (|fl| . (lig 3))
    (|ffi| . (lig 4))
    (|ffl| . (lig 5))
    (|i| . (lig 6))
  ))

(define (TeX-out str font direction size . mode)
  (define (TeX-token str mode)
    (define TeX-lookup
      (let ((code (compile `(access XXX ,TeX-environment))))
        (lambda (name)
          (set-car! (member 'XXX (cadddr code)) name)
          (%execute code))))
    (define (TeX-font name)
      (lambda (s str font direction size)
        (TeX-out str (string->symbol
                       (list->string (map char-upcase (string->list name))))
                 direction (if s s size))))
    (define (TeX-raise amount)
      (lambda (str font direction size)
        (set-font font direction size)
        (let* ((delta (* (cdr (text-size "I")) amount))
               (move (if (equal? direction 'vert)
                         (cons (- delta) 0)
                         (cons 0 (- delta))))
               (back (if (equal? direction 'vert)
                         (cons delta 0)
```

```
                    (cons 0 delta))))
      (move-rel move)
      (let* ((newfont
               (if (Font-Big? size)
                   font
                   (let ((try (assoc font font_siblings)))
                     (if try (cdr try) font))))
             (newsize (if (equal? newfont font)
                          (Font- size)
                          (Font* size)))
             (tail (TeX-out str newfont direction newsize 'single 'char)))
        (move-rel back)
        tail))))
(define (TeX-begin str font direction size)
  (TeX-out str font direction size))
(define (TeX-end str font direction size)
  (list str))
(letrec ((len (string-length str))
         (first-separator (substring-find-next-char-in-set
                            str 0 len
                            (if (member 'intoken mode)
                                "     \\^_{} "
                                "\\^_{}")))
         (first-len (cond ((member 'char mode) 1)
                          (first-separator first-separator)
                          (else len)))
         (trim (lambda (str)
                 (cond ((string-null? str) str)
                       ((char-whitespace? (string-ref str 0))
                        (trim (substring str 1 (string-length str))))
                       (else str)))))
  (if (equal? first-separator 0)
      (let ((tail (substring str 1 len)))
        (case (string-ref str 0)
          ((#\^) (cons (TeX-raise .6) tail))
          ((#\_) (cons (TeX-raise -.5) tail))
          ((#\{) (cons TeX-begin tail))
          ((#\}) (cons TeX-end tail))
          ((#\\) (if (and (not (string-null? tail))
                          (member (string-ref tail 0) '(#\\ #\^ #\_)))
                     (cons (lambda args
                             (apply |raw_handler|
                                    (cons (substring tail 0 1) args)))
                           (substring tail 1 (string-length tail)))
                     (let* ((next (TeX-token tail '(intoken)))
                            (first-token (car next))
                            (first-len (string-length first-token))
                            (double (substring-find-next-char-in-set
                                      first-token 0 first-len "@"))
                            (first-atom (if double
                                            (substring first-token 0 double)
                                            first-token))
```

```
                              (first-symbol (string->symbol first-atom))
                              (arg-string (if double
                                              (substring first-token
                                                         (1+ double) first-len)
                                              " "))
                              (argument (if (equal? (string-ref arg-string 0)
                                                    #\()
                                            (begin
                                              (string-set!
                                                arg-string
                                                (substring-find-next-char-in-set
                                                  arg-string 0
                                                  (- first-len double 1) ",")
                                                #\space)
                                              (read (open-input-string
                                                      arg-string)))
                                            (string->number arg-string)))
                              (TeX-cmd (assoc first-symbol TeX-codes))
                              (handler
                                (if TeX-cmd
                                    (apply (case (cadr TeX-cmd)
                                             (SYM o_sym)
                                             (SVAR o_svar)
                                             (S10 o_s10)
                                             (S20 o_s20)
                                             (LIG o_lig)
                                             (GRK o_grk))
                                           (cddr TeX-cmd))
                                    (TeX-lookup (string->symbol first-atom)))))
                         (cons (lambda args
                                 (apply (if (equal? handler #!unassigned)
                                            (TeX-font first-atom)
                                            handler)
                                        (cons argument args)))
                               (trim (cdr next))))))))
        (cons (substring str 0 first-len)
              (substring str first-len len)))))
(define (set-font font direction size)
  (if (not (assoc font (access font-l bgi-environment)))
      (install-user-font (symbol->string font)))
  (if (Font-Composite? size)
      (begin (set-text-style font direction 0)
             (set-user-char-size (cons (Font-X size) 1)
                                 (cons (Font-Y size) 1)))
      (set-text-style font direction size)))

(let ((next (if (member 'raw mode)
                (cons str "")
                (TeX-token str mode))))
  (if (string? (car next))
      (when (not (string-null? (car next)))
            (set-font font direction size)
```

```
                    (set-text-justify (if (equal? direction 'vert)
                                          'right
                                          'left)
                                      'bottom)
;               (writeln "out `" (car next) "' in " font " at " size)
                    (if draw_it
                        (out-text (car next))
                        (move-rel (cons (car (text-size (car next))) 0)))
                    (if (equal? direction 'vert)
                        (move-rel (cons 0 (- (car (text-size (car next)))))))
                    (if (member 'single mode)
                        (cdr next)
                        (TeX-out (cdr next) font direction size)))
              (let ((tail ((car next) (cdr next) font direction size)))
                (if (not (string? tail))
                    (car tail)
                    (if (member 'single mode)
                        tail
                        (TeX-out tail font direction size)))))))
      )))
```

This routine is the only one intended to be called by the user. It can receive one or two arguments, the first one being the string to display, and the second optional one being the starting position. It can be understood as a replacement for both `out-text` and `out-text-xy`.

