File created: 5-Dec-2023 00:12:04 **{WMEDLEY}<library>sketch>SKETCH-EDIT.;1**

edit by: rmk

changes to: (RECORDS TEXTELTSELECTION)

previous date: 21-Aug-2021 20:50:04 {WMEDLEY}<library>sketch>SKETCHEDIT.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ SKETCH-EDITCOMS
       ((COMS (* selection functions)
              (FNS BUTLAST CHAR.BEGIN CLOSEST.CHAR CLOSEST.LINE FLASHW HILITE.LINE HILITE.TEXT IN.TEXT.EXTEND
                   INIMAGEOBJ INTEXT NEW.TEXT.EXTEND NEW.TEXT.SELECTIONP NTHCHARWIDTH NTHLOCALREGION ONCHAR
                   SHOW.EXTENDED.SELECTION.FEEDBACK SHOW.FEEDBACK SHOW.FEEDBACK.BOX SELECTION.POSITION
                   SKED.CLEAR.SELECTION SKETCH.CLEANUP SK.ENTER.EDIT.CHANGE SKED.REMOVE.OTHER.SELECTIONS
                   SKED.EXTEND.SELECTION SKED.MOVE.SELECTION CREATE.TEXT.SELECTION SKED.SELECTION.FEEDBACK
                   SKED.SET.EXTENDSELECTION SKED.SET.SELECTION LINE.BEGIN SELECTION.GREATERP SK.WORD.BREAK.CLASS
                   SK.GETSYNTAX)
              (DECLARE%: DONTCOPY (RECORDS TEXTELTSELECTION))
              (UGLYVARS IN.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.SHADE
                     SELECTION.HIGHLIGHT.SHADE)
              (GLOBALVARS IN.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.SHADE
                     SELECTION.HIGHLIGHT.SHADE))
        (COMS (* editting functions)
              (FNS WB.EDITOR SK.TTYENTRYFN SK.TTYEXITFN SKED.INSERT \SKED.INSERT FIRST.N.ELEMENTS
                   SKED.CREATE.NEW.TEXTBOX SKED.CHARACTERPOSITION SKED.LINE.AND.CHAR#
                   \SKED.DELETE.WORD.FROM.STRING \SKED.INSERT.CHARS.TO.STR JOINCHARS STRINGFROMCHARACTERS
                   GETALLCHARS CLEANUP.EDIT SKED.NEW.TEXTELT))
        (COMS (* line adding functions)
              (FNS MAP.SCREEN.POSITION.ONTO.GRID NEAREST.ON.GRID SK.MIDDLE.TITLEFN WB.BUTTON.HANDLER
                   WB.ADD.NEW.POINT WB.DRAWLINE WB.RUBBERBAND.POSITION SK.RUBBERBAND.FEEDBACKFN
                   RESET.LINE.BEING.INPUT)
              [P (* Was MODERNIZE loaded before?)
                 (CL:WHEN (GETD 'MODERNWINDOW.SETUP)
                     (MODERNWINDOW.SETUP 'WB.BUTTON.HANDLER))]
              (FNS NEAREST.EXISTING.POSITION WB.NEARPT LASTMOUSEPOSITION))))
```

(* * selection functions)

```
(DEFINEQ
```

(**BUTLAST**

```
  [LAMBDA (LST)                                              (* rrb "17-JUL-83 13:58")
```

(* returns a list that has everything but the last element of the list.)

```
    (COND
       ((OR (NULL LST)
            (NULL (CDR LST)))
        NIL)
       (T (CONS (CAR LST)
                (BUTLAST (CDR LST])
```

(**CHAR.BEGIN**

```
  [LAMBDA (CHAR# LINE# TEXTELT STRM)                         (* rrb "14-Jan-85 15:40")
```

(* determines the x position of the first bit of character CHAR# in LINE# of TEXTELT.)

```
    (PROG ((LTEXT (fetch (SCREENELT LOCALPART) of TEXTELT))
           TEXT XPOS LFONT LREGION)
          (SETQ TEXT (CAR (NTH (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of LTEXT)
                               LINE#)))
          [SETQ XPOS (fetch (REGION LEFT) of (SETQ LREGION (CAR (NTH (fetch (LOCALTEXT LINEREGIONS) of LTEXT)
                                                                     LINE#]
          (COND
             ((EQ CHAR# 0)                                   (* before the first character.)
              (RETURN XPOS)))
          (SETQ LFONT (fetch (LOCALTEXT LOCALFONT) of LTEXT))
          (RETURN (IPLUS XPOS (COND
                                 ((IMAGESTREAMTYPEP STRM 'HARDCOPY)
                                                             (* hardcopy streams must pass the stream so correction in widths is accounted for.)
                                  (DSPFONT LFONT STRM)
                                  (STRINGWIDTH (SUBSTRING TEXT 1 CHAR#)
                                         STRM))
                                 ((FONTP LFONT)
                                  (for I from 1 to CHAR# sum (CHARWIDTH (CHCON1 (NTHCHAR TEXT I))
                                                                        LFONT)))
                                 (T
```

(* if it is printed in shade, put cursor a percentage of the way across the area.)

```
                                    (IQUOTIENT (ITIMES CHAR# (fetch (REGION WIDTH) of LREGION))
                                           (NCHARS TEXT])
```

(CLOSEST.CHAR

```
  [LAMBDA (XPOS LINE# TEXTELT STRM)                                    (* rrb "28-Apr-85 15:48")
```

(* * determines the the slot between characters that is closest to XPOS.
it will return 0 if the position is in the first half of the first character or before the first character)

```
    (PROG ((LTEXT (fetch (SCREENELT LOCALPART) of TEXTELT))
           TEXT LREGION LFONT LEFT THISCHARWIDTH)
          (SETQ TEXT (CAR (NTH (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of LTEXT)
                               LINE#)))
          (SETQ LFONT (fetch (LOCALTEXT LOCALFONT) of LTEXT))
          (RETURN (COND
                     ((IGREATERP [SETQ LEFT (fetch (REGION LEFT) of (SETQ LREGION
                                                                      (CAR (NTH (fetch (LOCALTEXT LINEREGIONS)
                                                                                       of LTEXT)
                                                                                LINE#]
                            XPOS)                                      (* before the first character.)
                      0)
                     ((IGEQ XPOS (IPLUS LEFT (fetch (REGION WIDTH) of LREGION)))
                                                                       (* past the rightmost character.)
                      (NCHARS TEXT))
                     [(IMAGESTREAMTYPEP STRM 'HARDCOPY)
```

(* hardcopy stream code is cobbled from STRINGWIDTH. Must be done because widths of characters is not an integral number of points and error would accumulate through the line.)

```
                      (PROG ([FONT (FONTCREATE LFONT NIL NIL NIL (STREAMPROP STRM 'HARDCOPYIMAGETYPE]
                             (LEFTMICAPOS (ITIMES (CONSTANT IMICASPERPT)
                                                  LEFT))
                             (XMICAPOS (ITIMES (CONSTANT IMICASPERPT)
                                               XPOS)))
                            (COND
                               [(STRINGP TEXT)                         (* assumes PRIN1 mode.)
                                (RETURN (for C instring TEXT as CHAR# from 1
                                           when (IGREATERP [SETQ LEFTMICAPOS (IPLUS LEFTMICAPOS
                                                                                    (SETQ THISCHARWIDTH
                                                                                     (CHARWIDTH C FONT]
                                                           XMICAPOS)
                                           do (RETURN (COND
                                                         ((IGREATERP (IDIFFERENCE LEFTMICAPOS (LRSH
                                                                                                THISCHARWIDTH
                                                                                                1))
                                                                     XMICAPOS)
                                                          (SUB1 CHAR#))
                                                         (T CHAR#)))
                                           finally (RETURN (SUB1 CHAR#]
                                (T (\ILLEGAL.ARG TEXT]
                      (T (for CHAR# from 1 to (NCHARS TEXT) when (IGREATERP [SETQ LEFT
                                                                            (IPLUS LEFT (SETQ THISCHARWIDTH
                                                                                         (NTHCHARWIDTH TEXT
                                                                                                       CHAR# LFONT]
                                                                 XPOS)
                          do                                           (* have the hotspot be between the characters.)
                             (RETURN (COND
                                        ((IGREATERP (IDIFFERENCE LEFT (LRSH THISCHARWIDTH 1))
                                                    XPOS)
                                         (SUB1 CHAR#))
                                        (T CHAR#)))
                          finally (RETURN (SUB1 CHAR#])
```

(CLOSEST.LINE

```
  [LAMBDA (TEXTELT Y)                                                  (* rrb " 1-MAY-83 10:15")
                                                                       (* determines the line of TEXTELT that Y is closest to.)
                                                                       (* assumes that the text elements are ordered from top to
                                                                       bottom.)
    (for LREGION in (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of TEXTELT)) as LINE#
       from 1 when (IGEQ Y (fetch (REGION BOTTOM) of LREGION)) do (RETURN LINE#) finally (RETURN (SUB1 LINE#])
```

(FLASHW

```
  [LAMBDA (WIN)                                                        (* flashs a window.)
    (INVERTW WIN)
    (DISMISS BELLRATE)
    (INVERTW WIN])
```

(HILITE.LINE

```
  [LAMBDA (TEXTELT LINE# MINLEFT MAXLEFT WINDOW)                       (* rrb " 1-MAY-83 09:54")
```

(* highlights within a single line of text between MINLEFT and MAXLEFT.
If MINLEFT is NIL it uses the beginning of the line. If MAXLEFT is NIL it uses the end of the line.)

```
    (PROG ((LREGION (NTHLOCALREGION TEXTELT LINE#)))
          (BITBLT NIL NIL NIL WINDOW (OR MINLEFT (SETQ MINLEFT (fetch (REGION LEFT) of LREGION)))
                 (fetch (REGION BOTTOM) of LREGION)
                 (IDIFFERENCE (OR MAXLEFT (fetch (REGION PRIGHT) of LREGION))
                        MINLEFT)
                 (fetch (REGION HEIGHT) of LREGION)
                 'TEXTURE
                 'INVERT SELECTION.HIGHLIGHT.SHADE])
```

(HILITE.TEXT

```
  [LAMBDA (TEXTELT SELLEFT SELLINE# EXTLEFT EXTLINE# WINDOW)      (* rrb "30-Dec-84 17:59")
                                                                  (* high lights between two positions in a text element.)
    (COND
       ((EQ SELLINE# EXTLINE#)                                    (* on the same line, highlight between them.)
        (PROG (MIN MAX)
              (COND
                 ((NULL SELLEFT)                                  (* SELLEFT is NIL during recursive calls and means use the
                                                                  beginning of the text.)
                  (SETQ MIN NIL)
                  (SETQ MAX EXTLEFT))
                 ((IGREATERP SELLEFT EXTLEFT)
                  (SETQ MIN EXTLEFT)
                  (SETQ MAX SELLEFT))
                 (T (SETQ MIN SELLEFT)
                    (SETQ MAX EXTLEFT)))
              (HILITE.LINE TEXTELT SELLINE# MIN MAX WINDOW)))
       ((IGREATERP EXTLINE# SELLINE#)                             (* fill from SEL to end of its line and recurse.)
        (HILITE.LINE TEXTELT SELLINE# SELLEFT NIL WINDOW)
        (HILITE.TEXT TEXTELT NIL (ADD1 SELLINE#)
               EXTLEFT EXTLINE# WINDOW))
       (T                                                         (* fill from EXT to the end of its line and recurse.)
          (HILITE.LINE TEXTELT EXTLINE# EXTLEFT NIL WINDOW)       (* always recurse to have highest selection first.)
          (HILITE.TEXT TEXTELT NIL (ADD1 EXTLINE#)
                 SELLEFT SELLINE# WINDOW])
```

(IN.TEXT.EXTEND

