```
filin 'userto.sm'!       "redefines 'to' for editing"
filin 'usertextwindow.sm'!      "the class of windows"
filin 'userfileseg.sm'!       "the file part of windows"
filin 'userparagraph.sm'!   "the text part of windows"
filin 'userparenck.sm'!      "parenthesis checker"
filin 'usermenu.sm'!         "the class of menus"
filin 'usercursors.sm'! "these are the cursor bits"

"make the system classes editable"
to fool hdr fn (☞hdr←8. ☞fn←hdr[1] eval.
     PUT fn ☞HEADER hdr[1 to hdr length-1].)
fool (number x y | | nprint)!
fool (atom x y)!
fool (vector x y | | substr)!
fool (string x y | | substr)!
fool (float x y | | fprint)!
fool (stream in | i s l)!
fool (obset i input | vec size end | each)!
fool (point t | x y)!
fool (rectangle t a b | origin extent)!
fool (turtle var | pen ink width dir x xf y yf frame | f)!
fool (textframe input | window frame last justtabspace scanmode reply buf font | defont)!
fool (dispframe n t | strm text)!
fool (file x | dirinst fname sadr rvec leader curadr bytec dirty status nextp backp lnused num
**ch pagen version sn1 sn2 | ncheck)!

to startup task    "one loop for loopless programs"
  (☞task ← :.
    (↞in⇒(☞GLOB ← :))
   task firsttime⇒
    (repeat
     (task eachtime⇒() ⇑task lasttime))
   ⇑false)!
☞firsttime ← ☞eachtime ← ☞lasttime ← nil!

☞gray←055132. ☞ltgray←0101202. ☞dkgray←076575.
to cursor p buff         "load cursor bits from string"
    (↞show⇒(mem 255 ← :.
        BLT 0 0 281 1 0 0 16 16 2+mem 255 1 0 0))!
to showcursor t x | | current          "recursive cursor"
    (☞t←current. cursor show ☞current←:.
     ☞x←:. cursor show ☞current←t. ⇑x)!
to quit (textwindow's instances map ☞
         (each's filepart close).          "close files"
    (↞on⇒(file (:) save⇒() dp0 open. ⇑nil))    "swap out"
    file (↞then⇒(:) 'Sys.Boot.') load)!      "swap in"
to flash (do 2(mem 59 ← 040000⊟mem 59. do 100().))!

to fontheight (⇑textframe's(defont[12] + defont[14]))!

disp clear. disp's text's frame  outline with ~1.!
☞disp ← dispframe rectangle point 260 542 point 250 140!
paragraph init!
textwindow init!
☞defs←obset!
☞sched ← obset!
sched ← textwindow file 'overview.sm.' in rectangle point 10 2 point 400 250!
☞fool ← GET USER ☞DO!
disp clear. PUT USER ☞DO ☞
    (☞TOPLEVEL ← AREC. sched map ☞ (startup ☞Active ← each in TOPLEVEL)
     PUT to ☞anchor nil. PUT showcursor ☞current normalcursor.
    showcursor normalcursor 0).!
```

```
"make 'to' know about changing title line"
☞fool←#to.!
to to i t fn hdr | | anchor to
 (☞hdr←stream of vector 4.
  repeat(☞t←⍬. t is atom⇒(hdr←t)
    t is vector⇒(hdr←t. done) error 'strange definition')
  ☞hdr←hdr contents.
  ☞fn←hdr[1] eval.
   (null #fn⇒()
    null ☞t←GET fn ☞HEADER⇒()
    for i to t length (t[i]=hdr[i]⇒() done)
    i≤t length⇒()
    i<hdr length⇒()
    ☞anchor←GET fn ☞DO.
    PUT fn ☞DO hdr[hdr length].
    ⇑{hdr[1] ☞updated})
  apply to to hdr in GLOB.
  ☞fn←hdr[1] eval.
  PUT fn ☞HEADER hdr[1 to hdr length-1].
  ⇑{hdr[1] ☞defined})!
PUT to ☞to #fool.!
```

```
to textwindow a b | textpart filepart frame title | instances margin whitey windowmenu
    (↞firsttime⇒
        (frame has mp⇒
            (filepart wakeup in rectangle frame'sorigin-point 23 0 point 26 frame'sextent y.
             ☞a ← frame inset margin point 3 3.
             frame paint 12 dkgray. a paint 12 0.
             textpart wakeup in a inset point 4 2 point 4 0.
             whitey put title at frame'sorigin+point 3 2)
         ⇑false)
     ↞eachtime⇒
        (frame has mp⇒
            (startup textpart.      "**typing, selection, edits"
             filepart closed⇒(⇑false)
             button 2⇒(☞(                "**window actions"
                        (⇑nil)
                        ("put textpart to sleep now so that wrong area
                          isn't complemented after moving"
                        textpart sleep.
                        ☞a ← frame'sorigin.
                        mem 0424 ← a x+32. mem 0425 ← a y.
                        showcursor bug3cursor repeat        "move"
                            (button 2⇒(done)
                             frame dragto point 1+mx\2 my))
                        (textpart unsel.
                        showcursor bug3cursor repeat        "grow"
                            (button 2⇒(done)
                             frame paint 12 0.
                             frame growto mp.
                             SELF adjust. frame paint 12 ltgray))
                        (filepart close.    "close"
                         frame paint 12 0.
                         instances delete SELF.
                         sched delete SELF. done)
                        (apply filepart to ☞(filin) in GLOB)   "read"
                        (showcursor waitcursor filepart backup)
                        (showcursor waitcursor filepart restore)                "backup/restor
**e"
                    ) [1+windowmenu bug] eval.
