File created: 5-Dec-2023 00:44:13 **{WMEDLEY}<library>sketch>SKETCH.;5**

edit by: rmk

changes to: (VARS SKETCHCOMS)

previous date: 19-Oct-2023 23:55:27 {WMEDLEY}<library>sketch>SKETCH.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ SKETCHCOMS
       [[DECLARE%: FIRST DOCOPY DONTEVAL@LOAD
              (P (PROG ((NOTECARDSFLG (GETPROP 'NOTECARDS 'FILEDATES))
                        (SKETCHFLG (AND (BOUNDP 'ALL.SKETCHES)
                                        ALL.SKETCHES))
                        TEDITFLG)
                       ;; current knows about SKETCH TEDIT and NOTECARDS.  Everyone else loses.
                       [MAP.PROCESSES (FUNCTION (LAMBDA (PROC PROCNAME PROCFORM)
                                                  (AND (EQ (CAR PROCFORM)
                                                           '\TEDIT1)
                                                       (SETQ TEDITFLG T]
                       (COND ((AND (BOUNDP 'ALL.SKETCHES)
                                   (OR SKETCHFLG NOTECARDSFLG TEDITFLG))
                              (ERROR (CONCAT "Please close" (COND (SKETCHFLG " all open Sketch windows,")
                                                                  (T ""))
                                             (COND (NOTECARDSFLG (CONCAT (COND (SKETCHFLG " and")
                                                                               (T ""))
                                                                         " any open notefiles,"))
                                                   (T ""))
                                             (COND (TEDITFLG (CONCAT (COND ((OR SKETCHFLG NOTECARDSFLG)
                                                                            " and")
                                                                           (T ""))
                                                                     " any TEDIT windows that have sketches in
                                                                     them,"))
                                                   (T ""))
                                             " then type 'RETURN'.
                                             To abort loading the new version of Sketch, type '^'."]
        (FNS SKETCH SKETCH.FROM.A.FILE SKETCHW.CREATE SKETCH.RESET SKETCHW.FIG.CHANGED SK.WINDOW.TITLE EDITSLIDE
             EDITSKETCH SK.PUT.ON.FILE SK.OUTPUT.FILE.NAME SKETCH.PUT SK.GET.FROM.FILE SK.INCLUDE.FILE
             SK.GET.IMAGEOBJ.FROM.FILE SKETCH.GET ADD.SKETCH.TO.VIEWER FILENAMELESSVERSION
             SK.ADD.ELEMENTS.TO.SKETCH SKETCH.SET.A.DEFAULT SK.POPUP.SELECTIONFN GETSKETCHWREGION SK.ADD.ELEMENT
             SK.ADD.PRIORITY.ELEMENT.TO.SKETCH SK.ELTS.BY.PRIORITY SK.ORDER.ELEMENTS
             SK.ADD.PRIORITY.LOCAL.ELEMENT.TO.SKETCH SK.ADD.ELEMENTS SK.CHECK.WHENADDEDFN SK.APPLY.MENU.COMMAND
             SK.DELETE.ELEMENT1 SK.MARK.DIRTY SK.MARK.UNDIRTY SK.MENU.AND.RETURN.FIELD SKETCH.SET.BRUSH.SHAPE
             SKETCH.SET.BRUSH.SIZE SKETCHW.CLOSEFN SK.CONFIRM.DESTRUCTION SKETCHW.OUTFN SKETCHW.REOPENFN
             MAKE.LOCAL.SKETCH MAP.SKETCHSPEC.INTO.VIEWER SKETCHW.REPAINTFN SKETCHW.REPAINTFN1 SK.DRAWFIGURE.IF
             SKETCHW.SCROLLFN SKETCHW.RESHAPEFN SK.UPDATE.EVENT.SELECTION LIGHTGRAYWINDOW SK.ADD.SPACES
             SK.SKETCH.MENU SK.CHECK.IMAGEOBJ.WHENDELETEDFN SK.APPLY.IMAGEOBJ.WHENDELETEDFN SK.RETURN.TTY
             SK.TAKE.TTY)
        (COMS                                                  ; fns for dealing with the sketch menu
              (FNS SKETCH.COMMANDMENU SKETCH.COMMANDMENU.ITEMS CREATE.SKETCHW.COMMANDMENU SKETCHW.SELECTIONFN
                   SKETCH.MONITORLOCK SK.EVAL.AS.PROCESS SK.EVAL.WITH.LOCK)
              (FNS SK.FIX.MENU SK.SET.UP.MENUS SK.INSURE.HAS.MENU SK.CREATE.STANDARD.MENU SK.ADD.ITEM.TO.MENU
                   SK.GET.VIEWER.POPUP.MENU SK.CLEAR.POPUP.MENU))
        (COMS                                                  ; fns for dealing with sketch structures
              (FNS SKETCH.CREATE GETSKETCHPROP PUTSKETCHPROP CREATE.DEFAULT.SKETCH.CONTEXT)
              (PROP ARGNAMES SKETCH.CREATE))
        (COMS                                                  ; fns for implementing copy and delete functions under keyboard
                                                               ; control.
              (FNS SK.COPY.BUTTONEVENTFN SK.BUTTONEVENT.MARK SK.BUILD.IMAGEOBJ SK.BUTTONEVENT.OVERP
                   SK.BUTTONEVENT.SAME.KEYS)
              (MACROS .DELETEKEYDOWNP. .MOVEKEYDOWNP.))
        (COMS                                                  ; fns for implementing the CHANGE command.
              (FNS SK.SEL.AND.CHANGE SK.CHECK.WHENCHANGEDFN SK.CHECK.PRECHANGEFN SK.CHANGE.ELT SK.CHANGE.THING
                   SKETCH.CHANGE.ELEMENTS SK.APPLY.SINGLE.CHANGEFN SK.DO.CHANGESPECS SK.VIEWER.FROM.SKETCH.ARG
                   SK.DO.CHANGESPEC1 SK.CHANGEFN SK.READCHANGEFN SK.DEFAULT.CHANGEFN CHANGEABLEFIELDITEMS
                   SK.APPLY.CHANGE.COMMAND SK.DO.AND.RECORD.CHANGES SK.APPLY.CHANGE.COMMAND1
                   SK.ELEMENTS.CHANGEFN READ.POINT.TO.ADD GLOBAL.KNOT.FROM.LOCAL SK.ADD.KNOT.TO.ELEMENT
                   SK.GROUP.CHANGEFN SK.GROUP.CHANGEFN1)
              (DECLARE%: DONTCOPY (RECORDS SKHISTORYCHANGESPEC)))
        (COMS                                                  ; fns for adding elements
              (FNS ADD.ELEMENT.TO.SKETCH ADD.SKETCH.VIEWER REMOVE.SKETCH.VIEWER ALL.SKETCH.VIEWERS
                   SKETCH.ALL.VIEWERS VIEWER.BUCKET ELT.INSIDE.REGION? ELT.INSIDE.SKWP SCALE.FROM.SKW
                   SK.ADDELT.TO.WINDOW SK.CALC.REGION.VIEWED SK.DRAWFIGURE SK.DRAWFIGURE1 SK.LOCAL.FROM.GLOBAL
                   SKETCH.REGION.VIEWED SKETCH.VIEW.FROM.NAME SK.UPDATE.REGION.VIEWED SKETCH.ADD.AND.DISPLAY
                   SKETCH.ADD.AND.DISPLAY1 SK.ADD.ITEM SKETCHW.ADD.INSTANCE))
                                                               ; fns for deleting things
        (FNS SK.SEL.AND.DELETE SK.ERASE.AND.DELETE.ITEM REMOVE.ELEMENT.FROM.SKETCH SK.DELETE.ELEMENT
             SK.DELETE.ELEMENT2 SK.DELETE.KNOT SK.SEL.AND.DELETE.KNOT SK.DELETE.ELEMENT.KNOT
             SK.CHECK.WHENDELETEDFN SK.CHECK.PREEDITFN SK.CHECK.END.INITIAL.EDIT SK.CHECK.WHENPOINTDELETEDFN
             SK.ERASE.ELT SK.DELETE.ELT SK.DELETE.ITEM DELFROMTCONC)
                                                               ; fns for copying stuff
        (FNS SK.COPY.ELT SK.SEL.AND.COPY SK.COPY.ELEMENTS SK.ADD.COPY.OF.ELEMENTS SK.GLOBAL.FROM.LOCAL.ELEMENTS
```

```
      SK.COPY.ITEM SK.INSERT.SKETCH)
(COMS                                                    ; fns for moving things.
      (FNS SK.MOVE.ELT SK.MOVE.ELT.OR.PT SK.APPLY.DEFAULT.MOVE SK.SEL.AND.MOVE SK.MOVE.ELEMENTS
           SKETCH.MOVE.ELEMENTS SKETCH.COPY.ELEMENTS \SKETCH.COPY.ELEMENT SK.TRANSLATE.ELEMENT
           SK.COPY.GLOBAL.ELEMENT SK.MAKE.ELEMENT.MOVE.ARG SK.MAKE.ELEMENTS.MOVE.ARG
           SK.MAKE.POINTS.AND.ELEMENTS.MOVE.ARG SK.SHOW.FIG.FROM.INFO SK.MOVE.THING
           UPDATE.ELEMENT.IN.SKETCH SK.UPDATE.ELEMENT SK.UPDATE.ELEMENTS SK.UPDATE.ELEMENT1
           SK.MOVE.ELEMENT.POINT)
                                                         ; fns for moving points or a collection of pts.
      (FNS SK.MOVE.POINTS SK.SEL.AND.MOVE.POINTS SK.DO.MOVE.ELEMENT.POINTS SK.MOVE.ITEM.POINTS
           SK.TRANSLATEPTSFN SK.TRANSLATE.POINTS SK.SELECT.MULTIPLE.POINTS SK.CONTROL.POINTS.IN.REGION
           SK.ADD.PT.SELECTION SK.REMOVE.PT.SELECTION SK.ADD.POINT SK.ELTS.CONTAINING.PTS
           SK.HOTSPOTS.NOT.ON.LIST)
      (MACROS .SHIFTKEYDOWNP.)
      (FNS SK.SET.MOVE.MODE SK.SET.MOVE.MODE.POINTS SK.SET.MOVE.MODE.ELEMENTS SK.SET.MOVE.MODE.COMBINED
           READMOVEMODE)
      (FNS SK.ALIGN.POINTS SK.SEL.AND.ALIGN.POINTS SK.ALIGN.POINTS.LEFT SK.ALIGN.POINTS.RIGHT
           SK.ALIGN.POINTS.TOP SK.ALIGN.POINTS.BOTTOM SK.EVEN.SPACE.POINTS.IN.X
           SK.EVEN.SPACE.POINTS.IN.Y SK.DO.ALIGN.POINTS SK.NTH.CONTROL.POINT
           SK.GET.SELECTED.ELEMENT.STRUCTURE SK.CORRESPONDING.CONTROL.PT SK.CONTROL.POINT.NUMBER
           SK.DO.ALIGN.SETVALUE))
(COMS                                                    ; stuff for supporting the GROUP sketch element.
      (FNS SKETCH.CREATE.GROUP SK.CREATE.GROUP1 SK.UPDATE.GROUP.AFTER.CHANGE SK.GROUP.ELTS
           SK.SEL.AND.GROUP SK.GROUP.ELEMENTS SK.UNGROUP.ELT SK.SEL.AND.UNGROUP SK.UNGROUP.ELEMENT
           SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS SK.LOCAL.REGION.OF.LOCAL.ELEMENTS
           SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS SK.UNIONREGIONS SKETCH.REGION.OF.SKETCH SK.FLASHREGION)
      (FNS INIT.GROUP.ELEMENT GROUP.DRAWFN GROUP.EXPANDFN GROUP.INSIDEFN GROUP.REGIONFN
           GROUP.GLOBALREGIONFN GROUP.TRANSLATEFN GROUP.TRANSFORMFN GROUP.READCHANGEFN)
      (FNS REGION.CENTER REMOVE.LAST)
                                                         ; moving the control point of a group
      (FNS SK.MOVE.GROUP.CONTROL.PT SK.SEL.AND.MOVE.CONTROL.PT SK.MOVE.GROUP.ELEMENT.CONTROL.POINT
           SK.READ.NEW.GROUP.CONTROL.PT)
      (RECORDS GROUP LOCALGROUP)
      (COMS                                              ; history and undo stuff for groups
            (FNS SK.DO.GROUP SK.CHECK.WHENGROUPEDFN SK.DO.UNGROUP SK.CHECK.WHENUNGROUPEDFN SK.GROUP.UNDO
                 SK.UNGROUP.UNDO)
            (IFPROP EVENTFNS GROUP UNGROUP)))
(COMS                                                    ; stuff for supporting the freezing of elements
      (FNS SK.FREEZE.ELTS SK.SEL.AND.FREEZE SK.FREEZE.ELEMENTS SK.UNFREEZE.ELT SK.SEL.AND.UNFREEZE
           SK.UNFREEZE.ELEMENTS SK.FREEZE.UNDO SK.UNFREEZE.UNDO SK.DO.FREEZE SK.DO.UNFREEZE)
      (IFPROP EVENTFNS FREEZE UNFREEZE))
(COMS                                                    ; programmer interface entries
      (FNS SKETCH.ELEMENTS.OF.SKETCH SKETCH.LIST.OF.ELEMENTS SKETCH.ADD.ELEMENT SKETCH.DELETE.ELEMENT
           DELFROMGROUPELT SKETCH.ELEMENT.TYPE SKETCH.ELEMENT.CHANGED SK.ELEMENT.CHANGED1
           SK.UPDATE.GLOBAL.IMAGE.OBJECT.ELEMENT))
                                                         ; utility routines for sketch windows.
(FNS INSURE.SKETCH LOCALSPECS.FROM.VIEWER SK.LOCAL.ELT.FROM.GLOBALPART SKETCH.FROM.VIEWER INSPECT.SKETCH
     ELT.INSIDE.SKETCHWP SK.INSIDE.REGION)
(FNS MAPSKETCHSPECS MAPCOLLECTSKETCHSPECS MAPSKETCHSPECSUNTIL MAPGLOBALSKETCHSPECS
     MAPGLOBALSKETCHELEMENTS)
(COMS                                                    ; multiple selection and copy select functions
      (FNS SK.ADD.SELECTION SK.COPY.INSERTFN SCREENELEMENTP SK.ITEM.REGION SK.ELEMENT.GLOBAL.REGION
           SK.LOCAL.ITEMS.IN.REGION SK.REGIONFN SK.GLOBAL.REGIONFN SK.REMOVE.SELECTION
           SK.SELECT.MULTIPLE.ITEMS SKETCH.GET.ELEMENTS SK.PUT.MARKS.UP SK.TAKE.MARKS.DOWN
           SK.TRANSLATE.GLOBALPART SK.TRANSLATE.ITEM SK.TRANSLATEFN TRANSLATE.SKETCH)
      (CONSTANTS (SK.NO.MOVE.DISTANCE 4))
      (DECLARE%: DONTCOPY (RECORDS SKFIGUREIMAGE)))
(COMS                                                    ; stuff for changing the input scale
      (FNS SK.INPUT.SCALE SK.UPDATE.SKETCHCONTEXT SK.SET.INPUT.SCALE SK.SET.INPUT.SCALE.CURRENT
           SK.SET.INPUT.SCALE.VALUE))
(COMS                                                    ; stuff for setting feedback amount
      (FNS SK.SET.FEEDBACK.MODE SK.SET.FEEDBACK.POINT SK.SET.FEEDBACK.VERBOSE SK.SET.FEEDBACK.ALWAYS)
      (INITVARS (SKETCH.VERBOSE.FEEDBACK T))
      (GLOBALVARS SKETCH.VERBOSE.FEEDBACK))
(COMS                                                    ; sketch icon support
      (FNS SKETCH.TITLE SK.SHRINK.ICONCREATE)
      (UGLYVARS SKETCH.TITLED.ICON.TEMPLATE))
(COMS                                                    ; fns for reading in various values
      (FNS READBRUSHSHAPE READ.FUNCTION READBRUSHSIZE READANGLE READARCDIRECTION)
      (FNS SK.CHANGE.DASHING READ.AND.SAVE.NEW.DASHING READ.NEW.DASHING READ.DASHING.CHANGE
           SK.CACHE.DASHING SK.DASHING.LABEL)
      (FNS READ.FILLING.CHANGE SK.CACHE.FILLING READ.AND.SAVE.NEW.FILLING SK.FILLING.LABEL)
      (INITVARS (SK.DASHING.PATTERNS)
             (SK.FILLING.PATTERNS))
      (GLOBALVARS SK.DASHING.PATTERNS SK.FILLING.PATTERNS)
      (P (SK.CACHE.DASHING '(2 4))
         (SK.CACHE.DASHING '(6 3 1 3))
         (SK.CACHE.FILLING BLACKSHADE)
         (SK.CACHE.FILLING GRAYSHADE)
         (SK.CACHE.FILLING HIGHLIGHTSHADE)))
(COMS                                                    ; stuff for reading input positions
      (FNS SK.GETGLOBALPOSITION SKETCH.TRACK.ELEMENTS SK.PICKOUT.WHOLE.MOVE.ELEMENTS
           MAP.SKETCH.ELEMENTS.INTO.VIEWER MAP.GLOBAL.POSITION.INTO.VIEWER SKETCH.TO.VIEWER.POSITION
           SKETCH.TRACK.IMAGE SK.TRACK.IMAGE1 MAP.VIEWER.XY.INTO.GLOBAL SK.SET.POSITION
           MAP.VIEWER.PT.INTO.GLOBAL VIEWER.TO.SKETCH.POSITION SK.INSURE.SCALE SKETCH.TO.VIEWER.REGION
           VIEWER.TO.SKETCH.REGION SK.READ.POINT.WITH.FEEDBACK SKETCH.GET.POSITION \CLOBBER.POSITION
           NEAREST.HOT.SPOT GETWREGION GET.BITMAP.POSITION SK.TRACK.BITMAP1)
```

```
            (RECORDS INPUTPT)
            (COMS                                                    ; stuff to allow reading positions from a number pad
                  (INITVARS (SKETCH.USE.POSITION.PAD NIL))
                  (GLOBALVARS SKETCH.USE.POSITION.PAD)
                  (FNS SK.BRING.UP.POSITION.PAD SK.PAD.READER.POSITION SK.POSITION.READER.REPAINTFN
                       SK.POSITION.PAD.FROM.VIEWER SK.INIT.POSITION.NUMBER.PAD.MENU
                       SK.READ.POSITION.PAD.HANDLER DISPLAY.POSITION.READER.TOTAL POSITION.PAD.READER.HANDLER
                       POSITIONPAD.HELDFN \POSITION.PAD.ADD.DIGIT.MENU \POSITION.READER.NUMBERPAD)))
      (INITVARS (ALL.SKETCHES)
             (INITIAL.SCALE 1.0)
             (DEFAULT.VISIBLE.SCALE.FACTOR 10.0)
             (MINIMUM.VISIBLE.SCALE.FACTOR 4.0))
      (VARS (SKETCH.ELEMENT.TYPES)
            (SKETCH.ELEMENT.TYPE.NAMES))
      (GLOBALVARS ALL.SKETCHES INITIAL.SCALE DEFAULT.VISIBLE.SCALE.FACTOR MINIMUM.VISIBLE.SCALE.FACTOR
             SKETCH.ELEMENT.TYPES SKETCH.ELEMENT.TYPE.NAMES SK.SELECTEDMARK SK.LOCATEMARK COPYSELECTIONMARK
             MOVESELECTIONMARK DELETESELECTIONMARK)
      (UGLYVARS SK.SELECTEDMARK SK.LOCATEMARK COPYSELECTIONMARK MOVESELECTIONMARK DELETESELECTIONMARK
             OTHERCONTROLPOINTMARK)
                                                                     ; accessing functions for the methods of a sketch type.
      (FNS SK.DRAWFN SK.TRANSFORMFN SK.EXPANDFN SK.INPUT SK.INSIDEFN SK.UPDATEFN)
      (INITRECORDS SKETCHTYPE)
      (DECLARE%: DONTCOPY (RECORDS SCREENELT GLOBALPART COMMONGLOBALPART INDIVIDUALGLOBALPART LOCALPART SKETCH
                                 SKETCHTYPE SKETCHCONTEXT))
      [ADDVARS (BackgroundMenuCommands (Sketch '(SKETCHW.CREATE NIL NIL (GETREGION)
                                                        NIL NIL T T)
                                               "Opens a sketch window for use."
                                               (SUBITEMS ("Page sized sketch" '(EDITSLIDE NIL)
                                                                "Opens a sketch window the size of a page.")
                                                      ("Landscaped sketch" '(EDITSLIDE NIL T)
                                                             "Opens a sketch window the size of a landscaped
                                                             page.")
                                                      ("Sketch, from a file" '(SKETCH.FROM.A.FILE)
                                                             "Reads a file name and opens a sketch window onto
                                                             the sketch it contains."]
      (VARS (BackgroundMenu))
      (FILES SKETCH-OPS SKETCH-ELEMENTS SKETCH-EDIT SKETCH-OBJ SKETCH-BMELT)
      (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                              SKETCH-OPS SKETCH-ELEMENTS SKETCH-OBJ
                                                              SKETCH-EDIT))
      (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (FROM LOADUPS)
                                              EXPORTS.ALL))
                                                                     ; recompute the sketch element types because loading SKETCH
                                                                     ; clobbers the previous ones.
      (P (INIT.BITMAP.ELEMENT)
         (INIT.SKETCH.ELEMENTS)
         (INIT.GROUP.ELEMENT))
      (COMS                                                          ; version checking stuff
            (CONSTANTS (SKETCH.VERSION 3))
            (FNS SK.CHECK.SKETCH.VERSION SK.INSURE.RECORD.LENGTH SK.INSURE.HAS.LENGTH SK.RECORD.LENGTH
                 SK.SET.RECORD.LENGTHS)
            (MACROS SK.SET.RECORD.LENGTHS.MACRO)
            (GLOBALVARS SKETCH.RECORD.LENGTHS)
            (P (SK.SET.RECORD.LENGTHS)))
      [COMS                                                          ; to correct for a bug in the file package that marks
                                                                     ; LOADCOMPed file as changed
            (P (UNMARKASCHANGED 'SKETCH 'FILE)
               (UNMARKASCHANGED 'SKETCH-ELEMENTS 'FILE)
               (UNMARKASCHANGED 'SKETCH-OPS 'FILE)
               (UNMARKASCHANGED 'SKETCH-EDIT 'FILE)
               (UNMARKASCHANGED 'SKETCH-OBJ 'FILE]
      (COMS                                                          ; add sketch as option to file browser edit command
            (FNS SK.ADD.EDIT.COMMAND.TO.FILE.BROWSER)
            (P (SK.ADD.EDIT.COMMAND.TO.FILE.BROWSER)))
      (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                           (NLAML)
                                                                           (LAMA SK.UNIONREGIONS SKETCH.CREATE
                                                                                 ])

(DECLARE%: FIRST DOCOPY DONTEVAL@LOAD

[PROG ((NOTECARDSFLG (GETPROP 'NOTECARDS 'FILEDATES))
       (SKETCHFLG (AND (BOUNDP 'ALL.SKETCHES)
                       ALL.SKETCHES))
       TEDITFLG)
 ;; current knows about SKETCH TEDIT and NOTECARDS.  Everyone else loses.
      [MAP.PROCESSES (FUNCTION (LAMBDA (PROC PROCNAME PROCFORM)
                                 (AND (EQ (CAR PROCFORM)
                                          '\TEDIT1)
                                      (SETQ TEDITFLG T]
      (COND
         ((AND (BOUNDP 'ALL.SKETCHES)
               (OR SKETCHFLG NOTECARDSFLG TEDITFLG))
          (ERROR (CONCAT "Please close" (COND
                                          (SKETCHFLG " all open Sketch windows,")
                                          (T ""))
```

```
                        (COND
                           (NOTECARDSFLG (CONCAT (COND
                                                    (SKETCHFLG " and")
                                                    (T ""))
                                                 " any open notefiles,"))
                           (T ""))
                        (COND
                           (TEDITFLG (CONCAT (COND
                                                ((OR SKETCHFLG NOTECARDSFLG)
                                                 " and")
                                                (T ""))
                                             " any TEDIT windows that have sketches in them,"))
                           (T ""))
                        " then type 'RETURN'.
                        To abort loading the new version of Sketch, type '^'."]
)

(DEFINEQ

(SKETCH
  [LAMBDA (SKETCH WINDOW)                                          (* rrb "17-Sep-86 10:21")
                                                                   (* opens a sketch window onto the sketch SKETCH)

    (COND
       [(AND SKETCH (LITATOM SKETCH))                              (* assume its a filename Get the region and scale from the file.)
        (PROG ((SKIMAGEOBJ (SK.GET.IMAGEOBJ.FROM.FILE SKETCH))
               SCREENREG READSKETCH)
              (SETQ SCREENREG (SK.SCALE.REGION (fetch (SKETCHIMAGEOBJ SKIO.REGION) of SKIMAGEOBJ)
                                               (fetch (SKETCHIMAGEOBJ SKIO.SCALE) of SKIMAGEOBJ)))
              (SETQ READSKETCH (fetch (SKETCHIMAGEOBJ SKIO.SKETCH) of SKIMAGEOBJ))
              (RETURN (SKETCHW.CREATE READSKETCH (fetch (SKETCHIMAGEOBJ SKIO.REGION) of SKIMAGEOBJ)
                             (OR WINDOW (GETBOXREGION (MIN (WIDTHIFWINDOW (fetch (REGION WIDTH) of SCREENREG))
                                                           (DIFFERENCE (BITMAPWIDTH (SCREENBITMAP))
                                                                       16))
                                                      (MIN (HEIGHTIFWINDOW (fetch (REGION HEIGHT) of SCREENREG)
                                                                           T)
                                                           (DIFFERENCE SCREENHEIGHT 16))
                                                      NIL NIL NIL "Position sketch window."))
                             (fetch (SKETCH SKETCHNAME) of READSKETCH)
                             (fetch (SKETCHIMAGEOBJ SKIO.SCALE) of SKIMAGEOBJ)
                             T
                             (fetch (SKETCHIMAGEOBJ SKIO.GRID) of SKIMAGEOBJ]
       (T (SKETCHW.CREATE SKETCH NIL (OR WINDOW (GETREGION))
                  NIL NIL T T])

(SKETCH.FROM.A.FILE
  [LAMBDA NIL                                                      (* rrb "24-Jun-86 11:40")
                                                                   (* reads a file name from the user and calls sketch on it.)

    (PROG ((NAME (PopUpWindowAndGetAtom "Sketch file name: ")))
          (RETURN (AND NAME (SKETCH NAME])

(SKETCHW.CREATE
  [LAMBDA (SKETCH SKETCHREGION SCREENREGION TITLE INITIALSCALE BRINGUPMENU INITIALGRID)
                                                                   ; Edited 25-Apr-88 15:18 by drc:

;;; creates a sketch window and returns it.

    (PROG (W SCALE SKPROC SKETCHSTRUCTURE)
          [SETQ SKETCHSTRUCTURE (SK.CHECK.SKETCH.VERSION (COND
                                                            ((NULL SKETCH)
                                                             (SKETCH.CREATE NIL))
                                                            ((LITATOM SKETCH)
                                                                ; treat it like a file name
                                                             (SKETCH.GET SKETCH))
                                                            ((type? SKETCH SKETCH)
                                                             SKETCH)
                                                            ((type? IMAGEOBJ SKETCH)
                                                                ; pull things out of the image object.
                                                             (SETQ SKPROC (IMAGEOBJPROP SKETCH 'OBJECTDATUM))
                                                             (OR (REGIONP SKETCHREGION)
                                                                 (SETQ SKETCHREGION (fetch (SKETCHIMAGEOBJ
                                                                                              SKIO.REGION)
                                                                                       of SKPROC)))
                                                             (OR (NUMBERP INITIALSCALE)
                                                                 (SETQ INITIALSCALE (fetch (SKETCHIMAGEOBJ
                                                                                              SKIO.SCALE)
                                                                                       of SKPROC)))
                                                             (OR (NUMBERP INITIALGRID)
                                                                 (SETQ INITIALGRID (fetch (SKETCHIMAGEOBJ
                                                                                             SKIO.GRID)
                                                                                      of SKPROC)))
                                                             (fetch (SKETCHIMAGEOBJ SKIO.SKETCH) of SKPROC))
                                                            ((AND (LITATOM (CAR SKETCH))
                                                                  (for ELT in (CDR SKETCH)
                                                                     always (GLOBALELEMENTP ELT)))
                                                                ; old form, probably written out by notecards, update to new
                                                                ; form.
```

```
                                                                (PROG (X)
                                                                      (SETQ X (SKIO.UPDATE.FROM.OLD.FORM SKETCH))
                                                                   ; smash sketch so this won't have to happen every time.
                                                                      (RPLACA SKETCH (CAR X))
                                                                      (RPLACD SKETCH (CDR X))
                                                                      (RETURN X)))
                                                               (T (\ILLEGAL.ARG SKETCH]
      [SETQ W (COND
                 ((WINDOWP SCREENREGION)
                  (AND TITLE (WINDOWPROP SCREENREGION 'TITLE TITLE))
                  SCREENREGION)
                 (T (CREATEW (COND
                                ((REGIONP SCREENREGION))
                                (T (CREATEREGION LASTMOUSEX LASTMOUSEY 20 20)))
                            (OR TITLE (SK.WINDOW.TITLE SKETCHSTRUCTURE))
                            NIL T]
      (SK.SET.UP.MENUS W (NOT (OPENWP SCREENREGION))
             BRINGUPMENU)
      (COND
         ((OR (REGIONP SCREENREGION)
              (WINDOWP SCREENREGION))                          ; user gave a region, don't interact
          NIL)
         (T                                                    ; let prompting for reshape show room for both menu and
                                                               ; window.
            (SHAPEW W)))
```

;; set the right margin so that text will never run into it. This can be removed when character positions are kept in points so \DSPPRINTCHAR
;; doesn't have to look at the right margin.

```
      (DSPRIGHTMARGIN 64000 W)
      (WINDOWPROP W 'SKETCH SKETCHSTRUCTURE)
      [WINDOWPROP W 'SCALE (SETQ SCALE (COND
                                          ((NUMBERP INITIALSCALE))
                                          [(REGIONP SKETCHREGION)
                                                               ; determine the scale and offsets so that the given region of the
                                                               ; sketch fits into the given window.
                                           (FQUOTIENT (fetch (REGION HEIGHT) of SKETCHREGION)
                                                  (fetch (REGION HEIGHT) of (DSPCLIPPINGREGION NIL W]
                                          ((NULL SKETCHREGION)
                                           INITIAL.SCALE)
                                          (T (\ILLEGAL.ARG SKETCHREGION]
                                                               ; check to make sure a context exists on the sketch because
                                                               ; before July 1985 it didn't exist.
      [WINDOWPROP W 'SKETCHCONTEXT (OR (GETSKETCHPROP SKETCHSTRUCTURE 'SKETCHCONTEXT)
                                       (PUTSKETCHPROP SKETCHSTRUCTURE 'SKETCHCONTEXT (
                                                                                  CREATE.DEFAULT.SKETCH.CONTEXT
                                                                                     ]
      (COND
         ((REGIONP SKETCHREGION)                               ; if given a region, translate to it.
          (WXOFFSET (IMINUS (FIX (QUOTIENT (fetch (REGION LEFT) of SKETCHREGION)
                                           SCALE)))
                    W)
          (WYOFFSET (IMINUS (FIX (QUOTIENT (fetch (REGION BOTTOM) of SKETCHREGION)
                                           SCALE)))
                    W)))
      (SK.UPDATE.REGION.VIEWED W)                              ; calculate the sketch region being viewed before mapping the
                                                               ; sketch into it.
      (MAP.SKETCHSPEC.INTO.VIEWER SKETCHSTRUCTURE W)
      (SK.CREATE.HOTSPOT.CACHE W)
      [WINDOWPROP W 'GRIDFACTOR (COND
                                   ((NUMBERP INITIALGRID)
                                    (LEASTPOWEROF2GT INITIALGRID))
                                   (T (SK.DEFAULT.GRIDFACTOR W]
      (WINDOWPROP W 'USEGRID (COND
                                (INITIALGRID T)))
      (WINDOWPROP W 'BUTTONEVENTFN (FUNCTION WB.BUTTON.HANDLER))
      (WINDOWPROP W 'COPYBUTTONEVENTFN (FUNCTION SK.COPY.BUTTONEVENTFN))
      (WINDOWPROP W 'COPYINSERTFN (FUNCTION SK.COPY.INSERTFN))
      (WINDOWPROP W 'RIGHTBUTTONFN (FUNCTION WB.BUTTON.HANDLER))
      (WINDOWPROP W 'CURSOROUTFN (FUNCTION SKETCHW.OUTFN))
      (WINDOWPROP W 'REPAINTFN (FUNCTION SKETCHW.REPAINTFN))
      (WINDOWADDPROP W 'RESHAPEFN (FUNCTION SKETCHW.RESHAPEFN))
      (WINDOWADDPROP W 'SHRINKFN (FUNCTION SK.RETURN.TTY))
      (WINDOWPROP W 'ICONFN (FUNCTION SK.SHRINK.ICONCREATE))
      (WINDOWADDPROP W 'EXPANDFN (FUNCTION SK.TAKE.TTY))
      (WINDOWPROP W 'SCROLLFN (FUNCTION SKETCHW.SCROLLFN))
      (WINDOWPROP W 'HARDCOPYFN (FUNCTION SKETCHW.HARDCOPYFN))
                                                               ; I'm not sure why this ever gets called but it did once so to be
                                                               ; sure, turn it off.
      (WINDOWPROP W 'PAGEFULLFN (FUNCTION NILL))
      [WINDOWPROP W 'PROCESS (SETQ SKPROC (ADD.PROCESS (LIST (FUNCTION WB.EDITOR)
                                                             (KWOTE W))
                                                       'RESTARTABLE T 'TTYENTRYFN 'SK.TTYENTRYFN 'TTYEXITFN
                                                       'SK.TTYEXITFN]
      (WINDOWPROP W 'SCROLLEXTENTUSE T)
      (WINDOWADDPROP W 'CLOSEFN (FUNCTION SKETCHW.CLOSEFN)
            T)
      (OPENW W)
```

```
        (ADD.SKETCH.VIEWER SKETCHSTRUCTURE W)
        (SKETCHW.REPAINTFN W)
        (RETURN W])
```

(SKETCH.RESET

```
  [LAMBDA (SKETCH)                                          (* rrb "11-Dec-85 11:24")
                                                            (* resets a sketch structure and all of the viewers onto it.)
    (PROG ((SKSTRUC (INSURE.SKETCH SKETCH)))                (* delete all sketch elements)
          (replace (SKETCH SKETCHTCELL) of SKSTRUC with (CONS))
          (for VIEWER in (ALL.SKETCH.VIEWERS SKSTRUC) do (SKED.CLEAR.SELECTION VIEWER)
                                                         (DSPRESET VIEWER)
                                                         (WINDOWPROP VIEWER 'SCALE INITIAL.SCALE)
                                                         (SK.UPDATE.REGION.VIEWED VIEWER)
                                                         (MAP.SKETCHSPEC.INTO.VIEWER SKSTRUC VIEWER)
                                                         (SK.CREATE.HOTSPOT.CACHE VIEWER)
                                                         (WINDOWPROP VIEWER 'GRIDFACTOR (SK.DEFAULT.GRIDFACTOR
                                                                                         VIEWER))
                                                         (WINDOWPROP VIEWER 'USEGRID NIL)
                                                         (WINDOWPROP VIEWER 'SKETCHHISTORY NIL)
                                                         (WINDOWPROP VIEWER 'SKETCHCHANGED NIL])
```

(SKETCHW.FIG.CHANGED

```
  [LAMBDA (W)                                               (* rrb "29-Nov-84 17:59")

        (* W is a sketch window that is being reshaped. Mark this fact in case it came out of a document.)

    (OR (WINDOWPROP W 'SKETCHCHANGED)
        (WINDOWPROP W 'SKETCHCHANGED 'OLD])
```

(SK.WINDOW.TITLE

```
  [LAMBDA (SKETCH)                                          (* rrb " 7-May-85 14:00")
                                                            (* returns the window title of a window onto a sketch.)
    (COND
       ((fetch (SKETCH SKETCHNAME) of SKETCH)
        (CONCAT "Viewer onto " (fetch (SKETCH SKETCHNAME) of SKETCH)))
       (T "Viewer onto a sketch"])
```

(EDITSLIDE

```
  [LAMBDA (SKETCH LANDSCAPE)                                ; Edited 20-Feb-87 10:44 by rrb
                                                            (* creates a sketch in a window the size of a screen.)
    (SKETCHW.CREATE SKETCH NIL (COND
                                  (LANDSCAPE (GETBOXREGION 780 612))
                                  (T (GETBOXREGION 612 770)))
          NIL NIL T 16.0])
```

(EDITSKETCH

```
  [LAMBDA (SLIDENAME)                                       (* rrb "14-Nov-84 17:15")
                                                            (* edits a named sketch)
    (SKETCHW.CREATE (SETQ SLIDENAME (OR SLIDENAME (GENSYM "SLIDE")))
          NIL NIL NIL NIL T 16.0)
    SLIDENAME])
```

(SK.PUT.ON.FILE

```
  [LAMBDA (SKETCHW)                                         ; Edited  6-Apr-87 18:18 by rrb
                                                            (* saves a sketch on a Tedit file.)

        (* also changes the name of the sketch to be the same as the name of the file.)

    (PROG ((SKETCH (INSURE.SKETCH (SKETCH.FROM.VIEWER SKETCHW)))
           NOWNAME NEWNAME TEXTSTREAM)
          (SETQ NOWNAME (SKETCH.TITLE SKETCH))
          (OR [SETQ NEWNAME (MKATOM (PROMPT.GETINPUT SKETCHW "File to PUT to:  " (SK.OUTPUT.FILE.NAME NOWNAME]
              (RETURN NIL))
          (SETQ NEWNAME (SKETCH.PUT NEWNAME SKETCH SKETCHW))
          [COND
             ((AND NEWNAME (NEQ NOWNAME NEWNAME))           (* change the name of the sketch to be the same as the file
                                                            name.)
              (replace (SKETCH SKETCHNAME) of SKETCH with NEWNAME)
                                                            (* change the titles of the viewers onto this sketch.)
              (for SKW in (ALL.SKETCH.VIEWERS SKETCH) do (WINDOWPROP SKW 'TITLE (SK.WINDOW.TITLE SKETCH]
          (RETURN NEWNAME])
```

(SK.OUTPUT.FILE.NAME

```
  [LAMBDA (SKETCHFILENAME)                                  (* rrb " 5-May-86 10:45")
    (COND
       ((STRPOS " " SKETCHFILENAME)                         (* don't put up dummy names that contain spaces)
        NIL)
       (T (FILENAMELESSVERSION SKETCHFILENAME])
```

(SKETCH.PUT

```
[LAMBDA (FILENAME SKETCH VIEWER REGION SCALE GRID)          ; Edited  1-Feb-2022 09:17 by rmk
                                                            ; Edited 17-Nov-87 17:47 by rrb
```

(* puts the sketch SKETCH on the file named FILENAME. VIEWER if given provides promptwindows and PUTFNs.)

```
   (PROG (TEXTSTREAM FILESTREAM)
         [COND
            ((NOT (DEFINEDP (FUNCTION OPENTEXTSTREAM)))
             (COND
                ((MOUSECONFIRM "TEDIT must be loaded to save sketches." "Click LEFT to load TEDIT now, RIGHT to
                       abort.")
                 (FILESLOAD TEDIT))
                (T (STATUSPRINT VIEWER "Sketch not saved.")
                   (RETURN NIL]
         [SETQ TEXTSTREAM (OPENTEXTSTREAM NIL NIL NIL NIL (AND VIEWER (LIST 'PUTFN (WINDOWPROP VIEWER
                                                                                    'TEDIT.PUTFN)
                                                                          'PROMPTWINDOW
                                                                          (GETPROMPTWINDOW VIEWER]
                                                            (* make a text stream with nothing in it except the sketch.)
         (TEDIT.INSERT.OBJECT [SKETCH.IMAGEOBJ (INSURE.SKETCH SKETCH)
                                    (COND
                                       ((REGIONP REGION))
                                       (VIEWER (SKETCH.REGION.VIEWED VIEWER)))
                                    (COND
                                       ((NUMBERP SCALE))
                                       (VIEWER (VIEWER.SCALE VIEWER)))
                                    (COND
                                       ((NUMBERP GRID))
                                       (VIEWER (SK.GRIDFACTOR VIEWER]
               TEXTSTREAM 1)
```

(* set the margins so that if the user hardcopies it directly the margins come out)

```
         (TEDIT.PARALOOKS TEXTSTREAM '(LEFTMARGIN 0 RIGHTMARGIN 0 QUAD CENTER)
                1 1)
         (TEDIT.PAGEFORMAT TEXTSTREAM (TEDIT.SINGLE.PAGEFORMAT NIL NIL NIL NIL NIL 0 0 0 0))
                                                            (* save the stream so that it can be closed.)
         (SETQ FILESTREAM (TEDIT.PUT TEXTSTREAM FILENAME))  (* grab the full file name if it is available.)
         (AND (OPENP FILESTREAM)
              (SETQ FILENAME (CLOSEF FILESTREAM)))
         (SK.MARK.UNDIRTY SKETCH)
         (RETURN FILENAME])
```

(SK.GET.FROM.FILE

```
[LAMBDA (SKETCHW)                                           (* rrb " 1-Oct-86 18:24")
                                                            (* retrieves a sketch from a file clobbering any existing sketch.)
   (COND
      ((SK.CONFIRM.DESTRUCTION SKETCHW "Press LEFT to delete current elements before GET.")
```

(* put the delete on the history list so that it can be undone. This leaves the gotten file there as well but seems better than nothing.)

```
       (SK.DELETE.ELEMENT2 (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH SKETCHW))
              SKETCHW)
       (SK.INCLUDE.FILE SKETCHW))
      ((SK.CONFIRM.DESTRUCTION SKETCHW "Press LEFT to include file, RIGHT to abort the GET.")
       (SK.INCLUDE.FILE SKETCHW))
      (T (STATUSPRINT SKETCHW "GET aborted.  The INCLUDE subcommand to GET doesn't delete."])
```

(SK.INCLUDE.FILE

```
[LAMBDA (SKETCHW)                                           (* rrb " 2-May-86 11:29")
                                                            (* retrieves a sketch from a file and includes it into the existing
                                                            sketch.)
```

(* also changes the name of the sketch to be the same as the name of the file.)

```
   (PROG ((SKETCH (SKETCH.FROM.VIEWER SKETCHW))
          NOWNAME FILENAME READSKETCH DIRTYSTATUS)
         (SETQ NOWNAME (fetch (SKETCH SKETCHNAME) of SKETCH))
         (SETQ FILENAME (MKATOM (PROMPT.GETINPUT SKETCHW "File to GET: ")))
         (COND
            ((MEMB FILENAME '(NIL %]))
             (CLOSEPROMPTWINDOW SKETCHW)
             (RETURN)))
         (STATUSPRINT SKETCHW " ...")
         (SETQ FILENAME (OR (INFILEP FILENAME)
                            (ERROR FILENAME "file not found.")))
         (OR (SETQ READSKETCH (SKETCH.GET FILENAME SKETCHW))
             (RETURN))
         [COND
            ((NEQ NOWNAME FILENAME)                         (* change the name of the sketch to be the same as the file
                                                            name.)
             (replace (SKETCH SKETCHNAME) of SKETCH with FILENAME)
                                                            (* change the name of the sketch to be the same as the file
                                                            name.)
```

```
        (for SKW in (ALL.SKETCH.VIEWERS SKETCH) do (WINDOWPROP SKW 'TITLE (SK.WINDOW.TITLE SKETCH]
    (ADD.SKETCH.TO.VIEWER READSKETCH SKETCHW (COND
                                              ((fetch (SKETCH SKETCHELTS) of SKETCH)
```

(* if the sketch has elements, ask about the defaults from the read file and set the status to leave the sketch marked dirty after the read.)

```
                                               (SETQ DIRTYSTATUS T)
                                               'ASK)
                                              (T
```

(* if the sketch doesn't have any elements, use the defaults from the read file and set the status to leave the sketch marked clean after the read.)

```
                                                 NIL)))
    (COND
       ((NULL DIRTYSTATUS)                                  (* if sketch was empty before, mark it as not needing to be dumped.)

        (SK.MARK.UNDIRTY SKETCH)))
    (STATUSPRINT SKETCHW " done."])
```

(SK.GET.IMAGEOBJ.FROM.FILE

```
  [LAMBDA (FILENAME VIEWER)                                 ; Edited 12-Feb-88 14:13 by rrb
                                                            (* reads the sketch image object datum from a file.)
    (RESETFORM (CURSOR WAITINGCURSOR)
           (PROG ([TEXTSTREAM (OPENTEXTSTREAM FILENAME NIL NIL NIL (AND VIEWER (LIST 'PROMPTWINDOW (
                                                                                    GETPROMPTWINDOW
                                                                                      VIEWER]
                  (READFILE (INFILEP FILENAME))
                  IMAGEOBJ READSKETCH)
                 (SETQ IMAGEOBJ (BIN TEXTSTREAM))
                 (CLOSEF TEXTSTREAM)
                 (COND
                    ((NOT (IMAGEOBJP IMAGEOBJ))
                     (STATUSPRINT (OR VIEWER PROMPTWINDOW)
                            FILENAME " is not a sketch file.")
                     (RETURN NIL)))
                 (COND
                    ([NOT (type? SKETCH (SETQ READSKETCH (fetch (SKETCHIMAGEOBJ SKIO.SKETCH)
                                                          of (IMAGEOBJPROP IMAGEOBJ 'OBJECTDATUM]
                     (STATUSPRINT (OR VIEWER PROMPTWINDOW)
                            FILENAME " is not a sketch file.")
                     (RETURN))
                    (T                                      (* save the name of where the sketch came from.)
                       (replace (SKETCH SKETCHNAME) of READSKETCH with (OR READFILE FILENAME))
                       (AND VIEWER (SK.CHANGE.GRID (fetch (SKETCHIMAGEOBJ SKIO.GRID) of (IMAGEOBJPROP
                                                                                          IMAGEOBJ
                                                                                          'OBJECTDATUM))
                                                   VIEWER))
                       (RETURN (IMAGEOBJPROP IMAGEOBJ 'OBJECTDATUM])
```

(SKETCH.GET

```
  [LAMBDA (FILENAME VIEWER)                                 (* rrb "29-Jan-86 11:21")
                                                            (* reads a sketch from a file.)
    (fetch (SKETCHIMAGEOBJ SKIO.SKETCH) of (SK.GET.IMAGEOBJ.FROM.FILE FILENAME VIEWER])
```

(ADD.SKETCH.TO.VIEWER

```
  [LAMBDA (SKETCHTOADD VIEWER ABOUTDEFAULTS?)               (* rrb "20-Mar-86 15:55")
                                                            (* adds the element in SKETCHTOADD to the sketch TOSKETCH)
    (PROG ([ADDSKETCH (COND
                         ((LITATOM SKETCHTOADD)             (* if it's an atom, assume its a file name.)
                          (SKETCH.GET SKETCHTOADD VIEWER))
                         ((INSURE.SKETCH SKETCHTOADD]
           (TOSKETCH (INSURE.SKETCH VIEWER))
           DEFAULTS)                                        (* set the default from the new sketch if appropriate)
          [AND (MEMB ABOUTDEFAULTS? '(NIL ASK))
               [NOT (EQUAL (SETQ DEFAULTS (GETSKETCHPROP ADDSKETCH 'SKETCHCONTEXT))
                           (GETSKETCHPROP TOSKETCH 'SKETCHCONTEXT]
               (COND
                  ((OR (NULL ABOUTDEFAULTS?)
                       (MENU (create MENU
                                     ITEMS _ '((Yes T "Will use the defaults of the retrieved sketch.")
                                               (No NIL "Will not change the defaults."))
                                     CENTERFLG _ T
                                     TITLE _ "Use the defaults from the retrieved sketch?"
                                     MENUCOLUMNS _ 2)))
                   (PUTSKETCHPROP TOSKETCH 'SKETCHCONTEXT DEFAULTS)
                   (WINDOWPROP VIEWER 'SKETCHCONTEXT DEFAULTS]
          (SK.ADD.ELEMENTS.TO.SKETCH (fetch (SKETCH SKETCHELTS) of ADDSKETCH)
                VIEWER)                                     (* copy properties from the read sketch.)
          (for SKPROP in (fetch (SKETCH SKETCHPROPS) of ADDSKETCH) by (CDDR SKPROP)
             do (SELECTQ SKPROP
                     (SKETCHCONTEXT
```

```
                    NIL)
                (VIEWS [PUTSKETCHPROP TOSKETCH 'VIEWS (UNION (GETSKETCHPROP ADDSKETCH 'VIEWS)
                                                             (GETSKETCHPROP TOSKETCH 'VIEWS])
                (PUTSKETCHPROP TOSKETCH SKPROP (GETSKETCHPROP ADDSKETCH SKPROP])
```

(FILENAMELESSVERSION

```
  [LAMBDA (FILENAME)                                        (* rrb "29-Jan-86 15:57")
                                                            (* strips the version number off of FILENAME if it has one.)
    (PACKFILENAME (CONS 'VERSION (CONS NIL (UNPACKFILENAME FILENAME])
```

(SK.ADD.ELEMENTS.TO.SKETCH

```
  [LAMBDA (ELTS SKW)                                        (* rrb "10-Mar-86 16:50")
                                                            (* adds a list of elements to a sketch)
    (for ELT in ELTS do

        (* clear the priority so that they get a priority based on their position in the new sketch.)

            (SK.SET.ELEMENT.PRIORITY ELT NIL)
            (SK.ADD.ELEMENT ELT SKW])
```

(SKETCH.SET.A.DEFAULT

```
  [LAMBDA (SKW)                                             (* rrb "14-Jul-86 13:43")
                                                            (* allows the user to set a default)
    (\CURSOR.IN.MIDDLE.MENU (create MENU
                                ITEMS _ '[(Line SKETCH.SET.BRUSH.SIZE "Sets the characteristics of the
                                        default brush." (SUBITEMS (Size SKETCH.SET.BRUSH.SIZE
                                                                  "Sets the size of the default
                                                                  brush")
                                                          (Shape SKETCH.SET.BRUSH.SHAPE "Sets
                                                                 the shape of the default
                                                                 brush")
                                                          (Add% arrowhead SK.SET.LINE.ARROWHEAD
                                                                 "Sets the arrowhead
                                                                 characteristics of new lines.")
                                                          ("Mouse line specs"
                                                                 SK.SET.LINE.LENGTH.MODE
                                                                 "Sets whether the lines drawn
                                                                 with the middle mouse button
                                                                 connect to each other.")))
                                (Arrowhead SK.SET.ARROWHEAD.LENGTH "Sets the characteristics of the
                                       default arrowhead." (SUBITEMS (Size SK.SET.ARROWHEAD.LENGTH)
                                                                     (Angle SK.SET.ARROWHEAD.ANGLE)
                                                                     (Type SK.SET.ARROWHEAD.TYPE)))
                                (Text SK.SET.TEXT.SIZE "Sets the size of newly added text."
                                      (SUBITEMS ("Font size" SK.SET.TEXT.SIZE "Sets the size of
                                                        newly added text.")
                                             ("Font family" SK.SET.TEXT.FONT "Sets the font family
                                                    of newly added text.")
                                             ("Horizontal justification" SK.SET.TEXT.HORIZ.ALIGN
                                                    "Sets the horizontal justification mode of new
                                                    text.")
                                             ("Vertical justification" SK.SET.TEXT.VERT.ALIGN
                                                    "Sets the vertical justification of new text.")
                                             ("Bold and/or italic" SK.SET.TEXT.LOOKS "Sets the bold
                                                    and italic look of new text.")))
                                (Text% Box SK.SET.TEXTBOX.HORIZ.ALIGN "Sets the alignment of text
                                       within new text boxes." (SUBITEMS ("Horizontal
                                                                          justification"
                                                                         SK.SET.TEXTBOX.HORIZ.ALIGN
                                                                                "Sets the
                                                                                horizontal
                                                                                alignment of text
                                                                                within new text
                                                                                boxes.")
                                                                        ("Vertical justification"
                                                                         SK.SET.TEXTBOX.VERT.ALIGN
                                                                         "Sets the vertical alignment
                                                                         of text within new text
                                                                         boxes.")))
                                (Arc SK.SET.ARC.DIRECTION "Sets the direction arcs go around their
                                     circle." (SUBITEMS ("Clockwise" SK.SET.ARC.DIRECTION.CW
                                                                "Makes new arcs go around in the
                                                                clockwise direction")
                                                        ("Counterclockwise" SK.SET.ARC.DIRECTION.CCW
                                                               "Makes new arcs go around in the
                                                               counterclockwise direction")))
                                ("Input scale" SK.SET.INPUT.SCALE "Sets the scale for newly added
                                       lines and text." (SUBITEMS ("Read new input scale"
                                                                         SK.SET.INPUT.SCALE
                                                                         "Reads a new input
                                                                         scale.")
                                                                 ("Make input scale current"
                                                                        SK.SET.INPUT.SCALE.CURRENT
                                                                        "makes the input scale be the
```

```
                                                                scale of the current view."))
                                        )
                                    (Feedback SK.SET.FEEDBACK.MODE "Controls the amount of feedback
                                           when adding new curves, circles, etc."
                                           (SUBITEMS ("Points only" SK.SET.FEEDBACK.POINT "Only the
                                                          control points will be shown when entering
                                                          elements.")
                                                  ("Fast figures" SK.SET.FEEDBACK.VERBOSE "Wires,
                                                         circles and ellipses are shown while they are
                                                         being entered.")
                                                  ("All figures" SK.SET.FEEDBACK.ALWAYS "Most elements
                                                         are shown while they are being entered.
                                                         This will be slow for arcs and curves."]
                                CENTERFLG _ T
                                WHENSELECTEDFN _ (FUNCTION SK.POPUP.SELECTIONFN)
                                MENUFONT _ (SK.FONTNAMELIST (FONTCREATE BOLDFONT])
```

(SK.POPUP.SELECTIONFN

```
[LAMBDA (ITEM MENU)                                          (* rrb " 3-Sep-85 14:27")
```

(* * calls the function appropriate for the item selected from the command menu associated with a figure window.)
(* uses SKW freely from enclosing call to MENU.)