```
  [LAMBDA (SELECTION SKW)                                         (* rrb "22-May-85 11:40")
                                                                  (* the user has right buttoned and the first selection was to in
                                                                  existing text.)
                                                                  (* current selection has already been undisplayed.)
    (PROG [(OLDLINE (fetch (TEXTELTSELECTION SKLINE#) of SELECTION))
           (OLDX (fetch (TEXTELTSELECTION SKLEFT) of SELECTION))
           (INTEXT (fetch (TEXTELTSELECTION SKTEXTELT) of SELECTION))
           FEEDBACKX FEEDBACKY FEEDBACKLINE FEEDBACKCHAR (REGION (DSPCLIPPINGREGION NIL SKW))
           (DSP (WINDOWPROP SKW 'DSP]
          (while (MOUSESTATE RIGHT) do                            (* track with the appropriate feedback)
                                       (COND
                                          ([NOT (INSIDEP REGION (SETQ FEEDBACKX (LASTMOUSEX DSP))
                                                        (SETQ FEEDBACKY (LASTMOUSEY DSP]
                                                                  (* cursor moved outside of the window.
                                                                  Reset selection and quit.)
                                           (SKED.SELECTION.FEEDBACK SKW)
                                           (RETURN)))
                                       (SETQ FEEDBACKLINE (CLOSEST.LINE INTEXT FEEDBACKY))
                                                                  (* inside of a text element.)
                                       (SETQ FEEDBACKX (CHAR.BEGIN (SETQ FEEDBACKCHAR (CLOSEST.CHAR FEEDBACKX
                                                                                                    FEEDBACKLINE INTEXT
                                                                                                    DSP))
                                                                   FEEDBACKLINE INTEXT DSP))
                                       (COND
                                          ((OR (NEQ OLDX FEEDBACKX)
                                               (NEQ OLDLINE FEEDBACKLINE))
                                           (HILITE.TEXT INTEXT OLDX OLDLINE (SETQ OLDX FEEDBACKX)
                                                  (SETQ OLDLINE FEEDBACKLINE)
                                                  SKW)))
            finally

          (* erase feedback. It will be put in as a result of setting the extention selection.)

                  (HILITE.TEXT INTEXT OLDX OLDLINE (fetch (TEXTELTSELECTION SKLEFT) of SELECTION)
                         (fetch (TEXTELTSELECTION SKLINE#) of SELECTION)
                         SKW)
                  (SKED.SET.EXTENDSELECTION (create TEXTELTSELECTION
                                                    SKTEXTELT _ INTEXT
                                                    SKLINE# _ OLDLINE
                                                    SKCHAR# _ FEEDBACKCHAR
                                                    SKLEFT _ OLDX
                                                    SKBOTTOM _ (LINE.BEGIN (OR FEEDBACKLINE 1)
                                                                           INTEXT))
                            SKW])
```

(INIMAGEOBJ

```
  [LAMBDA (SKIMAGEOBJSCREENELT X Y)                               (* rrb "31-Mar-84 11:43")
                                                                  (* return T if X Y is inside of the image object
                                                                  SKIMAGEOBJSCREENELT.)
```

```
    (INSIDEP (fetch (LOCALSKIMAGEOBJ SKIMOBJLOCALREGION) of (fetch (SCREENELT LOCALPART) of SKIMAGEOBJSCREENELT))
           X Y])
```

(INTEXT

```
  [LAMBDA (TEXTELT XORPT Y)                                          (* rrb " 6-MAY-83 19:37")
                                                                     (* determines which line if any a position is on.)
    (for LREGION in (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of TEXTELT)) as LINE#
       from 1 when (INSIDEP LREGION XORPT Y) do (RETURN LINE#])
```

(NEW.TEXT.EXTEND

```
  [LAMBDA (SELECTION SKW)                                            (* rrb "30-APR-83 16:25")
                                                                     (* the user has right buttoned and the first selection was to new
    text.)                                                           (* current selection has already been undisplayed.)
    (PROG (FEEDBACKX FEEDBACKY EXTENDEDSEL OLDX OLDY OLDCUR)
          (until (MOUSESTATE (NOT RIGHT)) do                         (* track with the appropriate feedback)
                                          (SETQ FEEDBACKX (LASTMOUSEX SKW))
                                          (SETQ FEEDBACKY (LASTMOUSEY SKW))
                                          (COND
                                             ((OR (NEQ OLDX FEEDBACKX)
                                                  (NEQ OLDY FEEDBACKY))
                                                                     (* erase previous feedback)
                                              (AND EXTENDEDSEL (SHOW.FEEDBACK.BOX SELECTION EXTENDEDSEL SKW)
                                                   )
                                              (SHOW.FEEDBACK.BOX SELECTION (SETQ EXTENDEDSEL
                                                                                 (create POSITION
                                                                                         XCOORD _ (SETQ OLDX
                                                                                                        FEEDBACKX)
                                                                                         YCOORD _ (SETQ OLDY
                                                                                                        FEEDBACKY)))
                                                                                 SKW)))
             finally (AND EXTENDEDSEL (SHOW.FEEDBACK.BOX SELECTION EXTENDEDSEL SKW))
                     (SKED.SET.EXTENDSELECTION EXTENDEDSEL SKW])
```

(NEW.TEXT.SELECTIONP

```
  [LAMBDA (SELECTION)                                                (* determines if a selection is pointing to new text location or
                                                                     existing text.)
    (POSITIONP SELECTION])
```

(NTHCHARWIDTH

```
  [LAMBDA (STR N FONT)                                               (* rrb "23-Aug-84 09:43")
                                                                     (* returns the character width of the Nth character in STR.)
    (CHARWIDTH (CHCON1 (NTHCHAR STR N))
           FONT])
```

(NTHLOCALREGION

```
  [LAMBDA (TEXTELT N)                                                (* rrb " 1-MAY-83 09:53")
    (CAR (NTH (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of TEXTELT))
              N])
```

(ONCHAR

```
  [LAMBDA (XPOS LINE# TEXTELT STRM)                                  (* rrb "24-Aug-84 11:14")

          (* determines the character number that XPOS is on in a particular line of a text element.)

          (* will return 1 if the position is in the first half of the first character.)

    (CLOSEST.CHAR XPOS LINE# TEXTELT STRM NIL])
```

(SHOW.EXTENDED.SELECTION.FEEDBACK

```
  [LAMBDA (SEL EXTENDSEL SKW)                                        (* rrb " 1-MAY-83 09:55")
                                                                     (* hi lights the selection between SEL and EXTENDSEL)
    (COND
       ((NEQ (fetch (TEXTELTSELECTION SKTEXTELT) of SEL)
             (fetch (TEXTELTSELECTION SKTEXTELT) of EXTENDSEL))      (* if the two selections aren't in the same text element, things
                                                                     are confused.)
        (SHOULDNT)))
    (HILITE.TEXT (fetch (TEXTELTSELECTION SKTEXTELT) of SEL)
           (fetch (TEXTELTSELECTION SKLEFT) of SEL)
           (fetch (TEXTELTSELECTION SKLINE#) of SEL)
           (fetch (TEXTELTSELECTION SKLEFT) of EXTENDSEL)
           (fetch (TEXTELTSELECTION SKLINE#) of EXTENDSEL)
           SKW])
```

(SHOW.FEEDBACK

```
  [LAMBDA (FEEDBACKCUR FEEDBACKX FEEDBACKY WINDOW)                   ; Edited  9-Jan-87 13:49 by rrb
                                                                     (* displays a cursor in XOR mode at a position.)
    (BITBLT (CAR FEEDBACKCUR)
           0 0 WINDOW (IDIFFERENCE FEEDBACKX (CADR FEEDBACKCUR))
           (IDIFFERENCE FEEDBACKY (CDDR FEEDBACKCUR))
```

```
          NIL NIL 'INPUT 'INVERT])
```

(SHOW.FEEDBACK.BOX

```
(SHOW.FEEDBACK.BOX
  [LAMBDA (P1 P2 WINDOW)                                    ; Edited 21-Aug-2021 19:08 by larry
                                                            (* draws a box between two points.)

    (PROG ((X1 (fetch (POSITION XCOORD) of P1))
           (Y1 (fetch (POSITION YCOORD) of P1))
           (X2 (fetch (POSITION XCOORD) of P2))
           (Y2 (fetch (POSITION YCOORD) of P2)))
          (BITBLT NIL NIL NIL WINDOW (IMIN X1 X2)
                 (IMIN Y1 Y2)
                 (ABS (IDIFFERENCE X1 X2))
                 (ABS (IDIFFERENCE Y1 Y2))
                 'TEXTURE
                 'INVERT NEW.TEXT.FEEDBACK.SHADE)           (* put cursor where the center would be.)
          (SHOW.FEEDBACK NEW.TEXT.FEEDBACK.CURSOR (IQUOTIENT (IPLUS X1 X2)
                                                             2)
                 (IQUOTIENT (IPLUS Y1 Y2)
                        2)
                 WINDOW])
```

(SELECTION.POSITION

```
(SELECTION.POSITION
  [LAMBDA (FIRSTPT SECONDPT)                                (* rrb " 6-MAY-83 18:09")

          (* returns the place where the text should go from one or two selections in open space.)

    (COND
       (SECONDPT (create POSITION
                         XCOORD _ (IQUOTIENT (IPLUS (fetch (POSITION XCOORD) of FIRSTPT)
                                                    (fetch (POSITION XCOORD) of SECONDPT))
                                             2)
                         YCOORD _ (IQUOTIENT (IPLUS (fetch (POSITION YCOORD) of FIRSTPT)
                                                    (fetch (POSITION YCOORD) of SECONDPT))
                                             2)))
       (T FIRSTPT])
```

(SKED.CLEAR.SELECTION

```
(SKED.CLEAR.SELECTION
  [LAMBDA (SKW DONTDISPLAYFLG DONTMAKEHISTEVENTFLG)         (* rrb " 5-Dec-85 14:30")
                                                            (* clears the selection and removes it from the display.)

    (OR DONTMAKEHISTEVENTFLG (SKETCH.CLEANUP SKW))
    (COND
       ((OR DONTDISPLAYFLG (SKED.SELECTION.FEEDBACK SKW))
        (WINDOWPROP SKW 'SELECTION NIL)
        (WINDOWPROP SKW 'EXTENDSELECTION NIL])
```

(SKETCH.CLEANUP

```
(SKETCH.CLEANUP
  [LAMBDA (SKETCHWINDOW)                                    ; Edited 20-Feb-87 17:51 by rrb
                                                            (* finishes up the currently being edited element.)

    (PROG (INITSELECTION NEWELT)
          (COND
             ((NOT (WINDOWP SKETCHWINDOW))

          (* only one of the viewers should have any changes but do this in lieu of figuring out which one.)

              (for VIEWER in (ALL.SKETCH.VIEWERS (INSURE.SKETCH SKETCHWINDOW)) do (SKETCH.CLEANUP VIEWER)))
             ((SETQ INITSELECTION (WINDOWPROP SKETCHWINDOW 'CHANGEDTEXTELT NIL))

          (* also checks to see if the current selection was edited and makes a history event if necessary.)