                ⇑false))
            showcursor scrollcursor startup filepart⇒() "**file scrolling"
            ⇑false)
     ↞lasttime⇒(filepart sleep. textpart sleep)
     ↞adjust⇒(frame's(origin x ← 1+origin x\2.  "alignment for gray"
            extent x ← 1+extent x\4))
     ↞init⇒(☞margin ← point 3 fontheight+4.
        ☞whitey ← textframe rectangle margin margin.
        whitey's(☞scanmode←4).
        ☞instances ← obset. ☞windowmenu ← menu '  move
 grow
 close
 read
 backup
 restore ' 2)
     ↞'s⇒(⇑8 eval)
     isnew⇒(☞b ← :. ☞title ← b'sfname.
        (↞in⇒(☞frame←:)
            showcursor  bug2cursor
                repeat (☞a ← mp. button 1⇒(done))
            ☞frame ← rectangle a point 400 250)
        SELF adjust. frame paint 12 gray.
        ☞filepart ← fileseg b.
        ☞textpart ← paragraph.
        repeat (button 1⇒() done)
        instances ← SELF))!
```

```
to fileseg a b | f base len winoff winlen bits scrollrect isclosed | blip
    (☇next⇒(f set to 1 winoff.
        ☞winlen ←(len - winoff) min :.
        ⇑f next into string winlen)
     ☇firsttime⇒(scrollrect has mp⇒
            (☞bits ← scrollrect makebuff. SELF show)
        ⇑false)
     ☇eachtime⇒(scrollrect has mp⇒  "scrolling"
            (button 4⇒(SELF move textpart scrolln my)
             button 2⇒(SELF move 0 - textpart scrolln my)
             button 1⇒(☞a ← my - frame's origin y+8.
                 ☞b ← frame's extent y-10.
                 SELF move to((float a)/b) * len ipart))
        ⇑false)
     ☇lasttime⇒(scrollrect loadbuff ☞bits swap nil)
     ☇move⇒(☞winoff ← (☇to⇒(:) winoff+:)
        ☞winoff ← 0 max winoff min len.
        textpart show nosel. SELF show)
     ☇closed⇒(⇑isclosed)
     ☇wakeup⇒((☇in⇒(☞scrollrect←:))
        isclosed⇒(SELF open)
        f's(null sadr⇒
                (☞sadr ← string 512.
                 SELF set to pagen - 1 0)))
     ☇show⇒(☞a ← (float scrollrect'sextent y-10)/1+len.
        ☞b ← rectangle
            scrollrect'sorigin +point 11 8+a*winoff
            point 5 a*winlen.
        scrollrect paint 12 gray. b paint 12 0)
     ☇replace⇒(☞a←:. f set to 1 winoff.
        a length=winlen⇒(f←a. f flush) "same length"
        ☞len←len-winoff+winlen.
        (a length>winlen⇒
        (f←a[1 to winlen].                "growing"
         ☞a←a[winlen+1 to a length].
         ☞winlen←winlen+a length.
         repeat (☞b←f next into
                    string len min a length max 500-a length.
             f skipnext - b length. f←a.
             0=☞len←len-b length⇒(f←b. done)
             f←b[1 to b length-a length].
             ☞a←b[1+b length-a length to b length]))
         f←a. ☞a←winlen-a length.     "shrinking"
         ☞winlen←winlen-a.
         repeat (f skipnext a.
             ☞b←f next into string 500 min len.
             f skipnext -b length+a. f←b.
             0=☞len←len-b length⇒(done)))
         f shorten to 1·☞len←f's(bytec+512*pagen-1) f flush)
     ☇sleep⇒(☞scrollrect ← nil.
        f's(☞sadr ← nil))
     ☇filin⇒(f set to 1 0.
        showcursor readcursor filin f)
     ☇open⇒((☇first⇒() f's(☞sadr ← string 512) f reopen)
        ☞isclosed ← false. f set to end. ☞winoff ← 0.
        ☞len ← f's(bytec + 512 * pagen - 1))
     ☇close⇒(SELF sleep. f close. ☞isclosed ← true)
     ☇backup⇒(☞a ← file f'sfname+'().'.
        f reset. f copyto a. a close)
     ☇restore⇒(☞a ← file f'sfname+'().' old⇒
        (f reset. a copyto f. a close. SELF open))
     isnew⇒(☞f ← :. SELF open first. SELF sleep))!
```

```
to paragraph t | frame pdisp char changed buf loc1 loc2 p1 p2 | bs del esc dfcomp editmenu Scr
**ap Deletion Newplace
    (☙showsel⇒(pdisp's(☞buf←:. ☞last←buf length) buf.
        ☞p1 ← pdisp ptofchar loc1+1.
        ☞p2 ← (loc2=loc1⇒(p1+point 1 0)
            pdisp ptofchar loc2+1)
        pdisp show. dfcomp p1 p2)
    ☙catchup⇒(☞t ← char contents. t length=0⇒() char reset.
        loc1=loc2⇒(SELF replace t nosave)
            SELF replace t)
    ☙firsttime⇒(⇑frame has mp)
    ☙eachtime⇒
        (frame has mp⇒
            (kbck⇒              "**type-in"
                (☞Scrap ← (Newplace⇒(☞Newplace ← false. loc1)
                    loc2-Scrap length)
                ☞char←stream. repeat
                    (bs=☞t←kbd⇒
                        (SELF catchup. ☞t←0.          "backspace"
                        repeat((loc2>t⇒(☞t←t+1))
                            kbck⇒(kmap[kbck]=bs⇒(kbd) done)
                            done)
                        ☞buf ← buf[loc2+1-t to loc2] replace ''.