```
  (CLOSEPROMPTWINDOW SKW)
  (SK.APPLY.MENU.COMMAND (CADR ITEM)
         SKW])
```

(GETSKETCHWREGION

```
[LAMBDA (SKETCHWINDOW)                                       (* rrb "11-Jul-86 15:48")
  (UNSCALE.REGION (GETWREGION SKETCHWINDOW)
         (VIEWER.SCALE SKETCHWINDOW])
```

(SK.ADD.ELEMENT

```
[LAMBDA (GELT SKETCHW DONTCLEARCURSOR GROUPFLG DONTCALLWHENADDEDFN)
                                                             (* rrb "30-Aug-86 15:08")
```

(* adds a new element to a sketch window and handles propagation to all other figure windows)

```
  (COND
     (GELT (PROG ([GELTTOADD (COND
                                (DONTCALLWHENADDEDFN GELT)
                                (T (SK.CHECK.WHENADDEDFN SKETCHW GELT]
                  (SKETCH (SKETCH.FROM.VIEWER SKETCHW))
                  ADDEDELT)                                  (* take down the caret.)
                 (OR GELTTOADD (RETURN))
                 (OR DONTCLEARCURSOR (SKED.CLEAR.SELECTION SKETCHW))
                                                             (* add the element to the sketch.)
                 (ADD.ELEMENT.TO.SKETCH GELT SKETCH)         (* do the window that the interaction occurred in first.)
                 (SETQ ADDEDELT (SK.ADD.AND.DISPLAY1 GELT SKETCHW NIL GROUPFLG))
                                                             (* propagate to other windows.)
                 (for SKW in (ALL.SKETCH.VIEWERS SKETCH) when (AND (NEQ SKW SKETCHW)
                                                                   (ELT.INSIDE.SKETCHWP GELT SKW))
                    do (SKETCH.ADD.AND.DISPLAY1 GELT SKW GROUPFLG))
                 (RETURN ADDEDELT])
```

(SK.ADD.PRIORITY.ELEMENT.TO.SKETCH

```
[LAMBDA (SKETCH ELEMENT PRIORITY)                            (* rrb "10-Mar-86 18:48")
```

(* * adds an element to a sketch at its place according to PRIORITY.)

```
  (PROG ((SKELTSCELL (fetch (SKETCH SKETCHTCELL) of SKETCH)))
        (RETURN (COND
                   ([OR (NULL (CAR SKELTSCELL))
                        (NOT (LESSP PRIORITY (SK.ELEMENT.PRIORITY (CADR SKELTSCELL]
```

(* special cases of no elements or this element being greater than any others.
This means the other part of the COND doesn't have to worry about the TCONC format.)

```
                    (TCONC SKELTSCELL ELEMENT))
                   [(LESSP PRIORITY (SK.ELEMENT.PRIORITY (CAAR SKELTSCELL)))
```

(* special check for first element. This allows the others to be handled by replacing the tail of the element before.)

```
                    (RPLACA SKELTSCELL (CONS ELEMENT (CAR SKELTSCELL]
                   (T (for SKELTTAIL on (CAR SKELTSCELL) when (LESSP PRIORITY (SK.ELEMENT.PRIORITY
                                                                   (CADR SKELTTAIL)))
                         do (RPLACD SKELTTAIL (CONS ELEMENT (CDR SKELTTAIL)))
                            (RETURN ELEMENT])
```

(SK.ELTS.BY.PRIORITY

```
[LAMBDA (GELTA GELTB)                                        (* rrb "10-Mar-86 17:57")
```

(* * sort function for sketch global elements that sorts by priority.)

```
    (ILESSP (SK.ELEMENT.PRIORITY GELTA)
           (SK.ELEMENT.PRIORITY GELTB])
```

(SK.ORDER.ELEMENTS

```
  [LAMBDA (GSKETCHELEMENTS)                                    (* rrb "10-Mar-86 16:30")
```

(* * puts a list of sketch global elements in order by priority.)

```
    (SORT GSKETCHELEMENTS (FUNCTION SK.ELTS.BY.PRIORITY])
```

(SK.ADD.PRIORITY.LOCAL.ELEMENT.TO.SKETCH

```
  [LAMBDA (LOCALSKETCHELTS LOCALELEMENT)                       (* rrb "26-Mar-86 10:21")
```

(* * adds an element to a sketch at its place according to PRIORITY.)

```
    (PROG [(PRIORITY (SK.ELEMENT.PRIORITY (fetch (SCREENELT GLOBALPART) of LOCALELEMENT]
          (RETURN (COND
                      ([OR (NULL (CDAR LOCALSKETCHELTS))
                           (NOT (LESSP PRIORITY (SK.ELEMENT.PRIORITY (fetch (SCREENELT GLOBALPART)
                                                                        of (CADR LOCALSKETCHELTS]
```

(* special cases of no elements in which case the local elements has only a name or this element being greater than any others. This means the other part of the COND doesn't have to worry about the TCONC format.)

```
                       (TCONC LOCALSKETCHELTS LOCALELEMENT))
                      (T                                       (* the first element of LOCALSKETCHELTS is the name of the sketch.)
                         (for SKELTTAIL on (CAR LOCALSKETCHELTS) when [LESSP PRIORITY (SK.ELEMENT.PRIORITY
                                                                                      (fetch (SCREENELT
                                                                                                GLOBALPART)
                                                                                         of (CADR SKELTTAIL]
                            do (RPLACD SKELTTAIL (CONS LOCALELEMENT (CDR SKELTTAIL)))
                               (RETURN LOCALELEMENT])
```

(SK.ADD.ELEMENTS

```
  [LAMBDA (ELEMENTS SKW)                                       (* rrb "10-Mar-86 17:57")
```

(* adds a list of global elements to a viewer but doesn't make an entry on the history list.)

```
                                                               (* sorts the elements so that their relative priority remains the same.)
    (for ELT in (SK.ORDER.ELEMENTS ELEMENTS) do (SK.SET.ELEMENT.PRIORITY ELT NIL)
                                                (SK.ADD.ELEMENT ELT SKW])
```

(SK.CHECK.WHENADDEDFN

```
  [LAMBDA (VIEWER GELT)                                        (* rrb "19-Oct-85 17:36")
```

(* checks if the sketch has a when added fn and if so, calls it and interprets the result.
Returns a list of the elements that should be deleted.)

```
    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           ADDFN RESULT)
          (COND
             ([NULL (SETQ ADDFN (GETSKETCHPROP SKETCH 'WHENADDEDFN]
              (RETURN GELT)))
          (SETQ RESULT (APPLY* ADDFN VIEWER GELT))
          (COND
             ((EQ RESULT 'DON'T)
              (RETURN NIL))
             ((GLOBALELEMENTP RESULT)
              (RETURN RESULT))
             (T (RETURN GELT])
```

(SK.APPLY.MENU.COMMAND

```
  [LAMBDA (COMMAND SKETCHW)                                    (* rrb " 3-Jan-85 13:17")
```

(* calls the function appropriate for the item selected from the command menu associated with a figure window.)

(* This is a separate function so it can be called by both pop up and fixed menu operations.)

```
    (COND
       ((NULL COMMAND)
        NIL)
       ((type? SKETCHTYPE COMMAND)                             (* if the selected item is an element type, add an instance.)
        (SKETCHW.ADD.INSTANCE COMMAND SKETCHW))
       [(LISTP COMMAND)                                        (* EVAL it)
        (EVAL (APPEND COMMAND (CONS (KWOTE SKETCHW]
       (T (APPLY* COMMAND SKETCHW])
```

(SK.DELETE.ELEMENT1

```
  [LAMBDA (OLDGELT SKETCHW GROUPFLG)                           (* rrb "19-Oct-85 17:09")
```

(* deletes an element to a sketch window and handles propagation to all other figure windows)

(* GROUPFLG indicates that this is part of a group operation and hence display and image object deleted fns don't need to be called.)

```
(PROG ((SKETCH (SKETCH.FROM.VIEWER SKETCHW))
       LOCALELT)                                          (* delete the element to the sketch.)
      (OR (REMOVE.ELEMENT.FROM.SKETCH OLDGELT SKETCH)
          (RETURN NIL))                                   (* do the window that the interaction occurred in first.)
      (SK.ERASE.AND.DELETE.ITEM (SK.LOCAL.ELT.FROM.GLOBALPART OLDGELT SKETCHW)
            SKETCHW GROUPFLG)                             (* propagate to other windows.)
      (for SKW in (ALL.SKETCH.VIEWERS SKETCH) when (AND (NEQ SKW SKETCHW)
                                                        (SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART
                                                                        OLDGELT SKW)))
         do (SK.ERASE.AND.DELETE.ITEM LOCALELT SKW GROUPFLG))
      (OR GROUPFLG (SK.CHECK.IMAGEOBJ.WHENDELETEDFN OLDGELT SKETCHW))
      (RETURN OLDGELT])
```

(SK.MARK.DIRTY

```
[LAMBDA (SKETCH)                                          (* rrb " 1-Oct-86 18:15")
                                                          (* marks a sketch as having been changed.
                                                          Puts a flag on its viewers.)
                                                          (* checks first because this is faster than always putting.)
   (for SKW in (ALL.SKETCH.VIEWERS SKETCH) do (OR (EQ (WINDOWPROP SKW 'SKETCHCHANGED)
                                                      T)
                                                  (WINDOWPROP SKW 'SKETCHCHANGED T])
```

(SK.MARK.UNDIRTY

```
[LAMBDA (SKETCH)                                          (* rrb "29-Nov-84 18:03")
                                                          (* marks a sketch as having been changed.
                                                          Puts a flag on its viewers.)
   (for SKW in (ALL.SKETCH.VIEWERS SKETCH) do (WINDOWPROP SKW 'SKETCHCHANGED 'OLD])
```

(SK.MENU.AND.RETURN.FIELD

```
[LAMBDA (ELEMENTTYPE)                                     (* rrb "11-May-84 16:03")
                                                          (* returns a field list of the field to be changed.)
   (PROG ((ITEMS (CHANGEABLEFIELDITEMS ELEMENTTYPE)))
         (RETURN (COND
                    ((NULL ITEMS)
                     NIL)
                    [(NULL (CDR ITEMS))
                     (EVAL (CADR (CAR ITEMS]
                    (T (MENU (create MENU
                                     ITEMS _ ITEMS
                                     CENTERFLG _ T
                                     TITLE _ "Choose which property to change"])
```

(SKETCH.SET.BRUSH.SHAPE

```
[LAMBDA (W)                                               (* rrb "11-Dec-84 15:31")
                                                          (* Sets the shape of the current brush)
   (PROG [(NEWSHAPE (PAINTW.READBRUSHSHAPE))
          (NOWBRUSH (fetch (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP W 'SKETCHCONTEXT]
         (RETURN (AND NEWSHAPE (replace (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP W 'SKETCHCONTEXT)
                                  with (create BRUSH using NOWBRUSH BRUSHSHAPE _ NEWSHAPE])
```

(SKETCH.SET.BRUSH.SIZE

```
[LAMBDA (W)                                               (* rrb "12-Jan-85 10:13")
                                                          (* sets the size of the current brush)
   (SK.SET.DEFAULT.BRUSH.SIZE [READBRUSHSIZE (fetch (BRUSH BRUSHSIZE) of (fetch (SKETCHCONTEXT SKETCHBRUSH)
                                                                             of (WINDOWPROP W 'SKETCHCONTEXT]
          W])
```

(SKETCHW.CLOSEFN

```
[LAMBDA (SKW)                                             (* rrb " 1-Oct-86 17:44")
                                                          (* close function for a viewer. Removes itself from the list of
                                                          viewers.)
   (PROG (PROCINFO)
         [COND
            [(SETQ PROCINFO (WINDOWPROP SKW 'DOCUMENTINFO))     (* this window came from a tedit document.)
             [COND
                ((WINDOWPROP SKW 'SKETCHCHANGED)
                 (COND
                    ((EQ (UPDATE.IMAGE.IN.DOCUMENT SKW)
                         'DON'T)
                     (RETURN 'DON'T]
             (COND
                ([OR (TTY.PROCESSP (THIS.PROCESS))
                     (TTY.PROCESSP (WINDOWPROP SKW 'PROCESS]
```

(* if this process or the sketch process has the tty, give it back to the Tedit that this window came from.)

```
                (AND [PROCESSP (SETQ PROCINFO (WINDOWPROP (fetch (SKETCHDOCUMENTINFO FROMTEDITWINDOW)
                                                                 of PROCINFO)
                                                          'PROCESS]
                               (TTY.PROCESS PROCINFO]
           ((NULL (SK.CONFIRM.DESTRUCTION SKW "unsaved changes ... press LEFT to close anyway"))
            (RETURN 'DON'T]
       (REMOVE.SKETCH.VIEWER (WINDOWPROP SKW 'SKETCH)
              SKW)                                          (* kill the process that supports the typing.)
       (DEL.PROCESS (WINDOWPROP SKW 'PROCESS NIL))
       (WINDOWADDPROP SKW 'OPENFN 'SKETCHW.REOPENFN])
```

(SK.CONFIRM.DESTRUCTION

```
[LAMBDA (VIEWER MSG)                                        (* rrb " 1-Oct-86 17:37")

        (* some destructive operation is about to take place, if the viewer is dirty, confirm that this is what is intended.
        If so, return T. If not, NIL.)

   (COND
      ((OR (WINDOWPROP VIEWER 'DONTQUERYCHANGES)
           (NEQ (WINDOWPROP VIEWER 'SKETCHCHANGED)
                T)))
      (T                                                    (* ask if user really wants to close)
         (STATUSPRINT VIEWER "
                ")
         (COND
            ((MOUSECONFIRM (OR MSG "unsaved changes ... press LEFT to do operation anyway")
                    T
                    (GETPROMPTWINDOW VIEWER))               (* close the prompt window which MOUSECONFIRM brought
                                                            up.)
             (CLOSEPROMPTWINDOW VIEWER)
             T)
            (T NIL])
```

(SKETCHW.OUTFN

```
[LAMBDA (SKW)                                               (* rrb "24-Jan-85 10:06")

        (* the cursor is leaving the window, updates any structures that may be spread out for efficiency.)

   NIL])
```

(SKETCHW.REOPENFN

```
[LAMBDA (SKW)                                               (* rrb " 7-Feb-84 11:31")
                                                            (* reopenfn for viewers. Adds it back onto the list of global
                                                            viewers.)
   (ADD.SKETCH.VIEWER (WINDOWPROP SKW 'SKETCH)
          SKW)
   (WINDOWPROP SKW 'PROCESS (ADD.PROCESS (LIST (FUNCTION WB.EDITOR)
                                               (KWOTE SKW])
```

(MAKE.LOCAL.SKETCH

```
[LAMBDA (SKETCH SKETCHREGION SCALE STREAM EVERYTHINGFLG)    (* rrb "22-Apr-85 16:45")

        (* * calculate the local parts for the region of the sketch at a given scale.
        EVERYTHINGFLG provides a way to override the inside check.
        This is necessary because the inside check works on local elements.
        When the inside check is change to work on global elements, this can be removed.)

   (for SKELT in (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH SKETCH)) when (OR EVERYTHINGFLG (SK.INSIDE.REGION
                                                                                              SKELT
                                                                                              SKETCHREGION))
      collect (SK.LOCAL.FROM.GLOBAL SKELT STREAM SCALE])
```

(MAP.SKETCHSPEC.INTO.VIEWER

```
[LAMBDA (SKETCH SKW)                                        (* rrb "12-May-85 17:02")
                                                            (* creates the local parts of a sketch and puts it onto the viewer.)
   (PROG ((SKREGION (WINDOWPROP SKW 'REGION.VIEWED))
          SPECS)

        (* local specs are kept as a TCONC cell so that additions to the end are fast.)

        (RETURN (WINDOWPROP SKW 'SKETCHSPECS (CONS [SETQ SPECS (CONS (fetch (SKETCH SKETCHNAME) of SKETCH)
                                                                     (for SKELT
                                                                        in (fetch (SKETCH SKETCHELTS)
                                                                              of SKETCH)
                                                                        when (SK.INSIDE.REGION SKELT SKREGION)
                                                                        collect (SK.LOCAL.FROM.GLOBAL SKELT
                                                                                       SKW]
                                                   (LAST SPECS])
```

(SKETCHW.REPAINTFN

```
[LAMBDA (W REG STOPIFMOUSEDOWN NEWGRIDFLG)                    (* rrb "21-Feb-86 10:38")
                                                              (* redisplays the sketch in a window)
                                                              (* for now ignore the region.)
```

(* if STOPIFMOUSEDOWN is T, it displays some but stops if the button left or middle button is still down and returns STOPPED)

```
  (DSPOPERATION 'PAINT W)
  (DSPRIGHTMARGIN 65000 W)                                    (* I don't know exactly how scrolling ever gets turned on but it has.)
  (DSPSCROLL 'OFF W)
  (PROG1 (SKETCHW.REPAINTFN1 W REG (AND STOPIFMOUSEDOWN (SETUPTIMER AUTOZOOM.REPAINT.TIME))
                NEWGRIDFLG)
         (SKED.SELECTION.FEEDBACK W])
```

(SKETCHW.REPAINTFN1

```
[LAMBDA (SKW REGION TIMER NEWGRIDFLG)                         (* rrb "11-Jul-86 15:51")
```

(* Draws all of the local elements in the sketch window SKW. internal function to SKETCHW.REPAINTFN This entry is provided so that SK.DRAWFIGURE.IF can RETFROM it if the timer has expired and a button is down.)

```
  (MAPSKETCHSPECS (LOCALSPECS.FROM.VIEWER SKW)
         (COND
            (TIMER                                            (* call a version of SK.DRAWFIGURE that checks the time.)
                   (FUNCTION SK.DRAWFIGURE.IF))
            (T (FUNCTION SK.DRAWFIGURE)))
          SKW REGION (VIEWER.SCALE SKW))
  (COND
     ((WINDOWPROP SKW 'GRIDUP)                                (* if grid is up, redisplay it)
      (SK.DISPLAY.GRID.POINTS SKW NEWGRIDFLG])
```

(SK.DRAWFIGURE.IF

```
[LAMBDA (SCREENELT STREAM REGION SCALE)                       (* rrb "22-Jan-85 11:34")
```

(* draws an element of a sketch in a window. If the free variable TIMER has expired and a button is down, it RETFROMs the repainting function.)

```
  (PROG1 (SK.DRAWFIGURE SCREENELT STREAM REGION SCALE)
      (AND TIMER (MOUSESTATE (OR LEFT MIDDLE))
           (TIMEREXPIRED? TIMER)
           (RETFROM 'SKETCHW.REPAINTFN1 'STOPPED)))])
```

(SKETCHW.SCROLLFN

```
[LAMBDA (SKW XDELTA YDELTA CONTINUOUSFLG)                      (* rrb "11-Jul-86 15:51")
```

(* scroll function for a sketch window. It must check to see which elements need to get added and deleted from the ones currently viewed as a result of the scrolling. Also if an element gets added, the clipping region must be expanded because part of the display of the object may be in the already visible part of the window.)

```
  (PROG ([SKETCH (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH (SKETCH.FROM.VIEWER SKW]
         (NOWREG (DSPCLIPPINGREGION NIL SKW))
         NEWREGION NEWLOCALREGION INNEW? NEWONES LOCALELT SCALE)
                                                              (* clear the caret.)
        (SKED.CLEAR.SELECTION SKW)
        [COND
           (CONTINUOUSFLG                                     (* set XDELTA and YDELTA for continuous scrolling)
                  [COND
                     ((AND XDELTA (NEQ XDELTA 0))
                      (COND
                         ((IGREATERP XDELTA 0)
                          (SETQ XDELTA 12))
                         (T (SETQ XDELTA -12]
                  (COND
                     ((AND YDELTA (NEQ YDELTA 0))
                      (COND
                         ((IGREATERP YDELTA 0)
                          (SETQ YDELTA 12))
                         (T (SETQ YDELTA -12]
        [SETQ NEWREGION (UNSCALE.REGION (SETQ NEWLOCALREGION (CREATEREGION (DIFFERENCE (fetch (REGION LEFT)
                                                                                           of NOWREG)
                                                                                        (COND
                                                                                           (XDELTA)
                                                                                           (0)))
                                                                           (DIFFERENCE (fetch (REGION BOTTOM)
                                                                                          of NOWREG)
                                                                                       (COND
                                                                                          (YDELTA)
                                                                                          (0)))
                                                                           (fetch (REGION WIDTH) of NOWREG)
                                                                           (fetch (REGION HEIGHT) of NOWREG)))
                                        (SETQ SCALE (VIEWER.SCALE SKW]
```

(* update the current image to contain the things that will be there after the scroll, then scroll.)

```
        [for GELT in SKETCH do (SETQ INNEW? (SK.INSIDE.REGION GELT NEWREGION))
                               (COND
                                  [(SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART GELT SKW))
                                                                  (* if it is not supposed to be in the new region, remove it.)
                                   (OR INNEW? (COND
                                                 ((REGIONSINTERSECTP NEWLOCALREGION (SK.ITEM.REGION LOCALELT))
                                                                  (* part of image may overlap the part of sketch that is still
                                                                  showing)
                                                  (SK.ERASE.AND.DELETE.ITEM LOCALELT SKW))
                                                 (T (SK.DELETE.ITEM LOCALELT SKW]
                                  (INNEW?                         (* just came in)
                                         (SETQ NEWONES (CONS GELT NEWONES]
        (SCROLLBYREPAINTFN SKW XDELTA YDELTA)
        (SKETCHW.FIG.CHANGED SKW)
        (SK.UPDATE.REGION.VIEWED SKW)
        (for GELT in NEWONES do (SKETCH.ADD.AND.DISPLAY1 GELT SKW SCALE])
```

(SKETCHW.RESHAPEFN

```
  [LAMBDA (SKW OLDIMAGE IMAGEREGION OLDSCREENREGION)              (* rrb "11-Jul-86 15:51")
```

(* reshape function for a sketch window. It must check to see which elements need to get added and deleted from the ones currently viewed as a result of the reshaping.)

```
    (PROG ([SKETCH (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH (SKETCH.FROM.VIEWER SKW]
           (NOWREG (DSPCLIPPINGREGION NIL SKW))
           NEWREGION NEWLOCALREGION INNEW? NEWONES LOCALELT SCALE)
                                                                  (* clear the caret.)
          (SKED.CLEAR.SELECTION SKW)
          (RESHAPEBYREPAINTFN SKW OLDIMAGE IMAGEREGION OLDSCREENREGION)
          [SETQ NEWREGION (UNSCALE.REGION (SETQ NEWLOCALREGION (DSPCLIPPINGREGION NIL SKW))
                                  (SETQ SCALE (VIEWER.SCALE SKW]
```

(* update the current image to contain the things that will be there after the scroll, then scroll.)

```
          [for GELT in SKETCH do (SETQ INNEW? (SK.INSIDE.REGION GELT NEWREGION))
                                 (COND
                                    [(SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART GELT SKW))
                                                                  (* if it is not supposed to be in the new region, remove it.)
                                     (OR INNEW? (COND
                                                   ((REGIONSINTERSECTP NEWLOCALREGION (SK.ITEM.REGION LOCALELT))
                                                                  (* part of image may overlap the part of sketch that is still
                                                                  showing)
                                                    (SK.ERASE.AND.DELETE.ITEM LOCALELT SKW))
                                                   (T (SK.DELETE.ITEM LOCALELT SKW]
                                    (INNEW?                       (* just came in)
                                           (SETQ NEWONES (CONS GELT NEWONES]
          (SKETCHW.FIG.CHANGED SKW)
          (SK.UPDATE.REGION.VIEWED SKW)
          (for GELT in NEWONES do (SKETCH.ADD.AND.DISPLAY1 GELT SKW SCALE])
```

(SK.UPDATE.EVENT.SELECTION

```
  [LAMBDA (HOTSPOTCACHE X1 Y1 X2 Y2 SCALE WINDOW COPYMODE DELETEMODE)
                                                                  (* rrb "31-Jan-85 11:35")
```

(* * internal function to SK.COPY.BUTTONEVENTFN that determines the elements within the given bounds and selects or deselects them.)

```
    (PROG (SELITEMS)
          (RETURN (COND
                     ((LASTMOUSESTATE UP)                         (* don't do anything with button up.)
                      NIL)
                     ((SETQ SELITEMS (SK.LOCAL.ITEMS.IN.REGION HOTSPOTCACHE (MIN X1 X2)
                                                               (MIN Y1 Y2)
                                                               (MAX X1 X2)
                                                               (MAX Y1 Y2)))
                                                                  (* OLD CODE (SETQ SELITEMS
                                                                  (SK.LOCAL.ITEMS.IN.REGION HOTSPOTCACHE
                                                                  (REGION.FROM.COORDINATES X1 Y1 X2 Y2) SCALE)))
                      (COND
                         [(LASTMOUSESTATE (OR (ONLY LEFT)
                                              (ONLY MIDDLE)))     (* left or middle only selects.)
                          (for SELITEM in SELITEMS do (SK.ADD.SELECTION SELITEM WINDOW (SK.BUTTONEVENT.MARK
                                                                                       COPYMODE DELETEMODE]
                         (T                                       (* anything but left only should cause deselect.)
                            (for SELITEM in SELITEMS do (SK.REMOVE.SELECTION SELITEM WINDOW (
                                                                                            SK.BUTTONEVENT.MARK
                                                                                             COPYMODE DELETEMODE
                                                                                             ])
```

(LIGHTGRAYWINDOW

```
  [LAMBDA (WINDOW)                                                (* rrb "28-Jun-84 10:27")
    (DSPFILL NIL 1 'INVERT WINDOW)
    WINDOW])
```

(SK.ADD.SPACES

```
  [LAMBDA (STRLST)                                          (* rrb "19-Jul-85 15:11")
                                                            (* adds eols between the elements of STRLST)
    (for STR in STRLST join (COND
                              ((EQUAL STR "")
                               NIL)
                              ((EQ (NTHCHARCODE STR -1)
                                   (CHARCODE EOL))          (* if it already ends in CR, don't add one.)
                               (LIST STR))
                              (T (LIST STR "
                                         "])
```

(SK.SKETCH.MENU

```
  [LAMBDA (SKW)                                             (* rrb "12-Sep-85 11:50")
                                                            (* brings up the normal sketch command menu.)
    (SK.MIDDLE.TITLEFN SKW T])
```

(SK.CHECK.IMAGEOBJ.WHENDELETEDFN

```
  [LAMBDA (GELT SKETCHW)                                    (* rrb "19-Oct-85 17:10")

          (* check to see if a when deleted function needs to be applied and applies it.)

    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (SKIMAGEOBJ                                         (* deleting an image object apply WHENDELETEDFN)
                    (SK.APPLY.IMAGEOBJ.WHENDELETEDFN GELT SKETCHW))
        (GROUP (for GELT in (fetch (GROUP LISTOFGLOBALELTS) of GELT) do (SK.CHECK.IMAGEOBJ.WHENDELETEDFN GELT
                                                                         SKETCHW)))
        NIL])
```

(SK.APPLY.IMAGEOBJ.WHENDELETEDFN

```
  [LAMBDA (GELT SKETCHW)                                    (* rrb "30-Jul-85 15:35")
                                                            (* applies the when deleted function for an image object.)
    (PROG (IMAGEOBJ FN)
          (COND
             ((AND (SETQ FN (IMAGEOBJPROP (SETQ IMAGEOBJ (fetch (SKIMAGEOBJ SKIMAGEOBJ)
                                                           of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                 of GELT)))
                                          'WHENDELETEDFN))
                   (NEQ FN 'NILL))                          (* documentation calls for passing text streams as well but there
              aren't any.)
              (APPLY* FN IMAGEOBJ SKETCHW])
```

(SK.RETURN.TTY

```
  [LAMBDA (W)                                               (* rrb "29-Aug-85 11:09")
                                                            (* gives up the tty when the window is shrunken.)
    (AND (TTY.PROCESSP (WINDOWPROP W 'PROCESS))
         (TTY.PROCESS T])
```

(SK.TAKE.TTY

```
  [LAMBDA (W)                                               (* rrb "29-Aug-85 11:10")
                                                            (* takes the tty when the window is expanded)
    (TTY.PROCESS (WINDOWPROP W 'PROCESS])

)

;; fns for dealing with the sketch menu

(DEFINEQ
```

(SKETCH.COMMANDMENU

```
  [LAMBDA (ITEMS TITLE)                                     (* rrb "14-Jul-86 13:43")
    (create MENU
            ITEMS _ ITEMS
            CENTERFLG _ T
            WHENSELECTEDFN _ (FUNCTION SKETCHW.SELECTIONFN)
            MENUFONT _ (SK.FONTNAMELIST (FONTCREATE BOLDFONT))
            TITLE _ TITLE])
```

(SKETCH.COMMANDMENU.ITEMS

```
  [LAMBDA (ADDFIXITEM ELEMENTTYPES VIEWER)                  (* rrb "24-Sep-86 18:11")
                                                            (* returns a list of the items that are in the sketch command
                                                            menu.)
    (APPEND '[(Delete SK.DELETE.ELT "Deletes one or more elements from the sketch." (SUBITEMS ("Delete
                                                                                               element(s)"
                                                                                                    SK.DELETE.ELT
                                                                                                      "Deletes
                                                                                                      one or more
                                                                                                      elements
                                                                                                      from the
                                                                                                      sketch.")
                                                                                              ("Delete point"
```

```
                                                                                                    SK.DELETE.KNOT
                                                                                                    "Deletes a
                                                                                                    control point
                                                                                                    from a wire or
                                                                                                    curve."]
        '[(Move SK.APPLY.DEFAULT.MOVE "Moves a control point, or one or more elements."
                (SUBITEMS (Move% point SK.MOVE.ELEMENT.POINT "Moves one of the control points.")
                       ("Move points" SK.MOVE.POINTS "Moves a collection of control points.")
                       ("Move elements" SK.MOVE.ELT "Moves one or more elements of the sketch.")
                       ("Two pt transform" SK.TWO.PT.TRANSFORM.ELTS "Moves one or more sketch elements with a
                              two point transformation.")
                       ("Three pt transform" SK.THREE.PT.TRANSFORM.ELTS "Moves one or more sketch elements
                              with a three point transformation.")
                       ("Set MOVE command mode" SK.SET.MOVE.MODE "changes whether the MOVE command applies to
                              points or elements." (SUBITEMS (Points SK.SET.MOVE.MODE.POINTS "Top level MOVE
                                                                    command will be the same as MOVE POINTS
                                                                    command.")
                                                             (Elements SK.SET.MOVE.MODE.ELEMENTS "Top level MOVE
                                                                    command will be the same as MOVE ELEMENTS
                                                                    command.")
                                                             (Combined SK.SET.MOVE.MODE.COMBINED "MOVE command
                                                                    will move points if a single point is
                                                                    clicked; elements otherwise"]
        '[(Copy SK.COPY.ELT "Copies a piece of the sketch." (SUBITEMS ("Copy elements" SK.COPY.ELT "copies
                                                                    one or more elements of the
                                                                    sketch.")
                                                                ("Copy w/2 pt trans"
                                                                    SK.COPY.AND.TWO.PT.TRANSFORM.ELTS
                                                                    "Copies one or more sketch elements
                                                                    with a two point transformation.")
                                                                ("Copy w/3 pt trans"
                                                                    SK.COPY.AND.THREE.PT.TRANSFORM.ELTS
                                                                    "Copies one or more sketch elements
                                                                    with a three point
                                                                    transformation."]
        '[(Align SK.ALIGN.POINTS.LEFT "Aligns a collection of points with the leftmost one."
                (SUBITEMS ("Align Left" SK.ALIGN.POINTS.LEFT "Aligns a collection of points with the
                                  leftmost one.")
                       ("Align Right" SK.ALIGN.POINTS.RIGHT "Aligns a collection of points with the
                              rightmost one.")
                       ("Align Top" SK.ALIGN.POINTS.TOP "Aligns a collection of points with the topmost
                              one.")
                       ("Align Bottom" SK.ALIGN.POINTS.BOTTOM "Aligns a collection of points with the
                              bottommost one.")
                       ("Move onto grid" SK.PUT.ELTS.ON.GRID "Moves control points to nearest grid point.")
                       ("Space evenly in X" SK.EVEN.SPACE.POINTS.IN.X "Moves points so that they are evenly
                              spaced between the leftmost and rightmost.")
                       ("Space evenly in Y" SK.EVEN.SPACE.POINTS.IN.Y "Moves points so that they are evenly
                              spaced between the topmost and bottommost."]
        '((Change SK.CHANGE.ELT "Changes a property of a piece or collection of pieces."))
        [AND (GETD 'SK.SEL.AND.SHOW.ANNOTE)
             (GETD 'NCP.CardFromWindow)
             (NCP.CardFromWindow VIEWER)
             '((Annotations SK.SEL.AND.SHOW.ANNOTE "Manipulates the annotations from a selected element."
                       (SUBITEMS (Add% Annotation SK.SEL.AND.ADD.ANNOTE "Adds an annotation to an element.")
                              (Delete% Annotation SK.SEL.AND.DELETE.ANNOTE "Deletes the annotation from an
                                     element.")
                              (Show% Annotation SK.SEL.AND.SHOW.ANNOTE "Shows the annotation of an element."]
        (for ELEMENT in (COND
                           ((EQ ELEMENTTYPES T)
                            SKETCH.ELEMENT.TYPE.NAMES)
                           (T ELEMENTTYPES))
           when [fetch (SKETCHTYPE LABEL) of (SETQ ELEMENT (GETPROP ELEMENT 'SKETCHTYPE]
           collect                                          (* add the sketch elements that have a label.)
                 (LIST (fetch (SKETCHTYPE LABEL) of ELEMENT)
                       ELEMENT
                       (fetch (SKETCHTYPE DOCSTR) of ELEMENT)))
        [AND (GETD 'SK.SEL.AND.SHOW.ANNOTE)
             '((Link SK.ADD.ANNOTATION "Adds an annotation object."]
        [AND FILLINGMODEFLG '(("Bury" SK.SEND.TO.BOTTOM "will put selected elements on the bottom of the
                                  display stack." (SUBITEMS ("Send to bottom" SK.SEND.TO.BOTTOM "same as
                                                                    BURY, puts selected elements on the
                                                                    bottom of the display stack.")
                                                             ("Bring to top" SK.BRING.TO.TOP "will cause
                                                                    selected elements to be displayed on top
                                                                    of ones now covering it.")
                                                             ("Reverse order" SK.SWITCH.PRIORITIES
                                                                    "reorders the display of elements.
                                                                    Selecting 2 will switch them."]
        '[(Group SK.GROUP.ELTS "groups a collection of elements into a single unit." (SUBITEMS ("Move group
                                                                                                    control
                                                                                                    point"
                                                                                        SK.MOVE.GROUP.CONTROL.PT

                                                                                                    "moves
                                                                                                    the
                                                                                                    control
```

```
                                                                                  point
                                                                                  of a
                                                                                  group
                                                                                  without
                                                                                  moving
                                                                                  the
                                                                                  group."
                                                                                  )
                                                                           (Group
                                                                               SK.GROUP.ELTS
                                                                                 "groups a
                                                                                 collection
                                                                                 of
                                                                                 elements
                                                                                 into a
                                                                                 single
                                                                                 unit.")
                                                                           (Freeze
                                                                              SK.FREEZE.ELTS

                                                                                 "freezes
                                                                                 elements
                                                                                 so that
                                                                                 they can
                                                                                 not be
                                                                                 moved,
                                                                                 changed,
                                                                                 copied or
                                                                                 deleted."]
          '[(UnGroup SK.UNGROUP.ELT "replaces a group element by its constituents." (SUBITEMS (UnGroup
                                                                              SK.UNGROUP.ELT

                                                                                 "replaces
                                                                                 a group
                                                                                 element by
                                                                                 its
                                                                                 constituen
ts.")
                                                                        (UnFreeze
                                                                             SK.UNFREEZE.ELT

                                                                              "unprotects
                                                                              an element so
                                                                              it can be
                                                                              moved,
                                                                              changed,
                                                                              copied or
                                                                              deleted."]
          '[(Undo SK.UNDO.LAST "undoes the previous event. Or the latest one that hasn't been undone."
                  (SUBITEMS (?Undo SK.SEL.AND.UNDO "allows selection of an event to undo.")
                         (Undo SK.UNDO.LAST "undoes the previous event. Or the latest one that hasn't been
                               undone."]
          '[(Defaults SKETCH.SET.A.DEFAULT "Changes one of the default characteristics."
                   (SUBITEMS (Line SKETCH.SET.BRUSH.SIZE "Sets the characteristics of the default brush."
                                   (SUBITEMS (Size SKETCH.SET.BRUSH.SIZE "Sets the size of the default brush")
                                          (Shape SKETCH.SET.BRUSH.SHAPE "Sets the shape of the default brush")
                                          (Add% arrowhead SK.SET.LINE.ARROWHEAD "Makes it so that new lines
                                                 automatically have Arrowheads.")
                                          ("Mouse line specs" SK.SET.LINE.LENGTH.MODE "Sets whether the lines
                                                 drawn with the middle mouse button connect to each other.")))
                          (Arrowhead SK.SET.ARROWHEAD.LENGTH "Sets the characteristics of the default
                                 arrowhead." (SUBITEMS ("set Size of default arrowhead" SK.SET.ARROWHEAD.LENGTH
                                                             )
                                                    ("set Angle of default arrowhead" SK.SET.ARROWHEAD.ANGLE)
                                                    ("set Type of default arrowhead" SK.SET.ARROWHEAD.TYPE)
                                                    ("default Add arrowheads" SK.SET.LINE.ARROWHEAD "Makes it
                                                           so that new lines automatically have Arrowheads.")))
                          (Text SK.SET.TEXT.SIZE "Sets the size of newly added text." (SUBITEMS ("Font size"

                                                                              SK.SET.TEXT.SIZE
                                                                                "Sets the size
                                                                                of newly added
                                                                                text.")
                                                                          ("Font family"
                                                                           SK.SET.TEXT.FONT
                                                                           "Sets the font
                                                                           family of newly
                                                                           added text.")
                                                                          ("Horizontal
                                                                           justification"

                                                                     SK.SET.TEXT.HORIZ.ALIGN
                                                                           "Sets the
                                                                           horizontal
                                                                           justification
                                                                           mode of new
                                                                           text.")
```

```
                                                                                                                        ("Vertical
                                                                                                                         justification"

                                                                                                                    SK.SET.TEXT.VERT.ALIGN
                                                                                                                         "Sets the
                                                                                                                         vertical
                                                                                                                         justification of
                                                                                                                         new text.")
                                                                                                                        ("Bold and/or
                                                                                                                         italic"
                                                                                                                         SK.SET.TEXT.LOOKS
                                                                                                                               "Sets the
                                                                                                                               bold and
                                                                                                                               italic look
                                                                                                                               of new
                                                                                                                               text.")))
                    (Text% Box SK.SET.TEXTBOX.HORIZ.ALIGN "Sets the alignment of text within new text
                           boxes." (SUBITEMS ("Horizontal justification" SK.SET.TEXTBOX.HORIZ.ALIGN
                                                       "Sets the horizontal alignment of text within new
                                                       text boxes.")
                                             ("Vertical justification" SK.SET.TEXTBOX.VERT.ALIGN "Sets the
                                                    vertical alignment of text within new text boxes.")))
                    (Arc SK.SET.ARC.DIRECTION "Sets the direction arcs go around their circle."
                         (SUBITEMS ("Clockwise" SK.SET.ARC.DIRECTION.CW "Makes new arcs go around in the
                                             clockwise direction")
                                  ("Counterclockwise" SK.SET.ARC.DIRECTION.CCW "Makes new arcs go around in
                                          the counterclockwise direction")))
                    ("Input scale" SK.SET.INPUT.SCALE "Sets the scale for newly added lines and text."
                           (SUBITEMS ("Read new input scale" SK.SET.INPUT.SCALE "Reads a new input
                                               scale.")
                                    ("Make input scale current" SK.SET.INPUT.SCALE.CURRENT "makes the input
                                           scale be the scale of the current view.")))
                    (Feedback SK.SET.FEEDBACK.MODE "Controls the amount of feedback when adding new
                           curves, circles, etc." (SUBITEMS ("Points only" SK.SET.FEEDBACK.POINT "Only
                                                                      the control points will be shown when
                                                                      entering elements.")
                                                          ("Fast figures" SK.SET.FEEDBACK.VERBOSE
                                                                 "Wires, circles and ellipses are shown
                                                                 while they are being entered.")
                                                          ("All figures" SK.SET.FEEDBACK.ALWAYS "Most
                                                                 elements are shown while they are being
                                                                 entered.
                                                                 This will be slow for arcs and curves."]
         '[(Grid SK.SET.GRID "Flips between using the grid and not using the grid." (SUBITEMS (|Turn grid ON|
                                                                                                SK.TURN.GRID.ON
                                                                                                "turns on a
                                                                                                grid.  Only pts
                                                                                                on the grid can
                                                                                                be selected.")
                                                                                              (|Turn grid OFF|
                                                                                               SK.TURN.GRID.OFF
                                                                                               "turns off the
                                                                                               grid.  Any point
                                                                                               can be selected.")
                                                                                              (LARGER% Grid
                                                                                               SK.MAKE.GRID.LARGER
                                                                                                     "doubles the
                                                                                                     distance
                                                                                                     between the
                                                                                                     grid
                                                                                                     points.")
                                                                                              (smaller% Grid

                                                                                              SK.MAKE.GRID.SMALLER
                                                                                               "halves the
                                                                                               distance between
                                                                                               the grid points.")
                                                                                              ("Display grid"
                                                                                               SK.DISPLAY.GRID
                                                                                               "XORs a point at
                                                                                               each grid point.
                                                                                               If grid is
                                                                                               visible, this will
                                                                                               erase it.")
                                                                                              ("Remove grid
                                                                                               display"
                                                                                                 SK.TAKE.DOWN.GRID
                                                                                                     "XORs a
                                                                                                     point at
                                                                                                     each grid
                                                                                                     point.  If
                                                                                                     grid is
                                                                                                     visible,
                                                                                                     this will
                                                                                                     erase it."]
         '[("Move view" SKETCH.ZOOM "makes a new region the part of the sketch visible."
                 (SUBITEMS ("Move view" SKETCH.ZOOM "changes the scale of the display.")
```

```
                        (AutoZoom SKETCH.AUTOZOOM "changes the scale around a selected point.")
                        (Home SKETCH.HOME "returns to the origin at the original scale")
                        ("Fit to window" SK.FRAME.IT "moves so that the entire sketch just fits in the
                               window" (SUBITEMS ("Fit to window" SK.FRAME.IT "moves so that the entire
                                                    sketch just fits in the window")
                                          ("Fit window to sketch" SK.FRAME.WINDOW.TO.SKETCH "reshapes the
                                                 window so that the entire sketch just fits")))
                        ("Restore view" SK.RESTORE.VIEW "Moves to a previously saved view."
                               (SUBITEMS ("Restore view" SK.RESTORE.VIEW "Moves to a previously saved view.")
                                      ("Save view" SK.NAME.CURRENT.VIEW "saves the current view (position and
                                             scale) of the sketch for easy return.")
                                      ("Forget view" SK.FORGET.VIEW "Deletes a previously saved view.")))
                        ("Coord window" ADD.GLOBAL.DISPLAY "creates a window that shows the cursor in global
                               coordinates." (SUBITEMS ("Coord window" ADD.GLOBAL.DISPLAY "creates a window
                                                    that shows the cursor position in global
                                                    coordinates.")
                                                 ("Grid coord window" ADD.GLOBAL.GRIDDED.DISPLAY
                                                        "creates a window that shows the grid position
                                                        nearest the cursor in global coordinates.")))
                        (New% window SKETCH.NEW.VIEW "opens another viewer onto this sketch"]
                '[(HardCopy SK.HARDCOPYIMAGEW "sends a copy of the current window contents on the default printer."
                          (SUBITEMS ("To a file" SK.HARDCOPYIMAGEW.TOFILE "Puts image on a file; prompts for filename
                                             and format")
                                 ("To a printer" SK.HARDCOPYIMAGEW.TOPRINTER "Sends image to a printer of your
                                        choosing")
                                 ("Whole sketch" SK.LIST.IMAGE "Sends the image of the whole sketch at the current
                                        scale to the printer." (SUBITEMS ("To a file" SK.LIST.IMAGE.ON.FILE "Sends the
                                                                          image of the whole sketch at the
                                                                          current scale on a file.")
                                                                   ("To a printer" SK.LIST.IMAGE "Sends the image
                                                                          of the whole sketch at the current scale
                                                                          to the printer.")))
                                 (Hardcopy% Display SK.SET.HARDCOPY.MODE "Makes the display correspond to the hardcopy
                                        image on the default printer.")
                                 (Normal% Display SK.UNSET.HARDCOPY.MODE "Changes the display to use display fonts."]
                '((Put SK.PUT.ON.FILE "saves this sketch on a file"))
                '[(Get SK.GET.FROM.FILE "gets a sketch from a file." (SUBITEMS (Get SK.GET.FROM.FILE "gets a sketch
                                                                                    from a file.")
                                                                          (Include SK.INCLUDE.FILE "adds the
                                                                                 contents of a file to the existing
                                                                                 sketch."]
                [AND ADDFIXITEM '((Fix% Menu SK.FIX.MENU "leaves up the menu of sketch operations."]
                (AND (EQUAL (USERNAME)
                            "BURTON.PA")
                     '((inspect INSPECT.SKETCH "Calls the Inspector on the figure data structures."])
```

(**CREATE.SKETCHW.COMMANDMENU**

```
  [LAMBDA (MENUTITLE ADDFIXITEM ELEMENTTYPES VIEWER)          (* rrb " 6-May-86 15:22")
                                                              (* returns the control menu for a figure window.)
    (SKETCH.COMMANDMENU (SKETCH.COMMANDMENU.ITEMS ADDFIXITEM ELEMENTTYPES VIEWER)
           MENUTITLE])
```

(**SKETCHW.SELECTIONFN**

```
  [LAMBDA (ITEM MENU)                                         (* rrb "31-Jan-86 11:34")
```

(* calls the function appropriate for the item selected from the command menu associated with a figure window.)

```
    (PROG [(SKW (WINDOWPROP (WFROMMENU MENU)
                        'MAINWINDOW]
           (RETURN (RESETLST
                       (COND
                          ((OBTAIN.MONITORLOCK (SKETCH.MONITORLOCK SKW)
                                T T)                          (* clear the prompt window if there is one.)
                           (CLOSEPROMPTWINDOW SKW)            (* reset the line being drawn if there is one.)
                           (RESET.LINE.BEING.INPUT SKW)
                           (SK.APPLY.MENU.COMMAND (CADR ITEM)
                                  SKW))
                          (T (STATUSPRINT SKW "
                                   " "Sketch operation in progress.  Please wait.")))))])
```

(**SKETCH.MONITORLOCK**

```
  [LAMBDA (VIEWER)                                            (* rrb "31-Jan-86 10:20")
                                                              (* returns the monitorlock for a sketch)
    (OR (WINDOWPROP VIEWER 'MONITORLOCK)
        (PROG [(LOCK (CREATE.MONITORLOCK (GENSYM "Sketch"]
              (WINDOWPROP VIEWER 'MONITORLOCK LOCK)
              (RETURN LOCK])
```

(**SK.EVAL.AS.PROCESS**

```
  [LAMBDA (FORM VIEWER)                                       (* rrb "31-Jan-86 11:23")
                                                              (* evals a form that grabs the sketch lock on its viewer in a
                                                              process.)
    (COND
```

```
       ((THIS.PROCESS)
        (ADD.PROCESS (LIST 'SK.EVAL.WITH.LOCK (KWOTE FORM)
                           (KWOTE VIEWER))
              'RESTARTABLE
              'NO))
       (T                                                    (* processes aren't on, don't bother with monitor locks.)
          (\EVAL FORM])
```

(SK.EVAL.WITH.LOCK

```
  [LAMBDA (FORM VIEWER)                                     (* rrb "31-Jan-86 11:22")
                                                            (* evals FORM in a context where it has the lock on VIEWER)
    (WITH.MONITOR (SKETCH.MONITORLOCK VIEWER)
           (EVAL FORM])

)

(DEFINEQ
```

(SK.FIX.MENU

```
  [LAMBDA (SKETCHW DONTOPENFLG)                             (* rrb "23-Sep-86 17:59")
                                                            (* attaches the menu on the right side of the viewer.)
    (PROG (MENUW)
          (OR (SETQ MENUW (SK.INSURE.HAS.MENU SKETCHW))
              (RETURN))                                     (* clear the popup menu cache.)
          (WINDOWPROP SKETCHW 'SKETCHPOPUPMENUCACHE NIL)
          (WINDOWPROP MENUW 'MINSIZE (CONS [BITMAPWIDTH (UPDATE/MENU/IMAGE (CAR (WINDOWPROP MENUW 'MENU]
                                           20))
          (COND
             ((NOT (MEMB MENUW (ATTACHEDWINDOWS SKETCHW)))
              (ATTACHWINDOW MENUW SKETCHW 'RIGHT 'TOP 'LOCALCLOSE)
              (WINDOWADDPROP MENUW 'CLOSEFN (FUNCTION DETACHWINDOW))
              (WINDOWADDPROP MENUW 'CLOSEFN (FUNCTION SK.CLEAR.POPUP.MENU)
                     T)
              (OR DONTOPENFLG (OPENW MENUW])
```

(SK.SET.UP.MENUS

```
  [LAMBDA (SKETCHW DONTOPENFLG MENUSPEC)                    (* rrb "23-Sep-86 17:59")
                                                            (* attached the sketch menu to the window.)
    (PROG (FIXEDMENUW POPUPMENUW FIXIT?)
          (COND
             ((NULL MENUSPEC)                               (* mark window so both menus will come up if needed.)
              (SETQ FIXEDMENUW (SETQ POPUPMENUW T)))
             ((type? MENU MENUSPEC)

          (* put the given menu as the fixed one and establish the standard one as the SKETCHPOPUPMENU)

              (SETQ FIXEDMENUW (MENUWINDOW MENUSPEC T))
              (SETQ POPUPMENUW T)
              (SETQ FIXIT? T))
             [(LISTP MENUSPEC)
              (SETQ FIXIT? (CADDR MENUSPEC))
              [SETQ FIXEDMENUW (SELECTQ (CAR MENUSPEC)
                                        ((T NIL)
                                             (CAR MENUSPEC))
                                        (COND
                                           ((type? MENU (CAR MENUSPEC))
                                            (MENUWINDOW (CAR MENUSPEC)
                                                   T))
                                           (T (\ILLEGAL.ARG (CAR MENUSPEC]
              (SETQ POPUPMENUW (SELECTQ (CADR MENUSPEC)
                                        ((T NIL)
                                             (CADR MENUSPEC))
                                        (COND
                                           ((type? MENU (CADR MENUSPEC))
                                            (MENUWINDOW (CADR MENUSPEC)
                                                   T))
                                           (T (\ILLEGAL.ARG (CADR MENUSPEC]
             (T                                             (* default is to bring up the standard menu)
                (SETQ FIXEDMENUW (SETQ POPUPMENUW T))
                (SETQ FIXIT? T)))                           (* clear the popup menu cache.)
          (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU FIXEDMENUW)
          (WINDOWPROP SKETCHW 'SKETCHPOPUPMENU POPUPMENUW)
          (AND FIXIT? (SK.FIX.MENU SKETCHW DONTOPENFLG])
```

(SK.INSURE.HAS.MENU

```
  [LAMBDA (SKETCHW)                                         (* rrb "23-Sep-86 17:59")
                                                            (* makes sure a sketch window has a menu.)
    (PROG [(FIXEDMENU (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU]
          [COND
             ((EQ (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU)
                  T)                                        (* no fixed menu yet but wants standard one, create it)
              (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU (SETQ FIXEDMENU (SK.CREATE.STANDARD.MENU SKETCHW]
          (RETURN FIXEDMENU])
```

(SK.CREATE.STANDARD.MENU

```
  [LAMBDA (VIEWER)                                          (* rrb "23-Sep-86 17:52")
                                                            (* creates the standard sketch viewer fixed menu window.)
    (RESETFORM (CURSOR WAITINGCURSOR)
           (MENUWINDOW (CREATE.SKETCHW.COMMANDMENU NIL NIL T VIEWER)
                  T])
```

(SK.ADD.ITEM.TO.MENU

```
  [LAMBDA (OLDMENU NEWITEM)                                 (* rrb "23-Sep-86 09:53")
                                                            (* returns a menu that is like OLDMENU but has one additional
                                                            item NEWITEM)
                                                            (* clober enough fields to get the menu to redraw itself correctly.)
    (create MENU using OLDMENU ITEMS _ (APPEND (fetch (MENU ITEMS) of OLDMENU)
                                               (LIST NEWITEM))
                       MENUCOLUMNS _ NIL MENUROWS _ NIL IMAGE _ NIL MENUGRID _
                       (create REGION
                              LEFT _ 0
                              BOTTOM _ 0])
```

(SK.GET.VIEWER.POPUP.MENU

```
  [LAMBDA (SKETCHW)                                         (* rrb "24-Sep-86 10:31")

          (* gets the popup menu for a viewer. If the sketch menu is open, it creates a standard one.
          If the sketch menu isn't open, it adds the fix menu item to it and pops it up.
          It is cleared each time the menu is fixed.)