              [SETQ NEWELT (fetch (SCREENELT GLOBALPART) of (fetch (TEXTELTSELECTION SKTEXTELT)
                                                                   of (OR (WINDOWPROP SKETCHWINDOW 'SELECTION)
                                                                          (RETURN]
              (COND
                 ((POSITIONP INITSELECTION)                 (* add an ADD event because previously there was nothing
                                                            here.)
                  (SK.ADD.HISTEVENT 'ADD (LIST NEWELT)
                         SKETCHWINDOW)
                  (SK.CHECK.END.INITIAL.EDIT SKETCHWINDOW NEWELT))
                 (T (SK.ENTER.EDIT.CHANGE SKETCHWINDOW (fetch (SCREENELT GLOBALPART) of (fetch (TEXTELTSELECTION
                                                                                                SKTEXTELT)
                                                                                           of INITSELECTION))
                           NEWELT])
```

(SK.ENTER.EDIT.CHANGE

```
(SK.ENTER.EDIT.CHANGE
  [LAMBDA (VIEWER OLDELT NEWELT)                            (* rrb " 3-Jan-86 18:48")

          (* adds a history event for the change operation that occurs at the end of a series of edits and calls the when changed
          function.)

    (SK.CHECK.WHENCHANGEDFN VIEWER OLDELT 'DATA (fetch (TEXT LISTOFCHARACTERS) of (fetch (GLOBALPART
                                                                                         INDIVIDUALGLOBALPART
                                                                                          )
```

```
                                                                                    of NEWELT))
          (fetch (TEXT LISTOFCHARACTERS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of OLDELT)))
    (SK.ADD.HISTEVENT 'CHANGE (LIST (create SKHISTORYCHANGESPEC
                                            OLDELT _ OLDELT
                                            NEWELT _ NEWELT
                                            PROPERTY _ 'DATA
                                            NEWVALUE _ (fetch (TEXT LISTOFCHARACTERS) of (fetch (GLOBALPART
                                                                                                 INDIVIDUALGLOBALPART
                                                                                                       )
                                                                                           of NEWELT))
                                            OLDVALUE _ (fetch (TEXT LISTOFCHARACTERS) of (fetch (GLOBALPART
                                                                                                 INDIVIDUALGLOBALPART
                                                                                                       )
                                                                                           of OLDELT)))
                                      (LIST OLDELT NEWELT))
          VIEWER])
```

(SKED.REMOVE.OTHER.SELECTIONS

```
  [LAMBDA (SKW)                                                    (* rrb " 5-Dec-85 12:12")
```

(* removes and undisplays any selections in any other windows onto the same sketch as SKW.)

```
    (for SKETCHWINDOW in (ALL.SKETCH.VIEWERS (SKETCH.FROM.VIEWER SKW)) when (NEQ SKETCHWINDOW SKW)
       do (SKED.CLEAR.SELECTION SKETCHWINDOW])
```

(SKED.EXTEND.SELECTION

```
  [LAMBDA (SKW)                                                    (* rrb "17-Jul-85 20:11")
                                                                   (* the user has left buttoned in a sketch window.
                                                                   Put the caret there.)
                                                                   (* take down the current selection.)
    (PROG [(SELECTION (WINDOWPROP SKW 'SELECTION))
           (EXTENSION (WINDOWPROP SKW 'EXTENDSELECTION]
          (RETURN (COND
                     [SELECTION (COND
                                   [(NEW.TEXT.SELECTIONP SELECTION)
```

(* if the previous selection was in new text, treat the right the same as the left.)

```
                                    (SKED.MOVE.SELECTION SKW (NOT (WINDOWPROP SKW 'USEGRID]
                                   (T (SKED.SELECTION.FEEDBACK SKW)
                                                                   (* extend within text.)
```

(* MAYBE SHOULD DO -
if there is already an extension, make the fixed point be the one of the two selections that is farthest from the current position.)

```
                                       (IN.TEXT.EXTEND SELECTION SKW]
                     (T
```

(* here is a right button before any left ones. Treat as if it were left.)

```
                        (SKED.MOVE.SELECTION SKW (NOT (WINDOWPROP SKW 'USEGRID])
```

(SKED.MOVE.SELECTION

```
  [LAMBDA (SKW USEGRID)                                            (* rrb "11-Jul-86 15:51")
                                                                   (* the user has left buttoned in a sketch window.
                                                                   Put the caret there.)
    (SKED.CLEAR.SELECTION SKW)
    (PROG (FEEDBACKX FEEDBACKY OLDGRIDX OLDGRIDY OLDX OLDY OLDCUR FEEDBACKCUR INTEXT INIMAGEOBJ STARTLINE
                 STARTCHAR X Y (DSP (WINDOWPROP SKW 'DSP))
                 (SCALE (VIEWER.SCALE SKW))
                 (GRID (SK.GRIDFACTOR SKW)))
          (until (MOUSESTATE UP)
             do
```

(* track with the appropriate caret depending upon whether the cursor is inside of existing text or not.)

```
                (SETQ X (LASTMOUSEX DSP))
                (SETQ Y (LASTMOUSEY DSP))
                (COND
                   ((OR (NEQ OLDX X)
                        (NEQ OLDY Y))                              (* only look for things when the cursor position has changed.)
                    (SETQ OLDX X)
                    (SETQ OLDY Y)
                    [COND
                       ([AND (SETQ INTEXT (for ELT in (LOCALSPECS.FROM.VIEWER SKW)
                                             when (SELECTQ (fetch (SCREENELT GTYPE) of ELT)
                                                           (TEXT (AND (NEQ (fetch (LOCALTEXT LOCALFONT)
                                                                              of (fetch (SCREENELT LOCALPART)
                                                                                    of ELT))
                                                                           'SHADE)
                                                                      (SETQ STARTLINE (INTEXT ELT X Y))))
                                                           (TEXTBOX (AND (NEQ (fetch (LOCALTEXTBOX LOCALFONT)
                                                                                 of (fetch (SCREENELT LOCALPART)
```

```
                                                                    of ELT))
                                                          'SHADE)
                                                    (INSIDE? (fetch (LOCALTEXTBOX
                                                                       LOCALTEXTBOXREGION)
                                                                of (fetch (SCREENELT LOCALPART)
                                                                      of ELT))
                                                             X Y)
                                                    (SETQ STARTLINE (CLOSEST.LINE ELT Y))))
                                              NIL)
                                        do (RETURN ELT)))
                                  (NOT (SK.ELEMENT.PROTECTED? (fetch (SCREENELT GLOBALPART) of INTEXT)
                                                'CHANGE]                    (* inside of a text element.)
                             (SETQ FEEDBACKCUR IN.TEXT.FEEDBACK.CURSOR)
                             (SETQ FEEDBACKX (CHAR.BEGIN (SETQ STARTCHAR (CLOSEST.CHAR X STARTLINE INTEXT DSP))
                                                 STARTLINE INTEXT DSP))
                             (SETQ FEEDBACKY (LINE.BEGIN STARTLINE INTEXT)))
                            (T (SETQ FEEDBACKCUR NEW.TEXT.FEEDBACK.CURSOR)
                               (COND
                                  (USEGRID (SETQ FEEDBACKX (MAP.WINDOW.ONTO.GRID X SCALE GRID))
                                           (SETQ FEEDBACKY (MAP.WINDOW.ONTO.GRID Y SCALE GRID)))
                                  (T                                        (* no grid)
                                     (SETQ FEEDBACKX X)
                                     (SETQ FEEDBACKY Y]
                         (COND
                            ((OR (NEQ OLDGRIDX FEEDBACKX)
                                 (NEQ OLDGRIDY FEEDBACKY)
                                 (NEQ OLDCUR FEEDBACKCUR))
                             (AND OLDGRIDX (SHOW.FEEDBACK OLDCUR OLDGRIDX OLDGRIDY SKW))
                             (SHOW.FEEDBACK (SETQ OLDCUR FEEDBACKCUR)
                                    (SETQ OLDGRIDX FEEDBACKX)
                                    (SETQ OLDGRIDY FEEDBACKY)
                                    SKW)))                                  (* give the coordinate display window a shot.)
                         (SKETCHW.UPDATE.LOCATORS SKW)))
                finally (AND OLDGRIDX (SHOW.FEEDBACK OLDCUR OLDGRIDX OLDGRIDY SKW))
                        (COND
                           ((EQ OLDCUR IN.TEXT.FEEDBACK.CURSOR)             (* selection is existing text)
                            (SKED.SET.SELECTION (CREATE.TEXT.SELECTION INTEXT STARTLINE STARTCHAR OLDGRIDX OLDGRIDY
                                                                       DSP)
                                    SKW))
                           (OLDGRIDX (SKED.SET.SELECTION (create POSITION
                                                                 XCOORD _ OLDGRIDX
                                                                 YCOORD _ OLDGRIDY)
                                              SKW])
```

(CREATE.TEXT.SELECTION

```
[LAMBDA (TEXTELT LINE# CHAR# LFT BTM STRM)                              (* rrb "23-Aug-84 13:48")
```

(* creates a text selection object. If LFT or BTM are NIL they are computed from the other arguments.)

```
   (create TEXTELTSELECTION
           SKTEXTELT _ TEXTELT
           SKLINE# _ LINE#
           SKCHAR# _ CHAR#
           SKLEFT _ (OR LFT (CHAR.BEGIN CHAR# LINE# TEXTELT STRM))
           SKBOTTOM _ (OR BTM (LINE.BEGIN LINE# TEXTELT])
```

(SKED.SELECTION.FEEDBACK

```
[LAMBDA (SKETCHW)                                                       (* rrb "14-Sep-84 18:20")
```

(* displays the feedback to the user about what the current selection is.
Returns NIL if there is no selection, T otherwise.)

```
   (PROG ((SELECTION (WINDOWPROP SKETCHW 'SELECTION))
          EXTENDSELECTION)
         (OR SELECTION (RETURN))
         (SETQ EXTENDSELECTION (WINDOWPROP SKETCHW 'EXTENDSELECTION))
         [COND
            [(NEW.TEXT.SELECTIONP SELECTION)                            (* outside of existing text region)
             (COND
                (EXTENDSELECTION
```

(* display a box whose center will be used as the center of mass of the text.)

```
                       (SHOW.FEEDBACK.BOX SELECTION EXTENDSELECTION SKETCHW))
                (T (SHOW.FEEDBACK NEW.TEXT.FEEDBACK.CURSOR (fetch (POSITION XCOORD) of SELECTION)
                          (fetch (POSITION YCOORD) of SELECTION)
                          SKETCHW]
            (T (COND
                  (EXTENDSELECTION
```

(* display a box whose center will be used as the center of mass of the text.)