                        ☞loc2 ← loc2-t. ☞loc1 ← loc1 min loc2.
                        ☞changed←true. kbck⇒() SELF showsel. done)
                    t=esc⇒(SELF catchup.
                        ☞loc1 ← loc2 min Scrap.  SELF showsel. done)
                    t=del⇒(☞loc1 ← Scrap. SELF replace ''. done)
                    char←t.        "char stream catches type-ahead"
                    kbck⇒() SELF catchup.
                    kbck⇒() done)
                ☞Scrap ← buf[1+Scrap min loc2 to loc2])
            button check⇒(☞t ← mp. ☞Newplace ← true. button 4⇒
                (dfcomp p1 p2.              "**select"
                ☞t ← ¯1+pdisp charofpt t.
                loc1=loc2=t⇒(repeat(loc1=0⇒(done)     "dbl-click for token"
                        tokenish buf[loc1]⇒(☞loc1←loc1-1) done)
                    repeat(loc2=buf length⇒(done)
                        tokenish buf[loc2+1]⇒(☞loc2←loc2+1) done)
                    dfcomp ☞p1 ← pdisp ptofchar loc1+1
                        ☞p2 ← (p1 + point 1 0) max
                            pdisp ptofchar loc2+1)
                ☞loc1← ☞loc2← t.
                dfcomp ☞p1 ← pdisp's reply ☞p2 ← p1+point 1 0.
                repeat
                    (button 4⇒              "**draw out selection"
                        (☞char ← ¯1+pdisp charofpt mp.
                        char=loc2⇒()
                        ☞t ← ☞p2 swap pdisp'sreply.
                        (char=loc1⇒(☞p2←p2+point 1 0))
                        char<☞loc2 swap char⇒(dfcomp p2 t)
                            dfcomp t p2)
                    done)
                loc2 < loc1⇒
                    (☞loc2 ← ☞loc1 swap loc2.
                    ☞p2 ← ☞p1 swap p2))
                button 1⇒(☞(()          "**editing actions"
                        ((eq Scrap Deletion⇒()    "undo"
                            ☞loc1 ← loc2-Scrap length)
                            SELF replace ☞Scrap ← Deletion andselect)
                        (☞Scrap ← ☞Deletion ← SELF selection)  "copy"
                        (SELF replace Scrap andselect)     "paste"
                        (SELF replace ''. ☞Scrap ← Deletion)                "cut"
                        (SELF lasttime. showcursor xeqcursor        "do it"
                            ☞t←evapply showcursor readcursor
                                read of SELF selection
                                to ☞(eval) in TOPLEVEL.
                            filepart closed⇒(⇑false)
                            ☞Scrap ← ☞Deletion ← stringof t)
```

```
                        (showcursor readcursor parenck. SELF showsel)
                        (☞t ← file SELF selection.          "open"
                            t⇒(sched ← textwindow t) flash)
                        (☞t ← file SELF selection old."include"
                            t⇒(SELF replace ☞Scrap ← t intostring andselect) flash)
                        )[1+editmenu bug] eval)
                ⇑false "button 2 out to window"))
        ⇑false)
    ↯replace⇒(☞t ← :.
        ☞changed ← true.
        (↯nosave⇒() ☞Deletion ← SELF selection)
        ☞buf ← buf[loc1+1 to loc2] replace t.
        ☞loc2 ← loc1+t length.
        (↯andselect⇒() ☞loc1 ← loc2)
        SELF showsel)
    ↯lasttime⇒(changed⇒
            (showcursor waitcursor filepart replace buf.
             ☞changed ← false))
    ↯contents⇒(⇑buf)
    ↯selection⇒(⇑buf[loc1 + 1 to loc2])
    ↯unsel⇒(☞loc1 ← ☞loc2 ← 0. ☞p2 ← p1)
    ↯wakeup⇒((↯in⇒(☞frame ← :))
        ☞pdisp ← textframe frame. SELF show)
    ↯sleep⇒(dfcomp p1 p2. "allow going to sleep twice" ☞p2 ← p1.        ☞pdisp ← ☞buf ← nil)
    ↯show⇒(☞buf ← filepart next
        frame's((extent x/8)*extent y/ fontheight) min
            20 max core-500.
        (↯nosel⇒(SELF unsel)) SELF showsel)
    ↯scrolln⇒(⇑¯1+pdisp charofpt point frame'sorigin x :)
    ↯'s⇒(⇑8 eval)
    ↯init⇒(☞editmenu ← menu '  undo
copy
paste
 cut
do it!
(...)
open
include ' 1. ☞bs ← 010. ☞del ← 0177. ☞esc ← 033.
        ☞Scrap← ☞Deletion← ''. ☞Newplace←true.
        to dfcomp p1 p2              "complement text range"
            °(☞p1 ← :. ☞p2 ← :.
                '(p1 y < p2 y⇒
                    ²((rectangle p1 point frame's(origin+extent-p1) x fontheight) comp.
                     ☞p1 ← point frame'sorigin x p1 y + fontheight.
                     p1 y < p2 y⇒
                        '((rectangle p1 point frame'sextent x p2 y - p1 y) comp))).
             p1 y > p2 y⇒'()
             '(rectangle point p1 x p2 y point p2 x - p1 x fontheight) comp))
    isnew⇒(SELF unsel. ☞p1 ← ☞changed ← ☞char ← false))!