    (OR (WINDOWPROP SKETCHW 'SKETCHPOPUPMENUCACHE)
        (PROG [(SKETCHMENU (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU]
              [COND
                 [(OR (NULL SKETCHMENU)
                      (OPENWP SKETCHMENU))

          (* window doesn't want a fixed menu or its fixed menu is already open, check for a popup one)

                  (COND
                     ((EQ (SETQ SKETCHMENU (WINDOWPROP SKETCHW 'SKETCHPOPUPMENU))
                          T)
                      (WINDOWPROP SKETCHW 'SKETCHPOPUPMENU (SETQ SKETCHMENU (SK.CREATE.STANDARD.MENU SKETCHW]
                     (T                                     (* use the fixed menu with an item added to fix the menu.)
                        [COND
                           ((EQ SKETCHMENU T)               (* no fixed menu yet but wants standard one, create it)
                            (WINDOWPROP SKETCHW 'SKETCHFIXEDMENU (SETQ SKETCHMENU (SK.CREATE.STANDARD.MENU SKETCHW
                                                                                                            ]
                           (SETQ SKETCHMENU (MENUWINDOW (SK.ADD.ITEM.TO.MENU (CAR (WINDOWPROP SKETCHMENU 'MENU))
                                                                             '(Fix% Menu SK.FIX.MENU "leaves up the menu of sketch
                                                                                                      operations."))
                                                        T]
                    (WINDOWPROP SKETCHW 'SKETCHPOPUPMENUCACHE SKETCHMENU)
                    (RETURN SKETCHMENU])
```

(SK.CLEAR.POPUP.MENU

```
  [LAMBDA (MENUW)                                           (* rrb "24-Sep-86 10:34")

          (* clears the cache of pop up window so that the fixed menu will be used if the user middle buttons.)

    (PROG NIL
          (WINDOWPROP (OR (MAINWINDOW MENUW)
                          (RETURN))
                 'SKETCHPOPUPMENUCACHE NIL])

)

;; fns for dealing with sketch structures

(DEFINEQ
```

(SKETCH.CREATE

```
  [LAMBDA ARGS                                              (* rrb " 6-Nov-85 11:16")
    (PROG [(SKETCH (create SKETCH
                           SKETCHNAME _ (AND (GREATERP ARGS 0)
                                             (ARG ARGS 1]
          (PUTSKETCHPROP SKETCH 'SKETCHCONTEXT (CREATE.DEFAULT.SKETCH.CONTEXT))
          (PUTSKETCHPROP SKETCH 'VERSION SKETCH.VERSION)    (* pick out the props that are context,)
          [COND
             ((GREATERP ARGS 1)
              (for I from 2 to ARGS by 2 do (PUTSKETCHPROP SKETCH (ARG ARGS I)
                                                           (ARG ARGS (ADD1 I]
          (RETURN SKETCH])
```

(GETSKETCHPROP

```
  [LAMBDA (SKETCH PROPERTY)                                 (* rrb " 3-Mar-86 14:37")
                                                            (* retrieves the property of a sketch)
```

```
    (PROG ((SKETCH (INSURE.SKETCH SKETCH))
           SKETCHCONTEXT)
          (SETQ SKETCHCONTEXT (LISTGET (fetch (SKETCH SKETCHPROPS) of SKETCH)
                                       'SKETCHCONTEXT))
          (RETURN (SELECTQ PROPERTY
                       (BRUSH (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT))
                       (SHAPE (fetch (BRUSH BRUSHSHAPE) of (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT)))
                       (SIZE (fetch (BRUSH BRUSHSIZE) of (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT)))
                       (COLOR (fetch (BRUSH BRUSHCOLOR) of (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT)))
                       (FONT (fetch (SKETCHCONTEXT SKETCHFONT) of SKETCHCONTEXT))
                       (TEXTALIGNMENT
                            (fetch (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of SKETCHCONTEXT))
                       (ARROWHEAD (fetch (SKETCHCONTEXT SKETCHARROWHEAD) of SKETCHCONTEXT))
                       (DASHING (fetch (SKETCHCONTEXT SKETCHDASHING) of SKETCHCONTEXT))
                       (USEARROWHEAD (fetch (SKETCHCONTEXT SKETCHUSEARROWHEAD) of SKETCHCONTEXT))
                       (DRAWINGMODE (OR (fetch (SKETCHCONTEXT SKETCHDRAWINGMODE) of SKETCHCONTEXT)
                                        'REPLACE))
                       (TEXTBOXALIGNMENT
                            (fetch (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of SKETCHCONTEXT))
                       (TEXTURE (fetch (SKFILLING FILLING.COLOR) of (fetch (SKETCHCONTEXT SKETCHFILLING)
                                                                           of SKETCHCONTEXT)))
                       ((FILLINGCOLOR BACKCOLOR)
                            (fetch (SKFILLING FILLING.TEXTURE) of (fetch (SKETCHCONTEXT SKETCHFILLING) of
                                                                                                  SKETCHCONTEXT
                                                                                                  )))
                       (LINEMODE (fetch (SKETCHCONTEXT SKETCHLINEMODE) of SKETCHCONTEXT))
                       (ARCDIRECTION (fetch (SKETCHCONTEXT SKETCHARCDIRECTION) of SKETCHCONTEXT))
                       (MOVEMODE (fetch (SKETCHCONTEXT SKETCHMOVEMODE) of SKETCHCONTEXT))
                       (ELEMENTS (fetch (SKETCH SKETCHELTS) of SKETCH))
                       (NAME (fetch (SKETCH SKETCHNAME) of SKETCH))
                       (LISTGET (fetch (SKETCH SKETCHPROPS) of SKETCH)
                              PROPERTY])
```

(PUTSKETCHPROP

```
  [LAMBDA (SKETCH PROPERTY VALUE)                                    (* rrb " 3-Mar-86 13:58")
```

(* stores a property on a sketch Returns VALUE. Knows about the form of a sketch and does value checking (or should.))

```
    (PROG ((SKETCH (INSURE.SKETCH SKETCH))
           SKETCHCONTEXT PLIST)
          (SETQ PLIST (fetch (SKETCH SKETCHPROPS) of SKETCH))
          (SETQ SKETCHCONTEXT (LISTGET (fetch (SKETCH SKETCHPROPS) of SKETCH)
                                       'SKETCHCONTEXT))
          [SELECTQ PROPERTY
               (BRUSH (replace (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT with VALUE))
               (SHAPE (replace (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT with (create BRUSH
                                                                                         using (fetch (SKETCHCONTEXT
                                                                                                       SKETCHBRUSH)
                                                                                                  of SKETCHCONTEXT)
                                                                                        BRUSHSHAPE _ VALUE)))
               (SIZE (replace (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT with (create BRUSH
                                                                                        using (fetch (SKETCHCONTEXT
                                                                                                      SKETCHBRUSH)
                                                                                                 of SKETCHCONTEXT)
                                                                                       BRUSHSIZE _ VALUE)))
               (COLOR (replace (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT with (create BRUSH
                                                                                         using (fetch (SKETCHCONTEXT
                                                                                                       SKETCHBRUSH)
                                                                                                  of SKETCHCONTEXT)
                                                                                        BRUSHCOLOR _ VALUE)))
               (FONT (replace (SKETCHCONTEXT SKETCHFONT) of SKETCHCONTEXT with VALUE))
               (TEXTALIGNMENT
                    (replace (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of SKETCHCONTEXT with VALUE))
               (ARROWHEAD (replace (SKETCHCONTEXT SKETCHARROWHEAD) of SKETCHCONTEXT with VALUE))
               (DASHING (replace (SKETCHCONTEXT SKETCHDASHING) of SKETCHCONTEXT with VALUE))
               (USEARROWHEAD (replace (SKETCHCONTEXT SKETCHUSEARROWHEAD) of SKETCHCONTEXT with VALUE))
               (DRAWINGMODE (replace (SKETCHCONTEXT SKETCHDRAWINGMODE) of SKETCHCONTEXT with VALUE))
               (TEXTBOXALIGNMENT
                    (replace (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of SKETCHCONTEXT with VALUE))
               (TEXTURE (replace (SKETCHCONTEXT SKETCHFILLING) of SKETCHCONTEXT with (create SKFILLING
                                                                                             using (fetch (SKETCHCONTEXT
                                                                                                           SKETCHFILLING)
                                                                                                      of SKETCHCONTEXT)
                                                                                            FILLING.TEXTURE _
                                                                                            VALUE)))
               ((BACKCOLOR FILLINGCOLOR)
                    (replace (SKETCHCONTEXT SKETCHFILLING) of SKETCHCONTEXT with (create SKFILLING
                                                                                         using (fetch (SKETCHCONTEXT
                                                                                                       SKETCHFILLING
                                                                                                       )
                                                                                                  of SKETCHCONTEXT)
                                                                                        FILLING.COLOR _ VALUE)))
               (LINEMODE (replace (SKETCHCONTEXT SKETCHLINEMODE) of SKETCHCONTEXT with VALUE))
               (ARCDIRECTION (replace (SKETCHCONTEXT SKETCHARCDIRECTION) of SKETCHCONTEXT with VALUE))
               (MOVEMODE (replace (SKETCHCONTEXT SKETCHMOVEMODE) of SKETCHCONTEXT with VALUE))
```

```
            (ELEMENTS (replace (SKETCH SKETCHTCELL) of SKETCH with (CONS VALUE (LAST VALUE))))
            (NAME (replace (SKETCH SKETCHNAME) of SKETCH with VALUE))
            (COND
               (PLIST (LISTPUT PLIST PROPERTY VALUE))
               (T (replace (SKETCH SKETCHPROPS) of SKETCH with (LIST PROPERTY VALUE]
         (RETURN VALUE])
```

(CREATE.DEFAULT.SKETCH.CONTEXT

```
  [LAMBDA NIL                                                    (* rrb "23-Sep-86 10:40")
                                                                 (* returns a default sketch context)
    (create SKETCHCONTEXT
            SKETCHBRUSH _ SK.DEFAULT.BRUSH
            SKETCHFONT _ [OR SK.DEFAULT.FONT (SK.FONT.LIST (DEFAULTFONT 'DISPLAY]
            SKETCHTEXTALIGNMENT _ SK.DEFAULT.TEXT.ALIGNMENT
            SKETCHARROWHEAD _ (create ARROWHEAD
                                      ARROWTYPE _ SK.DEFAULT.ARROW.TYPE
                                      ARROWANGLE _ SK.DEFAULT.ARROW.ANGLE
                                      ARROWLENGTH _ SK.DEFAULT.ARROW.LENGTH)
            SKETCHDASHING _ SK.DEFAULT.DASHING
            SKETCHUSEARROWHEAD _ NIL
            SKETCHTEXTBOXALIGNMENT _ SK.DEFAULT.TEXTBOX.ALIGNMENT
            SKETCHFILLING _ (SK.CREATE.DEFAULT.FILLING)
            SKETCHLINEMODE _ T
            SKETCHINPUTSCALE _ 1
            SKETCHDRAWINGMODE _ SK.DEFAULT.OPERATION])

)

(PUTPROPS SKETCH.CREATE ARGNAMES (NIL (NAME . DEFAULTS&VALUES) . U))

;; fns for implementing copy and delete functions under keyboard control.

(DEFINEQ
```

(SK.COPY.BUTTONEVENTFN

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

          (* * handles the button event when a copy key and/or the delete is held down.
          allows the user to select a group of the sketch elements from the sketch WINDOW.
          This is very similar to SK.SELECT.MULTIPLE.ITEMS)

          (* the selection protocol is left to add, right to delete. Multiple clicking in the same place upscales for both select and
          deselect. Sweeping will select or deselect all of the items in the swept out area.)

    (COND
       ([AND (TTY.PROCESSP (WINDOWPROP WINDOW 'PROCESS))
             (OR (.MOVEKEYDOWNP.)
                 (AND (.COPYKEYDOWNP.)
                      (.DELETEKEYDOWNP.]                         (* this is going to be a move command.)
        (SELECTQ (fetch (SKETCHCONTEXT SKETCHMOVEMODE) of (WINDOWPROP WINDOW 'SKETCHCONTEXT))
            (POINTS (SK.SEL.AND.MOVE.POINTS WINDOW))
            (SK.SEL.AND.MOVE WINDOW)))
       ((LASTMOUSESTATE (NOT UP))
        (PROG ((COPYMODE (OR (.COPYKEYDOWNP.)
                             (.MOVEKEYDOWNP.)))
               [DELETEMODE (AND (TTY.PROCESSP (WINDOWPROP WINDOW 'PROCESS))
                                (OR (.DELETEKEYDOWNP.)
                                    (.MOVEKEYDOWNP.]
               HOTSPOTCACHE
               (SCALE (VIEWER.SCALE WINDOW))
               OLDX ORIGX NEWX NEWY OLDY ORIGY MOVEDMUCHFLG SELITEMS RETURNVAL PREVMOUSEBUTTONS NOW
               MIDDLEONLYFLG OPERATION)
              [SETQ OPERATION (COND
                                 [COPYMODE (COND
                                             [(TTY.PROCESSP (WINDOWPROP WINDOW 'PROCESS))
                                                                 (* this is not a copy select operation)
                                              (COND
                                                 (DELETEMODE 'MOVE)
                                                 (T 'COPY]
                                             (T 'COPYSELECT]
                                 (DELETEMODE 'DELETE)
                                 (T                              (* keys aren't still down.)
                                    (RETURN]                     (* create the cache for the elements that allow the current
                                                                 operation.)
              (SETQ HOTSPOTCACHE (SK.HOTSPOT.CACHE.FOR.OPERATION WINDOW OPERATION))
              (COND
                 ((NOT (SK.HAS.SOME.HOTSPOTS HOTSPOTCACHE))      (* no items don't do anything.)
                  (RETURN)))
              (TOTOPW WINDOW)
              (SK.PUT.MARKS.UP WINDOW HOTSPOTCACHE)
              [STATUSPRINT WINDOW "
                     " "Select elements to " (COND
                                               [COPYMODE (COND
                                                           (DELETEMODE 'MOVE)
                                                           (T 'COPY]
                                               (DELETEMODE 'DELETE]
```

```
                                                        (* no selections have been made at this point.)
    STARTOVERLP
         (GETMOUSESTATE)
         (COND
            ((AND (LASTMOUSESTATE UP)
                  (SK.BUTTONEVENT.OVERP COPYMODE DELETEMODE))
             (SK.TAKE.MARKS.DOWN WINDOW HOTSPOTCACHE)
             (RETURN)))                                 (* MIDDLEONLYFLG is used to note case of picking characters
    out of a sketch.)
         (SETQ MIDDLEONLYFLG (LASTMOUSESTATE (ONLY MIDDLE)))
    SELECTLP
         (GETMOUSESTATE)
         (COND
            ((SK.BUTTONEVENT.OVERP COPYMODE DELETEMODE)  (* user let up copy key. Put sketch into input buffer.)
             (SETQ RETURNVAL (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))
             (GO EXIT))
            ([AND (LASTMOUSESTATE (NOT UP))
                  (OR (NOT (INSIDEP (WINDOWPROP WINDOW 'REGION)
                                    LASTMOUSEX LASTMOUSEY))
                      (NOT (SK.BUTTONEVENT.SAME.KEYS COPYMODE DELETEMODE]

    (* if a button is down, and either the keystate is different from entry or the cursor is out of the window, stop this event.)

             (SETQ RETURNVAL NIL)
             (GO EXIT)))                                (* cursor is still inside or buttons are up, leave sketch selected.)
         (SETQ NEWY (LASTMOUSEY WINDOW))
         (SETQ NEWX (LASTMOUSEX WINDOW))
         (COND
            ((NEQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS)

    (* a button has gone up or down, mark this as the origin of a new box to sweep.)

             (SETQ ORIGX NEWX)
             (SETQ ORIGY NEWY)
             (COND
                [(AND (EQ PREVMOUSEBUTTONS 0)
                      (NULL MOVEDMUCHFLG)
                      NOW)                              (* user double clicked and an element was selected.)
                 (SETQ NOW)
                 (COND
                    [[OR (AND (LASTMOUSESTATE (ONLY LEFT))
                              (NOT (SETQ MIDDLEONLYFLG)))
                         (AND MIDDLEONLYFLG (LASTMOUSESTATE (ONLY MIDDLE]
                                                        (* select the whole document.)
                     (for SELITEM in (LOCALSPECS.FROM.VIEWER WINDOW) do (SK.ADD.SELECTION SELITEM WINDOW
                                                                                (SK.BUTTONEVENT.MARK
                                                                                    COPYMODE DELETEMODE]
                    (T                                  (* thing selected is a the whole sketch, clear everything and start
                       over.)
                       (for SELITEM in (LOCALSPECS.FROM.VIEWER WINDOW)
                          do (SK.REMOVE.SELECTION SELITEM WINDOW (SK.BUTTONEVENT.MARK COPYMODE
                                                                                      DELETEMODE)))
                                                        (* set PREVMOUSEBUTTONS to cause reinitialization.)
                       (SETQ PREVMOUSEBUTTONS)
                       (GO STARTOVERLP]
                [(LASTMOUSESTATE (NOT UP))

    (* add or delete the element if any that the point is in. This uses a different method which takes into account the size of the
    selection knots which the area sweep doesn't.)

                 (COND
                    ((SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE (create POSITION
                                                                    XCOORD _ NEWX
                                                                    YCOORD _ NEWY)))
                     (COND
                        ([OR (AND (LASTMOUSESTATE (ONLY LEFT))
                                  (NOT (SETQ MIDDLEONLYFLG)))
                             (AND MIDDLEONLYFLG (LASTMOUSESTATE (ONLY MIDDLE]
                                                        (* left or middle selects.)
                         (SK.ADD.SELECTION NOW WINDOW (SK.BUTTONEVENT.MARK COPYMODE DELETEMODE)))
                        ((LASTMOUSESTATE RIGHT)         (* right cause deselect.)
                         (SK.REMOVE.SELECTION NOW WINDOW (SK.BUTTONEVENT.MARK COPYMODE DELETEMODE]
                    (T (SETQ MOVEDMUCHFLG)))
                 (SETQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS))
                ((COND
                    (MOVEDMUCHFLG (OR (NEQ OLDX NEWX)
                                      (NEQ OLDY NEWY)))
                    ((OR (IGREATERP (IABS (IDIFFERENCE ORIGX NEWX))
                                    SK.NO.MOVE.DISTANCE)
                         (IGREATERP (IABS (IDIFFERENCE ORIGY NEWY))
                                    SK.NO.MOVE.DISTANCE))   (* make the first pick move further so that it is easier to multiple
                     click.)
                     (SETQ MOVEDMUCHFLG T)))            (* cursor has moved more than the minimum amount since last
                                                        noticed.)
                                                        (* add or delete any with in the swept out area.)
                 (SK.UPDATE.EVENT.SELECTION HOTSPOTCACHE ORIGX ORIGY NEWX NEWY SCALE WINDOW COPYMODE
                        DELETEMODE)))
```

```
            (SETQ OLDX NEWX)
            (SETQ OLDY NEWY)
            (GO SELECTLP)
         EXIT
                                                    (* clear the selections from the window.)
            (for SEL in (WINDOWPROP WINDOW 'SKETCH.SELECTIONS) do (SK.REMOVE.SELECTION SEL WINDOW
                                                                   (SK.BUTTONEVENT.MARK COPYMODE
                                                                          DELETEMODE)))
            (SK.TAKE.MARKS.DOWN WINDOW HOTSPOTCACHE)
            (CLOSEPROMPTWINDOW WINDOW)              (* if middle was the only button used to select, return only the
                                                    text characters.)
            (RETURN (AND RETURNVAL (COND
                                      [(TTY.PROCESSP (WINDOWPROP WINDOW 'PROCESS))
                                                    (* the results will be going to this same window)
                                       (COND
                                          ((AND COPYMODE DELETEMODE)
                                                    (* move the elements)
                                           (SK.MOVE.ELEMENTS RETURNVAL WINDOW))
                                          [COPYMODE (* copy them)
                                                 (COND
                                                    (MIDDLEONLYFLG
                                                          (* if middle only, just get the characters.)
                                                          (COPYINSERT (SK.BUILD.IMAGEOBJ RETURNVAL WINDOW T))
                                                          )
                                                    (T (SK.COPY.ELEMENTS RETURNVAL WINDOW]
                                          (DELETEMODE (* delete them)
                                                 (SK.DELETE.ELEMENT RETURNVAL WINDOW]
                                       (T (COPYINSERT (SK.BUILD.IMAGEOBJ RETURNVAL WINDOW MIDDLEONLYFLG])
```

(SK.BUTTONEVENT.MARK

```
  [LAMBDA (COPYFLG DELETEFLG)                       (* rrb "29-Dec-84 19:02")
                                                    (* returns the mark that should be put on the points when they
                                                    are selected.)

    (COND
       (DELETEFLG (COND
                     (COPYFLG MOVESELECTIONMARK)
                     (T DELETESELECTIONMARK)))
       (T COPYSELECTIONMARK])
```

(SK.BUILD.IMAGEOBJ

```
  [LAMBDA (SCRELTS SKW CHARSONLYFLG)                ; Edited 20-Jun-92 15:28 by rmk:
                                                    (* builds an imageobj from the list of screen elements.)

    (COND
       [CHARSONLYFLG                                (* return only the text characters.)
        (PROG ((TEXTELTS (bind GELT for LOCALSKELT in SCRELTS
                            join (SELECTQ (fetch (GLOBALPART GTYPE) of (SETQ GELT (fetch (SCREENELT GLOBALPART)
                                                                                  of LOCALSKELT)))
                                          (TEXT (LIST (LIST (fetch (TEXT LOCATIONLATLON) of (SETQ GELT
                                                                                                  (fetch (GLOBALPART
                                                                                                         INDIVIDUALGLOBALPART
                                                                                                         )
                                                                                                    of GELT)))
                                                            GELT)))
                                          (TEXTBOX (LIST (LIST (SK.TEXTBOX.TEXT.POSITION (SETQ GELT
                                                                                                (fetch (GLOBALPART
                                                                                                       INDIVIDUALGLOBALPART
                                                                                                       )
                                                                                                  of GELT)))
                                                               GELT)))
                                          (SKIMAGEOBJ       (* grab the imageobj too.)
                                                      (LIST (LIST (create POSITION
                                                                          XCOORD _ [fetch (REGION LEFT)
                                                                                     of (fetch (SKIMAGEOBJ
                                                                                               SKIMOBJ.GLOBALREGION
                                                                                               )
                                                                                          of (SETQ GELT
                                                                                                (fetch (GLOBALPART
                                                                                                       INDIVIDUALGLOBALPART
                                                                                                       )
                                                                                                  of GELT]
                                                                          YCOORD _ (fetch (REGION BOTTOM)
                                                                                     of (fetch (SKIMAGEOBJ
                                                                                               SKIMOBJ.GLOBALREGION
                                                                                               )
                                                                                          of GELT)))
                                                                  GELT)))
                                          NIL)))
                CHARSLST)                           (* sort according to top from the left.)
               [SORT TEXTELTS (FUNCTION (LAMBDA (A B)
                                   (COND
                                      [(GREATERP (fetch (POSITION YCOORD) of (SETQ A (CAR A)))
                                                 (fetch (POSITION YCOORD) of (SETQ B (CAR B]
                                      ((EQUAL (fetch (POSITION YCOORD) of A)
                                              (fetch (POSITION YCOORD) of B))
                                       (LESSP (fetch (POSITION XCOORD) of A)
```

```
                                                    (fetch (POSITION XCOORD) of B]
          (RETURN (COND
                    ((EQUAL [CAR (LAST (SETQ CHARSLST (for TEXTELT in TEXTELTS
                                                       join
                                                         (* collect relevant parts.)
                                                         (COND
                                                            [(EQ 'SKIMAGEOBJ (fetch (
                                                                                INDIVIDUALGLOBALPART
                                                                                GTYPE)
                                                                             of (CADR TEXTELT)))

       (* copy image object so that copyfn is called. This also copies the part of the image object that are sketch relevent
       unnecessarily but it keeps copyfn call in one place.)

                                                              (LIST (COPY.IMAGE.OBJECT
                                                                     (fetch (SKIMAGEOBJ SKIMAGEOBJ)
                                                                        of (CADR TEXTELT]
                                                             (T (SK.ADD.SPACES (fetch (TEXT
                                                                                   LISTOFCHARACTERS
                                                                                      )
                                                                               of (CADR TEXTELT]
                                 "
                                 ")                                     (* strip off the trailing EOL that was added.)
                     (BUTLAST CHARSLST))
                    (T CHARSLST]
     [(AND (NOT (CDR SCRELTS))
           (EQ (fetch (GLOBALPART GTYPE) of (fetch (SCREENELT GLOBALPART) of (CAR SCRELTS)))
               'SKIMAGEOBJ))

      ;; RMK:  singelton imageobject.  Return an unencapsulated copy of it.  Don't need to worry about sketch transformations that might have
      ;; applied, since they don't affect imageobjects.

      (COPY.IMAGE.OBJECT (fetch (SKIMAGEOBJ SKIMAGEOBJ) of (FETCH (GLOBALPART INDIVIDUALGLOBALPART)
                                                              OF (fetch (SCREENELT GLOBALPART)
                                                                    of (CAR SCRELTS]
     (T
       (* return a sketch image object. The sketch is translated to bring its lower left coordinate to 0,0 so that when it is put in a
       document it is in a canonical place. Maybe don't need to do this anymore.)

       (SKETCH.IMAGEOBJ [create SKETCH using (INSURE.SKETCH SKW)
                                 SKETCHNAME _ NIL SKETCHELTS _
                                 (SK.SORT.GELTS.BY.PRIORITY
                                  (bind GELT for LOCALSKELT in SCRELTS
                                     collect (COND
                                                ((EQ (fetch (GLOBALPART GTYPE)
                                                        of (SETQ GELT (fetch (SCREENELT GLOBALPART)
                                                                         of LOCALSKELT)))
                                                     'SKIMAGEOBJ)
                                                     (* apply copy fn)
                                                 (SK.COPY.IMAGEOBJ GELT))
                                                (T (COPY GELT]
              (SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS SCRELTS (VIEWER.SCALE SKW))
              (VIEWER.SCALE SKW)
              (SK.GRIDFACTOR SKW])
```

```
(SK.BUTTONEVENT.OVERP
  [LAMBDA (COPYMODE DELETEMODE)                                        (* rrb " 1-Feb-85 18:39")

       (* determines if this button event is over by looking at the keys that are held down.
       COPYMODE and DELETEMODE indicate the keystate at the entry point.)

    (COND
       [DELETEMODE (AND (NOT (OR (.DELETEKEYDOWNP.)
                                 (.MOVEKEYDOWNP.)))
                        (OR (NULL COPYMODE)
                            (NULL (OR (.COPYKEYDOWNP.)
                                      (.MOVEKEYDOWNP.]
       (COPYMODE (NULL (.COPYKEYDOWNP.])
```

```
(SK.BUTTONEVENT.SAME.KEYS
  [LAMBDA (COPYMODE DELETEMODE)                                        (* rrb " 1-Feb-85 18:39")

       (* determines if the same keys are held down now as were held down at the start.
       If not, the event will be stopped. COPYMODE and DELETEMODE indicate the keystate at the entry point.)

    (COND
       [DELETEMODE (AND (OR (.DELETEKEYDOWNP.)
                            (.MOVEKEYDOWNP.))
                        (EQ COPYMODE (OR (.COPYKEYDOWNP.)
                                         (.MOVEKEYDOWNP.]
       (COPYMODE                                                        (* if we are not in delete mode, ignore the state of the delete
                                                                        key.)
              (.COPYKEYDOWNP.])
)
```

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS .DELETEKEYDOWNP. MACRO [NIL (OR (KEYDOWNP 'CTRL)
                                          (KEYDOWNP 'DELETE])

(PUTPROPS .MOVEKEYDOWNP. MACRO (NIL (KEYDOWNP 'MOVE)))
)

;; fns for implementing the CHANGE command.

(DEFINEQ

(SK.SEL.AND.CHANGE
  [LAMBDA (W)                                                    (* rrb "10-Dec-85 17:07")
                                                                 (* allows the user to select some elements and changes them.)
    (SK.CHANGE.THING (SK.SELECT.MULTIPLE.ITEMS W T NIL 'CHANGE)
          W])

(SK.CHECK.WHENCHANGEDFN
  [LAMBDA (VIEWER GELT PROPERTY NEWVALUE OLDVALUE)               (* rrb " 3-Jan-86 18:36")

          (* checks if the sketch has a whenchange fn and if so, calls it and interprets the result.
          Returns NIL if the change shouldn't be made.)

    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           RESULT WHENCHANGEDFN)
          (COND
             ([NULL (SETQ WHENCHANGEDFN (GETSKETCHPROP SKETCH 'WHENCHANGEDFN]
              (RETURN GELT)))
          (SETQ RESULT (APPLY* WHENCHANGEDFN VIEWER GELT PROPERTY NEWVALUE OLDVALUE))
          (COND
             ((EQ RESULT 'DON'T)
              (RETURN NIL))
             (T (RETURN GELT])

(SK.CHECK.PRECHANGEFN
  [LAMBDA (VIEWER SCRELT CHANGESPEC)                             (* rrb "27-Jun-86 15:51")

          (* checks if the sketch has a prechange fn and if so, calls it and interprets the result.
          Returns NIL if the change shouldn't be made.)

    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           PRECHANGEFN)
          (COND
             ((SETQ PRECHANGEFN (GETSKETCHPROP SKETCH 'PRECHANGEFN))
              (RETURN (APPLY* PRECHANGEFN VIEWER (fetch (SCREENELT GLOBALPART) of SCRELT)
                              CHANGESPEC])

(SK.CHANGE.ELT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:46")
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.CHANGE (KWOTE W))
          W])

(SK.CHANGE.THING
  [LAMBDA (ELTSTOCHANGE W)                                       (* rrb " 6-Jan-85 19:23")
                                                                 (* ELTSTOCHANGE is a sketch element that was selected for a
    CHANGE operation.)                                           (* Change according to the first one on the list)
    (PROG (FIRSTTYPE READCHANGEFN)                               (* find the first thing that has a change function.)
          (OR (for ELT in ELTSTOCHANGE when (AND [SETQ READCHANGEFN (SK.READCHANGEFN (SETQ FIRSTTYPE
                                                                                          (fetch (SCREENELT GTYPE)
                                                                                             of ELT]
                                                    (NEQ READCHANGEFN 'NILL))
                 do (RETURN T))
              (RETURN))
          (RETURN (SK.APPLY.CHANGE.COMMAND (SK.CHANGEFN FIRSTTYPE)
                        (APPLY* READCHANGEFN W ELTSTOCHANGE)
                        ELTSTOCHANGE W])

(SKETCH.CHANGE.ELEMENTS
  [LAMBDA (ELEMENTS CHANGESPECS SKETCHTOUPDATE ADDHISTORY?)      (* rrb " 2-Oct-86 16:38")

          (* Changes the elements ELEMENTS according to the change specifications CHANGESPECs.
          If SKETCHTOUPDATE is a viewer or a sketch. it will be updated.
          If ADDHISTORY is non-NIL, the changes will be added to the history list of SKETCHTOUPDATE which should be a viewer.
          CHANGESPECs can be a list of the line, brush, text or arc properties, e.g.
          ((TEXT BOLD) (SIZE LARGER) (DASHING (3 1 2 1))%. The changes will be applied to any elements for which they make
          sense.))

    (PROG ((VIEWER (SK.VIEWER.FROM.SKETCH.ARG SKETCHTOUPDATE))
           RESULT)
          (RETURN (SK.DO.AND.RECORD.CHANGES (for ELEMENT in ELEMENTS when (SETQ RESULT (SK.DO.CHANGESPECS
                                                                                          ELEMENT CHANGESPECS
```

```
                                                                                              VIEWER))
                                                       collect RESULT)
                         VIEWER NIL NIL (NULL ADDHISTORY?])

(SK.APPLY.SINGLE.CHANGEFN
  [LAMBDA (GELEMENT CHANGEFN CHANGESPEC VIEWER)                    (* rrb " 2-Oct-86 10:49")

        (* applies a single change to an element. It returns a change structure that contains the old and new elements.)

    (COND
       ((EQ (fetch (GLOBALPART GTYPE) of GELEMENT)
            'GROUP)                                                (* handle a group by propagating it)
        (SK.GROUP.CHANGEFN GELEMENT CHANGEFN CHANGESPEC VIEWER))
       (T (APPLY* CHANGEFN GELEMENT CHANGESPEC VIEWER])

(SK.DO.CHANGESPECS
  [LAMBDA (ELEMENT CHANGESPECS VIEWER)                             (* rrb " 2-Oct-86 16:31")

        (* returns a change structure that is the combined effects of applying all CHANGESPECS to ELEMENT.)
                                                                   (* for now, pretty kludgy)
    (PROG (NEWELEMENT)
          (COND
             ((NULL CHANGESPECS)
              NIL))
          (for CHANGESPEC in CHANGESPECS do (SETQ NEWELEMENT (OR (SK.DO.CHANGESPEC1 (COND
                                                                                       (NEWELEMENT
                                                                                        (fetch (
                                                                                            SKHISTORYCHANGESPEC
                                                                                             NEWELT)
                                                                                           of NEWELEMENT))
                                                                                       (T
                                                                   (* before one of the change specs applies, use the original
                                                                   element.)
                                                                                          ELEMENT))
                                                                         CHANGESPEC VIEWER)
                                                                 NEWELEMENT)))
          (RETURN (AND NEWELEMENT (create SKHISTORYCHANGESPEC
                                          OLDELT _ ELEMENT
                                          NEWELT _ (fetch (SKHISTORYCHANGESPEC NEWELT) of NEWELEMENT)
                                          PROPERTY _ CHANGESPECS])

(SK.VIEWER.FROM.SKETCH.ARG
  [LAMBDA (SKETCH)                                                 (* rrb " 2-Oct-86 10:57")

        (* returns the viewer that changes should be reflected in when SKETCH is passed in as a sketch argument.)

    (COND
       ((NULL SKETCH)
        NIL)
       ((WINDOWP SKETCH))
       ((SETQ SKETCH (INSURE.SKETCH SKETCH))
        (CAR (ALL.SKETCH.VIEWERS SKETCH])

(SK.DO.CHANGESPEC1
  [LAMBDA (ELEMENT CHANGESPEC VIEWER)                              (* rrb "23-Oct-86 14:21")
                                                                   (* applies a single change spec to a single element.)
    (PROG (CHANGEASPECTFN (CHANGEHOW (CADR CHANGESPEC)))
          (OR (SETQ CHANGEASPECTFN (SELECTQ (CAR CHANGESPEC)
                                       (SIZE (FUNCTION SK.CHANGE.BRUSH.SIZE))
                                       (SHAPE (FUNCTION SK.CHANGE.BRUSH.SHAPE))
                                       (ARROW (FUNCTION SK.CHANGE.ARROWHEAD))
                                       (FILLING (FUNCTION SK.CHANGE.FILLING))
                                       (DASHING (FUNCTION SK.CHANGE.DASHING))
                                       (ANGLE (FUNCTION SK.CHANGE.ANGLE))
                                       (DIRECTION (FUNCTION SK.CHANGE.ARC.DIRECTION))
                                       ((TEXT NEWFONT SETSIZE SAME FAMILY&SIZE)
                                            (SETQ CHANGEHOW CHANGESPEC)
                                            (FUNCTION SK.CHANGE.TEXT))
                                       (ADDPOINT                   (* handle this specially because it shouldn't go inside of a group
                                                 element.)
                                                 (RETURN (SK.ADD.KNOT.TO.ELEMENT ELEMENT CHANGEHOW)))
                                       (BRUSHCOLOR (FUNCTION SK.CHANGE.BRUSH.COLOR))
                                       (FILLINGCOLOR (FUNCTION SK.CHANGE.FILLING.COLOR))
                                       (FILLINGMODE (FUNCTION SK.CHANGE.FILLING.MODE))
                                      NIL))
              (RETURN))
          (RETURN (SK.APPLY.SINGLE.CHANGEFN ELEMENT CHANGEASPECTFN CHANGEHOW VIEWER])

(SK.CHANGEFN
  [LAMBDA (ELEMENTTYPE)                                            (* rrb " 8-Jan-86 17:15")

        (* returns the changefn for an element. The only one that isnt SK.ELEMENTS.CHANGEFN is image objects.)
```

```
                                                        (* the changefn should return a list of
                                                        SKHISTORYCHANGESPEC instances.)
    (OR (fetch (SKETCHTYPE CHANGEFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE))
        (FUNCTION SK.DEFAULT.CHANGEFN])
```

(SK.READCHANGEFN

```
  [LAMBDA (ELEMENTTYPE)                                 (* rrb " 6-Jan-85 18:29")
```

(* used to be (OR & (FUNCTION SK.DEFAULT.CHANGEFN)) If this really isn't necessary, clean out SK.DEFAULT.CHANGEFN and all the things only it calls. If it is necessary, update it to include a readchangefn.)

```
    (fetch (SKETCHTYPE READCHANGEFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.DEFAULT.CHANGEFN

```
  [LAMBDA (SCRNELT W FIELD)                             (* rrb "14-May-84 15:57")
    (PROG ([FIELD (OR FIELD (SK.MENU.AND.RETURN.FIELD (fetch (SCREENELT GTYPE) of SCRNELT]
           (INDVELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of SCRNELT))
           (NOSETVALUE "str")
           CURRENTVAL NEWPROPVALUE FIELDNAME)
          (COND
             ((NULL FIELD)
              (STATUSPRINT W "That element doesn't have any changeable parts.")
              (RETURN NIL)))
          (SETQ CURRENTVAL (RECORDACCESS (SETQ FIELDNAME (COND
                                                            ((LISTP FIELD)
                                                             (CAR FIELD))
                                                            (T FIELD)))
                                         INDVELT
                                         (RECLOOK (fetch (SCREENELT GTYPE) of SCRNELT))
                                         'FETCH))
          [COND
             ((LISTP FIELD)                             (* cadr is queryfunction which can do special input and return
                                                        value checking.)
              (SETQ NEWPROPVALUE (APPLY* (CADR FIELD)
                                         SCRNELT FIELD W NOSETVALUE)))
             (T                                         (* have NIL returned be no change.)
                (SETQ NEWPROPVALUE (OR (READ.FUNCTION [CONCAT "Enter new " (MKSTRING FIELD)
                                                              " value.  Current value is "
                                                              (MKSTRING (RECORDACCESS FIELD INDVELT
                                                                                      (RECLOOK (fetch (SCREENELT GTYPE)
                                                                                                  of SCRNELT))
                                                                                      'FETCH]
                                                       W)
                                                      NOSETVALUE]
          (OR (EQ NEWPROPVALUE NOSETVALUE)
              (RECORDACCESS FIELDNAME INDVELT (RECLOOK (fetch (SCREENELT GTYPE) of SCRNELT))
                     'REPLACE
                     (EVAL NEWPROPVALUE)))
          (RETURN (fetch (SCREENELT GLOBALPART) of SCRNELT])
```

(CHANGEABLEFIELDITEMS

```
  [LAMBDA (ELEMENTTYPE)                                 (* rrb "11-May-84 15:49")
```

(* returns the list of fields that element type allows to change. Each field should be of the form (FIELDNAMELABEL (QUOTE (FIELDNAME QUERYFN)) "helpstring") -
QUERYFN should be a function of four args%: the screen element being changed, the "field" returned from this function, the window the sketch is being displayed in, and a value to be returned if no change should be made.)

```
    (GETPROP ELEMENTTYPE 'CHANGEABLEFIELDITEMS])
```

(SK.APPLY.CHANGE.COMMAND

```
  [LAMBDA (CHANGEFN COMMAND SCRELTS SKW)                (* rrb "24-Sep-86 16:23")
                                                        (* applies a change command to the relevant elements in
                                                        SCRELTS.)
    (AND COMMAND (SK.DO.AND.RECORD.CHANGES (bind ELTCHANGE for SCRELT in SCRELTS
                                              when (SETQ ELTCHANGE (SK.APPLY.CHANGE.COMMAND1 CHANGEFN
                                                                                             COMMAND SCRELT SKW))
                                              collect ELTCHANGE)
                         SKW])
```

(SK.DO.AND.RECORD.CHANGES

```
  [LAMBDA (LSTOFCHANGESPECS VIEWER DONTUPDATEPRIORITYFLG DONTDISPLAYFLG DONTHISTORYFLG)
                                                        (* rrb " 2-Oct-86 16:22")
```

(* accepts a list of change specs and actually updates the sketch, viewer and history list.)

```
    (COND
       (LSTOFCHANGESPECS [SETQ LSTOFCHANGESPEC (COND
                                                  (DONTUPDATEPRIORITYFLG
                                                                  (* priority of new ones won't change, order by them.)
                                                         (SORT.CHANGESPECS.BY.NEW.PRIORITY LSTOFCHANGESPECS))
                                                  (T
```

```
        (* order so that new priorities are assigned in the same relative order as the old ones.)

                                                    (SORT.CHANGESPECS.BY.OLD.PRIORITY LSTOFCHANGESPECS]
            (SK.UPDATE.ELEMENTS LSTOFCHANGESPECS VIEWER DONTUPDATEPRIORITYFLG DONTDISPLAYFLG)
            (OR DONTHISTORYFLG (SK.ADD.HISTEVENT 'CHANGE LSTOFCHANGESPECS VIEWER))
            T])
```

(SK.APPLY.CHANGE.COMMAND1

```
  [LAMBDA (CHANGEFN COMMAND SCRELT VIEWER)                    (* rrb "27-Jun-86 15:48")

        (* applies a change command to a single screen element. Does the prechangefn and whenchangefn checks.)

    (PROG (FNRESULT CHANGES)
          (COND
             ((EQ (SETQ FNRESULT (SK.CHECK.PRECHANGEFN VIEWER SCRELT COMMAND))
                  'DON'T)
              (RETURN NIL))
             ((LISTP FNRESULT)                                (* result was a different change specification.)
              (SETQ COMMAND FNRESULT)))

          (* code was written to take a list but since prechangefn can change things at the elements level, every element is done
          individually.)

          (OR (SETQ CHANGES (APPLY* CHANGEFN (LIST SCRELT)
                                    VIEWER COMMAND))
              (RETURN))
          (SETQ CHANGES (CAR CHANGES))
          (RETURN (AND (SK.CHECK.WHENCHANGEDFN VIEWER (fetch (SKHISTORYCHANGESPEC OLDELT) of CHANGES)
                               (fetch (SKHISTORYCHANGESPEC PROPERTY) of CHANGES)
                               (fetch (SKHISTORYCHANGESPEC NEWVALUE) of CHANGES)
                               (fetch (SKHISTORYCHANGESPEC OLDVALUE) of CHANGES))
                       CHANGES])
```

(SK.ELEMENTS.CHANGEFN

```
  [LAMBDA (SCRELTS SKW HOW)                                   (* rrb " 2-Oct-86 16:18")
                                                              (* changefn for many sketch elements.)
    (PROG (CHANGEASPECTFN (CHANGEHOW (CADR HOW)))
          (OR (SETQ CHANGEASPECTFN (SELECTQ (CAR HOW)
                                            (SIZE (FUNCTION SK.CHANGE.BRUSH.SIZE))
                                            (SHAPE (FUNCTION SK.CHANGE.BRUSH.SHAPE))
                                            (ARROW (FUNCTION SK.CHANGE.ARROWHEAD))
                                            (FILLING (FUNCTION SK.CHANGE.FILLING))
                                            (DASHING (FUNCTION SK.CHANGE.DASHING))
                                            (ANGLE (FUNCTION SK.CHANGE.ANGLE))
                                            (DIRECTION (FUNCTION SK.CHANGE.ARC.DIRECTION))
                                            ((TEXT NEWFONT SETSIZE SAME FAMILY&SIZE)
                                                 (SETQ CHANGEHOW HOW)
                                                 (FUNCTION SK.CHANGE.TEXT))
                                            (ADDPOINT                (* handle this specially because it only works on the first
                                                                     element.)
                                                      (RETURN (LIST (SK.ADD.KNOT.TO.ELEMENT (fetch (SCREENELT
                                                                                                   GLOBALPART)
                                                                                                of (CAR SCRELTS))
                                                                                            CHANGEHOW))))
                                            (BRUSHCOLOR (FUNCTION SK.CHANGE.BRUSH.COLOR))
                                            (FILLINGCOLOR (FUNCTION SK.CHANGE.FILLING.COLOR))
                                            (FILLINGMODE (FUNCTION SK.CHANGE.FILLING.MODE))
                                            NIL))
              (RETURN))
          (RETURN (for SCRELT in SCRELTS collect (SK.APPLY.SINGLE.CHANGEFN (fetch (SCREENELT GLOBALPART)
                                                                                of SCRELT)
                                                                           CHANGEASPECTFN CHANGEHOW SKW])
```

(READ.POINT.TO.ADD

```
  [LAMBDA (SCRELT SKVIEWER)                                   (* rrb "20-May-86 10:52")

        (* asks where a point should be added and where it should be.
        Return a list (AfterPt NewPt))

    (PROG (AFTERPT NEWPT)
          (STATUSPRINT SKVIEWER "Select the point that the new point should follow.")
          (OR (SETQ AFTERPT (SK.SELECT.ITEM SKVIEWER NIL (LIST SCRELT)))
              (PROGN (CLOSEPROMPTWINDOW SKVIEWER)
                     (RETURN)))
          (STATUSPRINT SKVIEWER "Indicate where the new point should be.")
          (SETQ NEWPT (SK.READ.POINT.WITH.FEEDBACK SKVIEWER POINTREADINGCURSOR NIL NIL NIL NIL
                              SKETCH.USE.POSITION.PAD))
          (CLOSEPROMPTWINDOW SKVIEWER)
          (AND NEWPT (RETURN (LIST (GLOBAL.KNOT.FROM.LOCAL AFTERPT SCRELT)
                                   (SK.MAP.INPUT.PT.TO.GLOBAL NEWPT SKVIEWER])
```

(GLOBAL.KNOT.FROM.LOCAL

```
  [LAMBDA (LOCALKNOT SCRELT)                                        (* rrb "20-Nov-85 11:05")
                                                                    (* returns the global knot that corresponds to a local one.)
    (for LKNOT in (fetch (SCREENELT HOTSPOTS) of SCRELT) as GKNOT in (GETSKETCHELEMENTPROP (fetch (SCREENELT
                                                                                                   GLOBALPART)
                                                                                            of SCRELT)
                                                                                    'DATA)
       when (EQUAL LKNOT LOCALKNOT) do (RETURN GKNOT])
```

(SK.ADD.KNOT.TO.ELEMENT

```
  [LAMBDA (GELTWITHKNOTS PTS)                                       (* rrb "16-Jan-86 12:23")
                                                                    (* adds a point to a knot element.
                                                                    The point (CADR PTS) is added after
                                                                    (CAR PTS))
    (PROG ((OLDKNOTS (GETSKETCHELEMENTPROP GELTWITHKNOTS 'DATA))
           NEWKNOTS)
          [SETQ NEWKNOTS (for KNOT in OLDKNOTS join (COND
                                                      ((EQUAL KNOT (CAR PTS))
                                                       (LIST KNOT (CADR PTS)))
                                                      (T (LIST KNOT]
          (RETURN (create SKHISTORYCHANGESPEC
                          NEWELT _ (SK.CHANGE.ELEMENT.KNOTS GELTWITHKNOTS NEWKNOTS)
                          OLDELT _ GELTWITHKNOTS
                          PROPERTY _ 'DATA
                          NEWVALUE _ NEWKNOTS
                          OLDVALUE _ OLDKNOTS])
```

(SK.GROUP.CHANGEFN

```
  [LAMBDA (GROUPELT CHANGEASPECTFN CHANGEHOW SKW)                   (* rrb "10-Jan-86 12:15")

          (* maps a change function through all the elements of a group and returns a change spec event if it takes on any of them.)

    (PROG (NEWELT)
          (SETQ NEWELT (SK.GROUP.CHANGEFN1 GROUPELT CHANGEASPECTFN CHANGEHOW SKW))
          (OR NEWELT (RETURN))
          (RETURN (create SKHISTORYCHANGESPEC
                          NEWELT _ NEWELT
                          OLDELT _ GROUPELT
                          PROPERTY _ 'DATA
                          NEWVALUE _ (fetch (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                                of NEWELT))
                          OLDVALUE _ (fetch (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                                of GROUPELT])
```

(SK.GROUP.CHANGEFN1

```
  [LAMBDA (GROUPELT CHANGEASPECTFN CHANGEHOW SKW)                   (* rrb "27-Jun-86 16:19")

          (* maps a change function through all the elements of a group and returns a new element if it takes on any of them.)

    (PROG ((OLDSUBELTS (fetch (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT)))
           NEWSUBELTS NEWELT CHANGEDFLG)
          [SETQ NEWSUBELTS (for SUBELT in OLDSUBELTS
                              collect (COND
                                        ([SETQ NEWELT (COND
                                                        ((EQ (fetch (GLOBALPART GTYPE) of SUBELT)
                                                             'GROUP)
                                                                                (* handle a group by propagating it)
                                                         (SK.GROUP.CHANGEFN1 SUBELT CHANGEASPECTFN CHANGEHOW
                                                                SKW))
                                                        (T

          (* individual change functions return a change spec event, pull the new element out of it.
          This throws aways a lot of information about what was changed but I don't know any good way to save it so that it can be
          passed on undoing so don't save it.)

                                                                  (fetch (SKHISTORYCHANGESPEC NEWELT)
                                                                     of (APPLY* CHANGEASPECTFN SUBELT CHANGEHOW SKW]
                                                         (SETQ CHANGEDFLG T)
                                                         NEWELT]
          (OR CHANGEDFLG (RETURN))
          [SETQ NEWSUBELTS (for OLDSUBELT in OLDSUBELTS as NEWSUBELT in NEWSUBELTS
                              collect                               (* copy any unchanged elements so that user programs don't
                                                                    have to worry about them.)
                                                 (OR NEWSUBELT (SK.COPY.GLOBAL.ELEMENT OLDSUBELT]
          (RETURN (SK.UPDATE.GROUP.AFTER.CHANGE (create GLOBALPART
                                                        COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                                  of GROUPELT)
                                                        INDIVIDUALGLOBALPART _ (create GROUP
                                                                                       using (fetch (GLOBALPART
                                                                                                     INDIVIDUALGLOBALPART
                                                                                                     )
                                                                                              of GROUPELT)
                                                                                       LISTOFGLOBALELTS _
                                                                                       NEWSUBELTS])
```

```
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SKHISTORYCHANGESPEC (OLDELT NEWELT PROPERTY NEWVALUE OLDVALUE))
)
)
```