```
(define (out-hershey str . position)
  (case (length position)
    ((0) ())
    ((1) (move-to (car position)))
    (else (error "Invalid arguments" position)))
  (if (string? str)
      ((access TeX-out TeX-environment) str
       (access font_0 TeX-environment)
       (access direction_0 TeX-environment)
       (access size_0 TeX-environment))
      (error "String expected" str)))
```

This routine returns the space taken to display `str`. It does not perform any output, and is a replacement for `text-size`.

```
(define (size-hershey str)
  (set! (access draw_it TeX-environment) #f)
  (let ((here (get-xy)))
    ((access TeX-out TeX-environment) str
     (access font_0 TeX-environment)
     (access direction_0 TeX-environment)
     (access size_0 TeX-environment))
```

```
(let ((there (get-xy)))
  (begin0
    (cons (- (car there) (car here))
          (- (cdr there) (cdr here)))
    (set! (access draw_it TeX-environment) #t)
    (move-to here)))))
```

3 Testing

Here all the necessary settings are done to ease the testing process. This shouldn't make you think trial and error is a good programming scheme!

```
(define (test s)
  (split-screen 10)
  (clear-device)
  (move-to '(100 . 100))
  (out-hershey s))
```

4 User's Manual

To use the Hershey font system, a few initialisation steps must be followed:

```
(init-graph)                          ; switch to graphics mode
(set-text-justify 'LEFT 'BOTTOM)      ; 'standard'
```

Here 'standard' should be understood as a joke, because it is not the mode BGI starts in, contrary to obvious common sense. Now `Out-hershey` can be used, either like `Out-Text` or like `Out-Text-XY`. With the latter option, supply a point as second argument. In all cases, supply a string as first argument. The general format of the string is:

$$\{\langle command\rangle \mid \langle font\rangle \mid \langle symbol\rangle \mid (\hat{} \mid \_)\langle item\rangle \mid \text{text}\}$$

There $\langle command\rangle$ is

- `\small` selects a font 2 units smaller. A unit is about 8 pixels.
- `\large` selects a font 2 units larger.
- `\horiz` and `\vert` set the text direction.

A $\langle font\rangle$ is `\`$\langle name\rangle$ or `\`$\langle name\rangle$`@`$\langle scale\rangle$. Name can be one of `RM10`, `RM2L`, `RM2B` (roman simple, light and bold), `SL10`, `SL2L`, `SL2B` (same in *slanted*), `SS10`, `SS2L`, `SS2B` (same in sans-serif), `GR10`, `GR2L`, `GR2B` (same in $\gamma\rho\epsilon\epsilon\kappa$), `AN20`, `GO20`, `OE20` (antique, gothic and old english), `SY10`, `SY20` (symbols) and `CY20` (cyrillic). $\langle scale\rangle$ can be a number (size in multiples of 8 pixels) or a pair `(`$\langle x\rangle$`,`$\langle y\rangle$`)` (X- and Y-scaling). Sizes should range between 1 and 10.

```
(out-hershey␣"\\rm2b@1␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\rm2b@2␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\rm2b@3␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\rm2b@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\rm2b@5␣J.␣Bond␣007?")
```

J. Bond 00

```
(out-hershey␣"\\rm2b@(3,1)␣J.␣Bond␣007?")
```

J. Bo

```
(out-hershey␣"\\rm2b@(1,3)␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\rm2l@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\sl2b@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\ss2b@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\an20@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\go20@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\oe20@4␣J.␣Bond␣007?")
```

J. Bond 007?