```
                       (SHOW.EXTENDED.SELECTION.FEEDBACK SELECTION EXTENDSELECTION SKETCHW))
                (T (SHOW.FEEDBACK IN.TEXT.FEEDBACK.CURSOR (fetch (TEXTELTSELECTION SKLEFT) of SELECTION)
                          (fetch (TEXTELTSELECTION SKBOTTOM) of SELECTION)
```

```
                              SKETCHW]
          (RETURN T])
```

(SKED.SET.EXTENDSELECTION

```
  [LAMBDA (SELECTION SKETCHW)                                  (* rrb "30-APR-83 16:36")
                                                               (* stores the selection for a sketch window and displays its
                                                               feedback.)
    (WINDOWPROP SKETCHW 'EXTENDSELECTION SELECTION)
    (SKED.SELECTION.FEEDBACK SKETCHW])
```

(SKED.SET.SELECTION

```
  [LAMBDA (SELECTION SKETCHW)                                  (* rrb " 2-MAY-83 12:29")
                                                               (* stores the selection for a sketch window and displays its
                                                               feedback.)
    (WINDOWPROP SKETCHW 'SELECTION SELECTION)                  (* clear the extension also.)
    (WINDOWPROP SKETCHW 'EXTENDSELECTION NIL)
    (SKED.SELECTION.FEEDBACK SKETCHW])
```

(LINE.BEGIN

```
  [LAMBDA (LINE# TEXTELT)                                      (* rrb "30-APR-83 15:45")
                                                               (* returns the bottom of the LINE# th text element of TEXTELT.)
    (fetch (REGION BOTTOM) of (CAR (NTH (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of TEXTELT))
                                        LINE#])
```

(SELECTION.GREATERP

```
  [LAMBDA (SEL1 SEL2)                                          (* rrb " 1-MAY-83 11:31")
                                                               (* return T if SEL1 appears before SEL2)
    (PROG (L1 L2)
          (RETURN (COND
                     ((IGREATERP (SETQ L2 (fetch (TEXTELTSELECTION SKLINE#) of SEL2))
                                 (SETQ L1 (fetch (TEXTELTSELECTION SKLINE#) of SEL1)))
                      T)
                     ((EQ L1 L2)                               (* look at characters)
                      (IGREATERP (fetch (TEXTELTSELECTION SKCHAR#) of SEL2)
                                 (fetch (TEXTELTSELECTION SKCHAR#) of SEL1])
```

(SK.WORD.BREAK.CLASS

```
  [LAMBDA (CHCODE)                                             (* rrb "11-Jul-86 17:17")
                                                               (* version of TEDIT.WORDGET that makes sure TEDIT is
                                                               loaded.)
    (COND
       ((DEFINEDP (FUNCTION TEDIT.WORDGET))
        (TEDIT.WORDGET CHCODE))
       ((EQ CHCODE 32)                                         (* space, return the code tedit uses for word separators)
        22)
       (T                                                      (* this probably isn't right but should do something reasonable.)
          (GETSYNTAX CHCODE (GETREADTABLE])
```

(SK.GETSYNTAX

```
  [LAMBDA (CHARCODE)                                           (* rrb "11-Jul-86 17:18")

          (* version of getsyntax that uses the TEDIT table if it is available, otherwise the terminal.)

    (COND
       ((DEFINEDP (FUNCTION TEDIT.GETSYNTAX))
        (TEDIT.GETSYNTAX CHARCODE TEDIT.READTABLE))
       (T (GETSYNTAX CHARCODE (GETTERMTABLE])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD TEXTELTSELECTION (SKTEXTELT SKLINE# SKCHAR# SKLEFT SKBOTTOM))
)
)

(READVARS-FROM-STRINGS '(IN.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.SHADE
                                 SELECTION.HIGHLIGHT.SHADE)
       "(({(READBITMAP)(16 16
       %"@@@@%"
       %"@@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
```

```
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%"
       %"L@@@%")} 1 . 0)  ({(READBITMAP)(16 16
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@@@@%"
       %"@H@@%"
       %"AL@@%"
       %"CF@@%"
       %"FC@@%"
       %"LAH@%")} 4 . 4) 8 65535)
       ")

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS IN.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.CURSOR NEW.TEXT.FEEDBACK.SHADE SELECTION.HIGHLIGHT.SHADE)
)
```

(* * editting functions)

```
(DEFINEQ
```

(WB.EDITOR

```
  [LAMBDA (SKW)                                                    (* rrb "17-Jul-85 15:53")
```

(* the process that looks for characters and adds them to the white board as text elements.)

(* save the value of del as an interrupt character so it is restored when this process exits.
This is also done by TTYENTRYFN and TTYEXITFN on the process.)

```
    (RESETFORM (INTERRUPTCHAR 127 T)
           (PROG (CHARS EDITINPROGRESS)
                 (TTYDISPLAYSTREAM SKW)
             LP  (COND
                    ((\SYSBUFP)
```

(* a character has been typed, read all of the characters, delete the current selection if extended and insert the new characters.)

```
                       (RESET.LINE.BEING.INPUT SKW)
                       (SKED.INSERT (GETALLCHARS T)
                              SKW)
                       (SETQ EDITINPROGRESS T))
                      ((AND EDITINPROGRESS (NOT (INSIDEP (WINDOWPROP SKW 'REGION)
                                                         LASTMOUSEX LASTMOUSEY)))
                       (CLEANUP.EDIT SKW)
                       (SETQ EDITINPROGRESS NIL)))                 (* let the mouse process run.)
                   (BLOCK)
                   (GO LP])
```

(SK.TTYENTRYFN

```
  [LAMBDA (SKPROC)                                                 (* rrb "20-Jun-85 14:13")
                                                                   (* the sketch process just got the tty.
                                                                   Turns off DEL as an interrupt)
    (PROCESSPROP SKPROC 'OLDINTERRUPTVALUE (INTERRUPTCHAR 127 NIL])
```

(SK.TTYEXITFN

```
  [LAMBDA (SKPROC)                                                 (* rrb "20-Jun-85 13:55")
                                                                   (* the sketch process just got the tty.
                                                                   Turns off DEL as an interrupt)
    (INTERRUPTCHAR (PROCESSPROP SKPROC 'OLDINTERRUPTVALUE])
```

(SKED.INSERT

```
  [LAMBDA (CHARCODES SKW ATSCALE)                                  (* rrb "10-Feb-86 10:15")
```

(* deletes the characters in the extension and inserts characters into the currently selected position of SKW and leaves the selection at the end of the insertion.)

```
    (WITH.MONITOR (SKETCH.MONITORLOCK SKW)
           (\SKED.INSERT CHARCODES SKW ATSCALE])
```

(\SKED.INSERT

```
[LAMBDA (CHARCODES SKW ATSCALE)                                ; Edited 20-Feb-87 17:28 by rrb
  (COND
     ((GREATERP (LENGTH CHARCODES)
            200)

(* the maximum string length limits the number of characters that can be inserted at once.
This can happen from a shift select.)

      (SKED.INSERT (FIRST.N.ELEMENTS CHARCODES 200)
            SKW ATSCALE)
      (SKED.INSERT (NTH CHARCODES 201)
            SKW ATSCALE))
     (T (PROG ((SELECTION (WINDOWPROP SKW 'SELECTION))
               (EXTENSION (WINDOWPROP SKW 'EXTENDSELECTION))
               TEXTELT ELTTYPE GTEXTELT FIRSTLINE# FIRSTCHAR# LASTLINE# LASTCHAR# STRLST NEWSTRS NEWELT
               STRPIECE NEWLINE# NEWCHAR# SKCONTEXT PTRCHAR# CONTROLCHARTAIL)
              (COND
                 ((EQ (SK.GETSYNTAX (CAR CHARCODES))
                      'UNDO)

(* user typed an undo Avoid the overhead of inserting no characters and allow undo to be typed without a selection.)

                  (SETQ CONTROLCHARTAIL 'UNDO)
                  (GO UNDO)))
              (COND
                 ((NULL SELECTION)                             (* add a new text element with these characters.)
                  (STATUSPRINT SKW "
                        " "Indicate the position the typing should go with the left button.")
                  (RETURN)))
              (SKED.CLEAR.SELECTION SKW NIL T)
              (SKED.REMOVE.OTHER.SELECTIONS SKW)
              [COND
                 ((AND (OR (EQ (CAR CHARCODES)
                               (CHARCODE EOL))
                           (EQ (CAR CHARCODES)
                               (CHARCODE LINEFEED)))
                       (KEYDOWNP 'CTRL))                       (* user hit control CR. create a new text or textbox.)
                  (SKED.CREATE.NEW.TEXTBOX [COND
                                             ((NEW.TEXT.SELECTIONP SELECTION)
                                              NIL)
                                             (T (fetch (SCREENELT INDIVIDUALGLOBALPART)
                                                   of (fetch (TEXTELTSELECTION SKTEXTELT) of SELECTION]
                        SKW
                        (CDR CHARCODES))
                  (RETURN))
                 [(NEW.TEXT.SELECTIONP SELECTION)              (* selection is in open space, create a new text element.)
                                                               (* merge the characters into strings of each line.)
                  (SETQ ELTTYPE 'TEXT)
                  (SETQ CONTROLCHARTAIL (\SKED.INSERT.CHARS.TO.STR CHARCODES NIL SKW))
                  (COND
                     ((OR NEWSTRS STRPIECE)                    (* if there are any new characters, add a new text element.)

(* save the selection that marked the spot where the new text goes and add the text but not in a way that puts an event on
the history list. this is done during clean up.)

                      (WINDOWPROP SKW 'CHANGEDTEXTELT SELECTION)
                      (SETQ NEWELT (SK.ADD.ELEMENT (CREATE.TEXT.ELEMENT (SETQ NEWSTRS (NCONC1 NEWSTRS STRPIECE
                                                                                          ))
                                                                        (SK.MAP.INPUT.PT.TO.GLOBAL (create INPUTPT
                                                                                                           INPUT.ONGRID? _
                                                                                                           NIL
                                                                                                           INPUT.POSITION _
                                                                                                           (
                                                                                                            SELECTION.POSITION
                                                                                                            SELECTION
                                                                                                            EXTENSION))
                                                                                                   SKW)
                                                                        (OR (NUMBERP ATSCALE)
                                                                            (SK.INPUT.SCALE SKW))
                                                                        [fetch (SKETCHCONTEXT SKETCHTEXTALIGNMENT)
                                                                           of (SETQ SKCONTEXT (WINDOWPROP SKW
                                                                                                          'SKETCHCONTEXT]
                                                                        (fetch (SKETCHCONTEXT SKETCHFONT) of SKCONTEXT)
                                                                        (fetch (BRUSH BRUSHCOLOR) of (fetch (SKETCHCONTEXT
                                                                                                             SKETCHBRUSH)
                                                                                                    of SKCONTEXT)))
                                                   SKW T)))
                     (CONTROLCHARTAIL                          (* user typed control return to get textbox in the middle of no
                                                               where.)
                            (SKED.CREATE.NEW.TEXTBOX NIL SKW (CDR CONTROLCHARTAIL)
                                  ATSCALE)
                            (RETURN))
                     (T                                        (* user typed backspace, etc. when no text exists.
                                                               Put caret back in same place.)
                        (SKED.SET.SELECTION SELECTION SKW)
                        (RETURN)))                             (* put selection marker at the end.)
```