;; fns for adding elements

```
(DEFINEQ
```

(ADD.ELEMENT.TO.SKETCH

```
  [LAMBDA (GELT SKETCH)                                          (* rrb "23-Jun-87 13:29")
                                                                 (* changes the global sketch)
    (PROG [(REALSKETCH (INSURE.SKETCH SKETCH))
           (ELTPRI (\GETSKETCHELEMENTPROP1 GELT 'PRI]
          [COND
             ((EQ (fetch (GLOBALPART GTYPE) of GELT)
                  'SKIMAGEOBJ)                                   (* call the wheninsertedfn for this imageobj if there is one.)
              (PROG ((IMOBJ (fetch (SKIMAGEOBJ SKIMAGEOBJ) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                      of GELT)))
                     DATUM)
                    (COND
                       ((AND (SETQ DATUM (IMAGEOBJPROP IMOBJ 'WHENINSERTEDFN))
                             (NEQ DATUM 'NILL))                  (* call the image objects insertfn.)
                        (APPLY* DATUM IMOBJ (SK.VIEWER.FROM.SKETCH.ARG SKETCH)
                               NIL SKETCH)))
                    (RETURN]
          (COND
             ((NULL ELTPRI)                                      (* give the element a priority and put it at the end)
              (SK.SET.ELEMENT.PRIORITY GELT (SK.POP.NEXT.PRIORITY REALSKETCH))
              (TCONC (fetch (SKETCH SKETCHTCELL) of REALSKETCH)
                     GELT))
             (T (SK.ADD.PRIORITY.ELEMENT.TO.SKETCH SKETCH GELT ELTPRI)))
          (SK.MARK.DIRTY REALSKETCH])
```

(ADD.SKETCH.VIEWER

```
  [LAMBDA (SKETCH VIEWER)                                        (* rrb " 8-APR-83 17:56")
                                                                 (* adds VIEWER as a viewer of SKETCH.)
    (PROG (VIEWERS)
          (COND
             ((SETQ VIEWERS (ALL.SKETCH.VIEWERS SKETCH))         (* already has at least one viewer)
              (OR (FMEMB VIEWER VIEWERS)
                  (NCONC1 VIEWERS VIEWER)))
             (T                                                  (* doesn't have any viewers yet.)
                (SETQ ALL.SKETCHES (CONS (LIST SKETCH VIEWER)
                                         ALL.SKETCHES])
```

(REMOVE.SKETCH.VIEWER

```
  [LAMBDA (SKETCH VIEWER)                                        (* rrb "26-Apr-85 16:56")
                                                                 (* removes VIEWER as a viewer of SKETCH.)
    (PROG (VIEWERS)
          (COND
             ((SETQ VIEWERS (VIEWER.BUCKET SKETCH))              (* remove it from the list.)
              (COND
                 ((NULL (CDR (DREMOVE VIEWER VIEWERS)))          (* deleted the last viewer.)
                  (SETQ ALL.SKETCHES (REMOVE VIEWERS ALL.SKETCHES])
```

(ALL.SKETCH.VIEWERS

```
  [LAMBDA (SKETCH)                                               (* rrb " 8-APR-83 14:20")
                                                                 (* returns the list of all active viewers of a sketch)
    (CDR (VIEWER.BUCKET SKETCH])
```

(SKETCH.ALL.VIEWERS

```
  [LAMBDA (SKETCH)                                               (* returns all of the viewers onto a sketch.)
    (ALL.SKETCH.VIEWERS (INSURE.SKETCH SKETCH])
```

(VIEWER.BUCKET

```
  [LAMBDA (SKETCH)                                               (* rrb " 8-APR-83 14:20")
    (FASSOC SKETCH ALL.SKETCHES])
```

(ELT.INSIDE.REGION?

```
  [LAMBDA (GLOBALPART WORLDREG)                                  (* rrb " 4-AUG-83 14:51")
                                                                 (* determines if any part of an element is inside the region
                                                                 WORLDREG)
    (APPLY* (SK.INSIDEFN (fetch (GLOBALPART GTYPE) of GLOBALPART))
           GLOBALPART WORLDREG])
```

(ELT.INSIDE.SKWP

```
[LAMBDA (GLOBALPART SKETCHW)                                    (* rrb "25-Nov-85 17:46")
                                                                (* determines if a global element is in the world region of a map
    window.)
   (ELT.INSIDE.REGION? GLOBALPART (SKETCH.REGION.VIEWED SKETCHW])
```

(SCALE.FROM.SKW

```
[LAMBDA (WINDOW)                                                (* rrb "11-MAR-83 11:52")
                                                                (* gets the scale of a sketch window.)
   (WINDOWPROP WINDOW 'SCALE])
```

(SK.ADDELT.TO.WINDOW

```
[LAMBDA (PELT SKETCHW)                                          (* rrb "10-Mar-86 14:56")
                                                                (* adds a picture element to a sketch window.
                                                                Returns the element that was added.)
   (COND
      (PELT (SK.ADD.PRIORITY.LOCAL.ELEMENT.TO.SKETCH (WINDOWPROP SKETCHW 'SKETCHSPECS)
                PELT)
            [PROG ((CACHE (SK.HOTSPOT.CACHE SKETCHW)))
                  (COND
                     (CACHE                                     (* if there is a cache, adding an element will change it)
                           (SK.ADD.HOTSPOTS.TO.CACHE1 PELT CACHE))
                     (T                                         (* if this is the first, must set the window property too.)
                        (SK.SET.HOTSPOT.CACHE SKETCHW (SK.ADD.HOTSPOTS.TO.CACHE1 PELT CACHE]
            PELT])
```

(SK.CALC.REGION.VIEWED

```
[LAMBDA (WINDOW SCALE)                                          (* rrb "29-APR-83 08:37")
                                                                (* returns the region of the sketch visible in window.)
   (UNSCALE.REGION (DSPCLIPPINGREGION NIL WINDOW)
          SCALE])
```

(SK.DRAWFIGURE

```
[LAMBDA (SCREENELT STREAM REGION SCALE)                         (* rrb "30-Aug-84 14:31")

        (* draws an element of a sketch in a window. Makes sure the scale of the current drawing is with in the limits of the element.
        Returns SCREENELT)

   (PROG (GLOBALPART)
         [COND
            ([AND (NUMBERP SCALE)
                  (OR [LESSP SCALE (fetch (COMMONGLOBALPART MINSCALE) of (SETQ GLOBALPART (fetch (SCREENELT
                                                                                                  COMMONGLOBALPART
                                                                                                        )
                                                                                               of SCREENELT]
                      (GREATERP SCALE (fetch (COMMONGLOBALPART MAXSCALE) of GLOBALPART]
                                                                (* scale is out of bounds, don't draw it.)
             NIL)
            (T (SK.DRAWFIGURE1 SCREENELT STREAM (OR REGION (DSPCLIPPINGREGION NIL STREAM]
         (RETURN SCREENELT])
```

(SK.DRAWFIGURE1

```
[LAMBDA (ELT SKW REGION)                                        (* rrb "14-Sep-84 16:59")
                                                                (* displays a sketch element in a window)
   (APPLY* (SK.DRAWFN (fetch (SCREENELT GTYPE) of ELT))
          ELT SKW REGION])
```

(SK.LOCAL.FROM.GLOBAL

```
[LAMBDA (GELT SKSTREAM SCALE)                                   (* rrb "11-Jul-86 15:56")
                                                                (* returns the element instance of the global element GELT
                                                                expanded into the window SKW.)

        (* SKSTREAM can be deleted from call once TEXT.EXPANDFN no longer needs to distinquish INTERPRESS stream from
        windows.)

   (PROG ((SCRELT (APPLY* (SK.EXPANDFN (fetch (GLOBALPART GTYPE) of GELT))
                          GELT
                          (OR (NUMBERP SCALE)
                              (VIEWER.SCALE SKSTREAM))
                          SKSTREAM))
          ACTIVEREGION)                                         (* do the ACTIVEREGION which is common to all elements.)
         [AND SCRELT (SETQ ACTIVEREGION (GETSKETCHELEMENTPROP GELT 'ACTIVEREGION))
              (replace (LOCALPART LOCALHOTREGION) of (fetch (SCREENELT LOCALPART) of SCRELT)
                 with (SK.SCALE.REGION ACTIVEREGION (OR (NUMBERP SCALE)
                                                        (VIEWER.SCALE SKSTREAM]
         (RETURN SCRELT])
```

(SKETCH.REGION.VIEWED

```
[LAMBDA (VIEWER NEWREGION)                                      (* rrb "23-Apr-87 12:20")
```

```
                                                          (* returns the region in sketch coordinates of the area visible in
                                                          SKETCHW.)
    (COND
       [(IMAGEOBJP VIEWER)                                (* it is a sketch image object)
        (PROG ([SK? (LISTP (IMAGEOBJPROP VIEWER 'OBJECTDATUM]
               NEWVIEW)
              (COND
                 [(type? SKETCH (FETCH (SKETCHIMAGEOBJ SKIO.SKETCH) OF SK?))
                  (RETURN (PROG1 (fetch (SKETCHIMAGEOBJ SKIO.REGION) of SK?)
                                 [COND
                                    (NEWREGION (COND
                                                  ((REGIONP NEWREGION)
                                                   (replace (SKETCHIMAGEOBJ SKIO.REGION) of SK? with NEWREGION))
                                                  ((SETQ NEWVIEW (SKETCH.VIEW.FROM.NAME NEWREGION VIEWER))
                                                   (replace (SKETCHIMAGEOBJ SKIO.REGION) of SK? with NEWVIEW))
                                                  ((EQ NEWREGION 'HOME)
                                                                  (* change scale to 1.0 and set lower left of region viewed to
                                                                  (0,0)%.)
                                                   NIL)
                                                  (T              (* HOME and named views aren't supported for image object
                                                                  sketches.)
                                                     (\ILLEGAL.ARG NEWREGION]))]
                  (T (ERROR "not a sketch image object" VIEWER]
       [(WINDOWP VIEWER)
        (PROG1 (WINDOWPROP VIEWER 'REGION.VIEWED)
            [COND
               (NEWREGION (PROG (NEWVIEW)
                                (RETURN (COND
                                           ((REGIONP NEWREGION)
                                            (SKETCH.GLOBAL.REGION.ZOOM VIEWER NEWREGION))
                                           ((EQ NEWREGION 'HOME)
                                            (SKETCH.HOME VIEWER))
                                           ((SETQ NEWVIEW (SKETCH.VIEW.FROM.NAME NEWREGION VIEWER))
                                            (SK.MOVE.TO.VIEW VIEWER NEWVIEW))
                                           (T (\ILLEGAL.ARG NEWREGION])]
       (T (\ILLEGAL.ARG VIEWER])
```

(SKETCH.VIEW.FROM.NAME

```
  [LAMBDA (VIEWNAME SKETCHW)                              (* rrb "25-Nov-85 17:55")
                                                          (* returns the view structure for a view given its name.)
    (for SAVEDVIEW in (GETSKETCHPROP (INSURE.SKETCH SKETCHW)
                                     'VIEWS)
       when (EQUAL VIEWNAME (fetch (SKETCHVIEW VIEWNAME) of SAVEDVIEW)) do (RETURN SAVEDVIEW])
```

(SK.UPDATE.REGION.VIEWED

```
  [LAMBDA (SKW)                                           (* rrb "11-Jul-86 15:51")
                                                          (* updates the REGION.VIEWED property of a window.)
    (WINDOWPROP SKW 'REGION.VIEWED (SK.CALC.REGION.VIEWED SKW (VIEWER.SCALE SKW])
```

(SKETCH.ADD.AND.DISPLAY

```
  [LAMBDA (GELT SKETCHW DONTCLEARCURSOR)                  (* rrb "14-Nov-84 17:12")

          (* adds a new element to a sketch window and handles propagation to all other figure windows)

    (COND
       (GELT (SK.ADD.HISTEVENT 'ADD (LIST GELT)
                               SKETCHW)
             (SK.ADD.ELEMENT GELT SKETCHW DONTCLEARCURSOR])
```

(SKETCH.ADD.AND.DISPLAY1

```
  [LAMBDA (GELT SKETCHW SCALE NODISPLAYFLG)               (* rrb "11-Jul-86 15:51")
                                                          (* displays a sketch element and adds it to the window.)
    (COND
       (GELT (COND
                (NODISPLAYFLG (SK.ADD.ITEM GELT SKETCHW))
                (T (SK.DRAWFIGURE (SK.ADD.ITEM GELT SKETCHW)
                                  SKETCHW NIL (OR SCALE (VIEWER.SCALE SKETCHW])
```

(SK.ADD.ITEM

```
  [LAMBDA (GELT SKETCHW)                                  (* rrb "10-APR-83 13:38")

          (* adds a global element to a window. Returns the local element that was actually added.)

    (SK.ADDELT.TO.WINDOW (SK.LOCAL.FROM.GLOBAL GELT SKETCHW)
          SKETCHW])
```

(SKETCHW.ADD.INSTANCE

```
  [LAMBDA (TYPE SKW)                                      (* rrb "14-Nov-84 17:08")
                                                          (* reads an instance of type TYPE from the user and displays it
    in SKW.)
    (PROG ((ELT (SK.INPUT TYPE SKW)))
```

```
          (AND ELT (SKETCH.ADD.AND.DISPLAY ELT SKW))
          (RETURN ELT])

)

;; fns for deleting things

(DEFINEQ

(SK.SEL.AND.DELETE
  [LAMBDA (W)                                                    (* rrb "10-Dec-85 17:08")
                                                                 (* lets the user select elements and deletes them)
    (SK.DELETE.ELEMENT (SK.SELECT.MULTIPLE.ITEMS W T NIL 'DELETE)
          W])

(SK.ERASE.AND.DELETE.ITEM
  [LAMBDA (SELELT SKW NODISPLAYFLG)                              (* rrb "30-Jul-85 15:36")
                                                                 (* removes a sketch element from a viewer.)
    (COND
       (SELELT (OR NODISPLAYFLG (SK.ERASE.ELT SELELT SKW))
               (SK.DELETE.ITEM SELELT SKW])

(REMOVE.ELEMENT.FROM.SKETCH
  [LAMBDA (GELT SKETCH INSIDEGROUPFLG)                           (* rrb "26-Sep-86 13:24")

          (* changes the global sketch Returns the element or the group element containing the element if the element was found in
          the sketch. If INSIDEGROUPFLG is T, it will go inside of groups.)

    (PROG ((SKETCHDATA (INSURE.SKETCH SKETCH)))
          (COND
             ((DELFROMTCONC (fetch (SKETCH SKETCHTCELL) of SKETCHDATA)
                    GELT)
              (SK.MARK.DIRTY SKETCH)
              (RETURN T))
             [INSIDEGROUPFLG (RETURN (for ELT on (fetch (SKETCH SKETCHELTS) of SKETCHDATA)
                                        do                       (* look inside groups)
                                            (COND
                                               ((DELFROMGROUPELT GELT ELT)
                                                (SK.MARK.DIRTY SKETCH)
                                                (RETURN ELT]
             (T (RETURN NIL])

(SK.DELETE.ELEMENT
  [LAMBDA (ELTSTODEL SKETCHW ELTSFORHISTORY)                     (* rrb "30-Dec-85 16:19")

          (* deletes a list of element to a sketch window and handles propagation to all other figure windows)

    (SKED.CLEAR.SELECTION SKETCHW)
    (AND ELTSTODEL (SK.DELETE.ELEMENT2 (for SCRELT in ELTSTODEL collect (fetch (SCREENELT GLOBALPART) of SCRELT))
                          SKETCHW ELTSFORHISTORY])

(SK.DELETE.ELEMENT2
  [LAMBDA (GELTSTODEL SKETCHW ELTSFORHISTORY)                    (* rrb "30-Dec-85 16:18")

          (* deletes a list of global elements and adds it to the history list depending upon ELTSFORHISTORY)

    (PROG (DELETEDELTS)
          (SETQ DELETEDELTS (SK.CHECK.WHENDELETEDFN SKETCHW GELTSTODEL))
          (OR DELETEDELTS (RETURN))
          (OR (EQ ELTSFORHISTORY 'DON'T)
              (SK.ADD.HISTEVENT 'DELETE (OR ELTSFORHISTORY DELETEDELTS)
                     SKETCHW))
          (for GELT in DELETEDELTS do (SK.DELETE.ELEMENT1 GELT SKETCHW))
          (RETURN DELETEDELTS])

(SK.DELETE.KNOT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:47")
                                                                 (* lets the user select a knot in a curve or wire and deletes it.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.DELETE.KNOT (KWOTE W))
          W])

(SK.SEL.AND.DELETE.KNOT
  [LAMBDA (W)                                                    (* rrb "10-Dec-85 17:03")
                                                                 (* lets the user select a knot and deletes it.)
    (PROG [(KNOTELTS (SUBSET (LOCALSPECS.FROM.VIEWER W)
                             (FUNCTION (LAMBDA (SCRELT)
                                         (AND (MEMB (fetch (SCREENELT GTYPE) of SCRELT)
                                                    '(WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE))
                                              (NOT (SK.ELEMENT.PROTECTED? (fetch (SCREENELT GTYPE) of SCRELT)
                                                          'CHANGE]
          (COND
```

```
            ((NULL KNOTELTS)
             (STATUSPRINT W "There are no curve or wire elements to delete points from.")
             (RETURN)))
          (SK.DELETE.ELEMENT.KNOT (SK.SELECT.ITEM W NIL KNOTELTS)
                KNOTELTS W])
```

(SK.DELETE.ELEMENT.KNOT

```
  [LAMBDA (LOCALKNOT SCRELTS SKW)                                  (* rrb " 9-Jan-86 19:45")
                                                                   (* deletes a knot from a curve or wire element.)
    (SKED.CLEAR.SELECTION SKW)
    (COND
       ((NULL LOCALKNOT))
       ([OR (POSITIONP LOCALKNOT)
            (AND (NULL (CDR LOCALKNOT))
                 (POSITIONP (CAR LOCALKNOT))
                 (SETQ LOCALKNOT (CAR LOCALKNOT]
        (PROG ((SCREENELT (for SKELT in SCRELTS when (MEMBER LOCALKNOT (fetch (SCREENELT HOTSPOTS) of SKELT))
                             do (RETURN SKELT)))
               LOCALKNOTS GLOBALKNOT GLOBALKNOTS NEWKNOTS NEWELT CHANGES GLOBALPART)
              (COND
                 ((NULL SCREENELT)
                  (RETURN NIL)))
              (SETQ GLOBALPART (fetch (SCREENELT GLOBALPART) of SCREENELT))
              (SETQ GLOBALKNOT (for LKNOT in (SETQ LOCALKNOTS (fetch (SCREENELT HOTSPOTS) of SCREENELT))
                                  as GKNOT in (SETQ GLOBALKNOTS (GETSKETCHELEMENTPROP GLOBALPART 'DATA))
                                  when (EQUAL LKNOT LOCALKNOT) do (RETURN GKNOT)))
              (OR (SK.CHECK.WHENPOINTDELETEDFN SKW SCREENELT GLOBALKNOT)
                  (RETURN))
              (RETURN (COND
                         [(SETQ NEWKNOTS (REMOVE GLOBALKNOT GLOBALKNOTS))
                                                                   (* change the knots and update the element)
                          (COND
                             ((SETQ NEWELT (SK.CHANGE.ELEMENT.KNOTS GLOBALPART NEWKNOTS))
                                                                   (* call the when changed fn)
                              (OR (SK.CHECK.WHENCHANGEDFN SKW GLOBALPART 'DATA NEWKNOTS GLOBALKNOTS)
                                  (RETURN))                        (* make history entry and update screen)
                              (SK.UPDATE.ELEMENTS (SETQ CHANGES
                                                        (CONS (create SKHISTORYCHANGESPEC
                                                                      NEWELT _ NEWELT
                                                                      OLDELT _ GLOBALPART
                                                                      PROPERTY _ 'DATA
                                                                      NEWVALUE _ NEWKNOTS
                                                                      OLDVALUE _ GLOBALKNOTS)))
                                    SKW)
                              (SK.ADD.HISTEVENT 'CHANGE CHANGES SKW]
                         (T                                        (* delete the whole element.)
                            (SK.DELETE.ELEMENT (CONS SCREENELT)
                                  SKW])
```

(SK.CHECK.WHENDELETEDFN

```
  [LAMBDA (VIEWER GELTS)                                           (* rrb "30-Dec-85 16:15")
```

(* checks if the sketch has a when deleted fn and if so, creates the list of global elements and interprets the result.
Returns a list of the elements that should be deleted.)

```
    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           RESULT DELETEFN)
          (COND
             ([NULL (SETQ DELETEFN (GETSKETCHPROP SKETCH 'WHENDELETEDFN]
              (RETURN GELTS)))
          (SETQ RESULT (APPLY* DELETEFN VIEWER GELTS))
          (COND
             ((EQ RESULT 'DON'T)
              (RETURN NIL))
             ((LISTP RESULT)
              (RETURN RESULT))
             (T (RETURN GELTS])
```

(SK.CHECK.PREEDITFN

```
  [LAMBDA (VIEWER OLDELT)                                          (* rrb " 9-Dec-85 11:52")
                                                                   (* checks if the sketch has a preedit fn and if so, calls it)
    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           PREEDITFN)
          (COND
             ([NULL (SETQ PREEDITFN (GETSKETCHPROP SKETCH 'PREEDITFN]
              (RETURN T)))
          (RETURN (NEQ (APPLY* PREEDITFN VIEWER OLDELT)
                       'DON'T])
```

(SK.CHECK.END.INITIAL.EDIT

```
  [LAMBDA (VIEWER NEWELT)                                          (* rrb "15-Jan-86 15:20")
```

(* called when the edit of a newly created text element is ended.
Calls the when changed fn.)

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

(SK.CHECK.WHENPOINTDELETEDFN

```
  [LAMBDA (VIEWER SCRELT CONTROLPOINT)                                    (* rrb " 3-Jan-86 15:32")

          (* checks if the sketch has a prechange fn and if so, calls it and interprets the result.
          Returns NIL if the point should not be deleted.)

    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           RESULT PRECHANGEFN)
          (COND
             ([NULL (SETQ PRECHANGEFN (GETSKETCHPROP SKETCH 'PRECHANGEFN]
              (RETURN SCRELT)))
          (SETQ RESULT (APPLY* PRECHANGEFN VIEWER (LIST (fetch (SCREENELT GLOBALPART) of SCRELT))
                               (LIST 'DELETEPOINT CONTROLPOINT)))
          (COND
             ((EQ RESULT 'DON'T)
              (RETURN NIL))
             (T (RETURN SCRELT])
```

(SK.ERASE.ELT

```
  [LAMBDA (ELT WINDOW REGION)                                             (* rrb "30-Aug-86 15:08")
                                                                          (* erases a sketch element)
    (DSPOPERATION 'ERASE WINDOW)
    (SK.DRAWFIGURE ELT WINDOW REGION (VIEWER.SCALE WINDOW))
    (DSPOPERATION 'PAINT WINDOW])
```

(SK.DELETE.ELT

```
  [LAMBDA (W)                                                             (* rrb "31-Jan-86 10:48")
                                                                          (* lets the user select an element and deletes it.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.DELETE (KWOTE W))
           W])
```

(SK.DELETE.ITEM

```
  [LAMBDA (ELT SKETCHW)                                                   (* rrb "12-May-85 18:10")
                                                                          (* deletes an element from a window)
    (COND
       (ELT (DELFROMTCONC (WINDOWPROP SKETCHW 'SKETCHSPECS)
                  ELT)
            (SK.REMOVE.HOTSPOTS.FROM.CACHE1 ELT (SK.HOTSPOT.CACHE SKETCHW))
            ELT])
```

(DELFROMTCONC

```
  [LAMBDA (TCONCCELL ELEMENT)                                             (* rrb "26-Sep-86 13:24")

          (* deletes an element from a TCONC cell list. Returns T if the element was deleted, NIL if it wasn't a member.)

    (COND
       ((EQ ELEMENT (CAAR TCONCCELL))                                     (* first element)
        [COND
           ((EQLENGTH (CAR TCONCCELL)
                      1)                                                  (* only one element)
            (RPLACA TCONCCELL NIL)
            (RPLACD TCONCCELL NIL))
           (T                                                             (* remove first element.)
              (RPLACA TCONCCELL (CDAR TCONCCELL]
        T)
       ((EQ ELEMENT (CADR TCONCCELL))                                     (* elt to delete is the last one on the list, do special case.)
        (for TAIL on (CAR TCONCCELL) when (EQ (CDR TAIL)
                                              (CDR TCONCCELL))
           do                                                             (* update the TCONC last entry)
              (RPLACD TCONCCELL TAIL)                                     (* remove the last element)
              (RPLACD TAIL NIL)
              (RETURN))
        T)
       (T (for TAIL on (CAR TCONCCELL) when (EQ ELEMENT (CADR TAIL)) do (RPLACD TAIL (CDDR TAIL))
                                                                        (RETURN T)
             finally (RETURN NIL])

)

;; fns for copying stuff

(DEFINEQ

(SK.COPY.ELT
  [LAMBDA (W)                                                             (* rrb "31-Jan-86 10:49")
                                                                          (* lets the user select an element and copies it.)
```

```
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.COPY (KWOTE W))
           W])
```

(SK.SEL.AND.COPY

```
  [LAMBDA (W)                                                    (* rrb "10-Dec-85 17:08")
                                                                 (* lets the user select elements and copies them.)
    (SK.COPY.ELEMENTS (SK.SELECT.MULTIPLE.ITEMS W T NIL 'COPY)
           W])
```

(SK.COPY.ELEMENTS

```
  [LAMBDA (SCRELTS SKW)                                          (* rrb " 1-Oct-86 19:12")

          (* create a bitmap of the thing being moved and get its new position.
          Then translate all the pieces.)

    (AND SCRELTS (PROG (FIGINFO FIRSTHOTSPOT GHOTSPOT LOWLFT NEWGPOS DELTAPOS NEWELTS COPYFN SKETCH COPYARGS
                        COPYPLACEDYETFLG)                        (* call PRECOPYFN.)
                  [AND (SETQ COPYFN (GETSKETCHPROP (SETQ SKETCH (INSURE.SKETCH SKW))
                                                   'PRECOPYFN))
                       (SETQ DELTAPOS (APPLY* COPYFN SKW (SETQ COPYARGS (SK.GLOBAL.FROM.LOCAL.ELEMENTS
                                                                         SCRELTS]
                  [COND
                     ((EQ DELTAPOS 'DON'T)
                      (RETURN))
                     ((POSITIONP DELTAPOS)                       (* PRECOPYFN returned a position, don't bother to check for
                                                                 multiple copies.)

          (* value returned is the delta by which to move the point. Set up new position)

                      (RETURN (SK.ADD.COPY.OF.ELEMENTS SKW SCRELTS (OR COPYARGS (SETQ COPYARGS
                                                                                 (
                                                                    SK.GLOBAL.FROM.LOCAL.ELEMENTS
                                                                                  SCRELTS)))
                                    DELTAPOS]                    (* read new position from the user)
                  (SETQ FIGINFO (SK.FIGUREIMAGE SCRELTS (DSPCLIPPINGREGION NIL SKW)))
                  (SETQ LOWLFT (fetch (SKFIGUREIMAGE SKFIGURE.LOWERLEFT) of FIGINFO))
                  [SETQ FIRSTHOTSPOT (CAR (fetch (SCREENELT HOTSPOTS) of (CAR SCRELTS]
                  (SETQ GHOTSPOT (GETSKETCHELEMENTPROP (fetch (SCREENELT GLOBALPART) of (CAR SCRELTS))
                                 'POSITION))

          (* move the image by the first hotspot of the first element chosen.
          This will align the image on the grid correctly.)

                  PLACECOPYLP
                  (COND
                     ((SETQ NEWGPOS (SK.MAP.INPUT.PT.TO.GLOBAL [GET.BITMAP.POSITION
                                                                SKW
                                                                (fetch (SKFIGUREIMAGE SKFIGURE.BITMAP)
                                                                   of FIGINFO)
                                                                'PAINT "move the figure into place and
                                                                press the left button."
                                                                (FIXR (DIFFERENCE (fetch (POSITION XCOORD)
                                                                                     of LOWLFT)
                                                                                  (fetch (POSITION XCOORD)
                                                                                     of FIRSTHOTSPOT)))
                                                                (FIXR (DIFFERENCE (fetch (POSITION YCOORD)
                                                                                     of LOWLFT)
                                                                                  (fetch (POSITION YCOORD)
                                                                                     of FIRSTHOTSPOT]
                                                      SKW))
                      (CLOSEPROMPTWINDOW SKW))
                     (COPYPLACEDYETFLG                           (* already one copy down, close prompt window so user knows
                                                                 copy mode is over.)
                            (CLOSEPROMPTWINDOW SKW)
                            (RETURN NIL))
                     (T (STATUSPRINT SKW "Position was outside the window.  Copy not placed.")
                        (RETURN NIL)))
                  [SETQ DELTAPOS (create POSITION
                                         XCOORD _ (DIFFERENCE (fetch (POSITION XCOORD) of NEWGPOS)
                                                              (fetch (POSITION XCOORD) of GHOTSPOT))
                                         YCOORD _ (DIFFERENCE (fetch (POSITION YCOORD) of NEWGPOS)
                                                              (fetch (POSITION YCOORD) of GHOTSPOT]
                  (SK.ADD.COPY.OF.ELEMENTS SKW SCRELTS (OR COPYARGS (SETQ COPYARGS (
                                                                    SK.GLOBAL.FROM.LOCAL.ELEMENTS
                                                                                  SCRELTS)))
                        DELTAPOS)
                  (COND
                     ((.COPYKEYDOWNP.)
                      (SETQ COPYPLACEDYETFLG T)
                      (GO PLACECOPYLP))
                     (T (CLOSEPROMPTWINDOW SKW])
```

(SK.ADD.COPY.OF.ELEMENTS

```
  [LAMBDA (VIEWER SCRELEMENTS GLOBALELEMENTS NEWPOSDELTA)        (* rrb " 1-Oct-86 19:13")
```

```
        (* internal function for copying elements. Adds a copy of SCRELEMENTS moved by NEWPOSDELTA to VIEWER and calls
        the copyfn.)

    (PROG (SKETCH NEWELTS COPYFN X)
          (AND (SETQ COPYFN (GETSKETCHPROP (SETQ SKETCH (INSURE.SKETCH VIEWER))
                                   'WHENCOPIEDFN))
               (SETQ X (APPLY* COPYFN VIEWER GLOBALELEMENTS NEWPOSDELTA)))
          (COND
             ((EQ X 'DON'T)
              (RETURN))
             ((POSITIONP X)                                    (* value returned is the position to put the copy.
                                                               Set up new position)
              (SETQ NEWPOSDELTA X)))
          [SETQ NEWELTS (SK.SORT.GELTS.BY.PRIORITY (COND
                                                     ((AND (LISTP X)
                                                           (EVERY X (FUNCTION GLOBALELEMENTP)))
                                                                   (* value returns was a list of new global elements.)
                                                      X)
                                                     (T (MAPCOLLECTSKETCHSPECS SCRELEMENTS
                                                               (FUNCTION SK.COPY.ITEM)
                                                               NEWPOSDELTA VIEWER]
                                                                   (* add new elements to history list.)
          (SK.ADD.ELEMENTS NEWELTS VIEWER)
          (SK.ADD.HISTEVENT 'COPY NEWELTS VIEWER])
```

(SK.GLOBAL.FROM.LOCAL.ELEMENTS

```
  [LAMBDA (SCRELTS)                                            (* returns the global elements from a list of screen elements)
    (for SCRELT in SCRELTS collect (fetch (SCREENELT GLOBALPART) of SCRELT])
```

(SK.COPY.ITEM

```
  [LAMBDA (SELELT GLOBALDELTAPOS W)                            (* rrb "24-Jun-87 15:12")

        (* SELELT is a sketch element that was selected for a copy operation.
        GLOBALDELTAPOS is the amount the new item is to be offset from the old.)

    (PROG ((OLDGLOBAL (fetch (SCREENELT GLOBALPART) of SELELT)))
          [COND
             ((EQ (fetch (GLOBALPART GTYPE) of OLDGLOBAL)
                  'SKIMAGEOBJ)

        (* copying an image obj. Don't call its when copied fn. was changed to call the WHENINSERTEDFN instead when it acutally
        gets inserted.)

              (SETQ OLDGLOBAL (SK.COPY.IMAGEOBJ OLDGLOBAL W]
          (RETURN (SK.TRANSLATE.GLOBALPART OLDGLOBAL GLOBALDELTAPOS])
```

(SK.INSERT.SKETCH

```
  [LAMBDA (W SKETCH REGION SCALE)                              (* rrb "30-Sep-86 18:29")

        (* * inserts the sketch SKETCH into the sketch window W. Called by the copy insert function for sketch windows.)

    (AND SKETCH (PROG (LOCALSCRELTS FIGINFO FIRSTHOTSPOT LOWLFT NEWPOS WINDOWSCALE NEWELTS)
                                                               (* map inserted elements into new coordinate space.)
                      [COND
                         ([NOT (EQUAL SCALE (SETQ WINDOWSCALE (VIEWER.SCALE W]
                                                               (* change the scale of the sketch and the region.)
                          [SETQ SKETCH (create SKETCH using SKETCH SKETCHELTS _ (SK.TRANSFORM.GLOBAL.ELEMENTS
                                                                  (fetch (SKETCH SKETCHELTS)
                                                                     of SKETCH)
                                                                  (FUNCTION
                                                                SK.SCALE.POSITION.INTO.VIEWER.EXACT
                                                                  )
                                                                  (QUOTIENT SCALE WINDOWSCALE]
                          (SETQ REGION (SK.SCALE.REGION REGION (QUOTIENT SCALE WINDOWSCALE]
                      (OR (SETQ LOCALSCRELTS (MAKE.LOCAL.SKETCH SKETCH NIL WINDOWSCALE W T))
                          (RETURN))
                      (SETQ FIGINFO (SK.FIGUREIMAGE LOCALSCRELTS REGION))
                      [SETQ FIRSTHOTSPOT (CAR (fetch (SCREENELT HOTSPOTS) of (CAR LOCALSCRELTS]
                      (SETQ LOWLFT (fetch (SKFIGUREIMAGE SKFIGURE.LOWERLEFT) of FIGINFO))

        (* move the image by the first hotspot of the first element chosen.
        This will align the image on the grid correctly.)

                      (COND
                         ([SETQ NEWPOS (fetch (INPUTPT INPUT.POSITION) of (GET.BITMAP.POSITION
                                                                            W
                                                                            (fetch (SKFIGUREIMAGE SKFIGURE.BITMAP)
                                                                               of FIGINFO)
                                                                            'PAINT "move the figure into place and
                                                                            press the left button."
                                                                            (IDIFFERENCE (fetch (POSITION XCOORD)
                                                                                            of LOWLFT)
                                                                                   (fetch (POSITION XCOORD)
```

```
                                                                                    of FIRSTHOTSPOT))
                                                                          (IDIFFERENCE (fetch (POSITION YCOORD)
                                                                                         of LOWLFT)
                                                                                 (fetch (POSITION YCOORD)
                                                                                    of FIRSTHOTSPOT]
                                  (CLOSEPROMPTWINDOW W))
                                 (T (STATUSPRINT W "
                                                  " "Position was outside the window.  Copy not placed.")
                                    (RETURN NIL)))
                           (SETQ NEWELTS (MAPCOLLECTSKETCHSPECS LOCALSCRELTS (FUNCTION SK.COPY.ITEM)
                                                 (SK.MAP.FROM.WINDOW.TO.NEAREST.GRID (create POSITION
                                                                                            XCOORD _
                                                                                            (IDIFFERENCE
                                                                                             (fetch (POSITION XCOORD)
                                                                                                of NEWPOS)
                                                                                             (fetch (POSITION XCOORD)
                                                                                                of FIRSTHOTSPOT))
                                                                                            YCOORD _
                                                                                            (IDIFFERENCE
                                                                                             (fetch (POSITION YCOORD)
                                                                                                of NEWPOS)
                                                                                             (fetch (POSITION YCOORD)
                                                                                                of FIRSTHOTSPOT)))
                                                                                    WINDOWSCALE)
                                                                           W))
                           (SK.ADD.ELEMENTS NEWELTS W)
                           (SK.ADD.HISTEVENT 'COPY NEWELTS W)
                           (RETURN NEWELTS])

)

;; fns for moving things.

(DEFINEQ

(SK.MOVE.ELT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:49")
                                                                 (* lets the user select one or more elements and move them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE (KWOTE W))
           W])

(SK.MOVE.ELT.OR.PT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:49")
                                                                 (* lets the user select one or more elements and move them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE (KWOTE W)
                              T)
           W])

(SK.APPLY.DEFAULT.MOVE
  [LAMBDA (W)                                                    (* rrb " 2-Jun-85 12:52")
                                                                 (* applies the default move mode which can be either points,
                                                                 elements or both.)
    (SELECTQ (fetch (SKETCHCONTEXT SKETCHMOVEMODE) of (WINDOWPROP W 'SKETCHCONTEXT))
         (POINTS (SK.MOVE.POINTS W))
         (ELEMENTS (SK.MOVE.ELT W))
         (SK.MOVE.ELT.OR.PT W])

(SK.SEL.AND.MOVE
  [LAMBDA (W PTFLG)                                              (* rrb "10-Dec-85 17:06")

          (* lets the user select either a control point or one or more elements and move them.)

    (SK.MOVE.ELEMENTS [COND
                          ((EQ PTFLG 'ONLY)
                           (SK.SELECT.ITEM W NIL NIL 'MOVE))
                          (T (SK.SELECT.MULTIPLE.ITEMS W (NULL PTFLG)
                                    NIL
                                    'MOVE]
           W])

(SK.MOVE.ELEMENTS
  [LAMBDA (SCRELTS SKW)                                          (* rrb "11-Jul-86 15:51")
    (SKED.CLEAR.SELECTION SKW)
    (COND
       ((NULL SCRELTS))
       [[OR (POSITIONP SCRELTS)
            (AND (NULL (CDR SCRELTS))
                 (POSITIONP (CAR SCRELTS))
                 (SETQ SCRELTS (CAR SCRELTS]                     (* user selected a point, move just that point.)
        (PROG ((SKETCHELTS (SK.ELTS.FROM.HOTSPOT SCRELTS (SK.HOTSPOT.CACHE SKW)))
               SKETCHELT OTHERHOTSPOTS NEWPOS MOVEFN GDELTAPOS X MOVEARGS SKETCH)
              (COND
                 ((NULL SKETCHELTS)
```

```
                (RETURN NIL))
               ([NULL (SETQ SKETCHELT (for SCRELT in SKETCHELTS when (NOT (SK.ELEMENT.PROTECTED?
                                                                          (fetch (SCREENELT GLOBALPART)
                                                                             of SCRELT)
                                                                          'MOVE))
                                       do (RETURN SCRELT]    (* only protected elements at this point, shouldn't happen but
                                                             don't cause an error.)
                (RETURN NIL)))
            [COND
               ([NULL (SETQ OTHERHOTSPOTS (REMOVE SCRELTS (fetch (SCREENELT HOTSPOTS) of SKETCHELT]
                                                             (* only one control point, move it with the move element
                                                             function.)
                (RETURN (SK.MOVE.ELEMENTS (LIST SKETCHELT)
                                SKW]                         (* call sketch premovefn if given.)
            [AND (SETQ MOVEFN (GETSKETCHPROP (SETQ SKETCH (INSURE.SKETCH SKW))
                                    'PREMOVEFN))
                 (SETQ GDELTAPOS (APPLY* MOVEFN SKW (SETQ MOVEARGS (SK.MAKE.ELEMENT.MOVE.ARG SKETCHELT
                                                                          SCRELTS]
            [COND
               ((EQ GDELTAPOS 'DON'T)
                (RETURN))
               ((POSITIONP GDELTAPOS)
```

(* value returned is the delta by which to move the point. Set up new position)

```
                NIL)
               (T                                            (* read new position from the user)
                  (for PT in OTHERHOTSPOTS do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                  (CURSORPOSITION SCRELTS SKW)
                  (SETQ NEWPOS (SK.READ.POINT.WITH.FEEDBACK SKW NIL NIL NIL NIL NIL SKETCH.USE.POSITION.PAD))
                  (for PT in OTHERHOTSPOTS do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                                                             (* if user selected outside, don't move anything.)
                  (OR NEWPOS (RETURN NIL))                   (* calculate the delta that the selected point moves.)
                  (SETQ GDELTAPOS (SK.MAP.FROM.WINDOW.TO.NEAREST.GRID (create POSITION
                                                                          XCOORD _
                                                                          (IDIFFERENCE
                                                                           (fetch (POSITION XCOORD)
                                                                              of (fetch (INPUTPT
                                                                                                INPUT.POSITION
                                                                                                )
                                                                                        of NEWPOS))
                                                                           (fetch (POSITION XCOORD)
                                                                              of SCRELTS))
                                                                          YCOORD _
                                                                          (IDIFFERENCE
                                                                           (fetch (POSITION YCOORD)
                                                                              of (fetch (INPUTPT
                                                                                                INPUT.POSITION
                                                                                                )
                                                                                        of NEWPOS))
                                                                           (fetch (POSITION YCOORD)
                                                                              of SCRELTS)))
                                                    (VIEWER.SCALE SKW]
            (AND (SETQ MOVEFN (GETSKETCHPROP SKETCH 'WHENMOVEDFN))
                 (SETQ X (APPLY* MOVEFN SKW (OR MOVEARGS (SK.MAKE.ELEMENT.MOVE.ARG SKETCHELT SCRELTS))
                                GDELTAPOS)))
            (COND
               ((EQ X 'DON'T)
                (RETURN))
               ((POSITIONP X)
```

(* value returned is the delta by which to move the point. Set up new position)

```
                (SETQ GDELTAPOS X)))
            (RETURN (SK.MOVE.THING SKETCHELT SCRELTS GDELTAPOS SKW]
         (T
            (* create a bitmap of the thing being moved and get its new position.
            Then translate all the pieces.)

            (PROG (FIGINFO FIRSTHOTSPOT NEWPOS LOWLFT IMAGEPOSX IMAGEPOSY IMAGEBM DELTAPOS CHANGES MOVEFN X
                         GDELTAPOS)
                  [AND (SETQ MOVEFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                           'PREMOVEFN))
                       (SETQ GDELTAPOS (APPLY* MOVEFN SKW (SK.MAKE.ELEMENTS.MOVE.ARG SCRELTS]
                  [COND
                     ((EQ GDELTAPOS 'DON'T)
                      (RETURN))
                     ((POSITIONP GDELTAPOS)
```

(* value returned is the delta by which to move the point. Set up new position)

```
                      NIL)
                     (T                                      (* read new position from the user)
                        (SETQ FIGINFO (SK.FIGUREIMAGE SCRELTS (DSPCLIPPINGREGION NIL SKW)))
                        [SETQ FIRSTHOTSPOT (CAR (fetch (SCREENELT HOTSPOTS) of (CAR SCRELTS]
                        (SETQ IMAGEBM (fetch (SKFIGUREIMAGE SKFIGURE.BITMAP) of FIGINFO))
                        (SETQ LOWLFT (fetch (SKFIGUREIMAGE SKFIGURE.LOWERLEFT) of FIGINFO))
```

```
(* move the image by the first hotspot of the first element chosen.
This will align the image on the grid correctly.)

            (SETQ IMAGEPOSX (fetch (POSITION XCOORD) of LOWLFT))
            (SETQ IMAGEPOSY (fetch (POSITION YCOORD) of LOWLFT))
                                                    (* put the cursor on the hot spot)
            (CURSORPOSITION FIRSTHOTSPOT SKW)
            (COND
               ([NULL (ERSETQ (PROGN (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'ERASE SKW)
                                     (SETQ NEWPOS (fetch (INPUTPT INPUT.POSITION)
                                                    of (GET.BITMAP.POSITION SKW IMAGEBM 'PAINT
                                                              "Move image to its new position."
                                                              (IDIFFERENCE IMAGEPOSX
                                                                     (fetch (POSITION XCOORD)
                                                                        of FIRSTHOTSPOT))
                                                              (IDIFFERENCE IMAGEPOSY
                                                                     (fetch (POSITION YCOORD)
                                                                        of FIRSTHOTSPOT]
                                                    (* error happened, repaint the image.)
                (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT SKW)
                (CLOSEPROMPTWINDOW SKW)
                (ERROR!))
               ((NULL NEWPOS)
                (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT SKW)
                (STATUSPRINT SKW "Position was outside the window, copy not placed.")
                (RETURN NIL)))

(* GET.BITMAP.POSITION returns the position that the cursor was in which is the position of the first hotspot.)
                                                    (* calculate the delta that the selected point moves.)
            (SETQ GDELTAPOS (SK.MAP.FROM.WINDOW.TO.NEAREST.GRID [SETQ DELTAPOS
                                                                  (create POSITION
                                                                         XCOORD _
                                                                         (IDIFFERENCE
                                                                          (fetch (POSITION XCOORD)
                                                                             of NEWPOS)
                                                                          (fetch (POSITION XCOORD)
                                                                             of FIRSTHOTSPOT))
                                                                         YCOORD _
                                                                         (IDIFFERENCE
                                                                          (fetch (POSITION YCOORD)
                                                                             of NEWPOS)
                                                                          (fetch (POSITION YCOORD)
                                                                             of FIRSTHOTSPOT]
                                                (VIEWER.SCALE SKW]
                  (SKETCH.MOVE.ELEMENTS (for ELT in SCRELTS collect (fetch (SCREENELT GLOBALPART) of ELT))
                         GDELTAPOS SKW T)

(* I started noticing cases where the image was a point off on some lines and where the texture alignment was off so I
removed this (COND ((AND DELTAPOS (NOT (POSITIONP X)))
(* If the user was asked for a new position and the movefn didn't change it, redraw the image in case any of it was erased
by the calls to SK.TRANSLATE.ITEM) (SK.SHOW.FIG.FROM.INFO IMAGEBM
(IPLUS IMAGEPOSX (fetch (POSITION XCOORD) of DELTAPOS))
(IPLUS IMAGEPOSY (fetch (POSITION YCOORD) of DELTAPOS))
(QUOTE PAINT) SKW))))

            (CLOSEPROMPTWINDOW SKW])
```

(SKETCH.MOVE.ELEMENTS

```
  [LAMBDA (ELEMENTS DELTA SKETCHTOUPDATE ADDHISTORY?)          (* rrb " 2-Oct-86 11:09")

(* moves the elements ELEMENTS by the amount of position DELTA
(XCOORD gives x amount, YCOORD gives y delta) and updates the viewers on SKETCHTOUPDATE if it is given.)

    (PROG (X MOVEFN NEWGLOBALS SKETCH GDELTAPOS VIEWER)
          (OR (POSITIONP DELTA)
              (\ILLEGAL.ARG DELTA))
          (AND SKETCHTOUPDATE (SETQ SKETCH (INSURE.SKETCH SKETCHTOUPDATE))
               (SETQ VIEWER (SK.VIEWER.FROM.SKETCH.ARG SKETCHTOUPDATE)))
          (COND
             [[AND SKETCH (SETQ MOVEFN (GETSKETCHPROP SKETCH 'WHENMOVEDFN]

(* call the WHENMOVEDFN if any Pass the thing the user passed in if you can't find a viewer.)

              (COND
                 ((EQ (SETQ X (APPLY* MOVEFN VIEWER (for ELT in ELEMENTS collect (CONS T ELT))
                                      DELTA))
                      'DON'T)
                  (RETURN))
                 ((POSITIONP X)

(* value returned is the delta by which to move the point. Set up new position)

                  (SETQ GDELTAPOS X))
                 (T (SETQ GDELTAPOS DELTA]
             (T (SETQ GDELTAPOS DELTA)))
```

```
          (SETQ NEWGLOBALS (MAPGLOBALSKETCHSPECS (SK.SORT.GELTS.BY.PRIORITY ELEMENTS)
                                   (FUNCTION SK.TRANSLATE.ELEMENT)
                                   GDELTAPOS VIEWER))
          (AND ADDHISTORY? (SK.ADD.HISTEVENT 'MOVE (for NEWG in NEWGLOBALS as OLDG in ELEMENTS when NEWG
                                                      collect (LIST OLDG NEWG GDELTAPOS))
                                   VIEWER))
          (RETURN NEWGLOBALS])
```

(SKETCH.COPY.ELEMENTS

```
  [LAMBDA (ELEMENTS SKETCHTOUPDATE DELTA ADDHISTORY?)          (* rrb "15-Dec-86 15:58")
```

(* copies the elements ELEMENTS moving them by the amount of position DELTA
(XCOORD gives x amount, YCOORD gives y delta) and updates the viewers on SKETCHTOUPDATE if it is given.)

```
    (PROG (X COPYFN NEWGLOBALS SKETCH GDELTAPOS VIEWER)
          (COND
             ((NULL DELTA)
              (SETQ DELTA (CREATEPOSITION 0 0)))
             ((POSITIONP DELTA))
             (T (\ILLEGAL.ARG DELTA)))
          (AND SKETCHTOUPDATE (SETQ SKETCH (INSURE.SKETCH SKETCHTOUPDATE))
               (SETQ VIEWER (SK.VIEWER.FROM.SKETCH.ARG SKETCHTOUPDATE)))
          (COND
             [[AND SKETCH (SETQ COPYFN (GETSKETCHPROP SKETCH 'WHENCOPIEDFN]
```

(* call the WHENCOPIEFN if any Pass the thing the user passed in if you can't find a viewer.)

```
              (COND
                 ((EQ (SETQ X (APPLY* COPYFN VIEWER ELEMENTS DELTA))
                      'DON'T)
                  (RETURN))
                 ((POSITIONP X)
```

(* value returned is the delta by which to move the point. Set up new position)

```
                  (SETQ GDELTAPOS X))
                 (T (SETQ GDELTAPOS DELTA]
             (T (SETQ GDELTAPOS DELTA)))
          (SETQ NEWGLOBALS (MAPGLOBALSKETCHSPECS ELEMENTS (FUNCTION \SKETCH.COPY.ELEMENT)
                                   GDELTAPOS VIEWER))
          (AND SKETCH (for ELT in NEWGLOBALS do (SK.SET.ELEMENT.PRIORITY ELT NIL)
                                                (SKETCH.ADD.ELEMENT ELT SKETCHTOUPDATE)))
          (AND ADDHISTORY? VIEWER (SK.ADD.HISTEVENT 'COPY
                                        (for NEWG in NEWGLOBALS as OLDG in ELEMENTS when NEWG
                                           collect (LIST OLDG NEWG))
                                        VIEWER))
          (RETURN NEWGLOBALS])
```

(\SKETCH.COPY.ELEMENT

```
  [LAMBDA (GLOBALELEMENT GLOBALDELTAPOS W)                    (* rrb "24-Jun-87 15:05")
```

(* SELELT is a sketch element that was selected for a copy operation.
GLOBALDELTAPOS is the amount the new item is to be offset from the old.)

```
    (COND
       ((EQ (fetch (GLOBALPART GTYPE) of GLOBALELEMENT)
            'SKIMAGEOBJ)                                      (* copying an image obj. Calls its when copied fn.)
        (SK.TRANSLATE.GLOBALPART (SK.COPY.IMAGEOBJ GLOBALELEMENT W)
              GLOBALDELTAPOS))
       (T (SK.TRANSLATE.GLOBALPART GLOBALELEMENT GLOBALDELTAPOS])
```

(SK.TRANSLATE.ELEMENT

```
  [LAMBDA (GELT GLOBALDELTAPOS W)                             (* rrb "25-Sep-86 15:16")
```

(* * GELT is a sketch element to be moved. GLOBALDELTAPOS is the amount the item is to be translated.)

```
    (PROG (NEWGLOBAL)
          (COND
             ((SETQ NEWGLOBAL (SK.TRANSLATE.GLOBALPART GELT GLOBALDELTAPOS))
              (AND W (SK.UPDATE.ELEMENT GELT NEWGLOBAL W T T))
              (RETURN NEWGLOBAL])
```

(SK.COPY.GLOBAL.ELEMENT

```
  [LAMBDA (GLOBALELT)                                         (* returns a copy of a global element.)
    (SK.TRANSLATE.GLOBALPART GLOBALELT (CREATEPOSITION 0 0)
          T])
```

(SK.MAKE.ELEMENT.MOVE.ARG

```
  [LAMBDA (SCRELT SELPOS)                                     (* rrb " 5-Nov-85 14:35")
```

(* makes an argument structure that is suitable to be passed to the sketch movefn.
This is a list whose CAR is a list of the numbers of the control points being moved and whose CDR is the global sketch element.)

```
    (CONS (CONS (for I from 1 as PT in (fetch (SCREENELT HOTSPOTS) of SCRELT) when (EQUAL PT SELPOS)
                   do (RETURN I)))
          (fetch (SCREENELT GLOBALPART) of SCRELT])
```