```
(out-hershey␣"\\cy20@4␣L.Q.Cnpa‘web")
```

М.С.Горбачев

```
(out-hershey␣"\\gr2b@4␣Ledem␣acam")
```

Μεδεν αγαν

An ⟨*item*⟩ is either a character, or a symbol, or a group (`{}`). Text is any string of characters, except `^`'s and `_`'s must be prefixed by a backslash (they in effect are treated as symbols).

These are the symbols available:

`\langle`	⟨	`\div`	÷	`\int`	∫
`\rangle`	⟩	`\neq`	≠	`\oint`	∮
`\(`	(	`\equiv`	≡	`\sum`	∑
`\)`	)	`\leq`	≦	`\prod`	∏
`\[`	[	`\geq`	≧	`\infty`	∞
`\]`	]	`\propto`	∝	`\exists`	∃
`\{`	{	`\subset`	⊂	`\otimes`	⊗
`\}`	}	`\cup`	∪	`\perp`	⊥
`\rw`		`\supset`	⊃	`\angle`	∠
`\wr`	≀	`\cap`	∩	`\thatis`	∴
`\parallel`	‖	`\in`	∈	`\angstrom`	Å
`\pm`	±	`\nabla`	∇	`\hbar`	ℏ
`\mp`	∓	`\varsurd`	√	`\'`	´
`\times`	×	`\surd`	√	`` \` ``	`
`\cdot`	·	`\varint`	∫	`\u`	˘

\''	’
\``	‘
\`"	‘
\'"	’
\rightarrow	→
\uparrow	↑
\leftarrow	←
\downarrow	↓
\S	§
\dagger	†
\ddagger	‡
\box	⌑
\odot	⊙
\sun	☉
\mercury	☿
\venus	♀
\oplus	⊕
\earth	⊕
\mars	♂
\jupiter	♃
\saturn	♄
\uranus	⛢
\neptune	♆
\pluto	♇
\moon	☾
\comet	☄
\asteroid	✳
\ver	☊
\autumnis	☋
\bullet	•
\spadesuit	♤
\heartsuit	♡
\dash	---
\diamondsuit	♢
\sqcap	⊓
\clubsuit	♧
\wedge	∧
\varclub	♣
\underscore	_
\wp	℘
\scout	⚜
\bigtriangledown	▽
\circle	○
\square	□
\triangle	△
\diamond	◊
\star	☆
\smash	*
\CIRCLE	●
\SQUARE	■
\UTRIANGLE	▲
\LTRIANGLE	◀
\DTRIANGLE	▼
\RTRIANGLE	▶
\STAR	★
\FLAG	⚑
\anchor	⚓
\plane	✈
\work	⚒
\oil	
\boat	
\skew	
\christ	✠
\muslim	☪
\jew	✡
\bell	🔔
\palmtree	🌴
\firtree	🌲
\oaktree	🌳
\tree	
\sun	
\county	
\district	
\aries	♈
\taurus	♉
\gemini	♊
\cancer	♋
\leo	♌

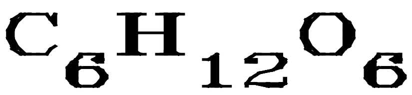
\virgo	♍
\libra	♎
\scorpio	♏
\sagittarius	♐
\capricorn	♑
\aquarius	♒
\pisces	♓
\steer	☸
\cent	¢
\verb*	
\mdot	·
\m‘	‘
\m’	’
\full	○
\half	○
\quarter	●
\sharp	♯
\natural	♮
\flat	♭
\rest	𝄻
\hrest	𝄼
\qrest	𝄽
\erest	𝄾
\Gclef	𝄞
\Fclef	𝄢
\tenorclef	
\aleph	$\aleph$
\Alpha	A
\Beta	B
\Gamma	Γ
\Delta	Δ
\Epsilon	E
\Zeta	Z
\Eta	H
\Theta	Θ
\Iota	I
\Kappa	K
\Lambda	Λ
\Mu	M
\Nu	N
\Xi	Ξ
\Omicron	O
\Pi	Π
\Rho	P
\Sigma	Σ
\Tau	T
\Upsilon	Υ
\Phi	Φ
\Khi	X
\Psi	Ψ
\Omega	Ω
\alpha	α
\beta	β
\gamma	γ
\delta	δ
\epsilon	ε
\zeta	ζ
\eta	η
\theta	ϑ
\iota	ι
\kappa	κ
\lambda	λ
\mu	μ
\nu	ν
\xi	ξ
\omicron	o
\pi	π
\rho	ρ
\sigma	σ
\tau	τ
\upsilon	υ
\phi	φ
\khi	χ
\psi	ψ
\omega	ω
\vardelta	∂
\varepsilon	ϵ
\vartheta	θ

`\varphi`	ϕ	`\fi`	ﬁ	`\ffl`	ﬄ
`\varsigma`	ς	`\fl`	ﬂ	`\i`	ı
`\ff`	ﬀ	`\ffi`	ﬃ		

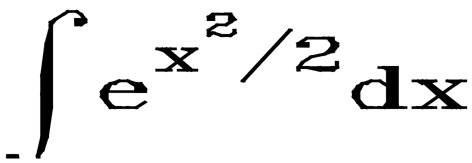
5 Sample output

These examples assume the screen has properly been initialised. There is no cheating: the code was actually typed in GeScheme!

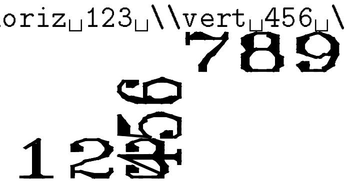
```
(out-hershey␣"\\rm2b@3␣C_6H_{12}O_6}")
```



```
(out-hershey␣"\\rm2b@3␣\\int␣e^{x^2/2}␣dx")
```



```
(out-hershey␣"\\rm2b@2␣\\horiz␣123␣\\vert␣456␣\\horiz␣789")
```



```
(out-hershey␣"\\rm2l@3␣S{\\small␣MALL}␣C{\\small␣APS}")
```

...But maybe the best examples are yours?