```
                    (SETQ NEWLINE# (LENGTH NEWSTRS))
                    (SETQ NEWCHAR# (NCHARS (CAR (LAST NEWSTRS]
                   (T [SETQ GTEXTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of (SETQ TEXTELT (fetch (
                                                                                    TEXTELTSELECTION
                                                                                       SKTEXTELT)
                                                                                  of SELECTION]
                      (SETQ ELTTYPE (fetch (INDIVIDUALGLOBALPART GTYPE) of GTEXTELT))
                      (SETQ STRLST (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of (fetch (SCREENELT LOCALPART)
                                                                                  of TEXTELT)))
                                                          (* set up points to beginning and end of selection.)
                      [COND
                         [(NULL EXTENSION)
                          (SETQ LASTCHAR# (SETQ FIRSTCHAR# (fetch (TEXTELTSELECTION SKCHAR#) of SELECTION)))
                          (SETQ LASTLINE# (SETQ FIRSTLINE# (fetch (TEXTELTSELECTION SKLINE#) of SELECTION]
                         ((SELECTION.GREATERP SELECTION EXTENSION)
                          (SETQ FIRSTLINE# (fetch (TEXTELTSELECTION SKLINE#) of SELECTION))
                          (SETQ FIRSTCHAR# (fetch (TEXTELTSELECTION SKCHAR#) of SELECTION))
                          (SETQ LASTLINE# (fetch (TEXTELTSELECTION SKLINE#) of EXTENSION))
                          (SETQ LASTCHAR# (fetch (TEXTELTSELECTION SKCHAR#) of EXTENSION))
                                                          (* make SELECTION be the candidate for the selection after the
                          deletion.)
                          (SETQ SELECTION EXTENSION))
                         (T (SETQ FIRSTLINE# (fetch (TEXTELTSELECTION SKLINE#) of EXTENSION))
                            (SETQ FIRSTCHAR# (fetch (TEXTELTSELECTION SKCHAR#) of EXTENSION))
                            (SETQ LASTLINE# (fetch (TEXTELTSELECTION SKLINE#) of SELECTION))
                            (SETQ LASTCHAR# (fetch (TEXTELTSELECTION SKCHAR#) of SELECTION]
                      [for STR in STRLST as LINE# from 1
                         do [COND
                               ((ILESSP LINE# FIRSTLINE#)         (* before the first, copy across)
                                (SETQ NEWSTRS (NCONC1 NEWSTRS STR)))
                               ((IGREATERP LINE# LASTLINE#)       (* After the last, copy across)
                                (SETQ NEWSTRS (NCONC1 NEWSTRS STR)))
                               ((EQ LINE# FIRSTLINE#)             (* on the first, save the part before.)
                                (SETQ STRPIECE (SUBSTRING STR 1 FIRSTCHAR#))
                                                          (* insert new text.)
                                (COND
                                   [CHARCODES (SETQ CONTROLCHARTAIL (\SKED.INSERT.CHARS.TO.STR
                                                                     CHARCODES
                                                                     (EQ ELTTYPE 'TEXTBOX)
                                                                     SKW))
                                         (SETQ NEWCHAR# (COND
                                                           (STRPIECE (NCHARS STRPIECE))
                                                           (T 0)))
                                         (SETQ NEWLINE# (ADD1 (LENGTH NEWSTRS]
                                   (T (SETQ NEWCHAR# FIRSTCHAR#)
                                      (SETQ NEWLINE# FIRSTLINE#]
                            (COND
                               ((EQ LINE# LASTLINE#)              (* on the last, copy the part before and the part after as one)
                                (SETQ NEWSTRS (COND
                                                 [STRPIECE (NCONC1 NEWSTRS (COND
                                                                              ((EQ LASTCHAR# (NCHARS STR))
                                                                                    (* special check because SUBSTRING returns NIL rather than
                                                                                    the empty string.)
                                                                               STRPIECE)
                                                                              (T (CONCAT STRPIECE
                                                                                         (SUBSTRING STR
                                                                                                (ADD1 LASTCHAR#]
                                                 [(NEQ LASTCHAR# (NCHARS STR))
                                                  (NCONC1 NEWSTRS (SUBSTRING STR (ADD1 LASTCHAR#]
                                                 (T NEWSTRS]
```

(* any other windows that had this selection have had it deleted already so this doesn't do anything for them.)

```
                      [COND
                         ((IGREATERP NEWLINE# (LENGTH NEWSTRS))
```

(* this corresponds to deleting every thing in a line. Make sure that if it is the last line that the selection is reset)

```
                          (COND
                             ((EQ (SETQ NEWLINE# (LENGTH NEWSTRS))
                                  0)
                              (SETQ NEWCHAR# 0)
                              (COND
                                 ((EQ ELTTYPE 'TEXT)
```

(* deleted everything in a text element, delete the text element and set the selection to new text cursor.)

```
                                  (COND
                                     [(WINDOWPROP SKW 'CHANGEDTEXTELT)
```

(* make the history event for this edit so that it will restore the original text element)

```
                                      (PROG ((INITSELECTION (WINDOWPROP SKW 'CHANGEDTEXTELT NIL)))
                                            (COND
                                               ((POSITIONP INITSELECTION)
```

(* this text element was typing that was never officially added, don't record the deletion either.)

```
                                        (SK.DELETE.ELEMENT (LIST TEXTELT)
                                               SKW
                                               'DON'T))
                                       (T                (* selection was existing text, record this as a delete event.)
                                          (SK.DELETE.ELEMENT (LIST TEXTELT)
                                                 SKW
                                                 (LIST (fetch (SCREENELT GLOBALPART)
                                                              of (fetch (TEXTELTSELECTION SKTEXTELT)
                                                                        of INITSELECTION]
                                    (T (SK.DELETE.ELEMENT (LIST TEXTELT)
                                                  SKW)))
                                 (SKED.SET.SELECTION (SK.SCALE.POSITION.INTO.VIEWER (fetch (TEXT LOCATIONLATLON)
                                                                                            of (fetch (SCREENELT

                                                                                                       INDIVIDUALGLOBALPART
                                                                                                             )
                                                                                             of TEXTELT))
                                                                                    (VIEWER.SCALE SKW))
                                                     SKW)
                                                (RETURN NIL))
                                               ((EQ ELTTYPE 'TEXTBOX)          (* deleted everything in a textbox)
                                                NIL)))
                                      (T (SETQ NEWCHAR# (NCHARS (CAR (LAST NEWSTRS]
                                (SETQ PTRCHAR# (SKED.CHARACTERPOSITION NEWSTRS NEWLINE# NEWCHAR#))
                                (COND
                                   ((WINDOWPROP SKW 'CHANGEDTEXTELT)
```

(* this is not the first change to the text element. Collect the changes so that only one element is put on the undo stack, not one for each character.)

```
                                    (SETQ NEWELT (SK.UPDATE.ELEMENT (fetch (SCREENELT GLOBALPART) of TEXTELT)
                                                                    (SK.REPLACE.TEXT.IN.ELEMENT (fetch (SCREENELT GLOBALPART)
                                                                                                       of TEXTELT)
                                                                                                NEWSTRS)
                                                                    SKW T)))
                                ((AND CONTROLCHARTAIL (NEQ CONTROLCHARTAIL 'UNDO))
```

(* user typed a character command to create a new text box. Create it and put the remaining characters in it and set the cursor there.)

(* this is done here so that no undo event is created for the textbox that the user was in when all they did was type a control-cr.)

```
                                    (SKED.CREATE.NEW.TEXTBOX (fetch (SCREENELT INDIVIDUALGLOBALPART) of NEWELT)
                                           SKW
                                           (CDR CONTROLCHARTAIL))
                                    (RETURN))
                                   (T
```

(* this is the first edit change to a new element, call the PREEDITFN and save old text element so undo event can be constructed when the selection changes.)

```
                                      (OR (SK.CHECK.PREEDITFN SKW (fetch (SCREENELT GLOBALPART) of TEXTELT))
                                          (RETURN NIL))
                                      (SETQ NEWELT (SK.UPDATE.ELEMENT (fetch (SCREENELT GLOBALPART) of TEXTELT)
                                                                      (SKED.NEW.TEXTELT (fetch (SCREENELT GLOBALPART) of TEXTELT)
                                                                                        NEWSTRS)
                                                                      SKW))
                                      (WINDOWPROP SKW 'CHANGEDTEXTELT SELECTION)))
```

(* recalculate the line %# and char %# of the insertion point as the textboxes at least do justification.)

```
                [SETQ NEWCHAR# (CDR (SETQ NEWLINE# (SKED.LINE.AND.CHAR# (fetch (LOCALTEXTBOX
                                                                                LOCALLISTOFCHARACTERS
                                                                                )
                                                                         of (fetch (SCREENELT LOCALPART)
                                                                                    of NEWELT))
                                                                   PTRCHAR#]
                (SETQ NEWLINE# (CAR NEWLINE#]
  UNDO
       (COND
          ((NULL CONTROLCHARTAIL)                          (* set the selection to where the characters were just inserted.)
           (SKED.SET.SELECTION (CREATE.TEXT.SELECTION NEWELT NEWLINE# NEWCHAR# NIL NIL
                                                      (WINDOWPROP SKW 'DSP))
                  SKW))
          [(EQ CONTROLCHARTAIL 'UNDO)
```

(* user types in an undo after some characters or while selection was in the middle of text.)

```
             (PROG (INITSELECTION EDITEDELT)
                   (COND
                      ((SETQ INITSELECTION (WINDOWPROP SKW 'CHANGEDTEXTELT NIL))
                                                           (* in the middle of editing, undo these edits.)
                       [SETQ EDITEDELT (fetch (SCREENELT GLOBALPART)
                                          of (OR NEWELT (fetch (TEXTELTSELECTION SKTEXTELT)
                                                               of (OR SELECTION (ERROR "NO SELECTION WHEN
```

```
                                                                    THERE SHOULD BE"]
                                                    (* add event to history list so the undo can be undone.)
                        (COND
                           ((POSITIONP INITSELECTION)      (* add an ADD event because previously there was nothing
                                                           here.)
                            (SK.ADD.HISTEVENT 'ADD (LIST EDITEDELT)
                                   SKW)
                            (SK.CHECK.END.INITIAL.EDIT SKW EDITEDELT))
                           (T (SK.ENTER.EDIT.CHANGE SKW (fetch (SCREENELT GLOBALPART)
                                                          of (fetch (TEXTELTSELECTION SKTEXTELT)
                                                                of INITSELECTION))
                                    EDITEDELT)))
                        (SK.UNDO.LAST SKW)
                        (SKED.SET.SELECTION INITSELECTION SKW))
                       (T                                  (* haven't edited any characters in the current element, just
                                                           undo the last thing.)
                          (SK.UNDO.LAST SKW]
                 (T
```

(* user typed a character command to create a new text box. Create it and put the remaining characters in it and set the cursor there.)

(* set the selection so that adding the new text box will create an undo event for the character change that took place in this text box before the control-cr was typed.)

```
                 (SKED.SET.SELECTION (CREATE.TEXT.SELECTION NEWELT NEWLINE# NEWCHAR# NIL NIL
                                            (WINDOWPROP SKW 'DSP))
                        SKW)
                 (SKED.CREATE.NEW.TEXTBOX (fetch (SCREENELT INDIVIDUALGLOBALPART) of NEWELT)
                        SKW
                        (CDR CONTROLCHARTAIL])
```

(FIRST.N.ELEMENTS

```
  [LAMBDA (LST N)                                   (* rrb "20-Jan-86 18:05")
                                                    (* returns a list of the first N elements of LST)
    (for I from 1 to N as ELT in LST collect ELT])
```

(SKED.CREATE.NEW.TEXTBOX

```
  [LAMBDA (TEXTELT SKW CHARSTOINSERT ATSCALE)                ; Edited 20-Feb-87 17:53 by rrb
```

(* create a new text box. Create it and put the remaining characters in it and set the cursor there.)

```
    (PROG (CURRENTREGION CURRENTPOS NEWELT)
          (COND
             ((EQ (fetch (INDIVIDUALGLOBALPART GTYPE) of TEXTELT)
                  'TEXT)                                (* current selection is text, move cursor so that a new text will be
              created.)
              (SETQ CURRENTREGION (APPLY (FUNCTION UNIONREGIONS)
                                         (fetch (TEXT LISTOFREGIONS) of TEXTELT)))
                                                        (* get the position of the text now in.)
              (SETQ CURRENTPOS (fetch (TEXT LOCATIONLATLON) of TEXTELT))
                                                        (* translate the position downward, leaving enough room for
                                                        another the same size.)
              [SETQ CURRENTPOS (CREATEPOSITION (fetch (POSITION XCOORD) of CURRENTPOS)
                                       (DIFFERENCE (fetch (POSITION YCOORD) of CURRENTPOS)
                                              (fetch (REGION HEIGHT) of CURRENTREGION]
```

(* transform the position into viewer coordinates, subtract 16 for room in between and set the selection.)