(SK.MAKE.ELEMENTS.MOVE.ARG

```
  [LAMBDA (SCRELTS)                                                    (* rrb " 5-Nov-85 14:34")

          (* makes an argument structure that is suitable to be passed to the sketch movefn.
          This is a list whose CAR is a list of the numbers of the control points being moved which is in this case T and whose CDR is
          the global sketch element.)

    (CONS T (for SCRELT in SCRELTS collect (fetch (SCREENELT GLOBALPART) of SCRELT])
```

(SK.MAKE.POINTS.AND.ELEMENTS.MOVE.ARG

```
  [LAMBDA (SCRELTS SELPTS)                                             (* rrb "21-Jan-86 17:38")

          (* makes an argument structure that is suitable to be passed to the sketch movefn.
          This is a list of lists each of whose CAR is a list of the numbers of the control points being moved and whose CDR is the
          global sketch element.)

    (for SCRELT in SCRELTS collect (CONS (bind NOTALL for I from 1 as PT in (fetch (SCREENELT HOTSPOTS) of SCRELT)
                                            when (COND
                                                    ((MEMBER PT SELPTS))
                                                    (T (SETQ NOTALL T)
                                                       NIL))
                                            collect I finally (OR NOTALL (RETURN T)))
                                         (fetch (SCREENELT GLOBALPART) of SCRELT])
```

(SK.SHOW.FIG.FROM.INFO

```
  [LAMBDA (IMAGEBM XOFFSET YOFFSET OPERATION WINDOW)                   (* rrb "14-Nov-84 14:20")
                                                                       (* puts a bitmap onto the sketch window.)
    (BITBLT IMAGEBM 0 0 WINDOW XOFFSET YOFFSET NIL NIL 'INPUT OPERATION])
```

(SK.MOVE.THING

```
  [LAMBDA (SKETCHELT LOCALPT GDELTAPOS SKW)                            (* rrb "27-Jun-86 14:04")
                                                                       (* moves a control point in a sketch element.)
    (PROG (OLDGLOBAL NEWGLOBAL)                                        (* calculate the delta that the selected point moves.)
          (SETQ NEWGLOBAL (SK.TRANSLATE.POINTS (LIST LOCALPT)
                                 GDELTAPOS SKETCHELT SKW))             (* moving a piece of an element.)
          (SK.UPDATE.ELEMENT (SETQ OLDGLOBAL (fetch (SCREENELT GLOBALPART) of SKETCHELT))
                NEWGLOBAL SKW)
          (SK.ADD.HISTEVENT 'MOVE (LIST (LIST OLDGLOBAL NEWGLOBAL GDELTAPOS))
                SKW)
          (RETURN NEWGLOBAL])
```

(UPDATE.ELEMENT.IN.SKETCH

```
  [LAMBDA (OLDGELT NEWGELT SKETCH SKW DONTUPDATEPRIORITYFLG)           (* rrb "26-Sep-86 13:35")
                                                                       (* changes the global sketch)

          (* returns NIL if the old global sketch element is not found in SKETCH.
          This can happen if things are undone out of order.)

    (PROG ((SKETCHSTRUCTURE (INSURE.SKETCH SKETCH))
           SKETCHELEMENTS)

          (* if old and new are the same, the change was done destructively;
          otherwise clobber the new one in.)

          [COND
             ((EQ OLDGELT NEWGELT))
             ((OR (NULL DONTUPDATEPRIORITYFLG)
                  (EQ (SK.ELEMENT.PRIORITY OLDGELT)
                      (SK.ELEMENT.PRIORITY NEWGELT)))

          (* same priorities so just clobber the old elements place in the list with the new one.)

              (OR (for GELTTAIL on (fetch (SKETCH SKETCHELTS) of SKETCHSTRUCTURE)
                     when (EQ (CAR GELTTAIL)
                              OLDGELT)
                     do [OR DONTUPDATEPRIORITYFLG (SK.SET.ELEMENT.PRIORITY NEWGELT (SK.ELEMENT.PRIORITY
                                                                                    (CAR GELTTAIL]
                        (RPLACA GELTTAIL NEWGELT)
                        (RETURN T))
                  (RETURN)))
             (T

          (* priority has changed so order of this element in the list may need to be changed.)

                (REMOVE.ELEMENT.FROM.SKETCH OLDGELT SKETCHSTRUCTURE)
                (SK.ADD.PRIORITY.ELEMENT.TO.SKETCH SKETCHSTRUCTURE NEWGELT (SK.ELEMENT.PRIORITY NEWGELT]
          (SK.MARK.DIRTY SKETCH)
```

```
          (RETURN T])
```

(SK.UPDATE.ELEMENT

```
  [LAMBDA (OLDGLOBAL NEWGLOBAL SKETCHW REDRAWIFSAMEFLG DONTUPDATEPRIORITYFLG DONTDISPLAYFLG)
                                                                    (* rrb "24-Sep-86 17:32")
```

(* replaces an old element with a new one. The global part of the old one may be the same as the new global part. This also handles propagation to other windows that have the same figure displayed.)

```
    (PROG ((SKETCH (SKETCH.FROM.VIEWER SKETCHW))
           UPDATEDELT)
```

(* update the element in the sketch first. If this returns NIL, the element was not found in the sketch.)

```
          (OR (UPDATE.ELEMENT.IN.SKETCH OLDGLOBAL NEWGLOBAL SKETCH SKETCHW DONTUPDATEPRIORITYFLG)
              (RETURN NIL))                    (* do the window that the interaction occurred in first.)
          (SETQ UPDATEDELT (SK.UPDATE.ELEMENT1 OLDGLOBAL NEWGLOBAL SKETCHW REDRAWIFSAMEFLG DONTDISPLAYFLG))
                                                (* propagate to other windows.)
          (for SKW in (ALL.SKETCH.VIEWERS SKETCH) when (NEQ SKW SKETCHW)
             do
```

(* the position may have changed which means that it may have moved in or out of a viewer.)

```
                (SK.UPDATE.ELEMENT1 OLDGLOBAL NEWGLOBAL SKW REDRAWIFSAMEFLG DONTDISPLAYFLG))
          (RETURN UPDATEDELT])
```

(SK.UPDATE.ELEMENTS

```
  [LAMBDA (CHANGEEVENTS WINDOW DONTUPDATEPRIORITYFLG DONTDISPLAYFLG)
                                                                    (* rrb "24-Sep-86 17:32")
```

(* replaces the global parts of a list of change events and handles updating the screen.)

```
    (for CHANGEEVENT in CHANGEEVENTS do (SK.UPDATE.ELEMENT (fetch (SKHISTORYCHANGESPEC OLDELT) of CHANGEEVENT)
                                                           (fetch (SKHISTORYCHANGESPEC NEWELT) of CHANGEEVENT)
                                                          WINDOW NIL DONTUPDATEPRIORITYFLG DONTDISPLAYFLG])
```

(SK.UPDATE.ELEMENT1

```
  [LAMBDA (OLDGELT NEWGELT SKETCHW REDRAWIFSAME DONTDISPLAYFLG)  (* rrb "24-Sep-86 17:32")
```

(* determines what action is needed wrt the viewer SKETCHW when the element OLDGELT is updated to NEWGELT. This works only in the given window.)

```
    (PROG (LOCALELT UPDATEFN NEWLOCAL)
          (COND
             [(SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART OLDGELT SKETCHW))
              (COND
                 (DONTDISPLAYFLG                                (* just do the update in the datastructure, don't change the display)
                        (SK.DELETE.ITEM LOCALELT SKETCHW)
                        (RETURN (SK.ADD.ITEM NEWGELT SKETCHW)))
                 ((EQ (SKETCH.ELEMENT.TYPE OLDGELT)
                      'SKIMAGEOBJ)                              (* handle imageobject case specially because changes are often in internal structure)
                  (SK.DELETE.ITEM LOCALELT SKETCHW)
```

(* erase the old image region because often the internal parts of the image object have been clobbered making it impossible to erase by redrawing)

```
                  (DSPFILL (fetch (LOCALSKIMAGEOBJ SKIMOBJLOCALREGION) of (fetch (SCREENELT LOCALPART)
                                                                             of LOCALELT))
                         WHITESHADE
                         'REPLACE SKETCHW)
                  (RETURN (SKETCH.ADD.AND.DISPLAY1 NEWGELT SKETCHW)))
                [[AND (EQUAL OLDGELT NEWGELT)
                      (NOT (MEMB (fetch (GLOBALPART GTYPE) of OLDGELT)
                                 '(TEXT TEXTBOX]
```

(* text and textbox are special because interactive editing reuses the same element after the first character but they need to use updatefns for speed.)

(* replacing something by something else that is identical. Check here because add will not add something that is already there and updatefn may call add first.)

```
                (COND
                   (REDRAWIFSAME
```

(* this entry is used from the WB.BUTTON.HANDLER and deals with image objects which we have no control over whether they give us something new or not.)

```
                          (SK.ERASE.AND.DELETE.ITEM LOCALELT SKETCHW))
                   (T (SK.DELETE.ITEM LOCALELT SKETCHW)
                      (RETURN (SK.ADD.ITEM NEWGELT SKETCHW]
                ((AND (SETQ UPDATEFN (SK.UPDATEFN (fetch (GLOBALPART GTYPE) of NEWGELT)))
                      (SETQ NEWLOCAL (APPLY* UPDATEFN LOCALELT NEWGELT SKETCHW)))
```

```
        (* if the old one is visible and the element has an updatefn, use it to update the display.
        Then delete the old one. The updatefn should have added the new one.)

                (SK.DELETE.ITEM LOCALELT SKETCHW)
                (RETURN NEWLOCAL))
               (T

        (* if this type doesn't have a updatefn or it returned NIL, do the erase and redraw method.)

                  (SK.ERASE.AND.DELETE.ITEM LOCALELT SKETCHW]
           ((NOT (MEMB NEWGELT (SKETCH.ELEMENTS.OF.SKETCH SKETCHW)))
                                                        (* this element isn't a member of this sketch, quit)
            (RETURN)))
        (RETURN (COND
                   ((ELT.INSIDE.SKWP NEWGELT SKETCHW)
                    (SKETCH.ADD.AND.DISPLAY1 NEWGELT SKETCHW])
```

(SK.MOVE.ELEMENT.POINT

```
  [LAMBDA (W)                                           (* rrb "31-Jan-86 10:50")
                                                        (* lets the user select an element and move it.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE (KWOTE W)
                              ''ONLY)
           W])

)

;; fns for moving points or a collection of pts.

(DEFINEQ
```

(SK.MOVE.POINTS

```
  [LAMBDA (W)                                           (* rrb "31-Jan-86 10:50")
                                                        (* lets the user select a collection of points and move them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE.POINTS (KWOTE W))
           W])
```

(SK.SEL.AND.MOVE.POINTS

```
  [LAMBDA (W)                                           (* rrb "17-Oct-85 11:11")

        (* * lets the user select a collection of control point and moves them.)

    (SK.DO.MOVE.ELEMENT.POINTS (SK.SELECT.MULTIPLE.POINTS W)
           W])
```

(SK.DO.MOVE.ELEMENT.POINTS

```
  [LAMBDA (SCRPTS SKW)                                  (* rrb "30-Sep-86 18:33")
                                                        (* moves a collection of points)
    (SKED.CLEAR.SELECTION SKW)
    (AND SCRPTS (PROG ((SCRELTS (SK.ELTS.CONTAINING.PTS SCRPTS SKW))
                       NONMOVEDHOTSPOTS ONEPTELTS FIGINFO FIRSTHOTSPOT NEWPOS LOWLFT IMAGEPOSX IMAGEPOSY IMAGEBM
                       DELTAPOS NEWGLOBALS CHANGES MOVEFN X MOVEARGS SKETCH GDELTAPOS)
                      [AND (SETQ MOVEFN (GETSKETCHPROP (SETQ SKETCH (INSURE.SKETCH SKW))
                                                       'PREMOVEFN))
                           (SETQ GDELTAPOS (APPLY* MOVEFN SKW (SETQ MOVEARGS (
                                                                SK.MAKE.POINTS.AND.ELEMENTS.MOVE.ARG
                                                                   SCRELTS SCRPTS]
                      (COND
                         ((EQ GDELTAPOS 'DON'T)
                          (RETURN))
                         ((POSITIONP GDELTAPOS)

        (* value returned is the delta by which to move the point. Set up new position)

                          NIL)
                         (T                             (* read new position from the user)

        (* create a bitmap of all of the elements that have any point being moved and get its new position.
        Use only the region that contains the points. points plus a boarder to catch the lines of a box as large as the region.)

                            (SETQ NONMOVEDHOTSPOTS (SK.HOTSPOTS.NOT.ON.LIST SCRPTS SCRELTS))
                            [SETQ ONEPTELTS (SUBSET SCRELTS (FUNCTION (LAMBDA (ELT)
                                                                (EQ (LENGTH (fetch (LOCALPART HOTSPOTS)
                                                                               of (fetch (SCREENELT
                                                                                         LOCALPART
                                                                                         )
                                                                                  of ELT)))
                                                                   1]
                            (SETQ FIGINFO (SK.FIGUREIMAGE SCRELTS NIL (INCREASEREGION
                                                                      (COND
                                                                         (ONEPTELTS
```

```
        (* include the regions of any elements that only have one control point.
        This picks up text and groups whose image is much larger than the point.)
```

```
                                                                    (SK.UNIONREGIONS (
                                                                          REGION.CONTAINING.PTS
                                                                             SCRPTS)
                                                                       (
                                                                    SK.LOCAL.REGION.OF.LOCAL.ELEMENTS
                                                                          ONEPTELTS)))
                                                                (T (REGION.CONTAINING.PTS SCRPTS)))
                                                            4)))
                                (SETQ FIRSTHOTSPOT (CAR SCRPTS))
                                (SETQ LOWLFT (fetch (SKFIGUREIMAGE SKFIGURE.LOWERLEFT) of FIGINFO))
                                (SETQ IMAGEBM (fetch (SKFIGUREIMAGE SKFIGURE.BITMAP) of FIGINFO))
```

(* move the image by the first hotspot of the first element chosen.
This will align the image on the grid correctly.)

```
                                (SETQ IMAGEPOSX (fetch (POSITION XCOORD) of LOWLFT))
                                (SETQ IMAGEPOSY (fetch (POSITION YCOORD) of LOWLFT))
                                                                    (* put the cursor on the hot spot)
                                (CURSORPOSITION FIRSTHOTSPOT SKW)
                                (COND
                                   ([NULL (ERSETQ (PROGN (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY
                                                         'ERASE SKW)
                                                         (for PT in NONMOVEDHOTSPOTS
                                                            do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                                                         (SETQ NEWPOS (fetch (INPUTPT INPUT.POSITION)
                                                                         of (GET.BITMAP.POSITION
                                                                             SKW IMAGEBM 'PAINT "Move image to its
                                                                             new position."
                                                                             (IDIFFERENCE IMAGEPOSX
                                                                                    (fetch (POSITION XCOORD)
                                                                                       of FIRSTHOTSPOT))
                                                                             (IDIFFERENCE IMAGEPOSY
                                                                                    (fetch (POSITION YCOORD)
                                                                                       of FIRSTHOTSPOT]
                                                                    (* error happened, repaint the image.)
                                    (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT SKW)
                                    (for PT in NONMOVEDHOTSPOTS do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                                    (CLOSEPROMPTWINDOW SKW)
                                    (ERROR!))
                                   ((NULL NEWPOS)
                                    (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT SKW)
                                    (for PT in NONMOVEDHOTSPOTS do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                                    (STATUSPRINT SKW "Position was outside the window, copy not placed.")
                                    (RETURN NIL)))
```

(* GET.BITMAP.POSITION returns the position that the cursor was in which is the position of the first hotspot.)

```
                                (for PT in NONMOVEDHOTSPOTS do (MARKPOINT PT SKW OTHERCONTROLPOINTMARK))
                                (SETQ GDELTAPOS (SK.MAP.FROM.WINDOW.TO.NEAREST.GRID
                                                   (create POSITION
                                                          XCOORD _ (IDIFFERENCE (fetch (POSITION XCOORD) of NEWPOS)
                                                                                (fetch (POSITION XCOORD) of FIRSTHOTSPOT))
                                                          YCOORD _ (IDIFFERENCE (fetch (POSITION YCOORD) of NEWPOS)
                                                                                (fetch (POSITION YCOORD) of FIRSTHOTSPOT)))
                                                   (VIEWER.SCALE SKW)))
                                                                    (* calculate the delta that the selected point moves.)
                             ))
                          (AND (SETQ MOVEFN (GETSKETCHPROP SKETCH 'WHENMOVEDFN))
                               (SETQ X (APPLY* MOVEFN SKW (OR MOVEARGS (SK.MAKE.ELEMENTS.MOVE.ARG SCRELTS))
                                               GDELTAPOS)))
                          (COND
                             ((EQ X 'DON'T)
                              (RETURN))
                             ((POSITIONP X)
```

(* value returned is the delta by which to move the point. Set up new position)

```
                                 (SETQ GDELTAPOS X)))
                          (SETQ NEWGLOBALS (MAPCOLLECTSKETCHSPECS SCRELTS (FUNCTION SK.MOVE.ITEM.POINTS)
                                                                  GDELTAPOS SKW SCRPTS))
                          (SK.ADD.HISTEVENT 'MOVE (for NEWG in NEWGLOBALS as OLDG in SCRELTS when NEWG
                                                     collect (LIST (fetch (SCREENELT GLOBALPART) of OLDG)
                                                                   NEWG))
                                  SKW)
                          (CLOSEPROMPTWINDOW SKW])
```

(SK.MOVE.ITEM.POINTS

```
  [LAMBDA (SELELT GLOBALDELTAPOS W LOCALPTS)                    (* rrb "11-Jul-85 13:44")
```

(* SELELT is a sketch element at least one of whose points was selected for a translate operation.
GLOBALDELTAPOS is the amount the item is to be translated.
LOCALPTS is the list of points that was selected. This function moves any of those that belong to SELELT and return the new global. If all of SELELT points are on LOCALPTS this is a SK.TRANSLATE.ITEM.)

```
    (PROG ((ELTHOTSPOTS (fetch (LOCALPART HOTSPOTS) of (fetch (SCREENELT LOCALPART) of SELELT)))
```

```
          MOVEDPTS NEWGLOBAL OLDGLOBAL NEWSCREENELT)            (* this shouldn't happen but don't cause an error if it does.)
         (OR (SETQ MOVEDPTS (INTERSECTION ELTHOTSPOTS LOCALPTS))
             (RETURN))
```

(* map the difference point onto a grid location that would have the same screen distance but will leave things on a power of two.)

```
         (SETQ OLDGLOBAL (fetch (SCREENELT GLOBALPART) of SELELT))
         (COND
            ((EQ (LENGTH MOVEDPTS)
                 (LENGTH ELTHOTSPOTS))                          (* all of its hot spots have been moved, just translate it)
             (OR (SETQ NEWGLOBAL (SK.TRANSLATE.GLOBALPART OLDGLOBAL GLOBALDELTAPOS W))
                 (RETURN NIL)))
            ((SETQ NEWGLOBAL (SK.TRANSLATE.POINTS MOVEDPTS GLOBALDELTAPOS SELELT W)))
            (T (RETURN NIL)))
         (SK.UPDATE.ELEMENT OLDGLOBAL NEWGLOBAL W T)
         (RETURN NEWGLOBAL])
```

(SK.TRANSLATEPTSFN

```
  [LAMBDA (ELEMENTTYPE)                                         (* rrb " 5-May-85 16:25")
                                                                (* goes from an element type name to its EXPANDFN)
    (fetch (SKETCHTYPE TRANSLATEPTSFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.TRANSLATE.POINTS

```
  [LAMBDA (SELPTS GLOBALDELTA SKETCHELT W)                      (* rrb " 6-May-86 11:01")
                                                                (* moves the selected points by a global amount.)
    (AND SKETCHELT (PROG ((NEWGLOBAL (APPLY* (SK.TRANSLATEPTSFN (fetch (SCREENELT GTYPE) of SKETCHELT))
                                             SKETCHELT SELPTS GLOBALDELTA W)))
                                                                (* copy the elements property list.)
                         (SK.COPY.ELEMENT.PROPERTY.LIST NEWGLOBAL (fetch (SCREENELT GLOBALPART) of SKETCHELT))
                         (RETURN NEWGLOBAL])
```

(SK.SELECT.MULTIPLE.POINTS

```
  [LAMBDA (SKW)                                                 (* rrb "10-Dec-85 16:41")
```

(* * allows the user to select a collection of control points.)

```
    (PROG ((INTERIOR (DSPCLIPPINGREGION NIL SKW))
           SELECTABLEITEMS HOTSPOTCACHE NOW OLDX ORIGX NEWX NEWY OLDY ORIGY SELPTS PREVMOUSEBUTTONS MOUSEINSIDE?
           )
          (COND
             [(SK.HAS.SOME.HOTSPOTS (SETQ HOTSPOTCACHE (SK.HOTSPOT.CACHE.FOR.OPERATION SKW 'MOVE]
             (T                                                 (* no items, don't do anything.)
                (RETURN)))
          (TOTOPW SKW)
          (SK.PUT.MARKS.UP SKW HOTSPOTCACHE)
          (until (MOUSESTATE (NOT UP)))
          (COND
             ((INSIDEP INTERIOR (LASTMOUSEX SKW)
                       (LASTMOUSEY SKW)))
             (T                                                 (* first press was outside of the window, don't select anything.)
                (SK.TAKE.MARKS.DOWN SKW HOTSPOTCACHE)
                (RETURN)))
      SELECTLP
          (COND
             ((MOUSESTATE UP)
              (GO SHIFTDOWNLP)))
```

(* this label provides an entry for the code that tests if the shift key is down.)

```
      SELAFTERTEST
          (SETQ NEWY (LASTMOUSEY SKW))
          (SETQ NEWX (LASTMOUSEX SKW))
          [COND
             [(NOT MOUSEINSIDE?)
```

(* mouse is outside, don't do anything other than wait for it to come back in.
If the user has let up all buttons, the branch to SELECTEXIT will have been taken.)

```
              (COND
                 ((INSIDEP INTERIOR NEWX NEWY)
                  (SETQ MOUSEINSIDE? T)                         (* restore the saved selected items.)
                  (for ELT in SELPTS do (SK.ADD.PT.SELECTION ELT SKW]
             ((NOT (INSIDEP INTERIOR NEWX NEWY))
```

(* mouse just went outside, remove selections but save them in case mouse comes back in.)

```
              (SETQ MOUSEINSIDE? NIL)
              (SETQ SELPTS (WINDOWPROP SKW 'SKETCH.SELECTIONS))
              (for ELT in SELPTS do (SK.REMOVE.PT.SELECTION ELT SKW)))
             [(NEQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS)
```

(* another button has gone down, mark this as the origin of a new box to sweep.)

```
            (SETQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS)
            (SETQ ORIGX (LASTMOUSEX SKW))
            (SETQ ORIGY (LASTMOUSEY SKW))                    (* add or delete the element that the button press occurred on if
            any.)
            (AND (SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE (create POSITION
                                                                XCOORD _ NEWX
                                                                YCOORD _ NEWY)
                                       T))
                 (COND
                    ((LASTMOUSESTATE (ONLY LEFT))            (* add selection.)
                     (SK.ADD.PT.SELECTION NOW SKW))
                    ((LASTMOUSESTATE RIGHT)                  (* remove selection.)
                     (SK.REMOVE.PT.SELECTION NOW SKW]
            ([AND (OR (NEQ NEWX OLDX)
                      (NEQ NEWY OLDY))
                 (SETQ SELPTS (SK.CONTROL.POINTS.IN.REGION HOTSPOTCACHE (MIN ORIGX NEWX)
                                    (MIN ORIGY NEWY)
                                    (MAX ORIGX NEWX)
                                    (MAX ORIGY NEWY]          (* add or delete any with in the swept out area.)
            (COND
               ((LASTMOUSESTATE (ONLY LEFT))                 (* left only selects.)
                (for SELPT in SELPTS do (SK.ADD.PT.SELECTION SELPT SKW)))
               ((LASTMOUSESTATE RIGHT)                       (* right cause deselect.)
                (for SELPT in SELPTS do (SK.REMOVE.PT.SELECTION SELPT SKW]
         (SETQ OLDX NEWX)
         (SETQ OLDY NEWY)
         (GO SELECTLP)
      SHIFTDOWNLP
         (COND
            ((MOUSESTATE (NOT UP))                           (* button went down again, initialize the button state and click
                                                             position.)
             (SETQ PREVMOUSEBUTTONS NIL)
             (GO SELAFTERTEST))
            ((.SHIFTKEYDOWNP.)
             [COND
                [(NOT MOUSEINSIDE?)                          (* mouse is outside%: if it comes back in, mark the selections.)
                 (COND
                    ((INSIDEP INTERIOR (LASTMOUSEX SKW)
                              (LASTMOUSEY SKW))
                     (SETQ MOUSEINSIDE? T)                   (* restore the saved selected items.)
                     (for ELT in SELPTS do (SK.ADD.PT.SELECTION ELT SKW]
                ((NOT (INSIDEP INTERIOR (LASTMOUSEX SKW)
                               (LASTMOUSEY SKW)))            (* mouse just went outside, remove marks but keep selections)
                 (SETQ MOUSEINSIDE? NIL)
                 (SETQ SELPTS (WINDOWPROP SKW 'SKETCH.SELECTIONS))
                 (for ELT in SELPTS do (SK.REMOVE.PT.SELECTION ELT SKW]
             (GO SHIFTDOWNLP)))
         (SETQ SELPTS (WINDOWPROP SKW 'SKETCH.SELECTIONS))
         (for SEL in SELPTS do (SK.REMOVE.PT.SELECTION SEL SKW))
         (SK.TAKE.MARKS.DOWN SKW HOTSPOTCACHE)
         (RETURN SELPTS])
```

(SK.CONTROL.POINTS.IN.REGION

```
  [LAMBDA (HOTSPOTCACHE LEFT BOTTOM RIGHT TOP)               (* rrb " 6-May-85 16:22")

          (* * returns a list of the control points that are within LOCALREGION)

    (PROG ((RLEFT (DIFFERENCE LEFT SK.POINT.WIDTH))
           (RBOTTOM (DIFFERENCE BOTTOM SK.POINT.WIDTH))
           (RRIGHT (PLUS RIGHT SK.POINT.WIDTH))
           (RTOP (PLUS TOP SK.POINT.WIDTH))
           ELTS)
          [for YBUCKET in HOTSPOTCACHE when (ILEQ (CAR YBUCKET)
                                                  RTOP)
             do (COND
                   ((ILESSP (CAR YBUCKET)
                            RBOTTOM)                         (* stop when Y gets too small.)
                    (RETURN)))
                (for XBUCKET in (CDR YBUCKET) when (ILEQ (CAR XBUCKET)
                                                         RRIGHT)
                   do (COND
                         ((ILESSP (CAR XBUCKET)
                                  RLEFT)                     (* stop when X gets too small.)
                          (RETURN)))                         (* collect the points if there are any elements cached there.)
                      (AND (CDR XBUCKET)
                           (SETQ ELTS (SK.ADD.POINT ELTS (CAR XBUCKET)
                                                    (CAR YBUCKET]
          (RETURN ELTS])
```

(SK.ADD.PT.SELECTION

```
  [LAMBDA (PT WINDOW MARKBM)                                 (* rrb " 9-May-85 10:18")
                                                             (* adds an item to the selection list of WINDOW.)
    (COND
       ([NOT (MEMBER PT (WINDOWPROP WINDOW 'SKETCH.SELECTIONS]
        (MARKPOINT PT WINDOW MARKBM)
```

```
        (WINDOWADDPROP WINDOW 'SKETCH.SELECTIONS PT])
```

(SK.REMOVE.PT.SELECTION

```
  [LAMBDA (PT WINDOW MARKBM)                                  (* rrb " 9-May-85 10:22")
                                                              (* removes an item from the selection list of WINDOW.)
    (COND
       ((MEMBER PT (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))
        (MARKPOINT PT WINDOW MARKBM)

          (* used to call WINDOWDELPROP but it has a bug that it only removes EQ things.)

        (WINDOWPROP WINDOW 'SKETCH.SELECTIONS (REMOVE PT (WINDOWPROP WINDOW 'SKETCH.SELECTIONS])
```

(SK.ADD.POINT

```
  [LAMBDA (PTLST X Y)                                         (* rrb " 6-May-85 16:22")
                                                              (* add the point X Y to PTLST unless it is already a member.)
    (COND
       ((for PT in PTLST thereis (AND (EQ (fetch (POSITION XCOORD) of PT)
                                          X)
                                      (EQ (fetch (POSITION YCOORD) of PT)
                                          Y)))
        PTLST)
       (T (CONS (create POSITION
                        XCOORD _ X
                        YCOORD _ Y)
                PTLST])
```

(SK.ELTS.CONTAINING.PTS

```
  [LAMBDA (PTLST SKW)                                         (* rrb " 4-May-85 15:38")
                                                              (* returns the list of elements that have any points on PTLST.)
    (bind (HOTSPOTCACHE _ (SK.HOTSPOT.CACHE SKW))
          ELTS for POS in PTLST do (SETQ ELTS (UNION (SK.ELTS.FROM.HOTSPOT POS HOTSPOTCACHE)
                                                     ELTS))
       finally                                                (* reverse them so the first selected pt has its element first.)
             (RETURN (REVERSE ELTS])
```

(SK.HOTSPOTS.NOT.ON.LIST

```
  [LAMBDA (PTLST ELTS)                                        (* rrb "19-Jul-85 13:18")

          (* returns a list of the hot spots on any of ELTS that aren't on PTLST.)

    (bind OTHERHOTSPOTS for ELT in ELTS do [for HOTSPOT in (fetch (SCREENELT HOTSPOTS) of ELT)
                                              do (OR (MEMBER HOTSPOT PTLST)
                                                     (MEMBER HOTSPOT OTHERHOTSPOTS)
                                                     (SETQ OTHERHOTSPOTS (CONS HOTSPOT OTHERHOTSPOTS]
       finally (RETURN OTHERHOTSPOTS])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS .SHIFTKEYDOWNP. MACRO [NIL (OR (KEYDOWNP 'LSHIFT)
                                         (KEYDOWNP 'RSHIFT])
)

(DEFINEQ
```

(SK.SET.MOVE.MODE

```
  [LAMBDA (SKW NEWMODE)                                       (* rrb " 2-Jun-85 12:52")

          (* * reads a value of move command mode and makes it the default)

    (PROG [(LOCALNEWMODE (OR NEWMODE (READMOVEMODE]
          (RETURN (AND LOCALNEWMODE (replace (SKETCHCONTEXT SKETCHMOVEMODE) of (WINDOWPROP SKW 'SKETCHCONTEXT)
                                       with (SELECTQ NEWMODE
                                                  ((POINTS ELEMENTS)
                                                       NEWMODE)
                                                  NIL])
```

(SK.SET.MOVE.MODE.POINTS

```
  [LAMBDA (SKW)                                               (* rrb " 2-Jun-85 12:47")
                                                              (* sets the default to move mode to points.)
    (SK.SET.MOVE.MODE SKW 'POINTS])
```

(SK.SET.MOVE.MODE.ELEMENTS

```
  [LAMBDA (SKW)                                               (* rrb " 2-Jun-85 12:48")
                                                              (* sets the default to move mode to elements)
    (SK.SET.MOVE.MODE SKW 'ELEMENTS])
```

(SK.SET.MOVE.MODE.COMBINED

```
  [LAMBDA (SKW)                                                  (* rrb " 2-Jun-85 12:49")
                                                                 (* sets the default to move mode to combined move.)
    (SK.SET.MOVE.MODE SKW 'COMBINED])
```

(READMOVEMODE

```
  [LAMBDA (MENUTITLE)                                            (* rrb " 6-Nov-85 09:54")
                                                                 (* interacts to get whether move mode should be points,
                                                                 elements or both.)
    (\CURSOR.IN.MIDDLE.MENU (create MENU
                                    TITLE _ (OR MENUTITLE "Top level MOVE command should apply to?")
                                    ITEMS _ '((Points 'POINTS "Top level MOVE command will be the same as MOVE
                                                    POINTS command.")
                                              (Elements 'ELEMENTS "Top level MOVE command will be the same as
                                                    MOVE ELEMENTS command.")
                                              (Combined 'COMBINED "MOVE command will move points if a single
                                                    point is clicked; elements otherwise"))
                                    CENTERFLG _ T])

)

(DEFINEQ

(SK.ALIGN.POINTS
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:50")
                                                                 (* lets the user select a collection of points and aligns them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE.POINTS (KWOTE W))
           W])

(SK.SEL.AND.ALIGN.POINTS
  [LAMBDA (ALIGNHOW W)                                           (* rrb "22-Jan-86 14:57")

          (* * lets the user select a collection of control point and aligns them.)

    (SK.DO.ALIGN.POINTS (SK.SELECT.MULTIPLE.POINTS W)
           ALIGNHOW W])

(SK.ALIGN.POINTS.LEFT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:51")
                                                                 (* lets the user select a collection of points and aligns them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''LEFT (KWOTE W))
           W])

(SK.ALIGN.POINTS.RIGHT
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:51")
                                                                 (* lets the user select a collection of points and aligns them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''RIGHT (KWOTE W))
           W])

(SK.ALIGN.POINTS.TOP
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:57")
                                                                 (* lets the user select a collection of points and aligns them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''TOP (KWOTE W))
           W])

(SK.ALIGN.POINTS.BOTTOM
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:58")
                                                                 (* lets the user select a collection of points and aligns them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''BOTTOM (KWOTE W))
           W])

(SK.EVEN.SPACE.POINTS.IN.X
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:58")
                                                                 (* lets the user select a collection of points and spaces them
                                                                 evenly in X)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''EVENX (KWOTE W))
           W])

(SK.EVEN.SPACE.POINTS.IN.Y
  [LAMBDA (W)                                                    (* rrb "31-Jan-86 10:58")
                                                                 (* lets the user select a collection of points and spaces them
                                                                 evenly in Y)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.ALIGN.POINTS ''EVENY (KWOTE W))
           W])

(SK.DO.ALIGN.POINTS
  [LAMBDA (SCRPTS ALIGNHOW SKW)                                  (* rrb "12-Sep-86 18:28")

          (* * aligns a collection of points according to ALIGNHOW which can be LEFT RIGHT TOP BOTTOM EVENX or EVENY)
```

```
(SKED.CLEAR.SELECTION SKW)                                  (* if there isn't at least two points, don't do anything.)
(AND (CDR SCRPTS)
     (PROG ((SELECTEDPTSTRUC (SK.GET.SELECTED.ELEMENT.STRUCTURE SCRPTS SKW))
            MOSTSELBUCK LEASTSELBUCK DIMENSION LEAST MOST PREMOVEFN X NEWGLOBALS)
           (AND (SETQ PREMOVEFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                               'PREMOVEFN))
                (EQ (APPLY* PREMOVEFN SKW (SK.MAKE.POINTS.AND.ELEMENTS.MOVE.ARG (SK.ELTS.CONTAINING.PTS
                                                                                  SCRPTS SKW)
                                                                                SCRPTS)
                                          ALIGNHOW)
                    'DON'T)
                (RETURN))
           (SETQ MOSTSELBUCK (CAR SELECTEDPTSTRUC))
           (SETQ LEASTSELBUCK (CAR SELECTEDPTSTRUC))        (* find the dimension of interest and do some error checking.)
           (SETQ DIMENSION (SELECTQ ALIGNHOW
                                    ((LEFT RIGHT)
                                        'HORIZONTAL)
                                    ((TOP BOTTOM)
                                        'VERTICAL)
                                    (EVENX (OR (CDDR SCRPTS)
                                               (RETURN))
                                           'HORIZONTAL)
                                    (EVENY (OR (CDDR SCRPTS)
                                               (RETURN))
                                           'VERTICAL)
                                    (SHOULDNT)))            (* calculate the extreme points.)
           [COND
              [(EQ DIMENSION 'HORIZONTAL)
               (for PT in (CDR SELECTEDPTSTRUC) do (COND
                                                     ((GREATERP (fetch (POSITION XCOORD) of (CAR PT))
                                                                (fetch (POSITION XCOORD) of (CAR MOSTSELBUCK)))
                                                      (SETQ MOSTSELBUCK PT))
                                                     ((LESSP (fetch (POSITION XCOORD) of (CAR PT))
                                                             (fetch (POSITION XCOORD) of (CAR LEASTSELBUCK)))
                                                      (SETQ LEASTSELBUCK PT]
              (T (for PT in (CDR SELECTEDPTSTRUC) do (COND
                                                       ((GREATERP (fetch (POSITION YCOORD)
                                                                     of (CAR PT))
                                                                  (fetch (POSITION YCOORD) of (CAR MOSTSELBUCK)))
                                                        (SETQ MOSTSELBUCK PT))
                                                       ((LESSP (fetch (POSITION YCOORD) of (CAR PT))
                                                               (fetch (POSITION YCOORD) of (CAR LEASTSELBUCK))
                                                               )
                                                        (SETQ LEASTSELBUCK PT]
                                                     (* find the extreme dimensions in global space)
           (SELECTQ DIMENSION
                (HORIZONTAL (SETQ LEAST (fetch (POSITION XCOORD) of (CAADR LEASTSELBUCK)))
                            [for GELTSTR in (CDDR LEASTSELBUCK) when (LESSP (fetch (POSITION XCOORD)
                                                                                   of (CAR GELTSTR))
                                                                            LEAST)
                               do (SETQ LEAST (fetch (POSITION XCOORD) of (CAR GELTSTR]
                            (SETQ MOST (fetch (POSITION XCOORD) of (CAADR MOSTSELBUCK)))
                            [for GELTSTR in (CDDR MOSTSELBUCK) when (GREATERP (fetch (POSITION XCOORD)
                                                                                     of (CAR GELTSTR))
                                                                              MOST)
                               do (SETQ MOST (fetch (POSITION XCOORD) of (CAR GELTSTR])
                (VERTICAL (SETQ LEAST (fetch (POSITION YCOORD) of (CAADR LEASTSELBUCK)))
                          [for GELTSTR in (CDDR LEASTSELBUCK) when (LESSP (fetch (POSITION YCOORD)
                                                                                 of (CAR GELTSTR))
                                                                          LEAST)
                             do (SETQ LEAST (fetch (POSITION YCOORD) of (CAR GELTSTR]
                          (SETQ MOST (fetch (POSITION YCOORD) of (CAADR MOSTSELBUCK)))
                          [for GELTSTR in (CDDR MOSTSELBUCK) when (GREATERP (fetch (POSITION YCOORD)
                                                                                   of (CAR GELTSTR))
                                                                            MOST)
                             do (SETQ MOST (fetch (POSITION YCOORD) of (CAR GELTSTR])
                NIL)                                        (* move the individual elements)
           (SETQ NEWGLOBALS
            (SELECTQ ALIGNHOW
                 ((LEFT BOTTOM)
                      (for SELBUCK in SELECTEDPTSTRUC join      (* apply the movement to each selected element)
                                                                (SK.DO.ALIGN.SETVALUE SELBUCK LEAST DIMENSION SKW)))
                 ((RIGHT TOP)
                      (for SELBUCK in SELECTEDPTSTRUC join      (* apply the movement to each selected element)
                                                                (SK.DO.ALIGN.SETVALUE SELBUCK MOST DIMENSION SKW)))
                 ((EVENX EVENY)
                      (for SELBUCK in [SORT SELECTEDPTSTRUC
                                            (COND
                                               [(EQ DIMENSION 'HORIZONTAL)
                                                                (* sort the selected points)
                                                (FUNCTION (LAMBDA (A B)
                                                            (OR (LESSP (fetch (POSITION XCOORD)
                                                                          of (CAR A))
                                                                       (fetch (POSITION XCOORD)
                                                                          of (CAR B)))
                                                                (AND (EQUAL (fetch (POSITION XCOORD)
```

```
                                                                    of (CAR A))
                                                                (fetch (POSITION XCOORD)
                                                                   of (CAR B)))
                                                            (LESSP (fetch (POSITION YCOORD)
                                                                      of (CAR A))
                                                                   (fetch (POSITION YCOORD)
                                                                      of (CAR B]
                                                    (T (FUNCTION (LAMBDA (A B)
                                                                   (OR (LESSP (fetch (POSITION YCOORD)
                                                                                 of (CAR A))
                                                                              (fetch (POSITION YCOORD)
                                                                                 of (CAR B)))
                                                                       (AND (EQUAL (fetch (POSITION YCOORD)
                                                                                      of (CAR A))
                                                                                   (fetch (POSITION YCOORD)
                                                                                      of (CAR B)))
                                                                            (LESSP (fetch (POSITION XCOORD)
                                                                                      of (CAR A))
                                                                                   (fetch (POSITION XCOORD)
                                                                                      of (CAR B]
                                as VALUE from LEAST to MOST by (FQUOTIENT (DIFFERENCE MOST LEAST)
                                                                          (SUB1 (LENGTH SELECTEDPTSTRUC)))
                                join                                    (* apply the movement to each selected element)
                                     (SK.DO.ALIGN.SETVALUE SELBUCK VALUE DIMENSION SKW)))
                          NIL))
                (AND NEWGLOBALS (SK.ADD.HISTEVENT 'MOVE NEWGLOBALS SKW))
                (CLOSEPROMPTWINDOW SKW])
```

(SK.NTH.CONTROL.POINT

```
[LAMBDA (ELEMENT N)                                     (* returns the nth control point of ELEMENT.)
  (SELECTQ N
       (1 (GETSKETCHELEMENTPROP ELEMENT 'POSITION))
       (2 (GETSKETCHELEMENTPROP ELEMENT '2NDCONTROLPT))
       (3 (GETSKETCHELEMENTPROP ELEMENT '3RDCONTROLPT))
       (CAR (NTH (GETSKETCHELEMENTPROP ELEMENT 'DATA)
                 N])
```

(SK.GET.SELECTED.ELEMENT.STRUCTURE

```
[LAMBDA (SELPTS SKW)                                    (* rrb "22-Jan-86 14:58")

          (* returns a list of the points and elements that each selected point on SELPTS corresponds to.
          Returns a list of lists of the form (SELPT (GPT1 GELT1) |...| (GPTn GELTn)))

    (bind (HOTSPOTCACHE _ (SK.HOTSPOT.CACHE SKW)) for POS in SELPTS
       collect (CONS POS (for ELT in (SK.ELTS.FROM.HOTSPOT POS HOTSPOTCACHE) collect (LIST (
                                                                                    SK.CORRESPONDING.CONTROL.PT
                                                                                      POS ELT)
                                                                                   ELT])
```

(SK.CORRESPONDING.CONTROL.PT

```
[LAMBDA (SELPT SCRELEMENT)                              (* rrb "22-Jan-86 14:59")

          (* returns the global control point of an element that corresponds to the screen point SELPT.)

    (for I from 1 as PT in (fetch (SCREENELT HOTSPOTS) of SCRELEMENT) when (EQUAL PT SELPT)
       do (RETURN (OR (SK.NTH.CONTROL.POINT (fetch (SCREENELT GLOBALPART) of SCRELEMENT)
                                            I)
                      (SHOULDNT])
```

(SK.CONTROL.POINT.NUMBER

```
[LAMBDA (SELPT SCRELT)                                  (* rrb "22-Jan-86 10:54")
                                                        (* returns the control point number that SELPT is on the element
    SCRELT)
    (for I from 1 as HOTPT in (fetch (SCREENELT HOTSPOTS) of SCRELT) when (EQUAL SELPT HOTPT)
       do (RETURN I])
```

(SK.DO.ALIGN.SETVALUE

```
[LAMBDA (SELBUCKET VALUE DIMENSION VIEWER)              (* rrb "22-Jan-86 17:23")
                                                        (* performs the alignment of a selection bucket structure.)
    (bind (SELPT _ (CAR SELBUCKET))
          (MOVEFN _ (GETSKETCHPROP (INSURE.SKETCH VIEWER)
                                   'WHENMOVEDFN))
       GDELTA X for GELTSTRUC in (CDR SELBUCKET)
       when (PROG NIL

          (* calculate the amount that this global element point should be moved and apply move fn)
                                                        (* don't move it if it moves 0.0)
                [SETQ GDELTA (create POSITION
                                     XCOORD _ (COND
                                                ((EQ DIMENSION 'HORIZONTAL)
                                                 (COND
                                                    ([ZEROP (SETQ X (DIFFERENCE VALUE (fetch (POSITION XCOORD)
```

```
                                                                                                    of (CAR GELTSTRUC]
                                                                  (RETURN))
                                                                 (T X)))
                                                             (T 0))
                                               YCOORD _ (COND
                                                             ((EQ DIMENSION 'VERTICAL)
                                                              (COND
                                                                 ([ZEROP (SETQ X (DIFFERENCE VALUE (fetch (POSITION YCOORD)
                                                                                                    of (CAR GELTSTRUC]
                                                                  (RETURN))
                                                                 (T X)))
                                                             (T 0]
                    (COND
                       ((NULL MOVEFN)
                        (RETURN T)))
                    (SETQ X (APPLY* MOVEFN VIEWER [LIST (LIST (SK.CONTROL.POINT.NUMBER SELPT (CADR GELTSTRUC))
                                                              (fetch (SCREENELT GLOBALPART) of (CADR GELTSTRUC]
                                  GDELTA))
                    (COND
                       ((EQ X 'DON'T)                                    (* if DON'T, don't move this guy.)
                        (RETURN NIL))
                       ((POSITIONP X)

            (* value returned is the delta by which to move the point. Set up new position)

                        (SETQ GDELTA X)))
                    (RETURN T))
          join

            (* build the history structure here because this is where the old screen element is known.)

             (AND (SETQ X (SK.MOVE.ITEM.POINTS (CADR GELTSTRUC)
                                 GDELTA VIEWER (LIST SELPT)))
                  (CONS (LIST (fetch (SCREENELT GLOBALPART) of (CADR GELTSTRUC))
                              X])

)
```

;; stuff for supporting the GROUP sketch element.

```
(DEFINEQ
```

(SKETCH.CREATE.GROUP

```
  [LAMBDA (LISTOFSKETCHELEMENTS CONTROLPOINT)                (* rrb " 4-Dec-85 21:38")
                                                             (* creates a sketch group element.)
    (SK.CREATE.GROUP1 LISTOFSKETCHELEMENTS (OR (POSITIONP CONTROLPOINT)
                                               (REGION.CENTER (SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS
                                                                   LISTOFSKETCHELEMENTS])
```

(SK.CREATE.GROUP1

```
  [LAMBDA (GELTS CONTROLPT)                                  (* rrb " 4-Dec-85 21:38")
                                                             (* creates a group element.)
    (SK.UPDATE.GROUP.AFTER.CHANGE (create GLOBALPART
                                          INDIVIDUALGLOBALPART _ (create GROUP
                                                                         LISTOFGLOBALELTS _ GELTS
                                                                         GROUPCONTROLPOINT _ CONTROLPT])
```

(SK.UPDATE.GROUP.AFTER.CHANGE

```
  [LAMBDA (GROUPELT)                                         (* rrb " 4-Dec-85 21:38")
                                                             (* updates the dependent field of a group element after a
                                                             change.)
    (PROG ((INDGROUPELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT))
           GROUPREGION)
          (SETQ GROUPREGION (SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS (fetch (GROUP LISTOFGLOBALELTS) of INDGROUPELT)
                                   ))
          (replace (GROUP GROUPREGION) of INDGROUPELT with GROUPREGION)
                                                             (* use same scales as a box would.)
          (BOX.SET.SCALES GROUPREGION GROUPELT)
          (RETURN GROUPELT])
```

(SK.GROUP.ELTS

```
  [LAMBDA (W)                                                (* rrb "31-Jan-86 10:58")
                                                             (* lets the user select a collection elements and groups them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.GROUP (KWOTE W))
           W])
```

(SK.SEL.AND.GROUP

```
  [LAMBDA (W)                                                (* rrb "10-Dec-85 17:08")
                                                             (* lets the user select elements and groups them.)
    (SK.GROUP.ELEMENTS (SK.SELECT.MULTIPLE.ITEMS W T NIL 'GROUP)
           W])
```

(SK.GROUP.ELEMENTS

```
[LAMBDA (SCRELTS SKW)                                   (* rrb "11-Jul-86 15:51")

        (* groups the collection of elements SCRELTS. Does this by creating a group element, adding it and deleting the individual
        elements.)

    (SKED.CLEAR.SELECTION SKW)
    (AND SCRELTS (PROG (GROUPELT LOCALGROUPELT)          (* call the group fn if there is one.)
                       (SETQ GROUPELT (SKETCH.CREATE.GROUP (for SCRELT in SCRELTS
                                                              collect (fetch (SCREENELT GLOBALPART) of SCRELT))
                                                           (MAP.GLOBAL.PT.ONTO.GRID (REGION.CENTER (
                                                                                       SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS
                                                                                             SCRELTS
                                                                                             (VIEWER.SCALE SKW)))
                                                                                    SKW)))
                                                        (* do grouping. This might return NIL if the when grouped
                                                        function says not to.)
                       (OR (SK.DO.GROUP GROUPELT SKW)
                           (RETURN))                    (* record it on the history list.)
                       (SK.ADD.HISTEVENT 'GROUP (LIST (LIST GROUPELT))
                              SKW)
                       (RETURN GROUPELT])
```

(SK.UNGROUP.ELT

```
[LAMBDA (W)                                             (* rrb "31-Jan-86 10:58")
                                                        (* lets the user select a collection elements and groups them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.UNGROUP (KWOTE W))
           W])
```

(SK.SEL.AND.UNGROUP

```
[LAMBDA (W)                                             (* rrb "10-Dec-85 18:03")
                                                        (* lets the user select elements and groups them.)
    (PROG NIL
          (RETURN (SK.UNGROUP.ELEMENT [SK.SELECT.MULTIPLE.ITEMS
                                       W T (COND
                                              [(SUBSET (LOCALSPECS.FROM.VIEWER W)
                                                       (FUNCTION (LAMBDA (SCRELT)
                                                                   (AND (EQ (fetch (SCREENELT GTYPE)
                                                                               of SCRELT)
                                                                            'GROUP)
                                                                        (NOT (SK.ELEMENT.PROTECTED?
                                                                               (fetch (SCREENELT GLOBALPART)
                                                                                  of SCRELT)
                                                                               'UNGROUP]
                                              (T                (* no group elements)
                                                 (STATUSPRINT W "There are no grouped elements to ungroup.")
                                                 (RETURN]
                                       W])
```

(SK.UNGROUP.ELEMENT

```
[LAMBDA (SCRELTS SKW)                                   (* rrb "15-Jan-86 16:12")
                                                        (* ungroups the first group element in SCRELTS.)
    (PROG ((GROUPELTS (for ELT in SCRELTS when (EQ (fetch (SCREENELT GTYPE) of ELT)
                                                  'GROUP)
                         collect (fetch (SCREENELT GLOBALPART) of ELT)))
           X)
          (OR GROUPELTS (RETURN))                       (* do the ungrouping. this may return NIL if the ungroup fn says
    don't.)
          (SETQ X (for GROUPELT in GROUPELTS when (SK.DO.UNGROUP GROUPELT SKW) collect (LIST GROUPELT)))
          (AND X (SK.ADD.HISTEVENT 'UNGROUP X SKW])
```

(SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS

```
[LAMBDA (SCRELTS SCALE)                                 (* rrb "30-Sep-86 18:33")
                                                        (* returns the global region occuppied by a list of local elements.)
    (PROG (GROUPREGION)
          [for SCRELT in SCRELTS do (SETQ GROUPREGION (COND
                                                        (GROUPREGION
                                                                (* first time because UNIONREGIONS doesn't handle NIL)
                                                          (SK.UNIONREGIONS GROUPREGION (SK.ITEM.REGION
                                                                                         SCRELT)))
                                                        (T (SK.ITEM.REGION SCRELT]
          (RETURN (UNSCALE.REGION GROUPREGION SCALE])
```

(SK.LOCAL.REGION.OF.LOCAL.ELEMENTS

```
[LAMBDA (SCRELTS SCALE)                                 (* rrb "30-Sep-86 18:33")
                                                        (* returns the local region occupied by a list of local elements.)
    (bind GROUPREGION for SCRELT in SCRELTS do [SETQ GROUPREGION (COND
                                                                   (GROUPREGION
                                                                (* first time because UNIONREGIONS doesn't handle NIL)
                                                                      (SK.UNIONREGIONS GROUPREGION
                                                                             (SK.ITEM.REGION SCRELT)))
                                                                   (T (SK.ITEM.REGION SCRELT]
       finally (RETURN GROUPREGION])
```

```
(SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS
  [LAMBDA (GELTS)                                          (* rrb "30-Sep-86 17:35")
                                                           (* returns the global region occuppied by a list of global
                                                           elements.)
    (COND
       [(LESSP (LENGTH GELTS)
               50)

          (* for smallish numbers of elements, only do the cons to create the args to SK.UNIONREGIONS.)