```

```
to parenck a b c d t lpar rpar evpar np
    (☞lpar←050. ☞rpar←051. ☞evpar ← 020. pdisp show.
     ☞c←pdisp'sreply.
     ☞b←loc1. ☞np←1. repeat
         (☞t←(buf[1 to b] find last lpar) max
              buf[1 to b] find last evpar.
          0<☞a←buf[t+1 to b] find last rpar⇒
               (☞np←np+1. ☞b←a-1)
          0=☞np←np-1⇒(☞loc1←t. done)
          0=☞b←t-1⇒(☞loc1←0. done))
     ☞b←loc2+1. ☞np←1. repeat
         (☞t←buf[b to c] find first rpar. (t=0⇒(☞t←c))
          ☞a←buf[b to t] find first lpar.
          ☞d←buf[b to t] find first evpar.
          (a=0⇒(☞a←d) d=0⇒() ☞a ← a min d)
          0<a⇒(☞np←np+1. ☞b←a+1)
          0=☞np←np-1⇒(☞loc2←t-1. done)
          c<☞b←t+1⇒(☞loc2←c-1. done))
     repeat(button 1⇒() button 4⇒() done))!

to tokenish t (☞t ← :. "test for token-chars"
     0141≤t≤0172⇒(⇑true)     "lower-case"
     060≤t≤071⇒(⇑true)  "digits"
     0101≤t≤0132⇒(⇑true)     "upper-case"
     t=025⇒(⇑true) "-" t=056⇒(⇑true) "."
     ⇑false)!
```

```
to menu pt i bits | str text thisline frame but | pt2
    (☞bug⇒(☞pt← mp-thisline center.
        text'sframe moveby pt. thisline moveby pt.
        frame moveby pt. ☞bits ← frame makebuff.
        frame paint 12 ¯1. text show. thisline comp.
        repeat(frame has ☞pt←mp⇒
            (button but⇒(thisline has pt⇒()
                text charofpt pt. ☞pt←text'sreply.
                thisline comp. thisline moveto
                    point text'sframe'sorigin x pt y.
                thisline comp)
            frame loadbuff bits.
            ⇑1+(thisline'sorigin-frame'sorigin)y/text fontheight)
            thisline comp. repeat
                (button but⇒(frame has mp⇒(thisline comp. done))
                frame loadbuff bits. ⇑0)))
    isnew⇒(☞str←:. ☞but←:. ☞text ← textframe
            rectangle ☞pt←point 0 0 point 1000 1000 with str.
        for i to str length+1
            (☞pt ← pt max text ptofchar i)
        text'sframe growto pt+point 0 text fontheight.
        ☞frame ← text'sframe inset ☞pt2←point ¯2 ¯2 pt2.
        ☞thisline←rectangle text'sframe'sorigin
            point text'sframe'sextent x text fontheight))!
```

```
normalcursor←'px|?pX_
```

should be in octal or binary !
(but smalltalk has no such convention —
how were these ever encoded ?)

```
@@⁘⁘'.!
☞bug2cursor←'✍✍☺←☺←←←☺←☺☺☺☺✍✍'.!
☞scrollcursor←'<?✍✍✍✍?<'.!
☞waitcursor←'a@sOsOsOa@'.!
☞readcursor←'
```

@≥ ≥ →(☺ { 5 {☺'.!

☞xeqcursor←'→(☺→(8q↗xF|D?|plXF_

```
@@88'.!

☞bug3cursor←'⚡⚡☺←☺←☺←←←☺☺☺☺⚡⚡'.!
```

```
"
SMALLTALK STRIKE FORMAT FONT EDITOR

**the current default font is 'st10.strike'**

**to change the current default font**
    filfont 'myfont.strike' in

**to create a font window for editing current default font at middle-click**
    showcursor bug2cursor repeat(button 1⇒(done))
    sched ← ☞Currentfont ← fontwindow textframe'sdefont at mp.

**to write the default font out**
    filfont 'myfont.strike' out

**to create a font window for editing a font other than the default font at middle-click**
    showcursor bug2cursor repeat(button 1⇒(done))
    sched ← ☞Currentfont ← fontwindow (file 'fancy14.strike' intostring) at mp.

**to put non-default font out**
        dsoff.
        ☞f ← file 'myfont.strike'.
        f ← Currentfont's font.
        f close shortened.
        dson.

**to view a non-default font in a window, past the following in the desired window and execute
****
    Active'stextpart'spdisp's(☞font ← (Currentfont's font)).
    Active'stextpart'spdisp show.

**examples of manual manipulation of Currentfont -- Currentfont is the font window the mouse c
**ursor was last in**

    Currentfont setascent 2.   **Deltas -- for entire font**
    Currentfont setascent ¯3.
    Currentfont setdescent 2.
    Currentfont setdescent ¯2.

    Currentfont setchar 0101.

    Currentfont setwidth manual 9.     **Absolute -- for char in window**

**Examples of  changing the widths of a number of characters --
    ! ! ! ! BE CAREFUL! ! ! !  -- A LARGE RANGE WILL RUN YOU OUT OF MEMORY.  If you use this f
**unction keep the ranges small and write your font often.

    setfontwidthglobal  01 to 020 fixed 10.
    setfontwidthglobal  01 to 020 delta ¯1.