```
              [SETQ CURRENTPOS (SK.SCALE.POSITION.INTO.VIEWER CURRENTPOS (OR (NUMBERP ATSCALE)
                                                                             (VIEWER.SCALE SKW]
              (SKED.SET.SELECTION (CREATEPOSITION (fetch (POSITION XCOORD) of CURRENTPOS)
                                         (DIFFERENCE (fetch (POSITION YCOORD) of CURRENTPOS)
                                                16))
                     SKW)                               (* insert a space so that there will be a text element here.)
              (SKED.INSERT [OR CHARSTOINSERT (CONSTANT (LIST (CHARCODE SPACE]
                     SKW ATSCALE)
              (RETURN))
             (T [SETQ CURRENTREGION (COND
                                       [(NULL TEXTELT)          (* create a region around the cursor)
                                        (UNSCALE.REGION (CREATEREGION (DIFFERENCE (LASTMOUSEX SKW)
                                                                                  50)
                                                                      (DIFFERENCE (LASTMOUSEY SKW)
                                                                                  35)
                                                                      100 70)
                                                        (OR (NUMBERP ATSCALE)
                                                            (VIEWER.SCALE SKW]
                                       (T                       (* create a region below the current element.)
```

(* should limit the dimensions of this box and put it to right if it would appear off the screen.
Later)

```
                                        (SETQ CURRENTREGION (fetch (TEXTBOX TEXTBOXREGION) of TEXTELT))
                                        (create REGION using CURRENTREGION BOTTOM _
                                                             (DIFFERENCE (fetch (REGION BOTTOM) of CURRENTREGION)
```

```
                                                                    (PLUS (fetch (REGION HEIGHT) of CURRENTREGION
                                                                                 )
                                                                          (TIMES (OR (NUMBERP ATSCALE)
                                                                                     (VIEWER.SCALE SKW))
                                                                                 16]
            (SETQ CURRENTREGION (MAP.GLOBAL.REGION.ONTO.GRID CURRENTREGION SKW))
            [SETQ NEWELT (COND
                                ((NULL TEXTELT)                    (* create a default textbox)
                                 (SK.TEXTBOX.CREATE CURRENTREGION (fetch (SKETCHCONTEXT SKETCHBRUSH)
                                                                         of (WINDOWPROP SKW 'SKETCHCONTEXT))
                                        (OR (NUMBERP ATSCALE)
                                            (SK.INPUT.SCALE SKW))
                                        SKW))
                                ((EQ (fetch (INDIVIDUALGLOBALPART GTYPE) of TEXTELT)
                                     'TEXTBOX)                     (* copy the characteristics of the current text box)
                                 (SK.TEXTBOX.CREATE1 CURRENTREGION (fetch (TEXTBOX TEXTBOXBRUSH) of TEXTELT)
                                        (LIST "")
                                        (fetch (TEXTBOX INITIALSCALE) of TEXTELT)
                                        (fetch (TEXTBOX TEXTSTYLE) of TEXTELT)
                                        (fetch (TEXTBOX FONT) of TEXTELT)
                                        (fetch (TEXTBOX TEXTBOXDASHING) of TEXTELT)
                                        (fetch (TEXTBOX TEXTBOXFILLING) of TEXTELT)
                                        (fetch (TEXTBOX TEXTCOLOR) of TEXTELT]
            (SKED.SET.SELECTION (CREATE.TEXT.SELECTION (SKETCH.ADD.AND.DISPLAY NEWELT SKW)
                                                       1 0 NIL NIL (WINDOWPROP SKW 'DSP))
                    SKW)))                                         (* put the remaining characters in the new textbox.)
          (AND CHARSTOINSERT (SKED.INSERT CHARSTOINSERT SKW ATSCALE])
```

(SKED.CHARACTERPOSITION

```
  [LAMBDA (STRLST LINE# CHAR#)                                     (* rrb "22-Jan-85 15:39")

          (* returns the character position of the character at line number LINE# and character position CHAR#)

    (PROG ((CHARPOS 0))
          [bind NCHARS for STR in STRLST as N from 1 to (SUB1 LINE#) do [SETQ CHARPOS (PLUS CHARPOS (SETQ NCHARS
                                                                                                        (NCHARS STR]
                                                                                  (COND
                                                                                     ((NEQ (NTHCHARCODE STR NCHARS)
                                                                                           (CHARCODE EOL))

          (* unless the last character in the string is CR, add one for the implied space or CR.)

                                                                                                 (SETQ CHARPOS (ADD1 CHARPOS]
          (RETURN (PLUS CHARPOS CHAR#])
```

(SKED.LINE.AND.CHAR#

```
  [LAMBDA (STRLST CHARPOS)                                         (* rrb "14-Jun-85 18:12")

          (* returns a dotted pair of the line number and character within line position of a character position.)

    (bind NCHARS (CHARSLEFT _ CHARPOS) for STR in STRLST as N from 1
       do [COND
             [(EQ (SETQ NCHARS (NCHARS STR))
                  CHARSLEFT)

          (* at end of a line. If the line ends in CR, return ptr to beginning of next line.)

              (COND
                 ((AND (EQ (NTHCHARCODE STR NCHARS)
                           (CHARCODE EOL))
                       (GREATERP (LENGTH STRLST)
                                 N))
                  (RETURN (CONS (ADD1 N)
                                0)))
                 (T (RETURN (CONS N CHARSLEFT]
            ((IGREATERP NCHARS CHARSLEFT)
             (RETURN (CONS N CHARSLEFT)))
            (T (SETQ CHARSLEFT (DIFFERENCE CHARSLEFT (COND
                                                        ((EQ (NTHCHARCODE STR NCHARS)
                                                             (CHARCODE EOL))
                                                                   (* if the line ends in CR, don't add one for the)
                                                         NCHARS)
                                                        (T (ADD1 NCHARS]
       finally                                                     (* return something that is after last character of last line.)
          (RETURN (CONS (LENGTH STRLST)
                        (NCHARS (CAR (LAST STRLST])
```

(\SKED.DELETE.WORD.FROM.STRING

```
  [LAMBDA (STRING)                                                 (* rrb "11-Jul-86 17:17")
                                                                   (* returns a string that has the last word of STRING deleted.)
    (PROG ((END (NCHARS STRING))
           CLASS)
      SKBLANKS
          (COND
```

```
            ((EQ END 0)                                          (* ran out of characters.)
             (RETURN))
            ((EQ (SETQ CLASS (SK.WORD.BREAK.CLASS (NTHCHARCODE STRING END)))
                 22)
             (SETQ END (SUB1 END))
             (GO SKBLANKS)))                                     (* now skip characters that have the same class as the first one
     encountered.)
      SKSAME
           (SETQ END (SUB1 END))
           (COND
              ((EQ END 0)                                        (* ran out of characters.)
               (RETURN))
              ((EQ (SK.WORD.BREAK.CLASS (NTHCHARCODE STRING END))
                   CLASS)
               (GO SKSAME))
              (T (RETURN (SUBSTRING STRING 1 END])
```

(**SKED.INSERT.CHARS.TO.STR**

```
  [LAMBDA (CHARCODES INCLUDECR SKW)                              (* rrb "11-Jul-86 17:18")
    (DECLARE (SPECVARS NEWSTRS STRPIECE))

          (* takes a list of characters and makes it into strings on the free variable NEWSTRS.
          The variable STRPIECE is set to the last line of characters. NEWSTRS is a list of the strings that precede this one which is
          used in the case of backspace onto the previous line.)

    (PROG (LINELST THISLINE REMAININGCHARS CLASS)
          [for CHAR in CHARCODES
             do (SELECTQ (SK.GETSYNTAX CHAR)
                     (CHARDELETE                                 (* delete the previous character.)
                                 [COND
                                    (THISLINE                    (* easy case of deleting type in.)
                                           (SETQ THISLINE (CDR THISLINE)))
                                    (LINELST                     (* deleting a typed in CR.)
                                           (SETQ THISLINE (CAR LINELST))
                                           (SETQ LINELST (CDR LINELST)))
                                    [STRPIECE                    (* remove the previous character from the current string.)
                                           (COND
                                              ((EQ (NCHARS STRPIECE)
                                                   1)
                                               (SETQ STRPIECE NIL))
                                              (T (SETQ STRPIECE (SUBSTRING STRPIECE 1 -2]
                                    [NEWSTRS (SETQ STRPIECE (CAR (LAST NEWSTRS)))
                                           (SETQ NEWSTRS (BUTLAST NEWSTRS))
                                           (COND
                                              ((EQ (NTHCHARCODE STRPIECE -1)
                                                   (CHARCODE EOL))
                                                                 (* remove previous eol)
                                               (COND
                                                  ((EQ (NCHARS STRPIECE)
                                                       1)
                                                   (SETQ STRPIECE NIL))
                                                  (T (SETQ STRPIECE (SUBSTRING STRPIECE 1 -2]
                                    (T                           (* no characters to delete)
                                       (FLASHW (TTYDISPLAYSTREAM])
                     (WORDDELETE                                 (* delete the previous word)

          (* use the TEdit word bounding readtable. Code are%: character = 21 -
          space = 22 -
          punctuation = 20)

                                 [COND
                                    [[OR THISLINE (PROG1 (SETQ THISLINE (CAR LINELST))
                                                         (SETQ LINELST (CDR LINELST)))]
                                                                 (* easy case of deleting type in.)

          (* if this line was empty, skip the cr that created it as part of the white space before the word.)
                                                                 (* skip any whitespace)
                                     (COND
                                        ([NULL (SETQ THISLINE (for TAIL on THISLINE
                                                                 while (EQ (SK.WORD.BREAK.CLASS (CAR TAIL))
                                                                           22)
                                                                 finally (RETURN TAIL]
                                                                 (* the whitespace backed up to the beginning of a line.
                                                                 quit there.)
                                         NIL)
                                        (T (SETQ CLASS (SK.WORD.BREAK.CLASS (CAR THISLINE)))
                                                                 (* skip all things of the same class as the first character before
                                                                 the whitespace)
                                           (SETQ THISLINE (for TAIL on THISLINE
                                                             until (NEQ (SK.WORD.BREAK.CLASS (CAR TAIL))
                                                                        CLASS)
                                                             finally (RETURN TAIL]
                                 (STRPIECE                       (* remove the previous character from the current string.)
                                           (SETQ STRPIECE (\SKED.DELETE.WORD.FROM.STRING STRPIECE)))
                                 (NEWSTRS [SETQ STRPIECE (\SKED.DELETE.WORD.FROM.STRING (CAR (LAST NEWSTRS]
                                           (SETQ NEWSTRS (BUTLAST NEWSTRS)))
```

```
                                    (T                                  (* no characters to delete)
                                       (FLASHW (TTYDISPLAYSTREAM])
                              (DELETE                                   (* delete selection. Here that means don't insert anything.))
                              (UNDO
```

(* by side effect this flushes any characters typed after the undo but it's not clear where they should go anyway.)