        (APPLY (FUNCTION SK.UNIONREGIONS)
               (for GELT in GELTS collect (SK.ELEMENT.GLOBAL.REGION GELT]
       (T (PROG (GROUPREGION)
                [for GELT in GELTS do (SETQ GROUPREGION (COND
                                                           (GROUPREGION
                                                           (* first time because UNIONREGIONS doesn't handle NIL)
                                                            (SK.UNIONREGIONS GROUPREGION (
                                                                                        SK.ELEMENT.GLOBAL.REGION
                                                                                          GELT)))
                                                           (T (SK.ELEMENT.GLOBAL.REGION GELT]
                (RETURN GROUPREGION])

(SK.UNIONREGIONS
  [LAMBDA REGIONS                                          (* rrb "30-Sep-86 18:14")

          (* returns the smallest region that encloses all of REGIONS Is different from UNIONREGIONS because it works in floating
          pt)

    (COND
       ((EQ 0 REGIONS)
        NIL)
       (T (PROG (REG LFT RGHT BTTM TP X NEWLFT NEWBTM)
                (SETQ REG (ARG REGIONS 1))
                (SETQ LFT (fetch (REGION LEFT) of REG))
                (SETQ RGHT (PLUS LFT (fetch (REGION WIDTH) of REG)))
                (SETQ BTTM (fetch (REGION BOTTOM) of REG))
                (SETQ TP (PLUS BTTM (fetch (REGION HEIGHT) of REG)))
                [for I from 2 thru REGIONS do (SETQ REG (ARG REGIONS I))
                                              (COND
                                                 ((LESSP (SETQ X (fetch (REGION LEFT) of REG))
                                                         LFT)
                                                  (SETQ LFT X)))
                                              (COND
                                                 ((GREATERP (SETQ X (PLUS X (fetch (REGION WIDTH) of REG)))
                                                            RGHT)
                                                  (SETQ RGHT X)))
                                              (COND
                                                 ((LESSP (SETQ X (fetch (REGION BOTTOM) of REG))
                                                         BTTM)
                                                  (SETQ BTTM X)))
                                              (COND
                                                 ((GREATERP (SETQ X (PLUS X (fetch (REGION HEIGHT) of REG)))
                                                            TP)
                                                  (SETQ TP X]
                (RETURN (create REGION
                                LEFT _ LFT
                                BOTTOM _ BTTM
                                WIDTH _ (DIFFERENCE RGHT LFT)
                                HEIGHT _ (DIFFERENCE TP BTTM])

(SKETCH.REGION.OF.SKETCH
  [LAMBDA (SKETCH)                                         (* rrb "23-Oct-85 11:17")
                                                           (* returns the global region of a sketch.)
    (SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH SKETCH])

(SK.FLASHREGION
  [LAMBDA (REGION WINDOW TEXTURE)                          (* rrb "30-Jul-85 15:47")
                                                           (* flashes a region)
    (DSPFILL REGION TEXTURE 'INVERT WINDOW)
    (DISMISS 400)
    (DSPFILL REGION TEXTURE 'INVERT WINDOW])

)

(DEFINEQ

(INIT.GROUP.ELEMENT
  [LAMBDA NIL                                              (* rrb "18-Oct-85 17:15")
                                                           (* initializes the text box element.)
    (COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'GROUP))
        (CREATE.SKETCH.ELEMENT.TYPE 'GROUP NIL "groups a collection of elements as a single element."
```

```
              (FUNCTION GROUP.DRAWFN)
              (FUNCTION GROUP.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION TEXTBOX.INPUTFN)
              (FUNCTION GROUP.INSIDEFN)
              (FUNCTION GROUP.REGIONFN)
              (FUNCTION GROUP.TRANSLATEFN)
              NIL
              (FUNCTION GROUP.READCHANGEFN)
              (FUNCTION GROUP.TRANSFORMFN)
              NIL
              (FUNCTION GROUP.GLOBALREGIONFN])

(GROUP.DRAWFN
  [LAMBDA (GROUPELT WINDOW REGION OPERATION)                 (* rrb "10-Dec-85 12:38")
                                                             (* draws a group element.)
    (for ELT in (fetch (LOCALGROUP LOCALELEMENTS) of (fetch (SCREENELT LOCALPART) of GROUPELT))
       do (APPLY* (SK.DRAWFN (fetch (SCREENELT GTYPE) of ELT))
                 ELT WINDOW REGION OPERATION])

(GROUP.EXPANDFN
  [LAMBDA (GROUPELT SCALE STREAM)                            (* rrb "30-Dec-85 17:30")
                                                             (* creates a local group screen element from a global group
                                                             element)
    (PROG ((GROUPINDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT))
           LOCALREGION)
          (SETQ LOCALREGION (SCALE.REGION.OUT (fetch (GROUP GROUPREGION) of GROUPINDVELT)
                                   SCALE))                   (* put the position in the center.)
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALGROUP
                                            GROUPPOSITION _ (SK.SCALE.POSITION.INTO.VIEWER (fetch (GROUP
                                                                                          GROUPCONTROLPOINT
                                                                                                 )
                                                                                        of GROUPINDVELT)
                                                                   SCALE)
                                            LOCALGROUPREGION _ LOCALREGION
                                            LOCALELEMENTS _ (for ELEMENT in (fetch (GROUP LISTOFGLOBALELTS)
                                                                               of GROUPINDVELT)
                                                              collect (SK.LOCAL.FROM.GLOBAL ELEMENT STREAM
                                                                             SCALE)))
                         GLOBALPART _ GROUPELT])

(GROUP.INSIDEFN
  [LAMBDA (GROUPELT WREG)                                    (* rrb "10-Jan-85 10:37")
                                                             (* determines if the global group element GROUPELT is inside
                                                             of WREG.)
    (REGIONSINTERSECTP (fetch (GROUP GROUPREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT))
           WREG])

(GROUP.REGIONFN
  [LAMBDA (GROUPSCRELT)                                      (* rrb "10-Dec-85 12:38")
                                                             (* returns the region occuppied by a group)
    (fetch (LOCALGROUP LOCALGROUPREGION) of (fetch (SCREENELT LOCALPART) of GROUPSCRELT])

(GROUP.GLOBALREGIONFN
  [LAMBDA (GGROUPELT)                                        (* rrb "18-Oct-85 17:13")
                                                             (* returns the global region occupied by a global group element.)
    (fetch (GROUP GROUPREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GGROUPELT])

(GROUP.TRANSLATEFN
  [LAMBDA (SKELT DELTAPOS)                                   (* rrb "28-Apr-85 18:43")

          (* * returns a group element which has been translated by DELTAPOS)

    (PROG ((GGROUPELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKELT))
           NEWREG)
          (SETQ NEWREG (REL.MOVE.REGION (fetch (GROUP GROUPREGION) of GGROUPELT)
                              (fetch (POSITION XCOORD) of DELTAPOS)
                              (fetch (POSITION YCOORD) of DELTAPOS)))

          (* makes a copy of the common global part because it includes the scales which may change for one of the instances.)

          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of SKELT))
                         INDIVIDUALGLOBALPART _ (create GROUP
                                                       GROUPREGION _ NEWREG
                                                       LISTOFGLOBALELTS _ (for SUBELT
                                                                            in (fetch (GROUP LISTOFGLOBALELTS)
                                                                                  of GGROUPELT)
                                                                            collect (SK.TRANSLATE.GLOBALPART
```

```
                                                                SUBELT DELTAPOS T))
                                                  GROUPCONTROLPOINT _ (PTPLUS (fetch (GROUP
                                                                                     GROUPCONTROLPOINT
                                                                                     )
                                                                        of GGROUPELT)
                                                                     DELTAPOS])
```

(GROUP.TRANSFORMFN

```
[LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb " 2-Jun-85 13:10")

        (* * returns a group element which has been transformed by TRANSFORMFN)

    (COND
       [(EQ TRANSFORMFN (FUNCTION SK.PUT.ON.GRID))            (* if putting things on a grid, move only the control point.)
        (PROG ((GGROUPELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
               NOWPOS)
              (SETQ NOWPOS (fetch (GROUP GROUPCONTROLPOINT) of GGROUPELT))
              (RETURN (GROUP.TRANSLATEFN GELT (PTDIFFERENCE (SK.TRANSFORM.POINT NOWPOS TRANSFORMFN
                                                                                TRANSFORMDATA)
                                                            NOWPOS]
       (T (PROG ((GGROUPELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
                 NEWREG)
```

(* this transforms the old region to get the new one. This is not as good as recalculating the new one from the transformed elements. The latter is hard because the region function only works on local elements and here we have only global ones.)

```
              (SETQ NEWREG (SK.TRANSFORM.REGION (fetch (GROUP GROUPREGION) of GGROUPELT)
                                TRANSFORMFN TRANSFORMDATA))
```

(* the control point could also profitably be put on a grid point but no other elements points are so done and it would be hard.)

```
              (RETURN (BOX.SET.SCALES NEWREG (create GLOBALPART
                                                     COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                           of GELT)
                                                     INDIVIDUALGLOBALPART _
                                                     (create GROUP
                                                            GROUPREGION _ NEWREG
                                                            LISTOFGLOBALELTS _
                                                            (for SUBELT in (fetch (GROUP LISTOFGLOBALELTS)
                                                                                  of GGROUPELT)
                                                               collect (SK.TRANSFORM.ELEMENT SUBELT TRANSFORMFN
                                                                                  TRANSFORMDATA SCALEFACTOR))
                                                            GROUPCONTROLPOINT _ (SK.TRANSFORM.POINT
                                                                                  (fetch (GROUP GROUPCONTROLPOINT
                                                                                                )
                                                                                     of GGROUPELT)
                                                                                  TRANSFORMFN TRANSFORMDATA])
```

(GROUP.READCHANGEFN

```
[LAMBDA (SKW SCRNELTS)                                        (* rrb "14-May-86 19:38")
                                                              (* reads how the user wants to change a textbox.)
    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ (SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                          TITLE _ "Change which part?"
                                                                          ITEMS _
                                                                          [APPEND
                                                                           (COND
                                                                              [(SKETCHINCOLORP)
                                                                               '(("Brush color" 'BRUSHCOLOR
                                                                                         "changes the color of any
                                                                                         lines or text in the
                                                                                         group.")
                                                                                 ("Filling color" 'FILLINGCOLOR
                                                                                         "changes the filling
                                                                                         color of any boxes or
                                                                                         text boxes in the
                                                                                         group."]
                                                                              (T NIL))
                                                                           '((Arrowheads 'ARROW "allows changing
                                                                                    of arrow head
                                                                                    charactistics.")
                                                                             (Shape 'SHAPE "changes the shape of
                                                                                    the brush")
                                                                             (Size 'SIZE "changes the size of the
                                                                                   lines")
                                                                             (Dashing 'DASHING "changes the
                                                                                    dashing property of the
                                                                                    elements with lines.")
                                                                             (Filling 'FILLING "allows changing
                                                                                    of the fillings.")
                                                                             (Text 'TEXT "allows changing the
                                                                                   properties of the text."]
                                                                          CENTERFLG _ T)))
                                (TEXT                         (* handle TEXT specially because it has several different cases.)
```

```
                              (AND (SETQ HOW (TEXT.READCHANGEFN SKW SCRNELTS T))
                                   (RETURN HOW)))
                         (SIZE (READSIZECHANGE "Change size how?"))
                         (SHAPE (READBRUSHSHAPE))
                         (ARROW (READ.ARROW.CHANGE SCRNELTS))
                         (DASHING (READ.DASHING.CHANGE))
                         (FILLING (READ.FILLING.CHANGE))
                         (BRUSHCOLOR (READ.COLOR.CHANGE "Change line color how?"))
                         (FILLINGCOLOR (READ.COLOR.CHANGE "Change filling color how?" T))
                        NIL))
          (RETURN (AND HOW (LIST ASPECT HOW])

)

(DEFINEQ

(REGION.CENTER
  [LAMBDA (REGION)                                          (* rrb "11-Jan-85 18:22")
                                                            (* returns the center of a region)
    (create POSITION
           XCOORD _ (PLUS (fetch (REGION LEFT) of REGION)
                          (QUOTIENT (fetch (REGION WIDTH) of REGION)
                                    2))
           YCOORD _ (PLUS (fetch (REGION BOTTOM) of REGION)
                          (QUOTIENT (fetch (REGION HEIGHT) of REGION)
                                    2])

(REMOVE.LAST
  [LAMBDA (LST)                                             (* removes the last element from a list.)
    (COND
       ((NULL (CDR LST))
        NIL)
       (T (for TAIL on LST when (NULL (CDDR TAIL)) do (RPLACD TAIL NIL)
                                                      (RETURN LST])

)

;; moving the control point of a group

(DEFINEQ

(SK.MOVE.GROUP.CONTROL.PT
  [LAMBDA (W)                                               (* rrb "31-Jan-86 10:59")
                                                            (* lets the user move the control point of a group.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.MOVE.CONTROL.PT (KWOTE W))
           W])

(SK.SEL.AND.MOVE.CONTROL.PT
  [LAMBDA (W)                                               (* rrb "23-Jan-86 18:11")
                                                            (* lets the user select a groups and move its control point.)
    (PROG NIL
          (RETURN (SK.MOVE.GROUP.ELEMENT.CONTROL.POINT
                   [SK.SELECT.ITEM W T (COND
                                       [(SUBSET (LOCALSPECS.FROM.VIEWER W)
                                                (FUNCTION (LAMBDA (SCRELT)
                                                            (AND (EQ (fetch (SCREENELT GTYPE) of SCRELT)
                                                                     'GROUP)
                                                                 (NOT (SK.ELEMENT.PROTECTED?
                                                                        (fetch (SCREENELT GLOBALPART)
                                                                           of SCRELT)
                                                                        'CHANGE]
                                       (T                   (* no group elements)
                                          (STATUSPRINT W "There are no grouped elements.")
                                          (RETURN]
                   W])

(SK.MOVE.GROUP.ELEMENT.CONTROL.POINT
  [LAMBDA (SCRGROUPELT SKW)                                 (* rrb "27-Jun-86 15:34")
                                                            (* reads a new location of the control point for a group element.)
    (PROG ((GELT (fetch (SCREENELT GLOBALPART) of SCRGROUPELT))
           (INDVGELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of SCRGROUPELT))
           OLDPOS NEWPOS NEWGROUPELT LOCALELT)
          (AND (EQ (SK.CHECK.PRECHANGEFN SKW GELT 'POSITION)
                   'DON'T)
               (RETURN))
          (SETQ OLDPOS (GETSKETCHELEMENTPROP GELT 'POSITION))
          (OR [SETQ NEWPOS (SK.READ.NEW.GROUP.CONTROL.PT SKW (fetch (LOCALGROUP LOCALGROUPREGION)
                                                                of (fetch (SCREENELT LOCALPART) of SCRGROUPELT]
              (RETURN))
          (OR (SK.CHECK.WHENCHANGEDFN SKW GELT 'POSITION NEWPOS OLDPOS)
              (RETURN))
          (SETQ NEWGROUPELT (SKETCH.CREATE.GROUP (fetch (GROUP LISTOFGLOBALELTS) of INDVGELT)
                                                 NEWPOS))
          (SK.SET.ELEMENT.PRIORITY NEWGROUPELT (SK.ELEMENT.PRIORITY GELT))
```

```
          (SK.DELETE.ELEMENT1 GELT SKW T)
          (SETQ LOCALELT (SK.ADD.ELEMENT NEWGROUPELT SKW T T T))
          (SK.FLASHREGION (fetch (LOCALGROUP LOCALGROUPREGION) of (fetch (SCREENELT LOCALPART) of LOCALELT))
                 SKW GRAYSHADE)
          (SK.ADD.HISTEVENT 'CHANGE (LIST (create SKHISTORYCHANGESPEC
                                                  NEWELT _ NEWGROUPELT
                                                  OLDELT _ GELT
                                                  PROPERTY _ 'POSITION
                                                  NEWVALUE _ NEWPOS
                                                  OLDVALUE _ OLDPOS))
                 SKW)
          (RETURN NEWGROUPELT])
```

(SK.READ.NEW.GROUP.CONTROL.PT

```
  [LAMBDA (VIEWER LOCALGROUPREGION)                                  (* rrb "14-Jul-86 13:51")
                                                                     (* reads where the user wants the new control point to be.)
    (PROG (PT)                                                       (* outline the group)
          (SK.DRAWBOX (fetch (REGION LEFT) of LOCALGROUPREGION)
                 (fetch (REGION BOTTOM) of LOCALGROUPREGION)
                 (fetch (REGION WIDTH) of LOCALGROUPREGION)
                 (fetch (REGION HEIGHT) of LOCALGROUPREGION)
                 1
                 'INVERT VIEWER 42405)
          (STATUSPRINT VIEWER "
                 " "Indicate position of the new control point.")
          (SETQ PT (SK.READ.POINT.WITH.FEEDBACK VIEWER NIL NIL NIL NIL NIL SKETCH.USE.POSITION.PAD))
                                                                     (* remove outline of the group)
          (SK.DRAWBOX (fetch (REGION LEFT) of LOCALGROUPREGION)
                 (fetch (REGION BOTTOM) of LOCALGROUPREGION)
                 (fetch (REGION WIDTH) of LOCALGROUPREGION)
                 (fetch (REGION HEIGHT) of LOCALGROUPREGION)
                 1
                 'INVERT VIEWER 42405)
          (RETURN (AND PT (SK.MAP.INPUT.PT.TO.GLOBAL PT VIEWER])

)

(DECLARE%: EVAL@COMPILE

(TYPERECORD GROUP (GROUPREGION LISTOFGLOBALELTS GROUPCONTROLPOINT))

(RECORD LOCALGROUP ((GROUPPOSITION)
                    LOCALHOTREGION LOCALGROUPREGION LOCALELEMENTS))
)
```

;; history and undo stuff for groups

```
(DEFINEQ
```

(SK.DO.GROUP

```
  [LAMBDA (GROUPELT SKW)                                             (* rrb "30-Sep-86 17:38")
                                                                     (* does a group event. Used to undo UNGROUP too.)
    (PROG (LOCALELT OKEDGELTS)
          (OR [SETQ OKEDGELTS (SK.CHECK.WHENGROUPEDFN SKW (fetch (GROUP LISTOFGLOBALELTS)
                                                                 of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                       of GROUPELT]
              (RETURN NIL))
          (replace (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT)
             with (SK.ORDER.ELEMENTS OKEDGELTS))
          (SK.UPDATE.GROUP.AFTER.CHANGE GROUPELT)
          (for GELT in OKEDGELTS do (SK.DELETE.ELEMENT1 GELT SKW T))
          (SETQ LOCALELT (SK.ADD.ELEMENT GROUPELT SKW T T T))        (* flash the grouped area to let user know something happened.)
          (SK.FLASHREGION (fetch (LOCALGROUP LOCALGROUPREGION) of (fetch (SCREENELT LOCALPART) of LOCALELT))
                 SKW GRAYSHADE)
          (RETURN LOCALELT])
```

(SK.CHECK.WHENGROUPEDFN

```
  [LAMBDA (VIEWER ELEMENTS)                                          (* rrb "15-Jan-86 16:07")
                                                                     (* checks the when grouped fn of a sketch viewer.)
    (PROG (GROUPFN X)
          (AND (SETQ GROUPFN (GETSKETCHPROP (INSURE.SKETCH VIEWER)
                                            'WHENGROUPEDFN))
               (SETQ X (APPLY* GROUPFN VIEWER ELEMENTS)))
          (RETURN (COND
                     ((EQ X 'DON'T)
                      NIL)
                     ((SKETCH.LIST.OF.ELEMENTSP X)
                      X)
                     (T ELEMENTS])
```

(SK.DO.UNGROUP

```
  [LAMBDA (GROUPELT SKW)                                             (* rrb "11-Jul-86 15:51")
                                                                     (* does a ungroup event. Used to undo GROUP too.)
    (PROG NIL
```

```
          (OR (SK.CHECK.WHENUNGROUPEDFN SKW GROUPELT)
              (RETURN))
          (SK.DELETE.ELEMENT1 GROUPELT SKW T)
          (for GELT in (fetch (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT))
             do (SK.ADD.ELEMENT GELT SKW T T T))
          (SK.FLASHREGION (SCALE.REGION.OUT (fetch (GROUP GROUPREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                                 of GROUPELT))
                                            (VIEWER.SCALE SKW))
                 SKW GRAYSHADE)
          (RETURN GROUPELT])
```

(SK.CHECK.WHENUNGROUPEDFN

```
  [LAMBDA (VIEWER GROUPELT)                                   (* rrb "15-Jan-86 16:19")
                                                              (* checks the when ungrouped fn of a sketch viewer.)
    (PROG (UNGROUPFN)
          (RETURN (OR [NULL (SETQ UNGROUPFN (GETSKETCHPROP (INSURE.SKETCH VIEWER)
                                                           'WHENUNGROUPEDFN]
                      (NEQ (APPLY* UNGROUPFN VIEWER GROUPELT)
                           'DON'T])
```

(SK.GROUP.UNDO

```
  [LAMBDA (EVENTARGS SKW)                                     (* rrb "15-Jan-86 16:12")
                                                              (* undoes a group event)
    (for GRP in EVENTARGS do (SK.DO.UNGROUP (CAR GRP)
                                            SKW))
    T])
```

(SK.UNGROUP.UNDO

```
  [LAMBDA (EVENTARGS SKW)                                     (* rrb "15-Jan-86 15:47")
                                                              (* undoes a ungroup event)
    (for GRP in EVENTARGS do (SK.DO.GROUP (CAR GRP)
                                          SKW))
    T])

)

(PUTPROPS GROUP EVENTFNS (SK.GROUP.UNDO SK.TYPE.OF.FIRST.ARG SK.UNGROUP.UNDO))

(PUTPROPS UNGROUP EVENTFNS (SK.UNGROUP.UNDO SK.TYPE.OF.FIRST.ARG SK.GROUP.UNDO))

;; stuff for supporting the freezing of elements

(DEFINEQ
```

(SK.FREEZE.ELTS

```
  [LAMBDA (W)                                                 (* rrb "31-Jan-86 10:59")
                                                              (* lets the user select a collection elements and freezes them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.FREEZE (KWOTE W))
           W])
```

(SK.SEL.AND.FREEZE

```
  [LAMBDA (W)                                                 (* rrb "11-Dec-85 15:30")
                                                              (* lets the user select elements and freezes them.)
    (SK.FREEZE.ELEMENTS (SK.SELECT.MULTIPLE.ITEMS W T NIL 'FROZEN)
           W])
```

(SK.FREEZE.ELEMENTS

```
  [LAMBDA (SCRELTS SKW)                                       (* rrb "11-Dec-85 15:30")
                                                              (* freezes the collection of elements SCRELTS.)
    (PROG (GELTS GELT)
          (OR (SETQ GELTS (for SCRELT in SCRELTS collect (fetch (SCREENELT GLOBALPART) of SCRELT)))
              (RETURN))
          (SK.DO.FREEZE GELTS SKW)
          (SK.ADD.HISTEVENT 'FREEZE GELTS SKW])
```

(SK.UNFREEZE.ELT

```
  [LAMBDA (W)                                                 (* rrb "31-Jan-86 10:59")
                                                              (* lets the user select a collection elements and unfreezes them.)
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.UNFREEZE (KWOTE W))
           W])
```

(SK.SEL.AND.UNFREEZE

```
  [LAMBDA (W)                                                 (* rrb "12-Dec-85 12:25")
                                                              (* lets the user select elements and freezes them.)
    (PROG NIL
          (RETURN (SK.UNFREEZE.ELEMENTS [SK.SELECT.MULTIPLE.ITEMS
                                         W T (COND
                                                [(SUBSET (LOCALSPECS.FROM.VIEWER W)
                                                         (FUNCTION (LAMBDA (SCRELT)
                                                                     (EQMEMB 'FROZEN (GETSKETCHELEMENTPROP
```

```
                                                                    (fetch (SCREENELT
                                                                              GLOBALPART)
                                                                       of SCRELT)
                                                                    'PROTECTION]
                                                  (T                (* no group elements)
                                                     (STATUSPRINT W "There are no frozen elements to unprotect.")
                                                     (RETURN]
                          W])
```

(SK.UNFREEZE.ELEMENTS

```
  [LAMBDA (SCRELTS SKW)                                             (* rrb "11-Dec-85 15:30")
                                                                    (* unfreezes the collection of elements SCRELTS.)
    (PROG (GELTS GELT)
          (OR (SETQ GELTS (for SCRELT in SCRELTS collect (fetch (SCREENELT GLOBALPART) of SCRELT)))
              (RETURN))
          (SK.DO.UNFREEZE GELTS SKW)
          (SK.ADD.HISTEVENT 'UNFREEZE GELTS SKW])
```

(SK.FREEZE.UNDO

```
  [LAMBDA (EVENTARGS SKW)                                           (* rrb "11-Dec-85 15:28")
                                                                    (* undoes a freeze event)
    (SK.DO.UNFREEZE EVENTARGS SKW])
```

(SK.UNFREEZE.UNDO

```
  [LAMBDA (EVENTARGS SKW)                                           (* rrb "11-Dec-85 15:28")
                                                                    (* undoes a unfreeze event)
    (SK.DO.FREEZE EVENTARGS SKW])
```

(SK.DO.FREEZE

```
  [LAMBDA (GELTS SKW)                                               (* rrb "11-Dec-85 15:27")
                                                                    (* does a freeze event. Used to undo UNFREEZE too.)
    (for GELT in GELTS do (ADDSKETCHELEMENTPROP GELT 'PROTECTION 'FROZEN))
    GELTS])
```

(SK.DO.UNFREEZE

```
  [LAMBDA (GELTS SKW)                                               (* rrb "11-Dec-85 15:27")
                                                                    (* does a unfreeze event. Used to undo FREEZE too.)
    (for GELT in GELTS do (REMOVESKETCHELEMENTPROP GELT 'PROTECTION 'FROZEN))
    GELTS])

)

(PUTPROPS FREEZE EVENTFNS (SK.FREEZE.UNDO SK.TYPE.OF.FIRST.ARG SK.UNFREEZE.UNDO))

(PUTPROPS UNFREEZE EVENTFNS (SK.UNFREEZE.UNDO SK.TYPE.OF.FIRST.ARG SK.FREEZE.UNDO))
```

;; programmer interface entries

```
(DEFINEQ
```

(SKETCH.ELEMENTS.OF.SKETCH

```
  [LAMBDA (SKETCH)                                                  (* rrb " 2-Aug-85 16:21")

          (* Returns the list of elements that are in SKETCH. SKETCH can be either a SKETCH structure, a sketch window
          (sometimes called a viewer) or a SKETCH stream (obtained via
          (OPENIMAGESTREAM (QUOTE name) (QUOTE SKETCH))%.
          If SKETCH is not a sketch, a sketch window or a sketch stream, it returns NIL.
          This can be used with sketch streams to determine the elements created by a call to a display function or series of functions
          by looking at the list differences; new elements are always added at the end.))

    (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH SKETCH T])
```

(SKETCH.LIST.OF.ELEMENTS

```
  [LAMBDA (SKETCH PREDICATE INSIDEGROUPSFLG)                        (* rrb "14-Aug-85 16:26")

          (* Returns a list of the sketch elements in SKETCH that satisfy PREDICATE.
          If INSIDEGROUPSFLG is T, elements that are members of a group will be considered too.
          Otherwise only top level objects are considered. Note%: PREDICATE will be applied to GROUP elements even when
          INSIDEGROUPSFLG is T.)
                                                                    (* FOR NOW, IGNORE INSIDEGROUPSFLG)
    (for ELT in (SKETCH.ELEMENTS.OF.SKETCH SKETCH) when (APPLY* PREDICATE ELT) collect ELT])
```

(SKETCH.ADD.ELEMENT

```
  [LAMBDA (ELEMENT SKETCH NODISPLAYFLG)                             (* rrb "30-Aug-86 15:09")

          (* Adds an element to a sketch. If NODISPLAYFLG is NIL, any windows currently displaying SKETCH will be updated to
          reflect ELEMENT's addition. If NODISPLAYFLG is T, the displays won't be updated.)

    (PROG [(SKSTRUC (COND
                        ((NULL SKETCH)
```

```
                    (SKETCH.CREATE NIL))
                   (T (INSURE.SKETCH SKETCH]
          (COND
             ((NULL ELEMENT)
              (RETURN SKSTRUC))
             ((NOT (GLOBALELEMENTP ELEMENT))
              (ERROR ELEMENT "is not a sketch element.")))      (* add the element to the sketch.)
          (ADD.ELEMENT.TO.SKETCH ELEMENT SKSTRUC)               (* propagate to the viewers.)
          (for SKW in (ALL.SKETCH.VIEWERS SKSTRUC) when (ELT.INSIDE.SKETCHWP ELEMENT SKW)
             do (SKETCH.ADD.AND.DISPLAY1 ELEMENT SKW NIL NODISPLAYFLG))
          (RETURN SKSTRUC])
```

(SKETCH.DELETE.ELEMENT

```
  [LAMBDA (ELEMENT SKETCH INSIDEGROUPSFLG NODISPLAYFLG)          (* rrb "19-Oct-85 17:09")
```

(* Deletes an element from a sketch. If INSIDEGROUPSFLG is T, the element will be deleted even if it is inside a group. Otherwise it will be deleted only if it is on the top level. If NODISPLAYFLG is NIL, any windows currently displaying SKETCH will be updated to reflect ELEMENT's deletion. If NODISPLAYFLG is T, the displays won't be updated. It returns ELEMENT if ELEMENT was deleted.)

```
    (PROG ((SKSTRUC (INSURE.SKETCH SKETCH))
           LOCALELT OLDGELT)                                     (* delete the element to the sketch.)
          (COND
             ((EQ T (SETQ OLDGELT (REMOVE.ELEMENT.FROM.SKETCH ELEMENT SKSTRUC INSIDEGROUPSFLG)))
                                                                (* element deleted was top level.)
              )
             (OLDGELT                                           (* element deleted was part of a group.)
                      (printout PROMPTWINDOW T "member of group deleted but group not redrawn."))
             (T (RETURN NIL)))                                  (* propagate to the viewers.)
          (for SKW in (ALL.SKETCH.VIEWERS SKSTRUC) when (SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART ELEMENT
                                                                                                     SKW))
             do (SK.ERASE.AND.DELETE.ITEM LOCALELT SKW NODISPLAYFLG))
          (SK.CHECK.IMAGEOBJ.WHENDELETEDFN ELEMENT SKETCH)
          (RETURN OLDGELT])
```

(DELFROMGROUPELT

```
  [LAMBDA (ELTTODEL GROUPELT)                                    (* rrb " 2-Aug-85 17:03")
                                                                 (* if ELTTODEL is a member of GROUPELT, this deletes it.)
    (AND (EQ (fetch (GLOBALPART GTYPE) of GROUPELT)
             'GROUP)
         (PROG ((INDVGROUPELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GROUPELT))
                SUBELTS)
               (SETQ SUBELTS (fetch (GROUP LISTOFGLOBALELTS) of INDVGROUPELT))
               (COND
                  ((MEMBER ELTTODEL SUBELTS)
                   (replace (GROUP LISTOFGLOBALELTS) of INDVGROUPELT with (REMOVE ELTTODEL SUBELTS))
                   (RETURN T))
                  (T (RETURN (for ELT in SUBELTS thereis (DELFROMGROUPELT ELTTODEL ELT])
```

(SKETCH.ELEMENT.TYPE

```
  [LAMBDA (ELEMENT)                                              (* rrb "14-Aug-85 16:35")
                                                                 (* returns the type of a global sketch element)
    (fetch (GLOBALPART GTYPE) of ELEMENT])
```

(SKETCH.ELEMENT.CHANGED

```
  [LAMBDA (SKETCH ELEMENT SKETCHWINDOW)                          (* rrb " 4-Feb-86 15:04")
```

(* If ELEMENT is an element of SKETCH, its local part is recalculated. This is normally used to notify sketch that an image object element has changed. Note%: this replaces the element with another one.)

```
    (PROG ((SKETCH (INSURE.SKETCH SKETCH))
           OLDREG)
          (OR (GLOBALELEMENTP ELEMENT)
              (ERROR ELEMENT " is not a sketch element."))      (* note that the sketch has changed.)
          (SK.MARK.DIRTY SKETCH)
          (SETQ OLDREG (fetch (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                           of ELEMENT)))
          (SK.UPDATE.GLOBAL.IMAGE.OBJECT.ELEMENT ELEMENT SKETCHWINDOW)
                                                                 (* do the window that the interaction occurred in first.)
          (AND SKETCHWINDOW (SK.ELEMENT.CHANGED1 ELEMENT OLDREG SKETCHWINDOW))
                                                                 (* propagate to other windows.)
          (for SKW in (ALL.SKETCH.VIEWERS SKETCH) when (NEQ SKW SKETCHWINDOW) do (SK.ELEMENT.CHANGED1 ELEMENT
                                                                                                     OLDREG SKW))
          (RETURN ELEMENT])
```

(SK.ELEMENT.CHANGED1

```
  [LAMBDA (SKIMAGEOBJELT OLDREGION SKETCHW)                      (* rrb "21-Aug-85 15:54")
                                                                 (* updates the display of an image object element in a window.)
    (PROG (LOCALELT)
          (COND
             ((SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART SKIMAGEOBJELT SKETCHW))
```

```
            (COND
               ((EQ (SKETCH.ELEMENT.TYPE SKIMAGEOBJELT)
                    'SKIMAGEOBJ)
                (SK.DELETE.ITEM LOCALELT SKETCHW)
                (DSPFILL OLDREGION WHITESHADE 'REPLACE SKETCHW)
                (RETURN (SKETCH.ADD.AND.DISPLAY1 SKIMAGEOBJELT SKETCHW])
```

(SK.UPDATE.GLOBAL.IMAGE.OBJECT.ELEMENT

```
  [LAMBDA (SKIMOBJELT VIEWER)                                (* rrb " 4-Feb-86 15:04")
                                                             (* updates the fields to reflect changes in the size of the image
                                                             object.)
    (PROG ((INDVSKIMOBJELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKIMOBJELT))
           IMOBJSIZE REGION SCALE)
          (SETQ IMOBJSIZE (IMAGEBOXSIZE (fetch (SKIMAGEOBJ SKIMAGEOBJ) of INDVSKIMOBJELT)
                                VIEWER))
          (SETQ REGION (fetch (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of INDVSKIMOBJELT))
          (SETQ SCALE (fetch (SKIMAGEOBJ SKIMOBJ.ORIGSCALE) of INDVSKIMOBJELT))
          (replace (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of INDVSKIMOBJELT with (CREATEREGION
                                                                                (fetch (REGION LEFT) of REGION)
                                                                                (fetch (REGION BOTTOM) of REGION)
                                                                                (TIMES (fetch (IMAGEBOX XSIZE)
                                                                                          of IMOBJSIZE)
                                                                                       SCALE)
                                                                                (TIMES (fetch (IMAGEBOX YSIZE)
                                                                                          of IMOBJSIZE)
                                                                                       SCALE)))
          (replace (SKIMAGEOBJ SKIMOBJ.OFFSETPOS) of INDVSKIMOBJELT with (create POSITION
                                                                                 XCOORD _ (fetch (IMAGEBOX XKERN)
                                                                                             of IMOBJSIZE)
                                                                                 YCOORD _ (fetch (IMAGEBOX YDESC)
                                                                                             of IMOBJSIZE)))
          (RETURN SKIMOBJELT])

)

;; utility routines for sketch windows.

(DEFINEQ

(INSURE.SKETCH
  [LAMBDA (SK NOERRORFLG)                                    (* rrb " 3-Oct-86 15:16")
                                                             (* returns the SKETCH structure from a window, sketch stream,
    or a structure.)
    (SK.CHECK.SKETCH.VERSION (COND
                                ((type? SKETCH SK)
                                 SK)
                                [(WINDOWP SK)
                                 (COND
                                    ((WINDOWPROP SK 'SKETCH))
                                    (T (AND (NULL NOERRORFLG)
                                            (ERROR SK "doesn't have a SKETCH property."]
                                [(IMAGESTREAMTYPEP SK 'SKETCH)  (* this is a sketch stream)
                                 (COND
                                    ((WINDOWPROP (\SKSTRM.WINDOW.FROM.STREAM SK)
                                                 'SKETCH))
                                    (T (AND (NULL NOERRORFLG)
                                            (ERROR "sketch stream window doesn't have SKETCH property" SK]
                                [(type? IMAGEOBJ SK)
                                 (PROG [(SK? (fetch (SKETCHIMAGEOBJ SKIO.SKETCH)
                                                of (LISTP (IMAGEOBJPROP SK 'OBJECTDATUM]
                                       (RETURN (COND
                                                  ((type? SKETCH SK?)
                                                   SK?)
                                                  (NOERRORFLG NIL)
                                                  (T (ERROR "not a sketch image object" SK]
                                ((AND (LISTP SK)
                                      (LITATOM (CAR SK))
                                      (for ELT in (CDR SK) always (GLOBALELEMENTP ELT)))
                                                             (* old form, probably written out by notecards, update to new
                                                             form.)
                                 (PROG (X)
                                       (SETQ X (SKIO.UPDATE.FROM.OLD.FORM SK))
                                                             (* smash sketch so this won't have to happen every time.)
                                       (RPLACA SK (CAR X))
                                       (RPLACD SK (CDR X))
                                       (RETURN X)))
                                ((NULL NOERRORFLG)
                                 (ERROR SK "not a SKETCH"])
```

(LOCALSPECS.FROM.VIEWER

```
  [LAMBDA (SKW)                                              (* rrb "12-May-85 16:46")
                                                             (* returns the sketch specification displayed in the window SKW.)
    (CDAR (WINDOWPROP SKW 'SKETCHSPECS])
```

```
(SK.LOCAL.ELT.FROM.GLOBALPART
  [LAMBDA (GLOBALPART SKW)                                    (* rrb "18-MAR-83 13:09")

          (* returns the local element from SKW that has global part GLOBALPART -
          NIL if there isn't one.)

    (for ELT in (LOCALSPECS.FROM.VIEWER SKW) when (EQ (fetch (SCREENELT GLOBALPART) of ELT)
                                                      GLOBALPART)
       do (RETURN ELT])

(SKETCH.FROM.VIEWER
  [LAMBDA (SKETCHW)                                           (* returns the sketch that the window views.)
    (WINDOWPROP SKETCHW 'SKETCH])

(INSPECT.SKETCH
  [LAMBDA (SKW)                                               (* rrb "18-Apr-84 14:44")
                                                              (* calls the inspector on the sketch specs of a sketch window.)
    (PROG ((SPECS (LOCALSPECS.FROM.VIEWER SKW)))
          (COND
             (SPECS (INSPECT/TOP/LEVEL/LIST SPECS])

(ELT.INSIDE.SKETCHWP
  [LAMBDA (GELT SKW)                                          (* rrb " 8-APR-83 13:18")
                                                              (* determines if a global element is in the region of a viewer)
    (SK.INSIDE.REGION GELT (WINDOWPROP SKW 'REGION.VIEWED])

(SK.INSIDE.REGION
  [LAMBDA (GELT REGION)                                       (* rrb "31-Aug-84 10:15")
                                                              (* determines if the element GELT is inside of the global region
                                                              REGION)
    (APPLY* (SK.INSIDEFN (fetch (GLOBALPART GTYPE) of GELT))
           GELT REGION])

)

(DEFINEQ
(MAPSKETCHSPECS
  [LAMBDA (SKSPECS SPECFN DATUM DATUM2 DATUM3)                (* rrb "10-Sep-84 14:58")

          (* walks through a sketch specification list and applies SPECFN to each of the individual elements.)

    (AND SKSPECS (COND
                   ((SCREENELEMENTP SKSPECS)
                    (APPLY* SPECFN SKSPECS DATUM DATUM2 DATUM3))
                   ((LISTP SKSPECS)
                    (for FIGSPEC in SKSPECS do (MAPSKETCHSPECS FIGSPEC SPECFN DATUM DATUM2 DATUM3)))
                   (T (ERROR "unknown figure specification" SKSPECS])

(MAPCOLLECTSKETCHSPECS
  [LAMBDA (SKSPECS SPECFN DATUM DATUM2 DATUM3 DATUM4)         (* rrb "26-Apr-85 09:29")

          (* walks through a sketch specification list and applies SPECFN to each of the individual
          (elements returning a list of the results.))

    (AND SKSPECS (COND
                   ((SCREENELEMENTP SKSPECS)
                    (APPLY* SPECFN SKSPECS DATUM DATUM2 DATUM3 DATUM4))
                   ((LISTP SKSPECS)
                    (for FIGSPEC in SKSPECS collect (MAPCOLLECTSKETCHSPECS FIGSPEC SPECFN DATUM DATUM2 DATUM3
                                                                           DATUM4)))
                   (T (ERROR "unknown figure specification" SKSPECS])

(MAPSKETCHSPECSUNTIL
  [LAMBDA (SKETCHSPECS SPECFN DATUM DATUM2)                   (* rrb " 4-AUG-83 15:22")

          (* walks through a sketch specification list and applies SPECFN to each of the individual elements.)

    (AND SKETCHSPECS (COND
                       ((SKETCH.ELEMENT.NAMEP (fetch (SCREENELT GTYPE) of SKETCHSPECS))
                        (APPLY* SPECFN SKETCHSPECS DATUM DATUM2))
                       ((LISTP SKETCHSPECS)
                        (for FIGSPEC in SKETCHSPECS bind VALUE when (SETQ VALUE (MAPSKETCHSPECSUNTIL FIGSPEC
                                                                                                     SPECFN DATUM DATUM2))
                           do (RETURN VALUE)))
                       (T (ERROR "unknown figure specification" SKETCHSPECS])

(MAPGLOBALSKETCHSPECS
  [LAMBDA (SKSPECS SPECFN DATUM DATUM2 DATUM3)                (* rrb "19-Feb-85 17:52")
```

```
        (* walks through a list of global sketch elements and applies SPECFN to each of the individual elements.)

    (AND SKSPECS (COND
                    ((GLOBALELEMENTP SKSPECS)
                     (APPLY* SPECFN SKSPECS DATUM DATUM2 DATUM3))
                    ((LISTP SKSPECS)
                     (for FIGSPEC in SKSPECS collect (MAPGLOBALSKETCHSPECS FIGSPEC SPECFN DATUM DATUM2 DATUM3)))
                    (T (ERROR "unknown global sketch element" SKSPECS])
```

(MAPGLOBALSKETCHELEMENTS

```
  [LAMBDA (SKSPECS SPECFN DATUM DATUM2 DATUM3)                    (* rrb "24-Apr-85 15:02")

        (* walks through a list of global sketch elements and applies SPECFN to each of the individual elements.
        Differes from MAPGLOBALSKETCHSPECS in that it know about and gets inside of GROUP elements.)

    (AND SKSPECS (COND
                    [(GLOBALELEMENTP SKSPECS)
                     (COND
                        ((EQ (fetch (GLOBALPART GTYPE) of SKSPECS)
                             'GROUP)                              (* map function down the individual elements.)
                         (MAPGLOBALSKETCHELEMENTS (fetch (GROUP LISTOFGLOBALELTS) of (fetch (GLOBALPART
                                                                                              INDIVIDUALGLOBALPART
                                                                                                     )
                                                                                       of SKSPECS))
                                 SPECFN DATUM DATUM2 DATUM3))
                        (T (APPLY* SPECFN SKSPECS DATUM DATUM2 DATUM3]
                    ((LISTP SKSPECS)
                     (for FIGSPEC in SKSPECS collect (MAPGLOBALSKETCHELEMENTS FIGSPEC SPECFN DATUM DATUM2
                                                                               DATUM3)))
                    (T (ERROR "unknown global sketch element" SKSPECS])

)

;; multiple selection and copy select functions

(DEFINEQ

(SK.ADD.SELECTION
  [LAMBDA (ITEM/POS WINDOW MARKBM FIRSTFLG)                       (* rrb " 9-May-85 10:42")
                                                                  (* adds an item to the selection list of WINDOW.)
    (COND
       ([NOT (MEMBER ITEM/POS (WINDOWPROP WINDOW 'SKETCH.SELECTIONS]

          (* must turning off the element's selection before adding it to the window selections because the display of the selection
          check to see if the points are already selected in another element.)

        (SK.SELECT.ELT ITEM/POS WINDOW MARKBM)
        (WINDOWADDPROP WINDOW 'SKETCH.SELECTIONS ITEM/POS FIRSTFLG])

(SK.COPY.INSERTFN
  [LAMBDA (IMAGEOBJ SKW)                                          (* rrb "23-Jun-87 13:25")

          (* * the function that gets called to insert a copy-selection into a sketch window.
          Knows how to insert sketches, everything else is text.)

    (PROG (IMAGEOBJYET SELECTION EXTENDSELECTION)

          (* bind the selection so that if the user has to place an image obj, it is restored before the characters are unBYSYSBUFed)

          [bind DATUM for IMOBJ inside IMAGEOBJ
             do (COND
                    ((STRINGP IMOBJ)
                     (BKSYSBUF IMOBJ))
                    ((EQ (fetch (IMAGEOBJ IMAGEOBJFNS) of IMOBJ)
                         SKETCHIMAGEFNS)                          (* this is a sketch imageobj)
                     [COND
                        ((NULL IMAGEOBJYET)                       (* save SELECTION and EXTENDSELECTION so they can be
                         restored)
                         (SETQ IMAGEOBJYET T)
                         (SETQ SELECTION (WINDOWPROP SKW 'SELECTION))
                         (SETQ EXTENDSELECTION (WINDOWPROP SKW 'EXTENDSELECTION]
                     (SETQ DATUM (IMAGEOBJPROP IMOBJ 'OBJECTDATUM))
                     (OR (SK.INSERT.SKETCH SKW (fetch (SKETCHIMAGEOBJ SKIO.SKETCH) of DATUM)
                                (fetch (SKETCHIMAGEOBJ SKIO.REGION) of DATUM)
                                (fetch (SKETCHIMAGEOBJ SKIO.SCALE) of DATUM))
                         (RETURN)))
                    (T                                            (* insert the image object whatever it is)
                       [COND
                          ((NULL IMAGEOBJYET)                     (* save SELECTION and EXTENDSELECTION so they can be
                           restored)
                           (SETQ IMAGEOBJYET T)
                           (SETQ SELECTION (WINDOWPROP SKW 'SELECTION))
                           (SETQ EXTENDSELECTION (WINDOWPROP SKW 'EXTENDSELECTION]
                                                                  (* if the user placed it outside, just return)
                       (OR (SK.INSERT.SKETCH SKW [SKETCH.CREATE 'DUMMYNAME 'ELEMENTS (LIST (SETQ DATUM
```

```
                                                                                    (
                                                                                SK.ELEMENT.FROM.IMAGEOBJ
                                                                                    IMOBJ SKW]
                                    (fetch (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of (fetch (GLOBALPART
                                                                                        INDIVIDUALGLOBALPART)
                                                                                  of DATUM))
                                    (VIEWER.SCALE SKW))
                              (RETURN]
          (COND
             (IMAGEOBJYET                                          (* restore the selection)
                    (WINDOWPROP SKW 'SELECTION SELECTION)
                    (WINDOWPROP SKW 'EXTENDSELECTION EXTENDSELECTION)
                    (SKED.SELECTION.FEEDBACK SKW])
```

(SCREENELEMENTP

```
  [LAMBDA (ELT?)                                                   (* rrb "26-Sep-86 14:53")

          (* * returns ELT? if it is a screen element.)

    (PROG (X)
          (RETURN (AND (LISTP ELT?)
                       (LISTP (CDR ELT?))
                       (SETQ X (fetch (SCREENELT GLOBALPART) of ELT?))
                       (SKETCH.ELEMENT.NAMEP (fetch (GLOBALPART GTYPE) of X))
                      ELT?])
```

(SK.ITEM.REGION

```
  [LAMBDA (SCRELT)                                                 (* rrb "24-Jan-85 17:46")
                                                                   (* SCRELT is a sketch element This function returns the region
    it occupies.)
    (PROG [(REGIONFN (SK.REGIONFN (fetch (SCREENELT GTYPE) of SCRELT]
          (RETURN (COND
                     ((OR (NULL REGIONFN)
                          (EQ REGIONFN 'NILL))
                      NIL)
                     ((APPLY* REGIONFN SCRELT])
```

(SK.ELEMENT.GLOBAL.REGION

```
  [LAMBDA (GELT)                                                   (* rrb "18-Oct-85 10:30")
                                                                   (* GELT is a global sketch element This function returns the
                                                                   global region it occupies.)
    (PROG [(REGIONFN (SK.GLOBAL.REGIONFN (fetch (GLOBALPART GTYPE) of GELT]
          (RETURN (COND
                     ((OR (NULL REGIONFN)
                          (EQ REGIONFN 'NILL))
                      NIL)
                     ((APPLY* REGIONFN GELT])
```

(SK.LOCAL.ITEMS.IN.REGION

```
  [LAMBDA (HOTSPOTCACHE LEFT BOTTOM RIGHT TOP)                     (* rrb "31-Jan-85 11:38")

          (* * returns a list of the LOCALITEMS that are within LOCALREGION)

          (* changed to take a hotspot cache instead of a list of local items.
          OLD ARGS were (HOTSPOTCACHE LOCALREGION SCALE) OLD CODE
          (PROG ((SKREGION (UNSCALE.REGION LOCALREGION SCALE)))
          (RETURN (for SCRELT in LOCALITEMS when (SK.INSIDE.REGION
          (fetch (SCREENELT GLOBALPART) of SCRELT) SKREGION) collect SCRELT))))

    (PROG ((RLEFT (DIFFERENCE LEFT SK.POINT.WIDTH))
           (RBOTTOM (DIFFERENCE BOTTOM SK.POINT.WIDTH))
           (RRIGHT (PLUS RIGHT SK.POINT.WIDTH))
           (RTOP (PLUS TOP SK.POINT.WIDTH))
           ELTS)
          [for YBUCKET in HOTSPOTCACHE when (ILEQ (CAR YBUCKET)
                                                  RTOP)
             do (COND
                   ((ILESSP (CAR YBUCKET)
                            RBOTTOM)                               (* stop when Y gets too small.)
                    (RETURN)))
                (for XBUCKET in (CDR YBUCKET) when (ILEQ (CAR XBUCKET)
                                                         RRIGHT)
                   do (COND
                         ((ILESSP (CAR XBUCKET)
                                  RLEFT)                           (* stop when X gets too small.)
                          (RETURN)))                               (* collect the elements.)
                      (SETQ ELTS (UNION (CDR XBUCKET)
                                        ELTS]
          (RETURN ELTS])
```

(SK.REGIONFN

```
  [LAMBDA (ELEMENTTYPE)                                            (* rrb " 5-Sep-84 16:06")
```

```
        (* * access fn for getting the function that returns the region of an item from its type.)

    (fetch (SKETCHTYPE REGIONFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.GLOBAL.REGIONFN

```
  [LAMBDA (ELEMENTTYPE)                                          (* rrb "18-Oct-85 10:30")

        (* * access fn for getting the function that returns the global region of a global sketch element from its type.)

    (fetch (SKETCHTYPE GLOBALREGIONFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.REMOVE.SELECTION

```
  [LAMBDA (ITEM/POS WINDOW MARKBM)                               (* rrb " 9-May-85 10:31")
                                                                 (* removes an item from the selection list of WINDOW.)
    (COND
       ((MEMBER ITEM/POS (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))

        (* must remove element from window selections before turning off its selection because the display of the selection check to
        see if the points are still selected in another element.)

        (WINDOWDELPROP WINDOW 'SKETCH.SELECTIONS ITEM/POS)
        (SK.DESELECT.ELT ITEM/POS WINDOW MARKBM])
```

(SK.SELECT.MULTIPLE.ITEMS

```
  [LAMBDA (WINDOW ITEMFLG SELITEMS OPERATION)                    (* rrb "10-Dec-85 17:34")

        (* * selects allows the user to select a group of the sketch elements from the sketch WINDOW.
        If ITEMFLG is NIL, the user is allows to select control points as well as complete items and the returned value may be the
        position of a control point. If SELITEMS is given it is used as the items to be marked and selected from.
        Keeps control and probably shouldn't)

        (* the selection protocol is left to add, right to delete. Multiple clicking in the same place upscales for both select and
        deselect. Sweeping will select or deselect all of the items in the swept out area.
        Also it keeps control as long as a shift key is down.)

