```
printing 0.
&( ***  DP0:stdnu   DP1:stdnu   stdnu ***)
to nifu nwch (
        repeat (( NE 0 "nbs←mouse 7)?(  done with cons frmse cons nbs cons mouse  **
8 mouse 9)
                -1<"nwch←rdch?( done with cons frkey nwch)
        ))

to winwat tstno nxin ( :tstno.
        repeat ("tstno←tstno-1. tstno=0?(done)
        "nxin←nifu.
        (nxin first)=frmse?("nbs←(nxin rest) first. &(check button state).
                            "nxin←(nxin rest) rest.
                            "mx←nxin first.
                            "my←nxin rest.
&(wfind finds relevant window and puts it in front of wlist;recog recognizes).
&(input command and puts all relevant info., incl. command name, in comvec).
                            (wfind mx my)?("comvec←recog mx my.
                                    &(***  (comvec[1]) comvec***).again)
                            ERROR (mouse not in existing window) . again
                            )
        (nxin first)=frkey?((wlist first) keyin nxin rest.
                                    again ))
        )

to wfind mx my (:mx. :my.
        (null #wlist)?(&(! )EMPTY)
        ((wlist first) msipt mx my)?(&(! )#wlist)
        "nlist←wlist rest. "onebe←#wlist
        repeat(
        (null #nlist)?(done with EMPTY)
        ((nlist first) msipt mx my)?(onebe rest←nlist rest.
                                    nlist rest←#wlist.
                                    "wlist←#nlist.
                                    done with #wlist)
        "onebe←#nlist. "nlist←nlist rest.
        )
      )

to chkmouse mx my  (:mx. :my. ulx<(mx)<ulx+wd?(uly<(my)<uly+ht))

to stwin lptr dchk : type ulx uly ht wd strvec curstrip curx cury  (
        "@INSTANCE←ALLOC 9.
        "type←#@stwin.
        :ulx. :uly. :ht. :wd.
        "lptr←#wlist.
         repeat (
          (null #lptr)?(done)
          (lptr first) ckbds ulx uly ht wd.
          "lptr←lptr rest
          )
        "wlist←cons #@INSTANCE #wlist.
        "dchk←dispwindow ulx uly ht wd.
        &((dchk first#0)?(**find error and correct**)).
      "strvec←vector[20].
      "curstrip←strip uly.
      curstrip init.
      #@INSTANCE
      )
to @stwin tmpx tmpy tmph tmpw inity lsu lsd index line notlaststrip: type ulx uly **
ht wd strvec curstrip curx cury(
```