"
to fontwindow "temps-" a b w x y mask newwidth | "inst vars-" frame font fontht fontl fontpt f
**ontwid fontxtabl char charx charwid cstr minchar | "class vars-" scale boxer
    (☟eachtime⇒
        (frame has mp⇒("while active"
            button 4⇒(SELF setbit mp on)     "make dot black"
            button 1⇒(SELF setbit mp off)    "make dot white"
            button 2⇒(SELF setwidth)         "grow character"
            kbck⇒(☞a ← kbd.
                a=0177⇒(SELF unshow. sched delete SELF. ⇑false)
                SELF setchar a.  a = 05 ⇒(ev)))
        ⇑false)
    ☟setbit⇒(☞a←(:)-frame'sorigin.
        ☞x←0 max (charwid-1) min a x/scale.
        ☞y←0 max (fontht-1) min a y/scale.
        boxer moveto frame'sorigin+point scale*x scale*y.
        boxer paint 12 ☞a←(☟on⇒(¯1) ☟off⇒(0)).
        BLT 12 a fontpt fontwid charx+x y 1 1 0 0 0 0)
    ☟firsttime⇒
        (frame has mp⇒(☞Currentfont ← SELF.
```

```
        SELF show "upon entry")
    ↑false)
◁lasttime⇒("upon exit")
◁show⇒(frame paint 12 0. frame outline.
    (textframe frame font font) show of cstr.
    blowup (rectangle frame'sorigin point charwid fontht) frame'sorigin scale)
◁unshow⇒(frame paint 12 0. frame outline with ~1)
◁setchar⇒(☞a ← :.                                    "011 = tab  015 = CR  040 = blank"
    a=011⇒() a=015⇒() a=040⇒()
    cstr[1] ← ☞char ← a. ☞charx ← fontl[fontxtabl+char-minchar]□
    ☞charwid ← fontl[1 + fontxtabl+char-minchar]-charx. SELF unshow.
    frame's(☞extent ← point charwid*scale fontht*scale).
    SELF show)
◁setwidth⇒((◁manual⇒(☞a ← :. frame'sextent x ← a*scale.)
                repeat(button 2⇒
                (frame outline with ~1.
                frame'sextent x ← (mx-frame'sorigin x)\scale.
                frame outline) done))
    ☞newwidth ← frame's extent x/scale.
    ☞a ← newwidth - charwid. a=0⇒()
    ☞y ← (fontwid +(a+15)/16).
    ☞w ← (fontxtabl *2)-3.
    ☞b ← font[1 to (w - 1) + 2*(fontht*(a+15)/16)].
    (☞x←leech b)[10]- y.
    x[6]- 7 +(y*fontht)+(fontl[4]- fontl[3])
    ☞b ← b[1 to b length] + font[w to font length].
    ☞x ← (PNT b)+11.
    BLT 0 0 x y 0 0 charx+charwid fontht fontpt fontwid 0 0.
    (a < 0 ⇒()
    BLT 12 0 x y charx+charwid 0 a fontht fontpt fontwid 0 0)
    BLT 0 0 x y charx+charwid+a 0 (fontwid*16)-charx+charwid
                    fontht fontpt fontwid charx+charwid 0.

    ((eq font (textframe'sdefont))⇒ (textframe's(☞defont←:) b.))

    SELF setfont b.
    showcursor waitcursor for x ← char+1 to 2+fontl[4]□
        (fontl[fontxtabl+x-minchar]-
            a+fontl[fontxtabl+x-minchar])
    "update maximum width if necessary"
    fontl[5]- fontl[5]-max newwidth.
    SELF setchar char.)
◁setascent⇒(☞a ← :. "ascent delta"
        ((fontl[7]□ + a) < 0 ⇒(☞a ← -(fontl[7])))
        ☞y ← fontl[10]□.
        (a > 0 ⇒(☞b ← string (y * a) * 2.
                b[1 to b length] ← all 0.
                ☞b ← font[1 to 18] + b[1 to b length] +
                                font[19 to font length])
            ☞b ← font[1 to 18] + font[(19+((y*-a)*2)) to font length])
        (☞x ← leech b)[7]□ ← fontl[7]□ + a.

    ((eq font (textframe'sdefont))⇒ (textframe's(☞defont←:) b.))

        SELF setfont b.
        x[6]- 7 +(y*fontht)+(fontl[4]- fontl[3])

        SELF setchar char.)
◁setdescent⇒(☞a ← :. "descent delta"
        ((fontl[8]□ + a) < 0 ⇒(☞a ← -(fontl[8]□ ))).
        ☞w ← (fontxtabl * 2 ) - 4.
        ☞y ← fontl[10]□.
        ☞b ← font[1 to 18] +font[19 to w + ((y*a)*2)].
        (w ≥ b length⇒() b[w+1 to b length]← all 0.)
        ☞b ← b[1 to b length] + font[w+1 to font length].
        (☞x ← leech b)[8]□ ← fontl[8]□ + a.

    ((eq font (textframe'sdefont))⇒ (textframe's(☞defont←:) b.))
```

```
            SELF setfont b.
            x[6]← 7 +(y*fontht)+(font1[4]← font1[3])

            SELF setchar char.)
    setfont⇒( font1 ← leech font←.
        fontpt ← (PNT font)+11.
        fontwid ← font1[10]
        ascent ← font1[7] descent ← font1[8]
        fontht ← ascent+descent.
        fontxtabl ← 11 +fontwid * fontht.
        minchar ← font1[3]

    's⇒(⇑ eval)   "in case you need to fish around"
    is⇒(ISIT eval)"for identification"
    isnew⇒(scale ← 9. SELF setfont :.
        boxer ← rectangle point 0 0 point scale-1 scale-1.
        frame ← rectangle (at. :) point scale scale*fontht.
        cstr ← string 1. SELF setchar 65 "set up an instance"))!