    (PROG ((INTERIOR (DSPCLIPPINGREGION NIL WINDOW))
           SELECTABLEITEMS HOTSPOTCACHE TIMER NOW OLDX ORIGX NEWX NEWY OLDY ORIGY OUTOFFIRSTPICK
           PREVMOUSEBUTTONS MOUSEINSIDE?)
          (COND
             (SELITEMS (SETQ SELECTABLEITEMS SELITEMS)           (* create a cache for the items to select from)
                    (SETQ HOTSPOTCACHE (SK.ADD.HOTSPOTS.TO.CACHE SELITEMS NIL)))
             [(AND (SETQ SELECTABLEITEMS (LOCALSPECS.FROM.VIEWER WINDOW))
                   (SK.HAS.SOME.HOTSPOTS (SETQ HOTSPOTCACHE (SK.HOTSPOT.CACHE.FOR.OPERATION WINDOW OPERATION]
             (T                                                  (* no items, don't do anything.)
                (RETURN)))
          (TOTOPW WINDOW)
          (SK.PUT.MARKS.UP WINDOW HOTSPOTCACHE)
          (until (MOUSESTATE (NOT UP)))
          (COND
             ((INSIDEP INTERIOR (LASTMOUSEX WINDOW)
                       (LASTMOUSEY WINDOW))
              (SETQ MOUSEINSIDE? T))
             (T                                                  (* first press was outside of the window, don't select anything.)
                (SK.TAKE.MARKS.DOWN WINDOW HOTSPOTCACHE)
                (RETURN)))
      SELECTLP
          (COND
             ((MOUSESTATE UP)
              (GO SELECTEXIT)))

          (* this label provides an entry for the code that tests if the shift key is down.)

      SELAFTERTEST
          (SETQ NEWY (LASTMOUSEY WINDOW))
          (SETQ NEWX (LASTMOUSEX WINDOW))
          [COND
             [(NOT MOUSEINSIDE?)

          (* mouse is outside, don't do anything other than wait for it to come back in.
          If the user has let up all buttons, the branch to SELECTEXIT will have been taken.)

              (COND
                 ((INSIDEP INTERIOR NEWX NEWY)
                  (SETQ MOUSEINSIDE? T)                          (* restore the saved selected items.)
                  (for ELT in SELITEMS do (SK.ADD.SELECTION ELT WINDOW]
             ((NOT (INSIDEP INTERIOR NEWX NEWY))

          (* mouse just went outside, remove selections but save them in case mouse comes back in.)

              (SETQ MOUSEINSIDE? NIL)
              (SETQ SELITEMS (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))
              (for ELT in SELITEMS do (SK.REMOVE.SELECTION ELT WINDOW)))
             [(NEQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS)
```

```
        (* another button has gone down, mark this as the origin of a new box to sweep.)

            (SETQ PREVMOUSEBUTTONS LASTMOUSEBUTTONS)
            (SETQ ORIGX (LASTMOUSEX WINDOW))
            (SETQ ORIGY (LASTMOUSEY WINDOW))
            [COND
               ((NULL ITEMFLG)                                    (* clear any selections that are of single points.)
                (for SEL in (WINDOWPROP WINDOW 'SKETCH.SELECTIONS) when (POSITIONP SEL)
                   do (SK.REMOVE.SELECTION SEL WINDOW]             (* add or delete the element that the button press occurred on if
            any.)
            (AND [SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE (create POSITION
                                                                XCOORD _ NEWX
                                                                YCOORD _ NEWY)
                                   (AND (NULL ITEMFLG)
                                        (LASTMOUSESTATE (ONLY LEFT))
                                        (NULL (WINDOWPROP WINDOW 'SKETCH.SELECTIONS]
                 (COND
                    ((LASTMOUSESTATE (ONLY LEFT))                 (* add selection.)
                     (SK.ADD.SELECTION NOW WINDOW))
                    ((LASTMOUSESTATE RIGHT)                       (* remove selection.)
                     (SK.REMOVE.SELECTION NOW WINDOW]
              ((COND
                  (OUTOFFIRSTPICK (OR (NEQ OLDX NEWX)
                                      (NEQ OLDY NEWY)))
                  ((OR (IGREATERP (IABS (IDIFFERENCE ORIGX NEWX))
                                  SK.NO.MOVE.DISTANCE)
                       (IGREATERP (IABS (IDIFFERENCE ORIGY NEWY))
                                  SK.NO.MOVE.DISTANCE))           (* make the first pick move further so that it is easier to multiple
                   click.)
                   (SETQ OUTOFFIRSTPICK T)))                      (* cursor has moved more than the minimum amount since last
                                                                  noticed.)
                                                                  (* add or delete any with in the swept out area.)
               (COND
                  ([AND (LASTMOUSESTATE (NOT UP))
                        (SETQ SELITEMS (SK.LOCAL.ITEMS.IN.REGION HOTSPOTCACHE (MIN ORIGX NEWX)
                                                                 (MIN ORIGY NEWY)
                                                                 (MAX ORIGX NEWX)
                                                                 (MAX ORIGY NEWY]

        (* if selecting multiple things, it must be whole items. Update NOW to be an item if it isn't already.)

                   [COND
                      ((POSITIONP NOW)
                       (SK.REMOVE.SELECTION NOW WINDOW)           (* if selecting, add the whole element in.)
                       (AND (LASTMOUSESTATE (ONLY LEFT))
                            (SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE NOW))
                            (SK.ADD.SELECTION NOW WINDOW]
                   (COND
                      ((LASTMOUSESTATE (ONLY LEFT))               (* left only selects.)
                       (for SELITEM in SELITEMS do (SK.ADD.SELECTION SELITEM WINDOW)))
                      ((LASTMOUSESTATE RIGHT)                     (* right cause deselect.)
                       (for SELITEM in SELITEMS do (SK.REMOVE.SELECTION SELITEM WINDOW]
            (SETQ OLDX NEWX)
            (SETQ OLDY NEWY)
            (GO SELECTLP)
        SELECTEXIT
            (COND
               (OUTOFFIRSTPICK (GO SHIFTDOWNLP)))                 (* wait for multiple clicks)
            (SETQ TIMER (SETUPTIMER CLICKWAITTIME TIMER))
        CLICKLP
            (COND
               [(AND (MOUSESTATE (NOT UP))
                     (ILESSP (IABS (IDIFFERENCE ORIGX (LASTMOUSEX WINDOW)))
                             SK.NO.MOVE.DISTANCE)
                     (ILESSP (IABS (IDIFFERENCE ORIGY (LASTMOUSEY WINDOW)))
                             SK.NO.MOVE.DISTANCE))
                (AND (LASTMOUSESTATE (ONLY LEFT))
                     (COND
                        ((POSITIONP NOW)                          (* thing selected is a point, select the whole item.)
                         (SK.REMOVE.SELECTION NOW WINDOW)
                         (SK.ADD.SELECTION (SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE NOW))
                                WINDOW))
                        ((SCREENELEMENTP NOW)

        (* thing now selected is an item, select all selectable items keeping the first one selected on the front.)

                         (for SELITEM in (SETQ NOW (CONS NOW (REMOVE NOW SELECTABLEITEMS)))
                            do (SK.ADD.SELECTION SELITEM WINDOW]
               ((NOT (TIMEREXPIRED? TIMER))
                (GO CLICKLP)))
        SHIFTDOWNLP
            (COND
               ((MOUSESTATE (NOT UP))                             (* button went down again, initialize the button state and click
                                                                  position.)
                (SETQ PREVMOUSEBUTTONS NIL)
                (SETQ OUTOFFIRSTPICK NIL)
```

```
            (GO SELAFTERTEST))
           ((.SHIFTKEYDOWNP.)
```

(* flip selection marks because if cursor is outside when shift key is let up, nothing is selected.)

```
        [COND
           [(NOT MOUSEINSIDE?)                                  (* mouse is outside%: if it comes back in, mark the selections.)
            (COND
               ((INSIDEP INTERIOR (LASTMOUSEX WINDOW)
                         (LASTMOUSEY WINDOW))
                (SETQ MOUSEINSIDE? T)                           (* restore the saved selected items.)
                (for ELT in SELITEMS do (SK.ADD.SELECTION ELT WINDOW]
           ((NOT (INSIDEP INTERIOR (LASTMOUSEX WINDOW)
                          (LASTMOUSEY WINDOW)))                 (* mouse just went outside, remove marks but keep selections)
            (SETQ MOUSEINSIDE? NIL)
            (SETQ SELITEMS (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))
            (for ELT in SELITEMS do (SK.REMOVE.SELECTION ELT WINDOW]
        (GO SHIFTDOWNLP)))
      (SETQ SELITEMS (WINDOWPROP WINDOW 'SKETCH.SELECTIONS))
      (COND
         (MOUSEINSIDE?                                          (* unmark and remove the selected items from the window property list.)
                (for SEL in SELITEMS do (SK.REMOVE.SELECTION SEL WINDOW)))
         (T                                                     (* they have already been unmarked, just remove them from the window.)
            (WINDOWPROP WINDOW 'SKETCH.SELECTIONS NIL)))
      (SK.TAKE.MARKS.DOWN WINDOW HOTSPOTCACHE)
      (RETURN SELITEMS])
```

(SKETCH.GET.ELEMENTS

```
  [LAMBDA (VIEWER SINGLEELEMENTFLG WHICHONES)                   (* rrb "17-Dec-85 15:35")
                                                                (* hilites the selection points and lets the user select one or more.)
    (PROG [[SELECTABLEITEMS (COND
                               ((LISTP WHICHONES)
                                (for ELT in WHICHONES collect (COND
                                                                 ((GLOBALELEMENTP ELT)
                                                                  (SK.LOCAL.ELT.FROM.GLOBALPART ELT VIEWER))
                                                                 (T (\ILLEGAL.ARG ELT]
           (OPERATION (SELECTQ (AND (NLISTP WHICHONES)
                                    WHICHONES)
                               ((MOVE COPY DELETE CHANGE GROUP UNGROUP COPYSELECT T FROZEN NIL)
                                  WHICHONES)
                               (\ILLEGAL.ARG WHICHONES]
      (RETURN (COND
                 (SINGLEELEMENTFLG (fetch (SCREENELT GLOBALPART) of (SK.SELECT.ITEM VIEWER T SELECTABLEITEMS
                                                                       OPERATION)))
                 (T (for SCRELT in (SK.SELECT.MULTIPLE.ITEMS VIEWER T SELECTABLEITEMS OPERATION)
                       collect (fetch (SCREENELT GLOBALPART) of SCRELT])
```

(SK.PUT.MARKS.UP

```
  [LAMBDA (SKETCHW HOTSPOTCACHE)                                (* rrb "29-Jan-85 17:40")
                                                                (* makes sure the selection points are up in a window.)
    (COND
       ((NULL (WINDOWPROP SKETCHW 'MARKS.UP))
        (SK.SHOWMARKS SKETCHW HOTSPOTCACHE)
        (WINDOWPROP SKETCHW 'MARKS.UP T])
```

(SK.TAKE.MARKS.DOWN

```
  [LAMBDA (SKETCHW HOTSPOTCACHE)                                (* rrb "29-Jan-85 17:41")
                                                                (* makes sure the selection points are down in a window.)
    (COND
       ((WINDOWPROP SKETCHW 'MARKS.UP)
        (SK.SHOWMARKS SKETCHW HOTSPOTCACHE)
        (WINDOWPROP SKETCHW 'MARKS.UP NIL])
```

(SK.TRANSLATE.GLOBALPART

```
  [LAMBDA (GLOBALELT DELTAPOS RETURNELTIFCANTFLG)               (* rrb "19-May-86 14:52")
```

(* GLOBALELT is a sketch element that was selected for a translate operation.
DELTAPOS is the amount the item is to be translated.)

```
    (PROG ((TRANSLATEFN (SK.TRANSLATEFN (fetch (GLOBALPART GTYPE) of GLOBALELT)))
           NEWGLOBAL OLDGLOBAL ACTIVEREGION)
          (RETURN (COND
                     ((OR (NULL TRANSLATEFN)
                          (EQ TRANSLATEFN 'NILL))               (* if can't translate, return the same thing.
                                                                This is probably an error condition.)
                      GLOBALELT)
                     ((SETQ NEWGLOBAL (APPLY* TRANSLATEFN GLOBALELT DELTAPOS))
```

(* copy the property list so that undoing works and because this code is used to make copies too.)

```
                    (SK.COPY.ELEMENT.PROPERTY.LIST NEWGLOBAL)
                    [COND
                       ([AND (SETQ ACTIVEREGION (GETSKETCHELEMENTPROP NEWGLOBAL 'ACTIVEREGION))
                             (EQUAL ACTIVEREGION (GETSKETCHELEMENTPROP GLOBALELT 'ACTIVEREGION]
```

(* update the ACTIVEREGION if the element has one and it is the same in the new element.)

```
                        (PUTSKETCHELEMENTPROP NEWGLOBAL 'ACTIVEREGION (REL.MOVE.REGION ACTIVEREGION
                                                                                (fetch (POSITION XCOORD)
                                                                                   of DELTAPOS)
                                                                                (fetch (POSITION YCOORD)
                                                                                   of DELTAPOS]
                     NEWGLOBAL)
                    (RETURNELTIFCANTFLG                          (* in the case of translating a whole sketch, need to return something.)
                          GLOBALELT])
```

(SK.TRANSLATE.ITEM

```
  [LAMBDA (SELELT GLOBALDELTAPOS W)                              (* rrb "21-Jan-85 18:35")
```

(* SELELT is a sketch element that was selected for a translate operation.
GLOBALDELTAPOS is the amount the item is to be translated.)

```
    (PROG (NEWGLOBAL OLDGLOBAL)
          (COND
             ((SETQ NEWGLOBAL (SK.TRANSLATE.GLOBALPART (SETQ OLDGLOBAL (fetch (SCREENELT GLOBALPART)
                                                                          of SELELT))
                                                       GLOBALDELTAPOS))
              (SK.UPDATE.ELEMENT OLDGLOBAL NEWGLOBAL W T)          (* don't include history for now. (SK.ADD.HISTEVENT
                                                                   (QUOTE TRANSLATE) (LIST OLDGLOBAL NEWGLOBAL) W))
              (RETURN NEWGLOBAL])
```

(SK.TRANSLATEFN

```
  [LAMBDA (ELEMENTTYPE)                                          (* rrb " 4-Sep-84 17:01")
    (fetch (SKETCHTYPE TRANSLATEFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(TRANSLATE.SKETCH

```
  [LAMBDA (SKETCH NEWXORG NEWYORG)                               (* rrb " 9-Jul-85 12:36")
```

(* * translates all the elements in a sketch to make the new {0, 0} be NEWXORG NEWYORG)

```
    (PROG [(DELTAPOS (create POSITION
                             XCOORD _ (MINUS NEWXORG)
                             YCOORD _ (MINUS NEWYORG]
          (RETURN (create SKETCH using SKETCH SKETCHELTS _ (for GELT in (fetch (SKETCH SKETCHELTS) of SKETCH)
                                                              collect (SK.TRANSLATE.GLOBALPART GELT DELTAPOS T])

)

(DECLARE%: EVAL@COMPILE

(RPAQQ SK.NO.MOVE.DISTANCE 4)

(CONSTANTS (SK.NO.MOVE.DISTANCE 4))
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SKFIGUREIMAGE (SKFIGURE.BITMAP SKFIGURE.LOWERLEFT))
)
)
```

;; stuff for changing the input scale

```
(DEFINEQ
```

(SK.INPUT.SCALE

```
  [LAMBDA (SKW)                                                  (* rrb " 4-Sep-85 15:35")
                                                                 (* returns the scale that input should be)
    (PROG [(SK (WINDOWPROP SKW 'SKETCHCONTEXT]
          (COND
             ((NULL SK)
              (ERROR SKW "arg not sketch window")
              (RETURN NIL)))
          (RETURN (COND
                     ((fetch (SKETCHCONTEXT SKETCHINPUTSCALE) of SK))
                     (T                                          (* early form of sketch that doesn't have an input scale.)
                        (SK.UPDATE.SKETCHCONTEXT SK)
                        (replace (SKETCHCONTEXT SKETCHINPUTSCALE) of SK with 1.0)
                        1.0])
```

(SK.UPDATE.SKETCHCONTEXT

```
  [LAMBDA (SKETCHCONTEXT)                                    (* rrb " 4-Sep-85 14:55")
                                                             (* updates an instance of a sketch context to have enough
                                                             fields.)

    (PROG ((NEWSK (CREATE.DEFAULT.SKETCH.CONTEXT)))
          [COND
             ((GREATERP (DIFFERENCE (LENGTH NEWSK)
                                    (LENGTH SKETCHCONTEXT))
                        0)                                   (* add fields to the sketch)
              (NCONC SKETCHCONTEXT (NTH NEWSK (ADD1 (LENGTH SKETCHCONTEXT]
          (RETURN SKETCHCONTEXT])
```

(SK.SET.INPUT.SCALE

```
  [LAMBDA (W)                                                (* rrb "19-Aug-86 11:52")
                                                             (* sets the size of the (input scale))
    (SK.SET.INPUT.SCALE.VALUE (RNUMBER (CONCAT "Input scale is now " (SK.INPUT.SCALE W)
                                               ".  Enter new input scale.  A larger scale will make new lines and
                                               text larger.")
                                       NIL NIL NIL T T)
          W])
```

(SK.SET.INPUT.SCALE.CURRENT

```
  [LAMBDA (W)                                                (* rrb "11-Jul-86 15:51")
                                                             (* sets the size of the input scale to the scale of the current
                                                             window.)

    (SK.SET.INPUT.SCALE.VALUE (VIEWER.SCALE W)
          W])
```

(SK.SET.INPUT.SCALE.VALUE

```
  [LAMBDA (NEWINPUTSCALE SKW)                                (* rrb "14-May-86 19:29")
                                                             (* sets the input scale to NEWINPUTSCALE)

    (AND (NUMBERP NEWINPUTSCALE)
         (NOT (ZEROP NEWINPUTSCALE))
         (replace (SKETCHCONTEXT SKETCHINPUTSCALE) of (WINDOWPROP SKW 'SKETCHCONTEXT) with (ABS NEWINPUTSCALE])

)

;; stuff for setting feedback amount

(DEFINEQ

(SK.SET.FEEDBACK.MODE
  [LAMBDA (VALUE)                                            (* rrb "19-Nov-85 13:25")

          (* sets the control on how much feedback to give the user as they are entering new figure elements.)

    [OR (MEMB VALUE '(POINTS T ALWAYS))
        (SETQ VALUE (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                    ITEMS _ '(("Points only" 'POINTS "Only the control points
                                                                    will be shown when entering elements.")
                                                              ("Fast figures" T "Wires, circles and ellipses are
                                                                    shown while they are being entered.")
                                                              ("All figures" 'ALWAYS "Most elements are shown
                                                                    while they are being entered.
                                                                    This will be slow for arcs and curves."))
                                                    CENTERFLG _ T]
    (AND VALUE (SETQ SKETCH.VERBOSE.FEEDBACK (SELECTQ VALUE
                                                      (POINTS NIL)
                                                      VALUE])
```

(SK.SET.FEEDBACK.POINT

```
  [LAMBDA NIL                                                (* sets the feedback to points only)
    (SK.SET.FEEDBACK.MODE 'POINTS])
```

(SK.SET.FEEDBACK.VERBOSE

```
  [LAMBDA NIL                                                (* sets the feedback to provide images on elements that are
                                                             fast.)

    (SK.SET.FEEDBACK.MODE T])
```

(SK.SET.FEEDBACK.ALWAYS

```
  [LAMBDA NIL                                                (* sets the feedback to give images on all figures.)
    (SK.SET.FEEDBACK.MODE 'ALWAYS])

)

(RPAQ? SKETCH.VERBOSE.FEEDBACK T)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCH.VERBOSE.FEEDBACK)
)
```

;; sketch icon support

```
(DEFINEQ
```

(SKETCH.TITLE

```
  [LAMBDA (SKW)                                                    (* rrb " 5-May-86 13:19")
                                                                   (* gets the title of the sketch being edited in SKW.)
    (fetch (SKETCH SKETCHNAME) of (INSURE.SKETCH SKW])
```

(SK.SHRINK.ICONCREATE

```
  [LAMBDA (W OLD-ICON POSITION)                                    ; Edited 25-Apr-88 15:44 by drc:
```

;;; Create the icon that represents this window.

```
    (LET [(ICONTITLE (WINDOWPROP W 'SKETCH.ICON.TITLE))
          (TITLE (SKETCH.TITLE W))
          (ICON (OR OLD-ICON (WINDOWPROP W 'ICON]
         (COND
            (ICON (CL:UNLESS (OR (EQUAL ICONTITLE TITLE)
                                 (NOT ICONTITLE))
                        ;; if we built this and the title is the same, or he has already put an icon on this, then we don't need to update it.
                        (SETQ ICONTITLE (OR TITLE ""))
                        (WINDOWPROP W 'SKETCH.ICON.TITLE ICONTITLE)
                        (ICONTITLE ICONTITLE NIL NIL ICON))
                  ICON)
            (T ;; make a new icon. Give it a title of '' so it can be distinguished from an ICON that the user supplied without an ICONTITLE.
               (SETQ ICONTITLE (OR TITLE ""))
               (WINDOWPROP W 'SKETCH.ICON.TITLE ICONTITLE)
               (TITLEDICONW SKETCH.TITLED.ICON.TEMPLATE ICONTITLE [COND
                                                                     ((NEQ TEDIT.ICON.FONT 'NOBIND)
                                                                      TEDIT.ICON.FONT)
                                                                     (T (DEFAULTFONT 'DISPLAY]
                     POSITION T NIL 'FILE])

)

(READVARS-FROM-STRINGS '(SKETCH.TITLED.ICON.TEMPLATE)
       "(({(READBITMAP)(87 95
       %"AOOOOOOOOOOOOOOOOOOOOL@@%"
       %"GOOOOOOOOOOOOOOOOOOOOL@@%"
       %"OKMHOHNCHNCHNCHNCHNCHN@@%"
       %"OOOOOOOOOOOOOOOOOOOOON@@%"
       %"ONJJCLGALGALGALGALGALF@@%"
       %"LOOOOOOOOOOOOOOOOOOOON@@%"
       %"NKOJCLGALGALGALGALGALF@@%"
       %"ONOOOOOOOOOOOOOOOOOOON@@%"
       %"NJJOOOOOOOOOOOOOOOOOON@@%"
       %"NNKNGALGALGALGALGALGAL@@%"
       %"OJJNOCLOCLOCLOCLOCLOCN@@%"
       %"NJJNFAHFAHFAHFAHFAHFAN@@%"
       %"NNJN@@@@@@@@@@@@@@@@@@N@@%"
       %"OJJN@@@@@@@@@@@@@@@@@@N@@%"
       %"OJKN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJKN@@@@@@@@@@@@@@@@@@N@@%"
       %"OKNN@@@@@@@@@@@@@@@@@@N@@%"
       %"OKJN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJJN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJNN@@@@@@@@@@@@@@@@@@N@@%"
       %"NKJN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJJN@@@@@@@@@@@@@@@@@@N@@%"
       %"NNKN@@@@@@@@@@@@@@@@@@N@@%"
       %"NNKN@@@@@@@@@@@@@@@@@@N@@%"
       %"OJNN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJNN@@@@@@@@@@@@@@@@@@N@@%"
       %"OJNN@@@@@@@@@@@@@@@@@@N@@%"
       %"OJJN@@@@@@@@@@@@@@@@@@N@@%"
       %"NNNN@@@@@@@@@@@@@@@@@@N@@%"
       %"NNNN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJNN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJKN@@@@@@@@@@@@@@@@@@N@@%"
       %"NJJN@@@@@@@@@@@@@@@@@@N@@%"
       %"L@@N@@@@@@@@@@@@@@@@@@N@@%"
       %"L@@N@@@@@@@@@@@@@@@@@@N@@%"
       %"L@BN@@@@@@@@@@@@@@@@@@N@@%"
       %"L@NN@@@@@@@@@@@@@@@@@@N@@%"
       %"LA@N@@@@@@@@@@@@@@@@@@N@@%"
       %"MM@N@@@@@@@@@@@@@@@@@@N@@%"
       %"LCBN@@@@@@@@@@@@@@@@@@N@@%"
       %"L@NN@@@@@@@@@@@@@@@@@@N@@%"
       %"L@BN@@@@@@@@@@@@@@@@@@N@@%"
       %"L@@N@@@@@@@@@@@@@@@@@@N@@%"
       %"L@@N@@@@@@@@@@@@@@@@@@N@@%"
       %"LB@N@@@@@@@@@@@@@@@@@@N@@%"
```

```
%"LDDN@@@@@@@@@@@@@@@@@N@@%"
%"LDBN@@@@@@@@@@@@@@@@@N@@%"
%"LBBN@@@@@@@@@@@@@@@@@N@@%"
%"LALN@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"LDDN@@@@@@@@@@@@@@@@@N@@%"
%"MLBN@@@@@@@@@@@@@@@@@N@@%"
%"LGBN@@@@@@@@@@@@@@@@@N@@%"
%"LDNN@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"LC@N@@@@@@@@@@@@@@@@@N@@%"
%"LDHN@@@@@@@@@@@@@@@@@N@@%"
%"LDJN@@@@@@@@@@@@@@@@@N@@%"
%"LCJN@@@@@@@@@@@@@@@@@N@@%"
%"LABN@@@@@@@@@@@@@@@@@N@@%"
%"L@NN@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"LD@N@@@@@@@@@@@@@@@@@N@@%"
%"LB@N@@@@@@@@@@@@@@@@@N@@%"
%"MBNN@@@@@@@@@@@@@@@@@N@@%"
%"MM@N@@@@@@@@@@@@@@@@@N@@%"
%"LCHN@@@@@@@@@@@@@@@@@N@@%"
%"L@FN@@@@@@@@@@@@@@@@@N@@%"
%"L@BN@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"LH@N@@@@@@@@@@@@@@@@@N@@%"
%"M@@N@@@@@@@@@@@@@@@@@N@@%"
%"MAHN@@@@@@@@@@@@@@@@@N@@%"
%"MBDN@@@@@@@@@@@@@@@@@N@@%"
%"MBDN@@@@@@@@@@@@@@@@@N@@%"
%"LLDN@@@@@@@@@@@@@@@@@N@@%"
%"L@DN@@@@@@@@@@@@@@@@@N@@%"
%"L@DN@@@@@@@@@@@@@@@@@N@@%"
%"L@LN@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"L@@N@@@@@@@@@@@@@@@@@N@@%"
%"NJJN@@@@@@@@@@@@@@@@@N@@%"
%"OJJN@@@@@@@@@@@@@@@@@N@@%"
%"NKKN@@@@@@@@@@@@@@@@@N@@%"
%"OJNN@@@@@@@@@@@@@@@@@N@@%"
%"CNNN@@@@@@@@@@@@@@@@@N@@%"
%"@OJN@@@@@@@@@@@@@@@@@N@@%"
%"@CNN@@@@@@@@@@@@@@@@@N@@%"
%"@@OOOOOOOOOOOOOOOOOOON@@%"
%"@@COOOOOOOOOOOOOOOOOON@@%"
%"@@@OOOOOOOOOOOOOOOOOON@@%")}  {(READBITMAP)(87 95
%"AOOOOOOOOOOOOOOOOOOOOH@@%"
%"GOOOOOOOOOOOOOOOOOOOOL@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
%"OOOOOOOOOOOOOOOOOOOOON@@%"
```

```
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"OOOOOOOOOOOOOOOOOOOOON@@%"
        %"COOOOOOOOOOOOOOOOOOOON@@%"
        %"@OOOOOOOOOOOOOOOOOOOON@@%"
        %"@COOOOOOOOOOOOOOOOOOON@@%"
        %"@@OOOOOOOOOOOOOOOOOOON@@%"
        %"@@COOOOOOOOOOOOOOOOOON@@%"
        %"@@@OOOOOOOOOOOOOOOOOON@@%")}  (16 4 64 77)))
        ")
```

;; fns for reading in various values

```
(DEFINEQ

(READBRUSHSHAPE
  [LAMBDA NIL                                          (* rrb " 6-Nov-85 09:57")
                                                       (* reads a brush shape from the user.)
    (\CURSOR.IN.MIDDLE.MENU (create MENU
                                    CENTERFLG _ T
                                    TITLE _ "pick a shape"
                                    ITEMS _ '(ROUND SQUARE VERTICAL HORIZONTAL DIAGONAL])

(READ.FUNCTION
  [LAMBDA (PRMPT W)                                    (* rrb "11-May-84 15:41")
    (PROG ((PROMPTWIN (GETPROMPTWINDOW W 3))
           OLDTTYDS LST)
          (SETQ OLDTTYDS (TTYDISPLAYSTREAM PROMPTWIN))
          (COND
             (PRMPT (printout PROMPTWIN PRMPT T ">> ")))   (* grab the tty.)
          (TTY.PROCESS NIL)
          (SETQ LST (CONS (READ T)
                          (READLINE)))
          (CLOSEW (TTYDISPLAYSTREAM OLDTTYDS))
          (RETURN (CAR LST])
```

```
(READBRUSHSIZE
  [LAMBDA (NOWSIZE)                                          (* rrb "19-May-86 15:44")
    (PROG ((N (RNUMBER (COND
                         (NOWSIZE (CONCAT "Current size is " NOWSIZE ".  Enter new brush size."))
                         (T "Enter new brush size."))
                       NIL NIL NIL T T T T)))
          (RETURN (COND
                     ((EQUAL N 0)
                      NIL)
                     (N (ABS N])

(READANGLE
  [LAMBDA NIL                                                (* rrb "14-May-86 19:29")
                                                             (* interacts to get an angle from the user.)
    (PROG ((NEWVALUE (RNUMBER "Enter arc angle in degrees." NIL NIL NIL T NIL T)))
          (RETURN (COND
                     ((EQ NEWVALUE 0)
                      NIL)
                     (T NEWVALUE])

(READARCDIRECTION
  [LAMBDA (MENUTITLE)                                        (* rrb " 6-Nov-85 09:53")
                                                             (* interacts to get whether an arc should go clockwise or
                                                             counterclockwise)
    (\CURSOR.IN.MIDDLE.MENU (create MENU
                                    TITLE _ (OR MENUTITLE "Which way should the arc go?")
                                    ITEMS _ '(("Clockwise" 'CLOCKWISE "The arc will be drawn clockwise from the
                                                                      first point to the second point.")
                                              ("Counterclockwise" 'COUNTERCLOCKWISE "The arc will be drawn
                                                                      counterclockwise from the first point to the second point.")
                                              )
                                    CENTERFLG _ T])

)

(DEFINEQ

(SK.CHANGE.DASHING
  [LAMBDA (ELTWITHLINE DASHING SKW)                          (* rrb " 9-Jan-86 16:58")
                                                             (* changes the line dashing of ELTWITHLINE if it has one)
                                                             (* knows about the various types of sketch elements and
                                                             shouldn't.)
    (PROG (SIZE GLINELT TYPE NEWDASHING NOWDASHING NEWELT)
          (COND
             ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHLINE))
                    '(WIRE BOX CLOSEDWIRE CLOSEDCURVE OPENCURVE CIRCLE ELLIPSE TEXTBOX ARC))
                                                             (* only works for things of wire type.)
              (SETQ GLINELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHLINE))
                                                             (* the dashing may be stored in different places for the element
              types.)
              [SETQ NEWDASHING (COND
                                  ((EQ DASHING 'NONE)        (* no dashing is marked with NIL)
                                   NIL)
                                  ((DASHINGP DASHING))
                                  (T (ERROR "illegal dashing" DASHING]
              (SETQ NOWDASHING (SELECTQ TYPE
                                   (WIRE (fetch (WIRE OPENWIREDASHING) of GLINELT))
                                   (BOX (fetch (BOX BOXDASHING) of GLINELT))
                                   (ARC (fetch (ARC ARCDASHING) of GLINELT))
                                   (TEXTBOX (fetch (TEXTBOX TEXTBOXDASHING) of GLINELT))
                                   (CLOSEDWIRE (fetch (CLOSEDWIRE CLOSEDWIREDASHING) of GLINELT))
                                   (CLOSEDCURVE (fetch (CLOSEDCURVE DASHING) of GLINELT))
                                   (OPENCURVE (fetch (OPENCURVE DASHING) of GLINELT))
                                   (CIRCLE (fetch (CIRCLE DASHING) of GLINELT))
                                   (ELLIPSE (fetch (ELLIPSE DASHING) of GLINELT))
                                   (SHOULDNT)))
              (COND
                 ((EQUAL NEWDASHING NOWDASHING)              (* if dashing isn't changing, don't bother creating a new one and
                  repainting.)
                  (RETURN)))
              (SETQ NEWELT (SELECTQ TYPE
                                (WIRE (create WIRE using GLINELT OPENWIREDASHING _ NEWDASHING))
                                (BOX (create BOX using GLINELT BOXDASHING _ NEWDASHING))
                                (ARC (create ARC using GLINELT ARCDASHING _ NEWDASHING))
                                (TEXTBOX (create TEXTBOX using GLINELT TEXTBOXDASHING _ NEWDASHING))
                                (CLOSEDWIRE (create CLOSEDWIRE using GLINELT CLOSEDWIREDASHING _ NEWDASHING))
                                (CLOSEDCURVE (create CLOSEDCURVE using GLINELT DASHING _ NEWDASHING))
                                (OPENCURVE (create OPENCURVE using GLINELT DASHING _ NEWDASHING))
                                (CIRCLE (create CIRCLE using GLINELT DASHING _ NEWDASHING))
                                (ELLIPSE (create ELLIPSE using GLINELT DASHING _ NEWDASHING))
                                (SHOULDNT)))
              (RETURN (create SKHISTORYCHANGESPEC
                              NEWELT _ (create GLOBALPART
                                               COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART) of
                                                                                                   ELTWITHLINE
```

```
                                                    )
                                   INDIVIDUALGLOBALPART _ NEWELT)
                      OLDELT _ ELTWITHLINE
                      PROPERTY _ 'DASHING
                      NEWVALUE _ NEWDASHING
                      OLDVALUE _ NOWDASHING])
```

(READ.AND.SAVE.NEW.DASHING

```
  [LAMBDA NIL                                          (* rrb " 6-Nov-85 09:57")
                                                       (* reads a new dashing, confirms it with the user and adds it to
                                                       SK.DASHING.PATTERNS)
    (PROG (DASHING BM)
      LP  (COND
             ((NULL (SETQ DASHING (READ.NEW.DASHING)))  (* user aborted)
              (RETURN NIL)))
          (SETQ BM (SK.DASHING.LABEL DASHING))
      CONFIRM
          (SELECTQ (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                   ITEMS _ (LIST (LIST BM T "Will use this as the dashing
                                                                  pattern.")
                                                                 '(Yes T "Will accept this pattern.")
                                                                 '(No 'NO "Will ask you for another dashing
                                                                 pattern."))
                                                   CENTERFLG _ T
                                                   TITLE _ "Is this pattern OK?"))
               (NO (GO LP))
               (T                                       (* add dashing to global list and return it.)
                  (SK.CACHE.DASHING DASHING BM)
                  (RETURN DASHING))
               (PROGN (PROMPTPRINT "Please select 'Yes' if this pattern is what you want; 'No' if it isn't.")
                      (GO CONFIRM])
```

(READ.NEW.DASHING

```
  [LAMBDA NIL                                          (* rrb "14-May-86 19:30")
                                                       (* reads a value of dashing from the user.)
    (PROMPTPRINT "You will be prompted for a series of numbers which specify the number of points ON and OFF.
           Enter 0 to end the dashing pattern.
           Enter 'Abort' to leave the dashing unchanged.")
    (bind VAL DASHLST OFF? (ORIGPOS _ (create POSITION
                                             XCOORD _ LASTMOUSEX
                                             YCOORD _ LASTMOUSEY))
       until (OR (EQ (SETQ VAL (RNUMBER (CONCAT "Enter the number of points " (COND
                                                                                (OFF? 'OFF)
                                                                                (T 'ON))
                                               ". Enter 0 to end the dashing.")
                                        ORIGPOS NIL NIL T T T))
                     0)
                 (NULL VAL))
       do (SETQ DASHLST (CONS (ABS VAL)
                              DASHLST))
          (SETQ OFF? (NOT OFF?))
       finally (CLRPROMPT)
             (RETURN (COND
                        ((NULL VAL)                    (* abort selection)
                         NIL)
                        (T (REVERSE DASHLST])
```

(READ.DASHING.CHANGE

```
  [LAMBDA NIL
    (DECLARE (GLOBALVARS SK.DASHING.PATTERNS))         (* rrb " 6-Nov-85 09:57")

          (* gets a description of how to change the arrow heads of a wire or curve.)

    (PROG (DASHING)
          (SELECTQ [SETQ DASHING (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                 CENTERFLG _ T
                                                                 TITLE _ "New dashing pattern?"
                                                                 ITEMS _
                                                                 (APPEND (for DASHPAT in SK.DASHING.PATTERNS
                                                                            collect (LIST (CAR DASHPAT)
                                                                                          (KWOTE (CADR DASHPAT))
                                                                                          "changes dashing to this
                                                                                          pattern"))
                                                                         '(("other" 'OTHER "will prompt you for a
                                                                             new dashing pattern.")
                                                                           ("no dashing" 'NONE "removes dashing."]
               (OTHER (RETURN (READ.AND.SAVE.NEW.DASHING)))
               (RETURN DASHING])
```

(SK.CACHE.DASHING

```
  [LAMBDA (DASHING BITMAP)                             (* rrb " 3-May-85 14:33")
                                                       (* adds a dashing and its bitmap label to the global cache.)
    (OR (for DASH in SK.DASHING.PATTERNS when (EQUAL (CADR DASH)
```

```
                                                    DASHING)
          do (RETURN T))
        (COND
           (SK.DASHING.PATTERNS (NCONC1 SK.DASHING.PATTERNS (LIST (COND
                                                                     ((BITMAPP BITMAP))
                                                                     (T (SK.DASHING.LABEL DASHING)))
                                                                  DASHING)))
           (T (SETQ SK.DASHING.PATTERNS (LIST (LIST (COND
                                                       ((BITMAPP BITMAP))
                                                       (T (SK.DASHING.LABEL DASHING)))
                                                    DASHING])
```

(SK.DASHING.LABEL

```
  [LAMBDA (DASHING)                                                  (* rrb " 3-May-85 14:32")
                                                                     (* creates a bitmap label which shows a dashing pattern.)
    (PROG (DS BM)
          [SETQ DS (DSPCREATE (SETQ BM (BITMAPCREATE 50 1]
          (DRAWLINE 0 0 50 0 1 NIL DS NIL DASHING)
          (RETURN BM])

)

(DEFINEQ
```

(READ.FILLING.CHANGE

```
  [LAMBDA NIL                                                        (* rrb " 6-Nov-85 09:58")
                                                                     (* reads a shade for the filling texture.)
    (PROG (FILLING)
          (SELECTQ (SETQ FILLING (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                  CENTERFLG _ T
                                                                  TITLE _ "New filling?"
                                                                  ITEMS _
                                                                  [APPEND (for FILLPAT in SK.FILLING.PATTERNS
                                                                             collect (LIST (CAR FILLPAT)
                                                                                           (KWOTE (CADR FILLPAT))
                                                                                           "changes filling to this
                                                                                           pattern"))
                                                                          '(("4x4 shade" '4X4 "Allows creation of a
                                                                                   4 bits by 4 bits shade")
                                                                            ("16x16 shade" '16X16 "Allows creation
                                                                                   of a 16 bits by 16 bits shade")
                                                                            ("No filling" 'NONE "no filling will be
                                                                                   used.")]
                                                                  MENUBORDERSIZE _ 1)))
                   (4X4 (RETURN (READ.AND.SAVE.NEW.FILLING)))
                   (16X16 (RETURN (READ.AND.SAVE.NEW.FILLING T)))
                   (RETURN FILLING])
```

(SK.CACHE.FILLING

```
  [LAMBDA (FILLING)                                                  (* rrb " 8-Jun-85 14:58")
                                                                     (* adds a dashing and its bitmap label to the global cache.)
    (OR (for FILL in SK.FILLING.PATTERNS when (EQUAL (CADR FILL)
                                                     FILLING)
          do (RETURN T))
        (COND
           (SK.FILLING.PATTERNS (NCONC1 SK.FILLING.PATTERNS (LIST (SK.FILLING.LABEL FILLING)
                                                                  FILLING)))
           (T (SETQ SK.FILLING.PATTERNS (LIST (LIST (SK.FILLING.LABEL FILLING)
                                                    FILLING)))
              'ADDED])
```

(READ.AND.SAVE.NEW.FILLING

```
  [LAMBDA (16X16FLG)                                                 (* rrb " 8-Jun-85 14:58")
                                                                     (* reads a new filling, confirms it with the user and adds it to
                                                                     SK.FILLING.PATTERNS)
    (PROG (FILLING)
          (COND
             ([NULL (SETQ FILLING (EDITSHADE (COND
                                                (16X16FLG (BITMAPCREATE 16 16]
                                                                     (* user aborted)
              (RETURN NIL)))
          (SK.CACHE.FILLING FILLING)
          (RETURN FILLING])
```

(SK.FILLING.LABEL

```
  [LAMBDA (FILLING)                                                  (* rrb " 8-Jun-85 12:08")
                                                                     (* creates a bitmap label which fills it with the texture FILLING.)
    (PROG [(BM (BITMAPCREATE (PLUS 8 (STRINGWIDTH "16x16 shade" MENUFONT))
                     (FONTPROP MENUFONT 'HEIGHT]
          (BLTSHADE FILLING BM)
          (RETURN BM])

)
```

```
(RPAQ? SK.DASHING.PATTERNS )

(RPAQ? SK.FILLING.PATTERNS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SK.DASHING.PATTERNS SK.FILLING.PATTERNS)
)

(SK.CACHE.DASHING '(2 4))

(SK.CACHE.DASHING '(6 3 1 3))

(SK.CACHE.FILLING BLACKSHADE)

(SK.CACHE.FILLING GRAYSHADE)

(SK.CACHE.FILLING HIGHLIGHTSHADE)
```

;; stuff for reading input positions

```
(DEFINEQ
```

(SK.GETGLOBALPOSITION

```
  [LAMBDA (W CURSOR)                                                    (* rrb "20-May-86 10:56")
                                                                        (* gets a position from the user and returns the global value of it.)
    (SK.MAP.INPUT.PT.TO.GLOBAL (SK.READ.POINT.WITH.FEEDBACK W CURSOR)
           W])
```

(SKETCH.TRACK.ELEMENTS

```
  [LAMBDA (ELEMENTS VIEWER CONSTRAINTFN HOTSPOT PROMPTMSG CONSTRAINTDATA FEEDBACKFN NOINITIALERASEFLG
                NOFINALPAINTFLG)                                        (* rrb "22-Jul-86 14:41")

          (* gets a point from the user by displaying an image of ELEMENTS.
          It calls CONSTRAINTFN everytime the cursor moves to allow user constraints on where the image is displayed.
          All positions and elements are in sketch coordinates.)

    (PROG (SCRELTS FIGINFO FIRSTHOTSPOT GLOBALHOTSPOT NEWPOS LOWLFT IMAGEPOSX IMAGEPOSY IMAGEBM DELTAPOS
                NEWGLOBALS SKETCH GDELTAPOS)
          (OR ELEMENTS (RETURN))
          (SETQ FIGINFO (SK.FIGUREIMAGE (SETQ SCRELTS (MAP.SKETCH.ELEMENTS.INTO.VIEWER ELEMENTS VIEWER))
                                (DSPCLIPPINGREGION NIL VIEWER)))
          [SETQ FIRSTHOTSPOT (COND
                                ((POSITIONP HOTSPOT)
                                 (MAP.GLOBAL.POSITION.INTO.VIEWER HOTSPOT VIEWER))
                                (T (CAR (fetch (SCREENELT HOTSPOTS) of (CAR SCRELTS]
          [SETQ GLOBALHOTSPOT (COND
                                 ((POSITIONP HOTSPOT))
                                 (T (MAP.VIEWER.PT.INTO.GLOBAL (CAR (fetch (SCREENELT HOTSPOTS)
                                                                           of (CAR SCRELTS)))
                                                               VIEWER]
          (SETQ IMAGEBM (fetch (SKFIGUREIMAGE SKFIGURE.BITMAP) of FIGINFO))
          (SETQ LOWLFT (fetch (SKFIGUREIMAGE SKFIGURE.LOWERLEFT) of FIGINFO))
          (SETQ IMAGEPOSX (fetch (POSITION XCOORD) of LOWLFT))
          (SETQ IMAGEPOSY (fetch (POSITION YCOORD) of LOWLFT))     (* put the cursor on the hot spot)
          (CURSORPOSITION FIRSTHOTSPOT VIEWER)
          (COND
             ([NULL (ERSETQ (PROGN (OR NOINITIALERASEFLG (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY
                                                                                'ERASE VIEWER))
                                   (SETQ NEWPOS (SKETCH.TRACK.IMAGE VIEWER IMAGEBM 'PAINT PROMPTMSG
                                                      (IDIFFERENCE IMAGEPOSX (fetch (POSITION XCOORD)
                                                                                of FIRSTHOTSPOT))
                                                      (IDIFFERENCE IMAGEPOSY (fetch (POSITION YCOORD)
                                                                                of FIRSTHOTSPOT))
                                                     CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN]
                                                                        (* error happened, repaint the image.)
              (OR NOINITIALERASEFLG (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT VIEWER))
              (CLOSEPROMPTWINDOW VIEWER)
              (ERROR!))
             (T (OR NOFINALPAINTFLG (SK.SHOW.FIG.FROM.INFO IMAGEBM IMAGEPOSX IMAGEPOSY 'PAINT VIEWER))
                (RETURN (AND NEWPOS (PTDIFFERENCE NEWPOS GLOBALHOTSPOT])
```

(SK.PICKOUT.WHOLE.MOVE.ELEMENTS

```
  [LAMBDA (MOVEELTLST)                                                  (* rrb "13-Dec-85 11:54")

          (* returns from a list of sketch elements that are being moved, the ones that will be completely moved)

    (COND
       ((EQ (CAR MOVEELTLST)
            T)
        (CDR MOVEELTLST))
       ((EVERY (CAR MOVEELTLST)
               (FUNCTION NUMBERP))
        NIL)
```

```
      (T (for X in MOVEELTLST when (EQ (CAR X)
                                       T)
            collect (CDR X])
```

(MAP.SKETCH.ELEMENTS.INTO.VIEWER

```
  [LAMBDA (ELEMENTS VIEWER)                                   (* rrb "12-Dec-85 12:25")
                                                              (* maps a list of elements into a viewer)
    (for SKELT in ELEMENTS collect (SK.LOCAL.FROM.GLOBAL SKELT VIEWER])
```

(MAP.GLOBAL.POSITION.INTO.VIEWER

```
  [LAMBDA (GPOS VIEWER)                                       (* rrb "11-Jul-86 15:54")
                                                              (* maps a sketch coordinate into a viewer coordinate.)
    (SK.SCALE.POSITION.INTO.VIEWER GPOS (VIEWER.SCALE VIEWER])
```

(SKETCH.TO.VIEWER.POSITION

```
  [LAMBDA (POSITION VIEWERSCALE)                              (* rrb "11-Jul-86 15:54")

          (* Transforms a position from sketch coordinates into viewer coordinates.
          VIEWERSCALE can be a scale or a viewer.)

    (SK.SCALE.POSITION.INTO.VIEWER POSITION (SK.INSURE.SCALE VIEWERSCALE])
```

(SKETCH.TRACK.IMAGE

```
  [LAMBDA (WINDOW BITMAP OPERATION MSG XOFFSET YOFFSET CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN)
                                                              (* rrb "11-Jun-86 13:44")

          (* gets a position by tracking with a and calling a user provided constraint function.
          The spec returns is actually (ONGRID? position) so that caller can tell whether it was placed on grid or not.)

    (PROG (WIDTH HEIGHT)
          (SETQ WIDTH (BITMAPWIDTH BITMAP))
          (SETQ HEIGHT (BITMAPHEIGHT BITMAP))
          (AND MSG (STATUSPRINT WINDOW "
                               " MSG))
          (RETURN (SK.TRACK.IMAGE1 WINDOW BITMAP (BITMAPCREATE WIDTH HEIGHT)
                         WIDTH HEIGHT (OR OPERATION 'PAINT)
                         XOFFSET YOFFSET CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN])
```

(SK.TRACK.IMAGE1

```
  [LAMBDA (W BITMAP BUFFER.BITMAP WIDTH HEIGHT OPERATION XOFFSET YOFFSET CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN)
                                                              (* rrb "11-Jun-86 13:59")

          (* tracks BITMAP until a button goes down and comes up. It calls CONSTRAINTFN to determine the position at which to
          display the image. Returns a point in global space that the image was placed.)
                                                              (* there is other code in BIGFONT that is probably better for
                                                              this.)
    (PROG (READPT)
          (SETQ READPT (SK.TRACK.BITMAP1 W BITMAP BUFFER.BITMAP WIDTH HEIGHT OPERATION XOFFSET YOFFSET
                                CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN))
          (RETURN (AND READPT (MAP.VIEWER.XY.INTO.GLOBAL (fetch (POSITION XCOORD) of (fetch (INPUTPT
                                                                                              INPUT.POSITION)
                                                                                         of READPT))
                                                         (fetch (POSITION YCOORD) of (fetch (INPUTPT INPUT.POSITION) of READPT))
                                                         W
                                                         (fetch (INPUTPT INPUT.ONGRID?) of READPT)
                                                         (create POSITION])
```

(MAP.VIEWER.XY.INTO.GLOBAL

```
  [LAMBDA (X Y VIEWER ONGRID? SCRATCHPT)                      (* rrb "11-Jul-86 15:52")

          (* maps from an x y pair in a window to the corresponding global position.
          ONGRID? is T if the X Y should be interpreted as being on the grid.
          SCRATCHPT is a scratch position that should be clobbered with the result.)

    (PROG ((SCALE (VIEWER.SCALE VIEWER))
           GRID)
          [COND
             (ONGRID? (SETQ GRID (SK.GRIDFACTOR VIEWER)))
             (T

          (* map the point onto a grid location that would have the same screen position as the given point.)

                (SETQ GRID (GREATESTPOWEROF2LT SCALE]
          (RETURN (SK.SET.POSITION (NEAREST.ON.GRID (TIMES X SCALE)
                                                    GRID)
                         (NEAREST.ON.GRID (TIMES Y SCALE)
                                GRID)
                         SCRATCHPT])
```

(SK.SET.POSITION

```
  [LAMBDA (X Y POSITION)                                          (* rrb "21-May-86 16:09")
                                                                  (* sets the x and y coordinate fields of a position.)
    (replace (POSITION XCOORD) of POSITION with X)
    (replace (POSITION YCOORD) of POSITION with Y)
    POSITION])
```

(MAP.VIEWER.PT.INTO.GLOBAL

```
  [LAMBDA (PT VIEWER ONGRID?)                                     (* rrb "11-Jul-86 15:52")

          (* maps from an PT in a window to the corresponding global position.
          ONGRID? is T if the PT should be interpreted as being on the grid.)

    (PROG ((SCALE (VIEWER.SCALE VIEWER))
           GRID)
          [COND
             (ONGRID? (SETQ GRID (SK.GRIDFACTOR VIEWER)))
             (T

          (* map the point onto a grid location that would have the same screen position as the given point.)

                (SETQ GRID (GREATESTPOWEROF2LT SCALE]
          (RETURN (create POSITION
                          XCOORD _ (NEAREST.ON.GRID (TIMES (fetch (POSITION XCOORD) of PT)
                                                           SCALE)
                                                    GRID)
                          YCOORD _ (NEAREST.ON.GRID (TIMES (fetch (POSITION YCOORD) of PT)
                                                           SCALE)
                                                    GRID])
```

(VIEWER.TO.SKETCH.POSITION

```
  [LAMBDA (POSITION VIEWERSCALE)                                  (* rrb "11-Jul-86 15:56")

          (* Transforms a position from viewer coordinates into sketch coordinates.
          VIEWERSCALE can be a scale or a viewer.)

    (SK.UNSCALE.POSITION.FROM.VIEWER POSITION (COND
                                                ((NUMBERP VIEWERSCALE))
                                                ((WINDOWP VIEWERSCALE)
                                                 (VIEWER.SCALE VIEWERSCALE))
                                                (T (\ILLEGAL.ARG VIEWERSCALE])
```

(SK.INSURE.SCALE

```
  [LAMBDA (VIEWERSCALE)                                           (* rrb "11-Jul-86 15:52")
    (COND
       ((NUMBERP VIEWERSCALE))
       ((WINDOWP VIEWERSCALE)
        (VIEWER.SCALE VIEWERSCALE))
       (T (\ILLEGAL.ARG VIEWERSCALE])
```

(SKETCH.TO.VIEWER.REGION

```
  [LAMBDA (REGION VIEWERSCALE)                                    (* rrb " 6-Jun-86 14:05")

          (* Transforms a region from sketch coordinates into viewer coordinates.
          VIEWERSCALE can be a scale or a viewer.)

    (PROG ((SCALE (SK.INSURE.SCALE VIEWERSCALE)))
          (RETURN (CREATEREGION (QUOTIENT (fetch (REGION LEFT) of REGION)
                                          SCALE)
                                (QUOTIENT (fetch (REGION BOTTOM) of REGION)
                                          SCALE)
                                (QUOTIENT (fetch (REGION WIDTH) of REGION)
                                          SCALE)
                                (QUOTIENT (fetch (REGION HEIGHT) of REGION)
                                          SCALE])
```