- Can only return 1 value so the cons? a bunch together
- operates w/in context of current window via ulx and uly
- needed to create a window of your own kind
- what window does when it's set up
- winds down window list to see if within limits since lptr is a window∈stwin it "knows" about token "ckbds" so apply value is used
- is this # ~ε or what
- ? does DISPWINDOW change ulx or uly
- curstrip is instance & class strip so token "init" known to it
- now start seeing window tokens ↓
- get block of storage

```
        %ckbds?(:tmpx. :tmpy. :tmph. :tmpw. &( **ck bds and adjust**))
        %msipt?(:tmpx. :tmpy.
                 (chkmouse tmpx tmpy)? (dispwindow ulx uly ht wd. &(ok this tim **
e)
                                        setxy curx cury. &(! )true) )
        %keyin?("lsu←"lsd←0.:newchar. "notlaststrip←EMPTY.
                "inity←getxy y.
                CA=curstrip insert curstrip length curx newchar?(shftlines.
                                curstrip finish."cury←getxy y.
                                "curstrip←strip cury. curstrip init )
                restorewind strvec)
        %insert?(:tmpx. :tmpy. "lsu←"lsd←0. &(# lines scrolled during edit)
                "inity←fndstrip. "notlaststrip←NE index strvec length.
                "newchar←rdch.&(really want nifu).
                CA=(strvec[index]) insert line tmpx newchar ?(shftlines.CA)
                restorewind strvec.RESET)
        )

to shftlines nlines limit index tmp (
        &(fix strvec to reflect lines scrolled up out of window)
        "nlines←lsu. "limit←strvec length. "index←0.
          repeat(0=nlines?(done)
                 limit<"index←index+1?(ERROR (scroll up).done)
                 nlines<"tmp←(strvec [index]) length?(done)
                 "nlines←nlines-tmp.
                 "tmp←(strvec length)-1.
                 &(shift strvec by tmp lines)
                 copyitems tmp from strvec at 2 to strvec at 1.
                 strvec length←tmp)
         nlines>0?((strvec[index]) shftlines up nlines)
         &(corresponding part for lines shifted down)
         "nlines←lsd."index←strvec length.
          repeat(0=nlines?(done)
                 0>"index←index-1?(ERROR (scroll down). done)
                 nlines<"tmp←(strvec[index]) length?(done)
                 "nlines←nlines-tmp.
                 strvec length←(strvec length)-1)
         nlines>0?((strvec[index]) shftlines down nlines))

to fndstrip limit (
         "line←(fontht+tmpy-uly)/fontht.
          "index←0. "limit←strvec length.
          repeat (limit<"index←index+1?(ERROR (can't find strip for mouse))
                 (line-1)<"inc←(strvec[index]) length?(done)
                 "line←line-inc)
         uly+fontht*line-1)


to restorewind strvec index limit sxy (:strvec.
         "limit←strvec length. "index←0.
         clear ulx uly ht wd.
         "sxy←getxy.
         setxy ulx uly.
         repeat (limit<"index←index+1?(done)
                 (strvec[index]) regenerate)
         setxy sxy first sxy rest)
to strip : type topy endy linevec newchar (
         "@INSTANCE←ALLOC 5.
         "type←#@strip.
         :topy.
         "endy←topy+fontht.
```