```
                                    (RETURN 'UNDO))
                              ((REDO FN CMD)
                                   (STATUSPRINT SKW "
                                          " "Not implemented in this editor. Sorry."))
                              (COND
                                 [(OR (EQ CHAR (CHARCODE EOL))
                                      (EQ CHAR (CHARCODE LINEFEED)))      (* eol)
                                  (COND
                                     ((KEYDOWNP 'CTRL)                    (* user entered control return, save remaining characters and
                                                                          return indicator)
                                      (SETQ REMAININGCHARS (MEMB CHAR CHARCODES))
                                      (RETURN))
                                     (T (SETQ LINELST (CONS (COND
                                                               (INCLUDECR
                                                                          (* text boxes need to have the CRs left in.)
                                                                  (CONS (CHARCODE EOL)
                                                                        THISLINE))
                                                               (T THISLINE))
                                                            LINELST))
                                        (SETQ THISLINE NIL]
                                 (T
```

(* add this character onto the front of this line; reversal will happen before conversion to string and return.)

```
                                  (SETQ THISLINE (CONS CHAR THISLINE]
                           (COND
                              [LINELST                                  (* had a cr in the character set.)
                                     [SETQ NEWSTRS (NCONC NEWSTRS [CONS (JOINCHARS STRPIECE (REVERSE (CAR (LAST LINELST]
                                                        (for CHLST in (REVERSE (BUTLAST LINELST))
                                                           collect (STRINGFROMCHARACTERS (REVERSE CHLST]
                                     (SETQ STRPIECE (STRINGFROMCHARACTERS (REVERSE THISLINE]
                              [THISLINE                                 (* no new lines, add these characters onto STRPIECE)
                                     (SETQ STRPIECE (JOINCHARS STRPIECE (REVERSE THISLINE]
                              (T                                        (* no new lines, or characters, leave STRPIECE alone.)
                                 NIL))
                           (RETURN REMAININGCHARS])
```

(JOINCHARS

```
[LAMBDA (STR CHARCODES)                                                 (* rrb "27-Dec-84 16:56")
```

(* makes a string by attaching the list of character codes CHARCODES onto the end of the string STR.)

```
   (COND
      ((NULL STR)
       (STRINGFROMCHARACTERS CHARCODES))
      (T (CONCAT STR (STRINGFROMCHARACTERS CHARCODES])
```

(STRINGFROMCHARACTERS

```
[LAMBDA (CHARS)                                                         (* rrb "27-Dec-84 16:56")
                                                                        (* makes a string from a list of characters)
   (MKSTRING (PACKC CHARS])
```

(GETALLCHARS

```
[LAMBDA (FILE)                                                          (* rrb "11-Jul-86 17:22")
                                                                        (* reads all of the characters that are on FILE.)
   (while (\SYSBUFP) collect (\GETKEY])
```

(CLEANUP.EDIT

```
[LAMBDA (SKW)                                                           (* rrb "12-Oct-84 11:23")
```

(* Place holder to propagates the current edit into other windows that might be viewing the same sketch.)

```
                                                                        (* called when the cursor leaves the sketch window.)
   NIL])
```

(SKED.NEW.TEXTELT

```
[LAMBDA (OLDGTEXTELT NEWSTRLST)                                         (* rrb "26-Apr-85 15:59")
```

(* creates a new text element by replacing only the list of characters of an old one.)

```
   (create GLOBALPART
           COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART) of OLDGTEXTELT)
           INDIVIDUALGLOBALPART _ (COND
                                     ((EQ (fetch (GLOBALPART GTYPE) of OLDGTEXTELT)
                                          'TEXT)
                                      (TEXT.SET.GLOBAL.REGIONS (create TEXT using (fetch (GLOBALPART
```

```
                                                                                    INDIVIDUALGLOBALPART
                                                                                    )
                                                                                **of** OLDGTEXTELT)
                                                                              LISTOFCHARACTERS _ NEWSTRLST)))
                                                    (T (TEXTBOX.SET.GLOBAL.REGIONS (create TEXTBOX using (fetch (GLOBALPART
                                                                                    INDIVIDUALGLOBALPART
                                                                                            )
                                                                                       of OLDGTEXTELT)
                                                                                    LISTOFCHARACTERS _
                                                                                    (OR NEWSTRLST
                                                                                        '("")])

)
```

(* * line adding functions)

```
(DEFINEQ
```

(MAP.SCREEN.POSITION.ONTO.GRID

```
  [LAMBDA (PT WINDOW FLIPGRIDSENSEFLG)                    (* rrb "11-Jul-86 15:51")
```

(* maps a point in screen coordinates into the screen coordinate that is closest to the grid in WINDOW. FLIPGRIDSENSEFLG is used to flip whether to use the grid or not and allows right buttoning to be the opposite of standard.)

```
    (COND
       [(COND
           ((WINDOWPROP WINDOW 'USEGRID)                  (* window is calling for grid. Use it unless grid sense is switched.)
            (NOT FLIPGRIDSENSEFLG))
           (T                                             (* window is not calling for grid, don't use it unless grid sense is
              switched.)
              FLIPGRIDSENSEFLG))
        (PROG ((GRID (SK.GRIDFACTOR WINDOW))
               (SCALE (VIEWER.SCALE WINDOW)))
              (RETURN (create POSITION
                              XCOORD _ (MAP.SCREEN.ONTO.GRID (fetch (POSITION XCOORD) of PT)
                                             SCALE GRID (DSPXOFFSET NIL WINDOW))
                              YCOORD _ (MAP.SCREEN.ONTO.GRID (fetch (POSITION YCOORD) of PT)
                                             SCALE GRID (DSPYOFFSET NIL WINDOW]
       (T                                                 (* avoid pt creation in the non-grid case.)
          PT])
```

(NEAREST.ON.GRID

```
  [LAMBDA (X GRIDSIZE)                                    (* rrb "23-Sep-86 10:51")
                                                          (* returns the point on a grid of size GRIDSIZE that is closest to
    X)
    (TIMES GRIDSIZE (FIX (FQUOTIENT [COND
                                       ((GREATERP X 0.0)          (* assymetry around 0.0)
                                        (FPLUS X (FQUOTIENT GRIDSIZE 2.0)))
                                       (T (FDIFFERENCE X (FQUOTIENT GRIDSIZE 2.0]
                                 GRIDSIZE])
```

(SK.MIDDLE.TITLEFN

```
  [LAMBDA (SKW STANDARDMENUFLG)                           (* rrb "23-Sep-86 17:59")
```

(* the middle button when down in the title bar. If the operations menu is not up, put it up, wait for a selection then take it down.)

```
    (PROG (OPMENUW)
          (OR [SETQ OPMENUW (COND
                               (STANDARDMENUFLG (MENUWINDOW (CREATE.SKETCHW.COMMANDMENU NIL NIL T SKW)
                                                            T))
                               (T (SK.GET.VIEWER.POPUP.MENU SKW]
              (RETURN))                                   (* move opmenu to near cursor)
          [MOVEW OPMENUW [IMIN (ADD1 LASTMOUSEX)
                               (IDIFFERENCE (BITMAPWIDTH (SCREENBITMAP SKW))
                                      (WINDOWPROP OPMENUW 'WIDTH]
                 (IMAX 0 (IMIN (IDIFFERENCE LASTMOUSEY (IQUOTIENT (WINDOWPROP OPMENUW 'HEIGHT)
                                                                  2))
                               (IDIFFERENCE (BITMAPHEIGHT (SCREENBITMAP SKW))
                                      (WINDOWPROP OPMENUW 'HEIGHT]
          (RETURN (PROG ([OPMENU (CAR (WINDOWPROP OPMENUW 'MENU]
                         (DSP (WINDOWPROP OPMENUW 'DSP))
                         SELCOMMAND)
                        [SETQ SELCOMMAND (CADR (CAR (RESETLST
                                                      (RESETSAVE (OPENW OPMENUW)
                                                             (LIST 'CLOSEW OPMENUW))
                                                      (MENU.HANDLER OPMENU DSP T T NIL))]
                                                      (* evaluate menu form after image has been taken down.)
                        (RETURN (SK.APPLY.MENU.COMMAND SELCOMMAND SKW])
```

(WB.BUTTON.HANDLER

```
  [LAMBDA (W)                                             ; Edited 19-Nov-87 11:37 by rrb
                                                          (* handles a button event in a whiteboard window.)
```

```
    (TOTOPW W)
    (RESETLST
        (COND
           ([AND (LASTMOUSESTATE RIGHT)
                 (NOT (INSIDEP (DSPCLIPPINGREGION NIL W)
                               (LASTMOUSEX W)
                               (LASTMOUSEY W]                    (* handle the right button with out the lock)
            (DOWINDOWCOM W))
           [(OBTAIN.MONITORLOCK (SKETCH.MONITORLOCK W)
                   T T)
```

(* * make sure nothing else is happening in the window.)

```
             (PROG (ACTIVEREGION BUTTONFN X Y IMAGEOBJ)
                   (RETURN
                    (COND
                       ((OR (.DELETEKEYDOWNP.)
                            (.MOVEKEYDOWNP.))
```

(* if the DELETE key is held down, handle it with the copy button event fn.)

```
                        (SK.COPY.BUTTONEVENTFN W))
                       [(AND [NOT (INSIDEP (DSPCLIPPINGREGION NIL W)
                                           (SETQ X (LASTMOUSEX W))
                                           (SETQ Y (LASTMOUSEY W]
                             (LASTMOUSESTATE (NOT UP)))          (* title bar or border action)
                        (COND
                           ((LASTMOUSESTATE MIDDLE)              (* offer command menu)
                            (SK.MIDDLE.TITLEFN W))
                           ((LASTMOUSESTATE RIGHT)
                            (DOWINDOWCOM W]
                       [(AND
                         (LASTMOUSESTATE (OR LEFT MIDDLE))
                         (for ELT in (LOCALSPECS.FROM.VIEWER W)
                            do (COND
                                  ((AND (SETQ ACTIVEREGION (fetch (SCREENELT LOCALHOTREGION) of ELT))
                                        (SETQ BUTTONFN (GETSKETCHPROP W 'BUTTONEVENTINFN))
                                        (INSIDEP ACTIVEREGION X Y))
```

(* look for an element with a hot region. This is done before the image object region so that the image objects function can be overridden.)

```
                                                                 (* inside of the hot region for an element with a button event fn.)
                                   (APPLY* BUTTONFN W (fetch (SCREENELT GLOBALPART) of ELT))
                                   (RETURN T))
                                  ([AND (type? SKIMAGEOBJ (fetch (SCREENELT INDIVIDUALGLOBALPART) of ELT))
                                        (INIMAGEOBJ ELT X Y)
                                        (NOT (SK.ELEMENT.PROTECTED? (fetch (SCREENELT GLOBALPART) of ELT)
                                                    'BUTTONEVENTINFN]
```

(* element is frozen, don't call it's imageobject fn. The rationale here is that the buttonineventfn is often used to edit the image object.)