to setfontwidthglobal a b w x y(a ← :. to. b ← :.
                    fixed⇒( x ← :.
                            for y ← a to b
                                (Currentfont setchar y.
                                Currentfont setwidth manual x.))
                        delta. x ←:.
                            for y ← a to b
                                (Currentfont setchar y.
                                w ← x + Currentfont's(charwid).
                                (w<0⇒(w←0))
                                Currentfont setwidth manual w.)
                    )!

to blowup r s w h dest inc sinc slice z z1 p
    (r←:. p←:. s←:.
    r's(w←extent x. h←extent y)
    z ← point 1 0. z1 ← point 0 1.
            do 2(          "first do horiz, then vert"
    inc ← z*¯1. sinc ← z*s.
    slice ← rectangle r'sorigin+z*w-1 z+z1*s*h.
    dest←p+z*s*w-1.
    do w (slice paint 0 dest. dest←dest-sinc.
        slice moveby inc)      "slice it up"
    slice ← rectangle p-inc (z1*s*h)+z*s-1.
    do 1+w/s (slice paint 12 0. slice moveby sinc)
    slice ← rectangle p (z1*s*h)+z*(s*w)-1.
    do s-2 (slice paint 1 p-inc)          "spread it out"
            z←z1 swap z. h←w swap h)   "flip to do vertical"
    )!

to filfont f
  (dsoff. f ← :.
    (in⇒
     ((f ← file f old⇒() dson. ⇑false)
      textframe's
       (defont ← f intostring)
        disp'stext's(font←defont)
        textwindow's(whitey ← textframe rectangle margin margin.
            whitey's(scanmode ← 4)).)
     out⇒
      (f ← file f.
       f ← textframe's defont)).
   f close shortened. dson.)!
```

```
to clocksource nsecs d t | day date | two16 (
        ↓next⇒(☞nsecs ←
                        (0 > ☞d ← mem 379⇒(two16 + d) float d)
                         + two16*(0 > ☞d ← mem 378⇒(two16 + d) d).
                (day ≠ ☞day ← (nsecs/86400.0) ipart-27392⇒
                        (SELF calcdate))"only recalculate date daily"
                 ↑date+SELF timeof (nsecs/86400.0) fpart)
        ↓calcdate⇒(☞date ← (0<day<367⇒('          ' +
                ☞('Mon' 'Tues' 'Wednes' 'Thurs' 'Fri' 'Satur' 'Sun')[1+(day+2) mod 7]+'day
'+              (for t ← 12 to 1 by ¯1 (day>☞d←☞(0 31 60 91 121 152 182 213 244 274 305 335 36
**6)[t]⇒(done)) ☞('January' 'February' 'March' 'April' 'May' 'June' 'July' 'August' 'September
**' 'October' 'November' 'December')[t]+ ' ' + (stringof day-d) + ', 1976
')) '       Date not set
'))
        ↓timeof⇒(☞t←:. ↑'       ' + (☞t←t*24. ☞d←(t<12⇒('am') 'pm') stringof (t<13⇒(t ipart) t
** ipart-12)) + ':' + (☞t←t fpart*60 ipart. t>9⇒(stringof t) '0'+stringof t)+d)
        ↓firsttime⇒(↑false) ↓closed⇒(↑false)
        ↓wakeup⇒(↓in.) ↓sleep⇒() ↓show⇒() ↓close⇒() ↓replace⇒(:)
        isnew⇒(☞day←0. ☞date←''. ☞two16←256.0*256))!

☞Clock ← textwindow file 'clock' in rectangle point 360 0 point 150 56. Clock's(filepart close.
** ☞filepart ← clocksource). sched ← Clock.
```

```
to EventNotice prop | ob msg etime prev next (

  ≪'s⇒(
    ☞prop ← 8.
    ≪←⇒(⇑prop ← :)
    ⇑prop eval)

  isnew⇒(
    ☞ob ← :.
    ☞msg ← :.
    ☞etime ← :.
    ☞prev ← :.
    ☞next ← :)

  ≪is⇒(ISIT eval)
)!

to SQS finger newOb time msg | now current set (

  isnew⇒
    (☞set ← EventNotice 0 0 0.0 0 0.
     set's next ← EventNotice 0 0 1.0e1000 set 0.
     ☞now ← 0.0)

  ≪schedule⇒
    (☞newOb ← :.
     ☞msg ← (≪for⇒(:) ☞(run)).
     ☞time ← (≪at⇒(now + :) now).
     ☞finger ← set's next.
     repeat
      (time ≥ finger's etime⇒
        (☞finger ← finger's next.
         again)
       ☞newOb ← EventNotice newOb msg time finger's prev finger.
       newOb's prev's next ← newOb.
       finger's prev ← newOb.
       done))

  ≪activate⇒
    (☞newOb ← SELF remove.
     ☞now ← newOb's etime.
     evapply ☞current ← newOb's ob to newOb's msg)

  ≪remove⇒
    (☞finger ← set's next.
     finger's next's prev ← finger's prev.
     finger's prev's next ← finger's next.
     ⇑finger)

  ≪full⇒(⇑0 ≠ set's next's ob)

  ≪print ⇒ (☞finger ← set's next.
            repeat (0 = finger's ob ⇒ (done)
                    finger's ob print. ☞finger ← finger's next))

  ≪'s⇒(⇑8 eval)

  ≪is⇒(ISIT eval)
)!

to blob | x y sides len 🐢
  (isnew⇒
    (☞sides ← 0. ☞x←rand mod 25. ☞y←500 - rand mod 25.