STDNU

```
        strvec append ← #@INSTANCE.
        "linevec←vector[10]. &(points to lines of strip).
        linevec append ← buffer[30]. &(char. string for current input)
        "newchar←buffer[1].newchar length←1.
        #@INSTANCE)
to @strip othy tmpx linel linen retlist tmp tmp2 : type topy endy linevec newchar **
 (
        %ckbds?(:othy. topy<(othy)<endy )
        %length?(linevec length)
        %init?(@INSTANCE print #PROMPT. "curx←getxy x. "cury←getxy y)
        %finish?(newline. (linevec last) append ←CR."endy←getxy y.
                &("index←0.lindx←linevec length.
                repeat(lindx<"index←index+1?(done)
                        bread linevec[index])))
        %insert?(:linel. :tmpx. "retlist←stins :tmp.
                (CA=retlist first?(update retlist rest))retlist first)
        %shftlines?(:tmp. "tmp2←(linevec length)-tmp.
                    %up?(copyitems tmp2 from linevec at tmp+1 to linevec at 1.
                        linevec length←tmp2)
                    %down?(linevec length←tmp2)
                    ERROR  (strip shift) )
        %regenerate?("tmp←linevec length. "tmp2←0.
                     repeat(tmp<"tmp2←tmp2+1?(done)
                            pstring linevec[tmp2]. newline))
        %print?(:bname. pstring #bname.
                "tmp←linevec last."tmp2←tmp length.
                tmp length←tmp2+bname length.
                copyitems bname length from bname at 1 to tmp at 1 + tmp2)

 )
to stins index taillen tailline taill tail2 headline insbuf nxin nlvec(
        "index←fndchar. edbufinit index. :nxin.
        repeat ( nxin=CA?(nlvfin. done with cons CA nlvec)
                 nxin=RESET?(done with cons RESET nil)
                 (nxin=BS?(0<insbuf length?(backspace insbuf last. printail.
                                            insbuf length←(insbuf length)-1)
                           "insbuf←nlvec last. backspace insbuf last.
                            insbuf length←(insbuf length)-1.
                            nlvec length←(nlvec length)-1.
                            &(fix disp. ptrs). setxy ulx (getxy y)-fontht.
                            setdisp 0. pstring insbuf. setdisp 1)
                  nxin=CR?(insbuf append←CR.nlvec append←insbuf.
                           "insbuf←buffer[30]. newline )
                  newchar[1]←nxin.                                          **
                  (0 < pstring #newchar?(newline. pstring #newchar.
                        insbuf append←CR. nlvec append←insbuf.
                                          "insbuf←buffer[30]))
                   insbuf append←nxin. printail)
                  "nxin←rdch.&(really want nifu)   )   )


  to nlvfin hlen iblen tllen t2len nlvl (
        "hlen←headline length. "iblen←insbuf length.
        "tllen←taill length."t2len←tail2 length.
        0=nlvec length?("nlvl←buffer[hlen+iblen+tllen].
                        copyitems hlen from headline at 1 to nlvl at 1.
                        copyitems iblen from insbuf at 1 to nlvl at hlen+1.
                        copyitems tllen from taill at 1 to nlvl at hlen+iblen+1.
                        nlvec append←nlvl.
                        0<t2len?(nlvec append←tail2))
        "nlvl←nlvec[1]."nlvllen←nlvl length. nlvl length←hlen+nlvllen.
        copyitems nlvllen from nlvl at 1 to nlvl at hlen+1.
        copyitems hlen from headline at 1 to nlvl at 1.
```

```
        insbuf length←iblen+tllen.
        copyitems tllen from taill at 1 to insbuf  at iblen+1.
        nlvec append←insbuf.
        0<t2len?(nlvec append tail2)
 )

to fndchar llimit (setxy ulx topy+(linel-1)*fontht.
                "headline←linevec[linel].
                 "index←0. "llimit←headline length.
                 setdisp 0.
                 repeat (llimit<"index←index+1?(done)
                        newchar[1]←headline[index].pstring newchar.
                        tmpx<getxy x?(done))
                 setdisp 1.index)

 to edbufinit index (:index
                "taillen←(headline length)-index-1.
                "tailline←buffer[taillen].
                tailline length←taillen.
                "taill←buffer[1]."tail2←buffer[1].
                copyitems taillen from headline at index to tailline at 1.
                headline length←index-1.