```
                                                                 (* inside of a imageobj, run its BUTTONEVENTINFN element.)
                                                                 (* there are miriad other arguments to the BUTTONEVENTINFN
                                   that are not applicable.)
                                   (COND
                                      ((EQ (RESETLST
                                               (RESETSAVE NIL (LIST 'DSPCLIPPINGREGION
                                                                    (DSPCLIPPINGREGION
                                                                     (INTERSECTREGIONS (fetch (LOCALSKIMAGEOBJ
                                                                                               SKIMOBJLOCALREGION)
                                                                                              of (fetch (SCREENELT
                                                                                                         LOCALPART)
                                                                                                        of ELT))
                                                                                       (DSPCLIPPINGREGION NIL W))
                                                                     W)
                                                                    W))
                                               (APPLY* [fetch (IMAGEFNS BUTTONEVENTINFN)
                                                          of (fetch (IMAGEOBJ IMAGEOBJFNS)
                                                                of (SETQ IMAGEOBJ (fetch (SKIMAGEOBJ SKIMAGEOBJ)
                                                                                     of (fetch (SCREENELT
                                                                                                INDIVIDUALGLOBALPART
                                                                                                )
                                                                                               of ELT]
                                                       IMAGEOBJ
                                                       (GETSTREAM W 'OUTPUT)
                                                       NIL
                                                       [DIFFERENCE X (fetch (REGION LEFT) of (SETQ ACTIVEREGION
                                                                                                  (SK.ITEM.REGION ELT]
                                                       (DIFFERENCE Y (fetch (REGION BOTTOM) of ACTIVEREGION))
                                                       W NIL (CAR (DECODEBUTTONS LASTMOUSEBUTTONS))))
                                             'CHANGED)
```

(* the extra arguments to the buttonineventfn are to make it look more like the call an image object gets from Tedit.)

```
                                                                 (* element changed, update the local fields and redisplay it.)
                                      (SKETCH.ELEMENT.CHANGED W (fetch (SCREENELT GLOBALPART) of ELT)
                                         W)))                    (* exit loop)
                                  (RETURN T]
```

```
                    [(LASTMOUSESTATE LEFT)                          (* move cursor)
                     (RESET.LINE.BEING.INPUT W)
                     (SKED.MOVE.SELECTION W (WINDOWPROP W 'USEGRID]
                    ((LASTMOUSESTATE RIGHT)                         (* extend selection)
                     (RESET.LINE.BEING.INPUT W)
                     (SKED.EXTEND.SELECTION W))
                    ((LASTMOUSESTATE MIDDLE)                        (* add a point to the current line)
                     (WB.ADD.NEW.POINT W]
          (T (STATUSPRINT W "
                    " "Sketch operation in progress.  Please wait.")))))])
```

(WB.ADD.NEW.POINT

```
[LAMBDA (WIN)                                                       (* rrb "12-May-86 18:25")
```

(* tracks the cursor with a point cursor drawing a rubberband line until the cursor is let up.
Nothing happens if cursor goes outside. Adds line segment to the currently being built line.)

```
   (PROG ((LINEPTS (WINDOWPROP WIN 'INPUTLINE))
          PT ELT)                                                   (* INPUTLINE are the points on the curve being input in most
                                                                    recently added first.)
         (COND
            [(NULL LINEPTS)                                         (* this is the first point being put down;)
             (COND
                ((SETQ PT (SK.READ.POINT.WITH.FEEDBACK WIN POINTREADINGCURSOR NIL NIL 'LEFT))
```

(* put the local wire element on the property list so that it can be added to.)

```
                 (WINDOWPROP WIN 'INPUTLINE PT]
            [(type? INPUTPT LINEPTS)                                (* this is the second point being put down, make a line for it)
             (COND
                ((SETQ PT (WB.RUBBERBAND.POSITION (fetch (INPUTPT INPUT.POSITION) of LINEPTS)
                                  WIN))
```

(* read the second point of the line on the upclick and make a sketch element)

```
                 (SETQ ELT (SKETCH.ADD.AND.DISPLAY (WIRE.INPUTFN WIN (LIST (SK.MAP.INPUT.PT.TO.GLOBAL LINEPTS
                                                                           WIN)
                                                                  (SK.MAP.INPUT.PT.TO.GLOBAL PT WIN)))
                                   WIN T))
                 (WINDOWPROP WIN 'INPUTLINE (COND
                                               ((fetch (SKETCHCONTEXT SKETCHLINEMODE) of (WINDOWPROP
                                                                                          WIN
                                                                                          'SKETCHCONTEXT))
                                                                    (* T indicates two point default. Start a new line.)
                                                NIL)
                                               (T
```

(* put the local wire element on the property list so that it can be added to.)

```
                                                  ELT]
            ((type? WIRE (fetch (SCREENELT INDIVIDUALGLOBALPART) of LINEPTS))
                                                                    (* add this point to all this sketch.)
             (COND
                ((SETQ PT (WB.RUBBERBAND.POSITION [CAR (LAST (fetch KNOTS of (fetch (SCREENELT LOCALPART)
                                                                                    of LINEPTS]
                                  WIN))
                 (WINDOWPROP WIN 'INPUTLINE (WIRE.ADD.POINT.TO.END LINEPTS PT WIN])
```

(WB.DRAWLINE

```
[LAMBDA (WIREELT STREAM REG OPERATION CLOSEDFLG DASHING BRUSH) (* rrb " 6-May-86 17:43")
                                                                (* draws a line from its white board element.)
   (PROG ((LWIRE (fetch (SCREENELT LOCALPART) of WIREELT))
          (GWIRE (fetch (SCREENELT INDIVIDUALGLOBALPART) of WIREELT))
          (BRUSHSIZE (IMAX (FIXR (fetch (BRUSH BRUSHSIZE) of BRUSH))
                           1))
          (BRUSHCOLOR (fetch (BRUSH BRUSHCOLOR) of BRUSH))
          PTS LOCALARROWPTS GARROWSPECS)
         (COND
            ((SETQ PTS (fetch (LOCALWIRE KNOTS) of LWIRE))
             (SETQ GARROWSPECS (fetch (WIRE WIREARROWHEADS) of GWIRE))
             (SETQ LOCALARROWPTS (fetch (LOCALWIRE ARROWHEADPTS) of LWIRE))
             (SETQ PTS (\SK.ADJUST.FOR.ARROWHEADS PTS LOCALARROWPTS GARROWSPECS STREAM))
             (COND
                [(NEQ (fetch (BRUSH BRUSHSHAPE) of BRUSH)
                      'ROUND)                                   (* use the slower curve drawing method that handles brushes)
                 (for PTTAIL on PTS while (CDR PTTAIL) do (DRAWCURVE (LIST (CAR PTTAIL)
                                                                           (CADR PTTAIL))
                                                                    NIL BRUSH DASHING STREAM)
                    finally (COND
                               (CLOSEDFLG                           (* if closed, finish with a line back to the origin.)
                                      (DRAWCURVE (LIST (CAR PTTAIL)
                                                       (CAR PTS))
                                                 NIL BRUSH DASHING STREAM))
                               (T                                   (* if not closed, check for arrowheads.)
                                  (DRAWARROWHEADS GARROWSPECS LOCALARROWPTS STREAM BRUSH OPERATION]
```

```
                (T (OR (EQ BRUSHSIZE 1)
                       (DRAWCURVE (LIST (CAR PTS))
                              NIL BRUSH NIL STREAM))
                   (for PTTAIL on PTS while (CDR PTTAIL) do (DRAWLINE (fetch (POSITION XCOORD) of (CAR PTTAIL))
                                                                      (fetch (POSITION YCOORD) of (CAR PTTAIL))
                                                                      (fetch (POSITION XCOORD) of (CADR PTTAIL))
                                                                      (fetch (POSITION YCOORD) of (CADR PTTAIL))
                                                                      BRUSHSIZE OPERATION STREAM BRUSHCOLOR DASHING)
                                                                (* put a round shape at each intersection point.)
                                                           (OR (EQ BRUSHSIZE 1)
                                                               (DRAWCURVE (LIST (CADR PTTAIL))
                                                                      NIL BRUSH NIL STREAM))
                      finally (COND
                                 (CLOSEDFLG                          (* if closed, finish with a line back to the origin.)
                                        (DRAWLINE (fetch (POSITION XCOORD) of (CAR PTTAIL))
                                                  (fetch (POSITION YCOORD) of (CAR PTTAIL))
                                                  (fetch (POSITION XCOORD) of (CAR PTS))
                                                  (fetch (POSITION YCOORD) of (CAR PTS))
                                                 BRUSHSIZE OPERATION STREAM BRUSHCOLOR DASHING))
                                 (T                                  (* if not closed, check for arrowheads.)
                                    (DRAWARROWHEADS GARROWSPECS LOCALARROWPTS STREAM BRUSH OPERATION])
```

(WB.RUBBERBAND.POSITION

```
  [LAMBDA (STARTPOSITION WINDOW)                                     (* rrb "11-Jul-86 15:51")
                                                                     (* gets the other end of a line via a rubberband prompting)
    (SK.READ.POINT.WITH.FEEDBACK WINDOW POINTREADINGCURSOR (FUNCTION SK.RUBBERBAND.FEEDBACKFN)
           (LIST STARTPOSITION (MAX (TIMES [fetch (BRUSH BRUSHSIZE) of (fetch (SKETCHCONTEXT SKETCHBRUSH)
                                                                          of (WINDOWPROP WINDOW 'SKETCHCONTEXT]
                                           (QUOTIENT (SK.INPUT.SCALE WINDOW)
                                                  (VIEWER.SCALE WINDOW)))
                                    1))
           'LEFT])
```

(SK.RUBBERBAND.FEEDBACKFN

```
  [LAMBDA (X Y WINDOW ORIGPT&SIZE)                                   (* rrb "12-May-86 18:01")
```

(* provides the rubberbanding feedback for the user inputting a point for an open wire from the middle button.)

```
    (SHOWSKETCHXY X Y WINDOW)
    (DRAWLINE (fetch (POSITION XCOORD) of (CAR ORIGPT&SIZE))
           (fetch (POSITION YCOORD) of (CAR ORIGPT&SIZE))
           X Y (CADR ORIGPT&SIZE)
           'INVERT WINDOW])
```

(RESET.LINE.BEING.INPUT

```
  [LAMBDA (SKW)                                                      (* rrb "31-MAR-83 18:03")
                                                                     (* resets the line that is being input by the user.)
    (WINDOWPROP SKW 'INPUTLINE NIL])

)
```

(* * Was MODERNIZE loaded before?)

```
(CL:WHEN (GETD 'MODERNWINDOW.SETUP)
    (MODERNWINDOW.SETUP 'WB.BUTTON.HANDLER))

(DEFINEQ
```

(NEAREST.EXISTING.POSITION

```
  [LAMBDA (SKW)
```

(* returns the existing position of skw that is closed to the lastmouse x and y.)

```
    (LASTMOUSEPOSITION SKW])
```

(WB.NEARPT

```
  [LAMBDA (TARGETPT HITPT)                                           (* rrb " 4-MAR-83 16:22")
                                                                     (* returns T if HITPT is "near" TARGETPT.)
    (PROG ((TX (fetch (POSITION XCOORD) of TARGETPT))
           (TY (fetch (POSITION YCOORD) of TARGETPT))
           (HX (fetch (POSITION XCOORD) of HITPT))
           (HY (fetch (POSITION YCOORD) of HITPT))
           (WB.POINT.WIDTH 4))
          (RETURN (AND (IGREATERP HX (IDIFFERENCE TX WB.POINT.WIDTH))
                       (IGREATERP (IPLUS TX WB.POINT.WIDTH)
                              HX)
                       (IGREATERP HY (IDIFFERENCE TY WB.POINT.WIDTH))
                       (IGREATERP (IPLUS TY WB.POINT.WIDTH)
                              HY])
```

(LASTMOUSEPOSITION

```
  [LAMBDA (WIN)
    (create POSITION
            XCOORD _ (LASTMOUSEX WIN)
            YCOORD _ (LASTMOUSEY WIN])

)
```

FUNCTION INDEX

RECORD INDEX