     ☞🐢 ← turtle.
     🐢's width ← 2.
     SELF draw.)
   ≪run⇒
    (SELF undraw.
     SELF draw.
```

```
    Simpula schedule SELF at rand mod 200)
  ⇦draw⇨
   (☺ penup goto ☞x ←(x + ¯25 + rand mod 100) mod 500 ☞y ←(¯25 + y
     + rand mod 100) mod 500 pendn up.
    poly (☞sides ←(sides + 1) mod 10 ) ☞len←rand mod 40.
    ☺ penup goto x y pendn up.)
  ⇦undraw⇨
   (☺ white.
    poly sides len.
    ☺ black)
  ⇦is⇨(ISIT eval))!

to poly s l (☞s ← :. ☞l ← :. do
    (s)
    (☺ go l turn 360 / s))!
```

```
" <Tesler>Projector.Ft "

" April 11, 1977  12:53 PM "

☞Projector ← class
    (
    (
        grayPattern ink keepRect proj rectVec
        screenDXY slideDXY screenHair1 screenHair2
        screenPt screenRect slidePt slideRect
    )
    (
        screenFrame para altoStyle charMinXY charDXY visibleScreenRect screen
        "The above instance variables must be in the order listed"
        visibleSlideRect slide symbolism reduction translation
    )
    ()()) ()()!

Projector'sinitcode ← ☞
    (↑new.
    ☞slide←:. ☞slidePt←:. ☞screen←:. ☞visibleScreenRect←:.
    ☞symbolism←:.  ☞reduction←:.
    ☞visibleSlideRect ← slidePt ⊡ slidePt + visibleScreenRect dXY*reduction.
    SELF ComputeTranslation.
    ☞screenFrame ← 0'0 ⊡ 0'0.
    ).☞new!

Projector understands ☞=☞ (↑eq SELF :)!

Projector understands ☞'s☞ (☞ink←8. ↑←⇒(↑ink←:) ↑ink eval)!

Projector understands ☞CacheText☞
    (
    ☞para←:. ☞slideRect←:. ☞altoStyle←:.
    ☞screenFrame ← SELF screenRectOf slideRect.
    )!

Projector understands ☞charNearScreenPt☞
    (CODE 16 SUB 1)!

Projector understands ☞charNearSlidePt☞
    (↑SELF charNearScreenPt SELF screenPtOf :)!

Projector understands ☞ComputeTranslation☞
    (☞translation ← visibleScreenRect minXY - visibleSlideRect minXY/reduction)!

Projector understands ☞FitText☞
    (
    screenFrame growby 0'9999.
    ☞ink ← (SELF screenHairBeforeChar 1+para length) maxY - screenFrame maxY.
    screenFrame growby 0'ink
    ↑ reduction y * ink+9999
    )!

Projector understands ☞HighlightChars☞
    (SELF HighlightScreenHairs SELF screenHairBeforeChar : SELF screenHairBeforeChar :)!

Projector understands ☞HighlightScreenHairs☞
    (
    ☞screenHair1←:. ☞screenHair2←:.
    screenHair1 minY = screenHair2 minY⇒
        (SELF HighlightScreenRect screenHair1 minXY ⊡ screenHair2 maxXY)
    ☞screenRect ← screenFrame minX'screenHair1 maxY ⊡ screenFrame maxX'screenHair2
    minY.
    SELF HighlightScreenRect screenHair1 minXY ⊡  screenRect maxXminY.
    SELF HighlightScreenRect screenRect.
    SELF HighlightScreenRect screenRect minXmaxY ⊡  screenHair2 maxXY.
    )!

Projector understands ☞HighlightScreenRect☞
    ((visibleScreenRect ⊡:) comp)!
```

```
Projector understands ☞HighlightSlideHairs☞
    (SELF HighlightScreenHairs SELF screenRectOf : SELF screenRectOf :)!

Projector understands ☞MakeScreenRectBe☞
    (
    ☞screenRect←:.
    screenFrame moveby screenRect minXY - visibleScreenRect minXY.
    ☞visibleScreenRect ← screenRect.
    SELF ComputeTranslation.
    visibleSlideRect growto SELF slidePtOf screenRect maxXY
    )!

Projector understands ☞MakeSlideRectBe☞
    (
    ☞slideRect←:.
    screenFrame moveby SELF screenDXYOf visibleSlideRect minXY - slideRect minXY.
    ☞visibleSlideRect ← slideRect.
    SELF ComputeTranslation.
    visibleScreenRect growto visibleSlideRect dXY/reduction
    )!

Projector understands ☞MoveIntoScreenRect☞
    (
    ☞screenRect←:.
    ☞rectVec ← visibleScreenRect ⊟ screenRect.
    ☞screenDXY ← screenRect minXY - visibleScreenRect minXY.
    ☞keepRect ← visibleScreenRect ⊡screenRect - screenDXY.
    SELF MakeScreenRectBe screenRect.
    SELF ShowBlting keepRect screenDXY rectVec.
    ⇑rectVec
    )!

Projector understands ☞Paint☞
    (
    ☞grayPattern←:. ☞slideRect←:. ☞ink←:.
    (SELF visibleScreenRectOf slideRect) paint ink+12 grayPattern
    )!

Projector understands ☞Receive☞
    (
    ☞slideRect ← :. ☞proj ← :. ☞slidePt ← :.
    ☞slideDXY ← slideRect minXY - slidePt.
    ☞destProj ← SELF subProjectorFrom slide slideRect.