                "linen←linel. &(index of last changed line)
                "nlvec←vector[5]. &(vector of changed lines).
                "insbuf←buffer[10] )

to printail sxy ("sxy←getxy.
                 (0="nchar←pstring tailline?(tail2 length←0.makecopy #tailline #t **
aill)
                     (1=nchar?(taill length←0. makecopy #tailline #tail2)
                       taill length←nchar-1.
                       copyitems nchar-1 from tailline at 1 to taill at 1.
                       tail2 length←(tailline length)-nchar.
                       copyitems tail2 length from tailline at nchar to tail2 at  **
1)
                      taill append←CR. newline. pstring tail2)
                  setxy sxy first sxy rest)

to makecopy buf1 buf2 blen (:buf1. :buf2.
                        "blen←buf1 length.
                        "buf2 length←blen.
                        copyitems blen from buf1 at 1 to buf2 at 1)

to update tmp tmp2 repvec (:repvec.
                "tmp←linevec length. "tmp2←repvec length.
                 copyitems tmp-linen from linevec at linen+1 to linevec at linel+ **
tmp2.
                 linevec length←tmp+tmp2-linen+1-linel.
                 0<repvec length?(copyitems tmp2 from repvec at 1 to linevec at 1 **
inel))

to newline (notlaststrip?(scrolldown.dispnewln)
            0>dispnewln?(scrollup))

to scrollup tmpy ("tmpy←getxy y."lsu←lsu+1.
                copy ulx uly+fontht tmpy-uly wd ulx uly 1. setxy ulx tmpy)

to scrolldown tmpy ("tmpy←fontht+getxy y. "lsd←lsd+1.
                  tmpy=uly+ht-fontht?("notlaststrip←EMPTY.clear ulx tmpy fontht  **
wd)
                   copy ulx tmpy uly+ht-tmpy+fontht wd ulx tmpy+fontht 1)
```

```
&(editing commands).

to INSERT xcoord ycoord (:xcoord. :ycoord. (wlist first) insert xcoord ycoord)

to insert (
  repeat( repeat(1=mouse 1?(done)). repeat(0=mouse 1?(done)).
          "xcoord←mouse 8."ycoord←mouse 9.
          &(showcursor at xcoord ycoord)
          (wfind xcoord ycoord)?(INSERT xcoord ycoord. done)
          ERROR (mouse not in existing window). again)   )

to iconinsert comvec (:comvec. INSERT (comvec[2]+comvec[3])/2 comvec[4]-fontht/2)
printing 1.

restart.
createglobals.
stwin 0 0 256 256.
```

```
printing 0.
to edit (ov "ocg)
to debug nn(:nn.print "D . print nn.repeat(eval read))
to winx ("x←stwin 0 0 127 256)
to winy ("y←stwin 0 128 127 256)
to & (:".0).  &(define comment brackets )

&(  ****INITIALIZATION****)

"true←1.
"PROMPT←buffer[2].      &(prompt char = ->)
PROMPT append←45.
PROMPT append←62.
"CA←27.           &(escape)
"RESET←92..             &(<backslash>)
"BS←127.                &(<backspace>)
"CR←13.         &(<CR>)
"DEL←255.               &(<DEL>)
"wlist←nil.
"maxht←256.
"maxwd←256.
"fontht←16.
"frmse←1.
"frkey←0.

&(****AUXILIARY PROCEDURES****)

to setup (ov "setup)
to init (setup. setdisp 1. )
to restart(init. "wlist←nil.)
to recog mx my (:mx. :my. 0)

to rdch (@CODE 60).             &(read a character from NOVA keybd)
to prch char (:char. @CODE 65). &(send char to NOVA keybd).

to cons : type first rest ("@INSTANCE←ALLOC 3. "type←#@CONS. :first . :rest. #@IN **
STANCE)
to @CONS : type first rest (
%first?(%←?(:first.)#first)
%rest?(%←?(:rest.)#rest)
#@SELF)

to ERROR msg ( :"msg. print msg)

to NE arg1 arg2 (:arg1 = :arg2 ?(EMPTY) true)

&(****INTERFACE TO DISPLAY MACHINE CODE****)

to dispwindow ulx uly ht wd (:ulx :uly :ht :wd . @CODE 50 . &(! )cons ulx uly)

to pstring sptr (:sptr. @CODE 53 . &(! )sptr)

to dispnewln retn over (@CODE 54 . &(! )retn)

to getxy xx yy (@CODE 52 .
            %x?(&(! )xx)
            %y?(&(! )yy)
            &(! )cons xx yy)

to setxy x y(:x :y . @CODE 51 . &(! )cons x y)
```

tells which break pt
reads from keyboard until say DONE
as long as no error else returns to very top

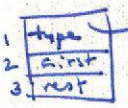
get binding & instance & return as return value

creates

pause

runmode

gives you the storage



code for cons called @cons

look it up but don't run it

SINIT

```
to setdisp n (:n . @CODE 57. &(! )n)

to clear ulx uly ht wd (:ulx :uly :ht :wd . @CODE 55 . &(! )cons ulx uly)

to copy orgx orgy ht wd desx desy clsw (
        :orgx :orgy :ht :wd :desx :desy :clsw .
         @CODE 56.
         &(! )cons orgx orgy)

to tab n (:n . @CODE 59. &(! )n)

to backspace chrcd (:chrcd . @CODE 58. &(! )chrcd)

printing 1.
```