    ☞keepRect ← proj visibleScreenRectOf destProj'svisibleSlideRect - slideDXY.
    ☞rectVec ← (proj visibleScreenRectOf slideRect - slideDXY) ⊟destProj'svisibleScreenRect.
    destProj ShowBlting keepRect SELF screenDXYOf slideDXY rectVec.
    ⇑rectVec
    )!

Projector understands ☞screenDXOf☞
    (⇑ :/reduction x)!

Projector understands ☞screenDXYOf☞
    (⇑ :/reduction)!

Projector understands ☞screenDYOf☞
    (⇑ :/reduction y)!

Projector understands ☞screenHairBeforeChar☞
    (
    CODE 16 SUB 2.
    ⇑rectangle charMinXY charDXY
    )!

Projector understands ☞screenHairBeforeThatChar☞
    (⇑rectangle charMinXY charDXY)!

Projector understands ☞screenPtOf☞
    (
    ☞slidePt←:.
```

```
    ↑(SELF screenXOf slidePt x) @ (SELF screenYOf slidePt y)
    )!

Projector understands 'screenRectOf'
    (
    'slideRect←:.
    ↑(SELF screenPtOf slideRect minXY) □ (SELF screenPtOf slideRect maxXY)
    )!

Projector understands 'screenXOf'
    (↑ translation x + :/reduction x)!

Projector understands 'screenYOf'
    (↑ translation y + :/reduction y)!

Projector understands 'ScrollBy'
    (SELF ScrollTo : + visibleSlideRect minXY)!

Projector understands 'ScrollTo'
    (
    'slidePt←:.
    'screenDXY ← visibleScreenRect minXY - SELF screenPtOf slidePt.
    visibleSlideRect moveto slidePt.
    SELF ComputeTranslation.
    screenFrame moveby screenDXY.
    'keepRect ← visibleScreenRect ⊃visibleScreenRect - screenDXY.
    'rectVec ← visibleScreenRect ⊟keepRect+screenDXY.
    SELF ShowBlting keepRect screenDXY rectVec
    )!

Projector understands 'Send'
    (
    'slideRect ← :. 'proj ← :. 'slidePt ← :.
    'slideDXY ← slidePt - slideRect minXY.
    'destProj ← proj subProjectorFrom slide slideRect + slideDXY.
    destProj TranslateSlide (0@0)-slideDXY.
    'keepRect ← SELF visibleScreenRectOf destProj'svisibleSlideRect.
    'rectVec ← (SELF visibleScreenRectOf slideRect) ⊟destProj'svisibleScreenRect.
    destProj ShowBlting keepRect SELF screenDXYOf slideDXY rectVec.
    ↑rectVec
    )!

Projector understands 'Show'
    (
    visibleScreenRect paint 12 0.
    slide ShowIn SELF
    )!

Projector understands 'ShowBlting'
    (
    'screenRect←:. 'screenDXY←:. 'rectVec←:.
    screenRect copyto screenRect minXY + screenDXY.
    for ink to rectVec length do
        (('regenerateRect←rectVec[ink]) nonEmpty⇒ (regenerateRect paint 12 0)).
    'rectVec ← visibleScreenRect ⊟screenRect.
    for ink to rectVec length do
        (('regenerateRect←rectVec[ink]) nonEmpty⇒ ((SELF subProjectorTo slide
        regenerateRect) Show)).
    )!

Projector understands 'ShowText'
    (CODE 16 SUB 0)!

Projector understands 'slide'
    (↑ slide)!

Projector understands 'slideDXOf'
    (↑ :*reduction x)!

Projector understands 'slideDXYOf'
    (↑ :*reduction)!
```

```
Projector understands ☞slideDYOf☞
    (⇑ :*reduction y)!

Projector understands ☞slideHairBeforeChar☞
    (⇑ SELF slideRectOf SELF screenHairBeforeChar :)!

Projector understands ☞slideHairBeforeThatChar☞
    (⇑ (SELF slidePtOf charMinXY) ⊡ ( SELF slidePtOf charDXY)).

Projector understands ☞slidePtOf☞
    (
    ☞screenPt←:.
    ⇑ (SELF slideXOf screenPt x) ' (SELF slideYOf screenPt y)
    )!

Projector understands ☞slideRectOf☞
    (
    ☞screenRect←:.
    ⇑ (SELF slidePtOf screenRect minXY) ⊡ (SELF slidePtOf screenRect maxXY)
    )!

Projector understands ☞slideXOf☞
    (⇑ reduction x * :-translation x)!

Projector understands ☞slideYOf☞
    (⇑ reduction y * :-translation y)!

Projector understands ☞subProjectorFrom☞
    (
    ☞ink ← :. ☞slideRect ← visibleSlideRect ⊠:.
    ⇑ Projector ink slideRect minXY screen SELF screenRectOf slideRect symbolism reduction
    )!

Projector understands ☞subProjectorTo☞
    (
    ☞ink ← :. ☞screenRect ← visibleScreenRect ⊠:.
    ⇑ Projector ink SELF slidePtOf screenRect minXY screen screenRect symbolism reduction
    )!

Projector understands ☞TranslateSlide☞
    (
    visibleSlideRect moveby :.
    SELF ComputeTranslation
    )!

Projector understands ☞visibleScreenRect☞
    (⇑ visibleScreenRect)!

Projector understands ☞visibleScreenRectOf☞
    (⇑ visibleScreenRect ⊠SELF screenRectOf :)!

Projector understands ☞visibleSlideRect☞
    (⇑ visibleSlideRect)!
```