(VIEWER.TO.SKETCH.REGION

```
  [LAMBDA (REGION VIEWERSCALE)                                    (* rrb " 6-Jun-86 14:05")

          (* Transforms a region from viewer coordinates into sketch coordinates.
          VIEWERSCALE can be a scale or a viewer.)

    (UNSCALE.REGION REGION (SK.INSURE.SCALE VIEWERSCALE])
```

(SK.READ.POINT.WITH.FEEDBACK

```
  [LAMBDA (WINDOW CURSOR FEEDBACKFN FEEDBACKFNDATA BUTTONFOREXISTINGPTS CONSTRAINTFN NUMBERPADTOOFLG)
                                                                  (* rrb "11-Jul-86 15:52")

          (* internal function that reads a point from the user. Each time the cursor moves, a feedback fn is called passing it the new
          X, new Y, WINDOW and FEEDBACKDATA It is expected to XOR something on the screen that tells the user something.)

    (RESETLST
```

```
        (RESETSAVE (CURSOR (OR CURSOR CROSSHAIRS)))
        (RESETSAVE NIL (LIST 'DSPOPERATION (DSPOPERATION 'INVERT WINDOW)
                             WINDOW))
        [PROG ((USEGRID (WINDOWPROP WINDOW 'USEGRID))
               (GRID (SK.GRIDFACTOR WINDOW))
               (SCALE (VIEWER.SCALE WINDOW))
               (HOTSPOTCACHE (SK.HOTSPOT.CACHE WINDOW))
               (SCRATCHPT (AND CONSTRAINTFN (create POSITION)))
               XSCREEN YSCREEN XGRID YGRID NEWX NEWY MOUSEDOWN LASTBUTTONSTATE ONGRID? NEARPOS CONSTRAINTPT
               POSITIONPAD)
              (OR FEEDBACKFN (SETQ FEEDBACKFN 'SHOWSKETCHXY))
              [COND
                 (NUMBERPADTOOFLG
```

(* IT WOULD BE NICER TO PUT THE POSITION READER OVERTOP OF THE MENU BUT THIS ROUTINE IS CALLED SEVERAL TIMES BY SEVERAL OF THE POINT READERS AND IT FLIPS UP AND DOWN SO STILL NEEDS MORE WORK TO GET RIGHT (* detach the window menu so that it doesn't come to top over the position reader.)
(AND (OPENWP (SETQ MENUW (SK.INSURE.HAS.MENU WINDOW)))
(RESETSAVE (DETACHWINDOW MENUW) (LIST (QUOTE SK.FIX.MENU) WINDOW))))

```
                  (RESETSAVE NIL (LIST 'CLOSEW (SETQ POSITIONPAD (SK.POSITION.PAD.FROM.VIEWER WINDOW]
        (RETURN (PROG1 (until [PROGN (GETMOUSESTATE)
                                     (COND
                                        [(AND POSITIONPAD (INSIDEP (WINDOWPROP POSITIONPAD 'REGION)
                                                                   LASTMOUSEX LASTMOUSEY))
                                         (COND
                                            ((AND XGRID (INSIDEP WINDOW XGRID YGRID))
                                                                 (* leaving the window, turn off the last feedback.)
                                             (APPLY* FEEDBACKFN XGRID YGRID WINDOW FEEDBACKFNDATA)
                                             (SETQ XGRID))) (* invoke position reader If it returns a position, return it.)
                                         (AND (SETQ YSCREEN (SK.READ.POSITION.PAD.HANDLER POSITIONPAD
                                                                 WINDOW FEEDBACKFN FEEDBACKFNDATA
                                                                 CONSTRAINTFN))
                                              (COND
                                                 [(EQ YSCREEN 'ABORT)
                                                  (COND
                                                     ((EQ NUMBERPADTOOFLG 'MULTIPLE)
```

(* if NUMBERPADTOOFLG is MULTIPLE, this is a context in which multiple values are being read and the only way to abort is to error. Note%: this leaves stuff on the screen.)

```
                                                      (ERROR!))
                                                     (T (RETURN NIL]
                                                 ((EQ YSCREEN 'STOP)
                                                  (RETURN NIL))
                                                 (T (RETURN YSCREEN]
                                        (MOUSEDOWN (LASTMOUSESTATE UP))
                                        ((LASTMOUSESTATE (OR LEFT MIDDLE RIGHT))
                                         (COND
                                            ((INSIDEP (WINDOWPROP WINDOW 'REGION)
                                                      LASTMOUSEX LASTMOUSEY)
                                             (SETQ MOUSEDOWN T)
                                             NIL)
                                            (T (RETURN]
                                  do (SETQ NEWX (LASTMOUSEX WINDOW))
                                     (SETQ NEWY (LASTMOUSEY WINDOW))
                                     [COND
                                        ((OR (NEQ NEWX XSCREEN)
                                             (NEQ NEWY YSCREEN)
                                             (NEQ LASTBUTTONSTATE LASTMOUSEBUTTONS))
                                                                 (* cursor changed position or a button went down, check if grid pt moved.)
                                         (SKETCHW.UPDATE.LOCATORS WINDOW)
                                         (SETQ XSCREEN NEWX)
                                         (SETQ YSCREEN NEWY)
                                         (SETQ LASTBUTTONSTATE LASTMOUSEBUTTONS)
                                         [COND
                                            ((AND HOTSPOTCACHE (SELECTQ BUTTONFOREXISTINGPTS
                                                                   (MIDDLE (LASTMOUSESTATE MIDDLE))
                                                                   (LEFT (LASTMOUSESTATE LEFT))
                                                                   NIL)
                                                 (SETQ NEARPOS (NEAREST.HOT.SPOT HOTSPOTCACHE NEWX NEWY)))
                                                                 (* on middle, pick the closest point)
                                             (SETQ NEWX (fetch (POSITION XCOORD) of NEARPOS))
                                             (SETQ NEWY (fetch (POSITION YCOORD) of NEARPOS))
                                             (SETQ ONGRID? NIL))
                                            ((SETQ ONGRID? (COND
                                                              ((LASTMOUSESTATE RIGHT)
                                                                   (* if right is down, flip sense of using grid)
                                                               (NOT USEGRID))
                                                              (T   (* otherwise use the grid if told to.)
                                                                 USEGRID)))
                                             (SETQ NEWX (MAP.WINDOW.ONTO.GRID NEWX SCALE GRID))
                                             (SETQ NEWY (MAP.WINDOW.ONTO.GRID NEWY SCALE GRID]
                                         (PROGN [COND
                                                   ([AND CONSTRAINTFN
                                                         (POSITIONP (SETQ CONSTRAINTPT
```

```
                                                    (APPLY* CONSTRAINTFN
                                                           (MAP.VIEWER.XY.INTO.GLOBAL NEWX NEWY
                                                                  VIEWER ONGRID? SCRATCHPT)
                                                           W FEEDBACKFNDATA]
                                                (SETQ NEWX (FIXR (QUOTIENT (fetch (POSITION XCOORD)
                                                                              of CONSTRAINTPT)
                                                                           SCALE)))
                                                (SETQ NEWY (FIXR (QUOTIENT (fetch (POSITION YCOORD)
                                                                              of CONSTRAINTPT)
                                                                           SCALE]
                                            (COND
                                               ((OR (NEQ XGRID NEWX)
                                                    (NEQ YGRID NEWY))
```

(* grid point has changed too. Call the feedback function if the point is in the window.
If it is outside, don't show anything.)

```
                                                (AND XGRID (INSIDEP WINDOW XGRID YGRID)
                                                     (APPLY* FEEDBACKFN XGRID YGRID WINDOW FEEDBACKFNDATA))
                                                (AND (INSIDEP WINDOW (SETQ XGRID NEWX)
                                                              (SETQ YGRID NEWY))
                                                     (APPLY* FEEDBACKFN XGRID YGRID WINDOW FEEDBACKFNDATA]
                                    finally (RETURN (COND
                                                       ((AND XGRID (INSIDEP WINDOW XGRID YGRID))
                                                                  (* if the cursor was outside the window when let up, return NIL)
                                                        (APPLY* FEEDBACKFN XGRID YGRID WINDOW FEEDBACKFNDATA)
                                                        (create INPUTPT
                                                                INPUT.ONGRID? _ ONGRID?
                                                                INPUT.POSITION _
                                                                (create POSITION
                                                                        XCOORD _ XGRID
                                                                        YCOORD _ YGRID])])
```

(SKETCH.GET.POSITION

```
[LAMBDA (VIEWER CURSOR FEEDBACKFN FEEDBACKFNDATA CONSTRAINTFN) (* rrb "21-May-86 16:51")
                                                               (* user available version of
                                                               SK.READ.POINT.WITH.FEEDBACK.)
```

(* reads a point from the user. Each time the cursor moves, a feedback fn is called passing it the new X, new Y, VIEWER and FEEDBACKDATA It is expected to XOR something on the screen that tells the user something. CONSTRAINTFN is called to constrain the read point.)

```
  (PROG (READPT)
        (SETQ READPT (SK.READ.POINT.WITH.FEEDBACK VIEWER CURSOR FEEDBACKFN FEEDBACKFNDATA 'MIDDLE
                          CONSTRAINTFN SKETCH.USE.POSITION.PAD))
        (RETURN (COND
                   ((NULL READPT)
                    (RETURN NIL))
                   ((EQ (fetch (INPUTPT INPUT.ONGRID?) of READPT)
                        'GLOBAL)                          (* user entered a global number directly.)
                    (fetch (INPUTPT INPUT.GLOBALPOSITION) of READPT))
                   (T (MAP.VIEWER.XY.INTO.GLOBAL (fetch (POSITION XCOORD) of (fetch (INPUTPT INPUT.POSITION)
                                                                                    of READPT))
                              (fetch (POSITION YCOORD) of (fetch (INPUTPT INPUT.POSITION) of READPT))
                              VIEWER
                              (fetch (INPUTPT INPUT.ONGRID?) of READPT)
                              (create POSITION])
```

(\CLOBBER.POSITION

```
[LAMBDA (X Y OLDPT)                                       (* rrb " 4-Apr-86 13:34")
```

(* returns a position with values x and y. Clobbers OLDPT if it is a POSITION.)

```
  (COND
     ((POSITIONP OLDPT)
      (replace (POSITION XCOORD) of OLDPT with X)
      (replace (POSITION YCOORD) of OLDPT with Y)
      OLDPT)
     (T (CREATEPOSITION X Y])
```

(NEAREST.HOT.SPOT

```
[LAMBDA (CACHE X Y)                                       (* rrb "31-Jul-85 10:14")
                                                          (* returns the nearest hot spot to X Y)
  (PROG ((BESTMEASURE 10000)
         BESTX BESTY YDIF THISDIF)
        [for YBUCKET in CACHE do (SETQ YDIF (ABS (DIFFERENCE (CAR YBUCKET)
                                                             Y)))
                                 (for XBUCKET in (CDR YBUCKET)
                                    do (COND
                                          ((CDR XBUCKET)          (* this bucket has entries)
                                                                  (* use Manhattan distance for efficiency.)
                                           [SETQ THISDIF (PLUS YDIF (ABS (DIFFERENCE (CAR XBUCKET)
                                                                                     X]
                                           (COND
```

```
                                        ((ILESSP THISDIF BESTMEASURE)
                                         (SETQ BESTMEASURE THISDIF)
                                         (SETQ BESTX (CAR XBUCKET))
                                         (SETQ BESTY (CAR YBUCKET]
        (RETURN (AND BESTX (create POSITION
                                XCOORD _ BESTX
                                YCOORD _ BESTY])
```

(GETWREGION

```
  [LAMBDA (W NEWREGIONFN NEWREGIONFNDATA MINWIDTH MINHEIGHT)     ; Edited 12-Jun-90 13:25 by mitani
                                                                 (* gets a region from a window)
    (PROG ((REG (GETREGION MINWIDTH MINHEIGHT NIL NEWREGIONFN NEWREGIONFNDATA)))
          (RETURN (CREATEREGION (IDIFFERENCE (fetch (REGION LEFT) of REG)
                                             (DSPXOFFSET NIL W))
                                (IDIFFERENCE (fetch (REGION BOTTOM) of REG)
                                       (DSPYOFFSET NIL W))
                                (fetch (REGION WIDTH) of REG)
                                (fetch (REGION HEIGHT) of REG])
```

(GET.BITMAP.POSITION

```
  [LAMBDA (WINDOW BITMAP OPERATION MSG XOFFSET YOFFSET)          (* rrb "11-Jul-85 11:00")

          (* gets a position by tracking with a bitmap The spec returns is actually
          (ONGRID? position) so that caller can tell whether it was placed on grid or not.)

    (PROG (BUFFER.BITMAP WIDTH HEIGHT)
          (SETQ WIDTH (BITMAPWIDTH BITMAP))
          (SETQ HEIGHT (BITMAPHEIGHT BITMAP))
          (SETQ BUFFER.BITMAP (BITMAPCREATE WIDTH HEIGHT))
          (STATUSPRINT WINDOW "
                 " MSG)
          (RETURN (SK.TRACK.BITMAP1 WINDOW BITMAP BUFFER.BITMAP WIDTH HEIGHT (OR OPERATION 'PAINT)
                         XOFFSET YOFFSET])
```

(SK.TRACK.BITMAP1

```
  [LAMBDA (W BITMAP BUFFER.BITMAP WIDTH HEIGHT OPERATION XOFFSET YOFFSET CONSTRAINTFN CONSTRAINTDATA FEEDBACKFN)
                                                                 (* rrb "11-Jul-86 15:52")

          (* tracks BITMAP until a button goes down and comes up. It calls CONSTRAINTFN to determine the position at which to
          display the image. Returns a list of (ongrid? position) so that caller can know whether the point chosen was on a grid or not.)
                                                                 (* there is other code in BIGFONT that might be better for this.)
    (PROG [DOWN LEFT BOTTOM NEW.LEFT NEW.BOTTOM GRID.LEFT GRID.BOTTOM ONGRID? NEARPOS CONSTRAINTPT
                (DSP (WINDOWPROP W 'DSP))
                (USEGRID (WINDOWPROP W 'USEGRID))
                (GRID (SK.GRIDFACTOR W))
                (SCALE (VIEWER.SCALE W))
                (HOTSPOTCACHE (SK.HOTSPOT.CACHE W))
                CONSTRAINTPT
                (SCRATCHPT (AND CONSTRAINTFN (create POSITION]
          (OR XOFFSET (SETQ XOFFSET 0))
          (OR YOFFSET (SETQ YOFFSET 0))
          (TOTOPW W)
          (RETURN (until (AND DOWN (LASTMOUSESTATE UP))
                     do (GETMOUSESTATE)
                        (COND
                           ((LASTMOUSESTATE (NOT UP))
                            (SETQ DOWN T)))
                        (SETQ NEW.LEFT (LASTMOUSEX DSP))
                        (SETQ NEW.BOTTOM (LASTMOUSEY DSP))
                        [COND
                           ((OR (NEQ NEW.LEFT LEFT)
                                (NEQ NEW.BOTTOM BOTTOM))         (* cursor changed position check if grid pt moved.)
                            (SKETCHW.UPDATE.LOCATORS W)
                            (SETQ LEFT NEW.LEFT)
                            (SETQ BOTTOM NEW.BOTTOM)
                            [COND
                               ((AND HOTSPOTCACHE (LASTMOUSESTATE MIDDLE)
                                     (SETQ NEARPOS (NEAREST.HOT.SPOT HOTSPOTCACHE NEW.LEFT NEW.BOTTOM)))
                                                                 (* on middle, pick the closest point)
                                (SETQ ONGRID? NIL)
                                (SETQ NEW.LEFT (fetch (POSITION XCOORD) of NEARPOS))
                                (SETQ NEW.BOTTOM (fetch (POSITION YCOORD) of NEARPOS)))
                               ((SETQ ONGRID? (COND
                                                 ((LASTMOUSESTATE RIGHT)
                                                                 (* if right is down, flip sense of using grid)
                                                  (NOT USEGRID))
                                                 (T              (* otherwise use the grid if told to.)
                                                    USEGRID)))
                                (SETQ NEW.LEFT (MAP.WINDOW.ONTO.GRID NEW.LEFT SCALE GRID))
                                (SETQ NEW.BOTTOM (MAP.WINDOW.ONTO.GRID NEW.BOTTOM SCALE GRID]
                                                                 (* check the constraintfn)
                            [COND
                               ([AND CONSTRAINTFN (POSITIONP (SETQ CONSTRAINTPT
                                                                   (APPLY* CONSTRAINTFN
```

```
                                                                    (MAP.VIEWER.XY.INTO.GLOBAL NEW.LEFT
                                                                          NEW.BOTTOM W ONGRID? SCRATCHPT)
                                                                    W CONSTRAINTDATA]
                                                                (* scale the returns global position into window coordinates)
                                    (SETQ NEW.LEFT (FIXR (QUOTIENT (fetch (POSITION XCOORD) of CONSTRAINTPT)
                                                                   SCALE)))
                                    (SETQ NEW.BOTTOM (FIXR (QUOTIENT (fetch (POSITION YCOORD) of CONSTRAINTPT)
                                                                     SCALE]
                                 (COND
                                    ((OR (NEQ NEW.LEFT GRID.LEFT)
                                         (NEQ NEW.BOTTOM GRID.BOTTOM))
                                                                (* grid location changed, move the text image.)
                                     [COND
                                        (GRID.LEFT (AND FEEDBACKFN (APPLY* FEEDBACKFN GRID.LEFT GRID.BOTTOM W
                                                                          CONSTRAINTDATA))
                                                   (BITBLT BUFFER.BITMAP 0 0 W (IPLUS GRID.LEFT XOFFSET)
                                                          (IPLUS GRID.BOTTOM YOFFSET)
                                                          WIDTH HEIGHT 'INPUT 'REPLACE]
                                     (SETQ GRID.LEFT NEW.LEFT)
                                     (SETQ GRID.BOTTOM NEW.BOTTOM)
                                     (BITBLT W (IPLUS GRID.LEFT XOFFSET)
                                            (IPLUS GRID.BOTTOM YOFFSET)
                                            BUFFER.BITMAP 0 0 NIL NIL 'INPUT 'REPLACE)
                                     (BITBLT BITMAP 0 0 DSP (IPLUS GRID.LEFT XOFFSET)
                                            (IPLUS GRID.BOTTOM YOFFSET)
                                            WIDTH HEIGHT 'INPUT OPERATION)
                                     (AND FEEDBACKFN (APPLY* FEEDBACKFN GRID.LEFT GRID.BOTTOM W CONSTRAINTDATA]
                           finally [COND
                                      (GRID.LEFT                (* restore screen)
                                             (AND FEEDBACKFN (APPLY* FEEDBACKFN GRID.LEFT GRID.BOTTOM W CONSTRAINTDATA))
                                             (BITBLT BUFFER.BITMAP 0 0 W (IPLUS GRID.LEFT XOFFSET)
                                                    (IPLUS GRID.BOTTOM YOFFSET)
                                                    WIDTH HEIGHT 'INPUT 'REPLACE]
                                                                (* return the position if any part of the bitmap is visible.)
                                 (RETURN (AND (REGIONSINTERSECTP (DSPCLIPPINGREGION NIL DSP)
                                                                 (CREATEREGION (IPLUS LEFT XOFFSET)
                                                                        (IPLUS BOTTOM YOFFSET)
                                                                        WIDTH HEIGHT))
                                              (create INPUTPT
                                                     INPUT.ONGRID? _ ONGRID?
                                                     INPUT.POSITION _ (create POSITION
                                                                              XCOORD _ GRID.LEFT
                                                                              YCOORD _ GRID.BOTTOM])

)

(DECLARE%: EVAL@COMPILE

[RECORD INPUTPT (INPUT.ONGRID? INPUT.POSITION INPUT.GLOBALPOSITION)
       (TYPE? (AND (LISTP DATUM)
                   (OR (NULL (CAR DATUM))
                       (EQ (CAR DATUM)
                           T))
                   (LISTP (CDR DATUM))
                   (POSITIONP (CADR DATUM]
)

;; stuff to allow reading positions from a number pad

(RPAQ? SKETCH.USE.POSITION.PAD NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCH.USE.POSITION.PAD)
)

(DEFINEQ

(SK.BRING.UP.POSITION.PAD
  [LAMBDA (VIEWER MSG OPENFLG)                                  (* rrb "10-Jun-86 15:26")

          (* * brings up a position reading number pad associated with VIEWER.
          Puts it over the menu if it is up.)

    (RESETFORM (RADIX 10)
           (PROG ((NUMBER/READER/MAXDIGITS 8)
                  (MARGIN 6)
                  (DIGITFONT (FONTCREATE 'MODERN 12 'BOLD))
                  (MSGFONT (FONTCREATE DEFAULTFONT))
                  (VIEWERREGION (WINDOWPROP VIEWER 'REGION))
                  WIN WINWIDTH WINHEIGHT TOTALSWIDTH TOTALSHEIGHT FONTHEIGHT MSGLINES XNUMBERPAD YNUMBERPAD
                  COMMANDPAD)
                 [SETQ TOTALSWIDTH (IPLUS 12 (ITIMES (ADD1 NUMBER/READER/MAXDIGITS)
                                                     (CHARWIDTH (CHARCODE 0)
                                                                DIGITFONT]
                 [SETQ TOTALSHEIGHT (PLUS 2 (FONTPROP DIGITFONT 'HEIGHT]
                 (SETQ XNUMBERPAD (\POSITION.READER.NUMBERPAD DIGITFONT TOTALSWIDTH))
```

```
                (SETQ YNUMBERPAD (\POSITION.READER.NUMBERPAD DIGITFONT TOTALSWIDTH))
                (SETQ COMMANDPAD (create MENU
                                         ITEMS _ '(abort enter quit)
                                         CENTERFLG _ T
                                         MENUFONT _ DIGITFONT
                                         WHENHELDFN _ (FUNCTION POSITIONPAD.HELDFN)
                                         WHENSELECTEDFN _ (FUNCTION POSITION.PAD.READER.HANDLER)
                                         MENUBORDERSIZE _ 1
                                         MENUOUTLINESIZE _ 2
                                         ITEMHEIGHT _ (PLUS 6 TOTALSHEIGHT)))
                                                        (* leave room for three lines and the number at the top)
                                                        (* use the numberpad's width so things look better.)
                (SETQ TOTALSWIDTH (fetch (MENU IMAGEWIDTH) of XNUMBERPAD))
                (SETQ WINWIDTH (IPLUS (TIMES 2 (PLUS MARGIN TOTALSWIDTH))
                                      MARGIN
                                      (fetch (MENU IMAGEWIDTH) of COMMANDPAD)
                                      MARGIN))
                (SETQ WINHEIGHT (IPLUS (COND
                                          [MSG
```

(* if there is a msg, leave room for it at the top. In any case, leave room for the labels X and Y.)

```
                                                (ITIMES (LENGTH (SETQ MSGLINES (BREAK.MSG.INTO.LINES MSG MSGFONT
                                                                                                     WINWIDTH)))
                                                        (FONTPROP MSGFONT 'HEIGHT]
                                                   (T 0))
                                                (FONTPROP DIGITFONT 'HEIGHT)
                                                (TIMES MARGIN 3)
                                                TOTALSHEIGHT MARGIN (fetch (MENU IMAGEHEIGHT) of XNUMBERPAD)))
                [SETQ WINHEIGHT (HEIGHTIFWINDOW WINHEIGHT NIL (WINDOWPROP VIEWER 'BORDER]
                (SETQ WIN (CREATEW (CREATEREGION 0 0 (WIDTHIFWINDOW WINWIDTH)
                                                 WINHEIGHT)
                                   NIL
                                   (WINDOWPROP VIEWER 'BORDER)
                                   T))
                (MOVEW WIN (SK.PAD.READER.POSITION VIEWER WIN))
                (WINDOWADDPROP WIN 'REPAINTFN (FUNCTION SK.POSITION.READER.REPAINTFN))
                [COND
                   (MSG                                          (* save msg on the window so repaintfn can get at it)
                        (WINDOWPROP WIN 'MESSAGE MSGLINES)
                        (WINDOWPROP WIN 'MESSAGEFONT MSGFONT)    (* note where the message begins.)
                        (MOVETOUPPERLEFT WIN)
                        (WINDOWPROP WIN 'MESSAGEBOTTOM (DSPYPOSITION NIL WIN]
                (WINDOWPROP WIN 'DIGITFONT DIGITFONT)
                (OPENW WIN)
```

(* window is opened because of bug in ADDMENU that it doesn't work unless window is open.)

```
                (\POSITION.PAD.ADD.DIGIT.MENU WIN MARGIN MARGIN 'X XNUMBERPAD TOTALSWIDTH TOTALSHEIGHT
                       NUMBER/READER/MAXDIGITS)
                (\POSITION.PAD.ADD.DIGIT.MENU WIN (PLUS MARGIN TOTALSWIDTH MARGIN)
                       MARGIN
                       'Y YNUMBERPAD TOTALSWIDTH TOTALSHEIGHT NUMBER/READER/MAXDIGITS)
                (REDISPLAYW WIN NIL T)
                [ADDMENU COMMANDPAD WIN (create POSITION
                                             XCOORD _ (PLUS MARGIN (TIMES 2 (PLUS MARGIN TOTALSWIDTH)))
                                             YCOORD _ (PLUS MARGIN (QUOTIENT (DIFFERENCE (fetch (MENU
                                                                                                 IMAGEHEIGHT
                                                                                                 )
                                                                                           of XNUMBERPAD)
                                                                                 (fetch (MENU IMAGEHEIGHT)
                                                                                    of COMMANDPAD))
                                                                             2]
                (OR OPENFLG (CLOSEW WIN))
                (RETURN WIN])
```

(SK.PAD.READER.POSITION

```
  [LAMBDA (VIEWER READERWINDOW)                                  (* rrb "10-Jun-86 12:24")
```

(* returns the lower left corner where a position reading pad should be placed for the sketch viewer VIEWER.)

```
    (PROG ((VIEWERREGION (WINDOWPROP VIEWER 'REGION))
           (READERREGION (WINDOWPROP READERWINDOW 'REGION))
           VLFT VBTM PWID)
          (SETQ VLFT (fetch (REGION LEFT) of VIEWERREGION))
          (SETQ VBTM (fetch (REGION BOTTOM) of VIEWERREGION))
          (SETQ PWID (fetch (REGION WIDTH) of READERREGION))
          (RETURN (COND
                     [(OR (GREATERP VLFT PWID)
                          (GREATERP VLFT VBTM)
                          (GREATERP PWID (fetch (REGION WIDTH) of VIEWERREGION)))
```

(* the position reader will fit to the left, or there is more room on the left, or the position pad reader is wider than the viewer.)

```
                      (create POSITION
                              XCOORD _ (DIFFERENCE (MAX 10 VLFT)
```

```
                                    PWID)
                            YCOORD _ (DIFFERENCE (fetch (REGION PTOP) of VIEWERREGION)
                                                 (fetch (REGION HEIGHT) of READERREGION]
                        (T                                              (* more room on the bottom)
                           (create POSITION
                                  XCOORD _ (MAX 10 VLFT)
                                  YCOORD _ (DIFFERENCE VBTM (fetch (REGION HEIGHT) of READERREGION])
```

(SK.POSITION.READER.REPAINTFN

```
[LAMBDA (POSITIONPAD)                                           (* rrb "11-Jun-86 13:28")
                                                                (* repaints a position pad reader)
   (PROG ((MSGLINES (WINDOWPROP POSITIONPAD 'MESSAGE))
          NUMBERMENU TOTALREGION)
         [COND
            (MSGLINES                                           (* if there is a msg, print it at the top.)
                   (DSPFONT (WINDOWPROP POSITIONPAD 'MESSAGEFONT)
                          POSITIONPAD)
                   (MOVETO 0 (WINDOWPROP POSITIONPAD 'MESSAGEBOTTOM)
                          POSITIONPAD)
                   (for LINE in MSGLINES do (PRIN3 LINE POSITIONPAD)
                                            (TERPRI POSITIONPAD]
         (DSPFONT (WINDOWPROP POSITIONPAD 'DIGITFONT)
                POSITIONPAD)                                    (* the actual displaying of the menus is done by the repaintfn
                                                                supplied by ADDMENU)
         (for LABEL in '(X Y) do (SETQ NUMBERMENU (WINDOWPROP POSITIONPAD LABEL))
                                 (SETQ TOTALREGION (GETMENUPROP NUMBERMENU 'TOTALREG))
                                 (\READNUMBER.OUTLINEREGION TOTALREGION POSITIONPAD 2)
                                 (CENTERPRINTINAREA LABEL (fetch (REGION LEFT) of TOTALREGION)
                                        (PLUS 6 (fetch (REGION TOP) of TOTALREGION))
                                        (fetch (REGION WIDTH) of TOTALREGION)
                                        (fetch (REGION HEIGHT) of TOTALREGION)
                                        POSITIONPAD)
                                 (DISPLAY.POSITION.READER.TOTAL NUMBERMENU])
```

(SK.POSITION.PAD.FROM.VIEWER

```
[LAMBDA (VIEWER)                                                (* rrb "11-Jun-86 14:17")
                                                                (* cache the position pad because it takes a while to create.
                                                                Opens it too.)
   (PROG (PAD)
         (COND
            ((SETQ PAD (WINDOWPROP VIEWER 'POSITION.PAD))
             (WINDOWPROP PAD 'FINISHEDFLG NIL)                  (* move the pad in case the window has moved or been
                                                                reshaped.)
             (MOVEW PAD (SK.PAD.READER.POSITION VIEWER PAD))
             (OPENW PAD)                                        (* initialize some values)
             (SK.INIT.POSITION.NUMBER.PAD.MENU (WINDOWPROP PAD 'X))
             (SK.INIT.POSITION.NUMBER.PAD.MENU (WINDOWPROP PAD 'Y))
             (RETURN PAD))
            (T                                                  (* flip cursor because this may require font search)
               (RESETFORM (CURSOR WAITINGCURSOR)
                      (SETQ PAD (SK.BRING.UP.POSITION.PAD VIEWER "Select the location of the desired position in
                                     the window or enter its X and Y coordinates here." T)))
               (WINDOWPROP VIEWER 'POSITION.PAD PAD)
               (RETURN PAD])
```

(SK.INIT.POSITION.NUMBER.PAD.MENU

```
[LAMBDA (MNU)                                                   (* rrb "21-May-86 15:29")
                                                                (* reinitializes a numberpad reader)
   (PUTMENUPROP MNU 'TOTAL 0)
   (PUTMENUPROP MNU 'DECIMALPOWER NIL)
   (DISPLAY.POSITION.READER.TOTAL MNU])
```

(SK.READ.POSITION.PAD.HANDLER

```
[LAMBDA (POSITIONPAD VIEWER FEEDBACKFN FEEDBACKFNDATA CONSTRAINTFN)
                                                                (* rrb "11-Jul-86 15:54")

        (* tracks the cursor while it is in the position pad and sets variables for SK.READ.POINT.WITH.FEEDBACK and returned T
        if it succeeded)
                                                                (* uses many variable freely from
                                                                SK.READ.POINT.WITH.FEEDBACK)
   (PROG (NEWX NEWY CONSTRX CONSTRY PREVX PREVY FINISHVAL (SCALE (VIEWER.SCALE VIEWER)))
         (SPAWN.MOUSE)
         (WINDOWADDPROP POSITIONPAD 'CLOSEFN (FUNCTION \NUMBERPAD.READER.CLOSEFN))
         (RETURN (until [PROGN (GETMOUSESTATE)
                               (OR (NOT (INSIDEP (WINDOWPROP POSITIONPAD 'REGION)
                                                 LASTMOUSEX LASTMOUSEY))
                                   (SETQ FINISHVAL (WINDOWPROP POSITIONPAD 'FINISHEDFLG NIL]
                     do                                         (* keep bringing the numberpad to the top.)
                        (TOTOPW POSITIONPAD)
                        (DISMISS 100)
                        (SETQ NEWX (GETMENUPROP (WINDOWPROP POSITIONPAD 'X)
                                                'TOTAL))
                        (SETQ NEWY (GETMENUPROP (WINDOWPROP POSITIONPAD 'Y)
```

```
                              'TOTAL))
                    [COND
                       ((OR (NEQ NEWX PREVX)
                            (NEQ NEWY PREVY))              (* user entered a new number)
                        (SETQ PREVX NEWX)
                        (SETQ PREVY NEWY)
```

(* this code is differerent from the code in SK.READ.POINT.WITH.FEEDBACK in that is works in sketch coordinates.)

```
                        [COND
                           ([AND CONSTRAINTFN (POSITIONP (SETQ CONSTRAINTPT (APPLY* CONSTRAINTFN
                                                                                    (SK.SET.POSITION NEWX
                                                                                           NEWY SCRATCHPT)
                                                                                    VIEWER FEEDBACKFNDATA]
                            (SETQ NEWX (fetch (POSITION XCOORD) of CONSTRAINTPT))
                            (SETQ NEWY (fetch (POSITION YCOORD) of CONSTRAINTPT]
                        (COND
                           ((OR (NEQ CONSTRX NEWX)
                                (NEQ CONSTRY NEWY))
```

(* grid point has changed too. Update the position numberpads and Call the feedback function if the point is in the window.
If it is outside, don't show anything.)

```
                            (PUTMENUPROP (WINDOWPROP POSITIONPAD 'X)
                                  'TOTAL NEWX)
                            (PUTMENUPROP (WINDOWPROP POSITIONPAD 'Y)
                                  'TOTAL NEWY)
                            (DISPLAY.POSITION.READER.TOTAL (WINDOWPROP POSITIONPAD 'X))
                            (DISPLAY.POSITION.READER.TOTAL (WINDOWPROP POSITIONPAD 'Y))
                            (AND CONSTRX (APPLY* FEEDBACKFN CONSTRX CONSTRY VIEWER FEEDBACKFNDATA))
                            (APPLY* FEEDBACKFN (QUOTIENT (SETQ CONSTRX NEWX)
                                                         SCALE)
                                   (QUOTIENT (SETQ CONSTRY NEWY)
                                          SCALE)
                                   VIEWER FEEDBACKFNDATA]
                  finally                                  (* remove the closefn so that it doesn't get run on the way out.)
                (WINDOWDELPROP POSITIONPAD 'CLOSEFN (FUNCTION \NUMBERPAD.READER.CLOSEFN))
                (AND CONSTRX (APPLY* FEEDBACKFN CONSTRX CONSTRY VIEWER FEEDBACKFNDATA))
                (RETURN (COND
                           ((MEMB FINISHVAL '(STOP ABORT))
```

(* means the numberpad reader was closed. If the number pad includes the ABORT command, do what it would do,
otherwise the program is not expecting NIL so cause an error.)

```
                            (RETURN FINISHVAL))
                           (FINISHVAL (AND CONSTRX (SETQ FINISHVAL
                                                         (create POSITION
                                                                 XCOORD _ CONSTRX
                                                                 YCOORD _ CONSTRY))
                                                   (create INPUTPT
                                                          INPUT.ONGRID? _ 'GLOBAL
                                                          INPUT.POSITION _ (SK.SCALE.POSITION.INTO.VIEWER
                                                                                FINISHVAL SCALE)
                                                          INPUT.GLOBALPOSITION _ FINISHVAL)))
                           (T                                  (* mouse left the window, return)
                              NIL])
```

(DISPLAY.POSITION.READER.TOTAL

```
  [LAMBDA (MNU)                                            (* rrb "19-May-86 17:09")
                                                           (* displays the number total in the box in the window.)
    (PROG ((TOTALREG (GETMENUPROP MNU 'TOTALREG))
           (DECIMALPLACES (GETMENUPROP MNU 'DECIMALPOWER))
           (WIN (WFROMMENU MNU)))
          (DSPFILL TOTALREG WHITESHADE 'REPLACE WIN)
          (RESETFORM (RADIX 10)
                 (CENTERPRINTINREGION
                  [COND
                     [DECIMALPLACES
```

(* printing a decimal number must check to make sure the correct number of decimal places print.)

```
                      (PROG ([TOTSTR (MKSTRING (GETMENUPROP MNU 'TOTAL]
                             DECPOS NAFTERDEC NCHARS)
                            (SETQ NCHARS (NCHARS TOTSTR))
                            (SETQ DECPOS (STRPOS "." TOTSTR))
                            (RETURN (COND
                                       ((EQ (SUB1 DECIMALPLACES)
                                            (SETQ NAFTERDEC (DIFFERENCE NCHARS DECPOS)))
                                                           (* right number of places)
                                        TOTSTR)
                                       [(GEQ NAFTERDEC DECIMALPLACES)
                                                           (* strip off the unwanted ones.)
                                        (SUBSTRING TOTSTR 1 (PLUS DECPOS (SUB1 DECIMALPLACES]
                                       (T                  (* not enough zeros on the end)
                                          (CONCAT TOTSTR (bind STR for I from 1
                                                            to (DIFFERENCE (SUB1 DECIMALPLACES)
```

```
                                                        NAFTERDEC)
                                                  do (COND
                                                        (STR (SETQ STR (CONCAT STR "0")))
                                                        (T (SETQ STR "0")))
                                                  finally (RETURN STR]
                    (T (GETMENUPROP MNU 'TOTAL]
                  TOTALREG WIN])
```

(POSITION.PAD.READER.HANDLER

```
[LAMBDA (DIGIT MNU)                                     (* rrb "10-Jun-86 15:50")
                                                        (* handles a key stroke or menu digit selection for a number pad
```

reader.)

```
(PROG (TOTAL POWER OPERATION TOPOFSTACK (WIN (WFROMMENU MNU)))
      (SETQ TOTAL (GETMENUPROP MNU 'TOTAL))
      [PUTMENUPROP MNU 'TOTAL
             (SELECTQ DIGIT
                  ((← bs)
                       (COND
                          ((NULL (GETMENUPROP MNU 'DIGITYET)) (* bs was the first key)
                           (PUTMENUPROP MNU 'DIGITYET T)
                           0)
                          [(SETQ POWER (GETMENUPROP MNU 'DECIMALPOWER))
                                                        (* have read decimal pt -
                                                        much harder)
                           (COND
                              ((EQ POWER 1)             (* backspace over the decimal point.)
                               (PUTMENUPROP MNU 'DECIMALPOWER NIL)
                               (FIX TOTAL))
                              (T (PUTMENUPROP MNU 'DECIMALPOWER (SETQ POWER (SUB1 POWER)))
                                                        (* dirty but effective.)
                                 (PROG ((TOTSTR (MKSTRING TOTAL)))
                                                        (* SUBSTRING will be NIL if the total has a trailing zero.)
                                       (RETURN (MKATOM (OR (SUBSTRING TOTSTR 1 (PLUS (STRPOS "." TOTSTR)
                                                                                     (SUB1 POWER)))
                                                           TOTSTR]
                          (T                            (* no decimal point)
                             (IQUOTIENT TOTAL 10))))
                  (±                                    (* +/- sign)
                     (MINUS TOTAL))
                  ((÷ × − + =)                          (* operation sign)
                       [COND
                          ((NULL (GETMENUPROP MNU 'DIGITYET)) (* last thing hit was an operation, just save this one.)
                           (PUTMENUPROP MNU 'OPERATION (COND
                                                          ((EQ DIGIT '=)
                                                           NIL)
                                                          (T DIGIT)))
                           (RETURN))
                          ((SETQ OPERATION (GETMENUPROP MNU 'OPERATION))
```

(* perform the operation that is stored between the top of stack and the current total)

```
                           (COND
                              [(SETQ TOPOFSTACK (GETMENUPROP MNU 'TOPOFSTACK))
                                                        (* a previous value exists)
                               (SETQ TOTAL (SELECTQ OPERATION
                                                (÷      (* divide, check for 0 divisor)
                                                   (COND
                                                      ((ZEROP TOTAL)
                                                       (PROMPTPRINT "Can't divide by zero"))
                                                      (T (QUOTIENT TOPOFSTACK TOTAL))))
                                                (×      (* times)
                                                   (TIMES TOPOFSTACK TOTAL))
                                                (−      (* minus)
                                                   (DIFFERENCE TOPOFSTACK TOTAL))
                                                (PLUS TOPOFSTACK TOTAL]
                                    (T TOTAL]
                       (PUTMENUPROP MNU 'TOPOFSTACK TOTAL)
                       (PUTMENUPROP MNU 'DIGITYET NIL)
                       (PUTMENUPROP MNU 'DECIMALPOWER NIL)
                       (PUTMENUPROP MNU 'OPERATION (COND
                                                      ((EQ DIGIT '=)
                                                       NIL)
                                                      (T DIGIT)))
                       TOTAL)
                  (%                                    (* empty key)
                     TOTAL)
                  (%.                                   (* decimal point)
                       (COND
                          ((GETMENUPROP MNU 'DECIMALPOWER)     (* already has a decimal pt, don't do anything)
                           (RETURN))
                          ((NULL (GETMENUPROP MNU 'DIGITYET)) (* first key hit is a decimal point.)
                           (PUTMENUPROP MNU 'DIGITYET T)
                           (PUTMENUPROP MNU 'DECIMALPOWER 1)
                           0.0)
                          (T (PUTMENUPROP MNU 'DECIMALPOWER 1)
                             (FLOAT TOTAL))))
```

```
                    (enter (WINDOWPROP WIN 'FINISHEDFLG T)
                           (RETURN))
                    (quit (WINDOWPROP WIN 'FINISHEDFLG 'STOP)
                          (RETURN))
                    (abort                                        (* abort key)
                           (WINDOWPROP WIN 'FINISHEDFLG 'ABORT)
                           (RETURN))
                    (C                                            (* clear key)
                       (PUTMENUPROP MNU 'DECIMALPOWER NIL)
                       (PUTMENUPROP MNU 'DIGITYET T)
                       (PUTMENUPROP MNU 'TOPOFSTACK NIL)
                       0)
                    (ce                                           (* clear key)
                        (PUTMENUPROP MNU 'DECIMALPOWER NIL)
                        (PUTMENUPROP MNU 'DIGITYET T)
                        0)
                    (COND
                       [(NUMBERP DIGIT)
                        (COND
                           ((NULL (GETMENUPROP MNU 'DIGITYET)) (* first key hit after an operation, note this and clear the total.)
                            (PUTMENUPROP MNU 'DIGITYET T)
                            (SETQ TOTAL 0)))
                        (COND
                           ((EQ (GETMENUPROP MNU 'MAXDIGITS)
                                (NCHARS (ABS TOTAL)))          (* don't take any more.)
                            (\READNUMBER.FLASHAREA 0 0 1000 1000 WIN)
                            TOTAL)
                           [(SETQ POWER (GETMENUPROP MNU 'DECIMALPOWER))
                                                               (* have read decimal pt)
                            (PUTMENUPROP MNU 'DECIMALPOWER (ADD1 POWER))
                            (SETQ POWER (bind (N _ 1.0) for I from 1 to POWER do (SETQ N (FTIMES N 0.1))
                                               finally (RETURN N)))
                            (COND
                               ((GEQ TOTAL 0)
                                (PLUS TOTAL (TIMES DIGIT POWER)))
                               (T (DIFFERENCE TOTAL (TIMES DIGIT POWER]
                           ((GEQ TOTAL 0)
                            (PLUS (TIMES TOTAL 10)
                                  DIGIT))
                           (T (DIFFERENCE (TIMES TOTAL 10)
                                          DIGIT]
                       (T                                      (* uninteresting key struck, ignore it)
                          (RETURN]
              (DISPLAY.POSITION.READER.TOTAL MNU])
```

(POSITIONPAD.HELDFN

```
  [LAMBDA (ITEM MENU BUTTON)                                   (* rrb "10-Jun-86 15:29")
                                                               (* prints the help information for a numberpad.)
    (PROMPTPRINT (SELECTQ ITEM
                    (enter "Indicates that you are through entering the position.")
                    (ce "Will reset the total to 0")
                    (C "Will clear the stack and set the total to 0")
                    (= "performs the previously specified operation between the memory and the current total")
                    (+ "Will read another number to be added to the current total")
                    (- "Will read another number to be subtracted to the current total")
                    (× "Will read another number to be multiplied by the current total")
                    (÷ "Will read another number and divides the current total by it")
                    (quit "Will stop prompting you for points.")
                    (abort "will abort this sketch operation.")
                    (± " will change the sign of the total")
                    (%. "will enter a decimal point.")
                    ((bs ←)
                        "Will erase the last digit entered.")
                    (%  "doesn't do anything.")
                   "Will put this digit on the right of the total."])
```

(\POSITION.PAD.ADD.DIGIT.MENU

```
  [LAMBDA (WIN LEFT MARGIN LABEL MENU TOTALSWIDTH TOTALSHEIGHT NUMBER/READER/MAXDIGITS)
                                                               (* rrb "10-Jun-86 12:06")
          (* * adds a menu which is a number pad menu to WIN, allocates the total region for it.)

    (PROG (TOTALREGION)
          (ADDMENU MENU WIN (create POSITION
                                   XCOORD _ LEFT
                                   YCOORD _ MARGIN))
          (PUTMENUPROP MENU 'TOTALREG (SETQ TOTALREGION (create REGION
                                                                LEFT _ LEFT
                                                                BOTTOM _ (PLUS (fetch (MENU IMAGEHEIGHT)
                                                                                  of MENU)
                                                                               MARGIN MARGIN)
                                                                WIDTH _ TOTALSWIDTH
                                                                HEIGHT _ TOTALSHEIGHT)))
          (PUTMENUPROP MENU 'TOTAL 0)
          (PUTMENUPROP MENU 'MAXDIGITS NUMBER/READER/MAXDIGITS)  (* put link to the menu so the window can eventually get the
                                                                 values.)
```

```
          (WINDOWPROP WIN LABEL MENU)
          (RETURN WIN])
```

(\POSITION.READER.NUMBERPAD

```
  [LAMBDA (DIGITFONT WIDTH)                                  (* rrb "10-Jun-86 15:33")
                                                             (* returns a menu which is a numberpad suitable for a position
                                                             reader.)
    (create MENU
            ITEMS _
            '(← ce C ÷ 1 2 3 × 4 5 6 - 7 8 9 + ± 0 %. =)
            MENUCOLUMNS _ 4
            CENTERFLG _ T
            MENUFONT _ DIGITFONT
            WHENHELDFN _ (FUNCTION POSITIONPAD.HELDFN)
            WHENSELECTEDFN _ (FUNCTION POSITION.PAD.READER.HANDLER)
            MENUOUTLINESIZE _ 2
            ITEMHEIGHT _ (IPLUS 2 (FONTPROP DIGITFONT 'HEIGHT))
            ITEMWIDTH _ (AND WIDTH (QUOTIENT (DIFFERENCE WIDTH 8)
                                             4])

)

(RPAQ? ALL.SKETCHES )

(RPAQ? INITIAL.SCALE 1.0)

(RPAQ? DEFAULT.VISIBLE.SCALE.FACTOR 10.0)

(RPAQ? MINIMUM.VISIBLE.SCALE.FACTOR 4.0)

(RPAQQ SKETCH.ELEMENT.TYPES NIL)

(RPAQQ SKETCH.ELEMENT.TYPE.NAMES NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS ALL.SKETCHES INITIAL.SCALE DEFAULT.VISIBLE.SCALE.FACTOR MINIMUM.VISIBLE.SCALE.FACTOR
       SKETCH.ELEMENT.TYPES SKETCH.ELEMENT.TYPE.NAMES SK.SELECTEDMARK SK.LOCATEMARK COPYSELECTIONMARK
       MOVESELECTIONMARK DELETESELECTIONMARK)
)

(READVARS-FROM-STRINGS '(SK.SELECTEDMARK SK.LOCATEMARK COPYSELECTIONMARK MOVESELECTIONMARK DELETESELECTIONMARK
                                 OTHERCONTROLPOINTMARK)
       "({(READBITMAP)(7 7
       %"ON@@%"
       %"ON@@%"
       %"ON@@%"
       %"ON@@%"
       %"ON@@%"
       %"ON@@%"
       %"ON@@%")}  {(READBITMAP)(11 11
       %"OON@%"
       %"OON@%"
       %"L@F@%"
       %"L@F@%"
       %"L@F@%"
       %"L@F@%"
       %"L@F@%"
       %"L@F@%"
       %"OON@%"
       %"OON@%")}  {(READBITMAP)(11 11
       %"@@@@%"
       %"EED@%"
       %"BJH@%"
       %"EED@%"
       %"BJH@%"
       %"EED@%"
       %"BJH@%"
       %"EED@%"
       %"BJH@%"
       %"EED@%"
       %"@@@@%")}  {(READBITMAP)(19 19
       %"OL@@@@@@%"
       %"N@@@@@@@%"
       %"O@@@@@@@%"
       %"KH@@@@@@%"
       %"I@@@@@@@%"
       %"H@@@@@@@%"
       %"@CH@@@@@%"
       %"@CL@@@@@%"
       %"@CN@@@@@%"
       %"@AO@@@@@%"
       %"@@OH@@@@%"
       %"@@GH@@@@%"
       %"@@CH@@@@%"
```

```
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%")}  {(READBITMAP)(13 13
       %"L@AH%"
       %"H@@H%"
       %"@@@@%"
       %"AHL@%"
       %"AML@%"
       %"@OH@%"
       %"@G@@%"
       %"@OH@%"
       %"AML@%"
       %"AHL@%"
       %"@@@@%"
       %"H@@H%"
       %"L@AH%")}  {(READBITMAP)(11 11
       %"@@@@%"
       %"@D@@%"
       %"BJH@%"
       %"AE@@%"
       %"BJH@%"
       %"EED@%"
       %"BJH@%"
       %"AE@@%"
       %"BJH@%"
       %"@D@@%"
       %"@@@@%")})
       ")
```

;; accessing functions for the methods of a sketch type.

```
(DEFINEQ
```

(SK.DRAWFN

```
  [LAMBDA (ELEMENTTYPE)                                              (* rrb "17-MAR-83 22:28")
                                                                     (* goes from an element type name to its DRAWFN)
    (fetch (SKETCHTYPE DRAWFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.TRANSFORMFN

```
  [LAMBDA (ELEMENTTYPE)                                              (* rrb " 7-Feb-85 12:08")
                                                                     (* goes from an element type name to its TRANSFORMFN)
    (fetch (SKETCHTYPE TRANSFORMFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.EXPANDFN

```
  [LAMBDA (ELEMENTTYPE)                                              (* goes from an element type name to its EXPANDFN)
    (fetch (SKETCHTYPE EXPANDFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])
```

(SK.INPUT

```
  [LAMBDA (ELEMENTTYPE SKETCHW)                                      (* rrb "11-MAR-83 09:54")
                                                                     (* applies an element types input function to a window.)
    (APPLY* (fetch (SKETCHTYPE INPUTFN) of ELEMENTTYPE)
           SKETCHW])
```

(SK.INSIDEFN

```
  [LAMBDA (ELEMENTTYPE)                                              (* rrb " 4-Oct-86 11:02")
                                                                     (* goes from an element type name to its inside predicate)
    (PROG (SKTYPE)
      LP  (COND
             ([NULL (SETQ SKTYPE (GETPROP ELEMENTTYPE 'SKETCHTYPE]
                                                                     (* unknown sketch type and this is the first place where such is
                                                                     encountered.)
              (ERROR ELEMENTTYPE "Unknown sketch type.
                     If you can load the file containing it, do so and type 'RETURN'.
                     Otherwise, type '^'.")
              (GO LP)))
           (RETURN (fetch (SKETCHTYPE INSIDEFN) of SKTYPE])
```

(SK.UPDATEFN

```
  [LAMBDA (ELEMENTTYPE)                                              (* rrb "21-Dec-84 11:28")

          (* goes from an element type name to its updatefn The update function is called when an element in a window has changed.
          It will get args of the old local screen element, the new global element and the window.
          If it can update the display more efficiently than erasing and redrawing, it should and return the new local sketch element.)

    (fetch (SKETCHTYPE UPDATEFN) of (GETPROP ELEMENTTYPE 'SKETCHTYPE])

)

(/DECLAREDATATYPE 'SKETCHTYPE
```

```
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '30)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SCREENELT (LOCALPART . GLOBALPART)
       (RECORD GLOBALPART (COMMONGLOBALPART INDIVIDUALGLOBALPART)
              (RECORD INDIVIDUALGLOBALPART (GTYPE . GOTHERINFO))
              (RECORD COMMONGLOBALPART (MINSCALE MAXSCALE SKELEMENTPROPLIST)))
       (RECORD LOCALPART (HOTSPOTS LOCALHOTREGION . OTHERLOCALINFO)))

(RECORD GLOBALPART (COMMONGLOBALPART INDIVIDUALGLOBALPART)
       (RECORD INDIVIDUALGLOBALPART (GTYPE . RESTOFGLOBALPART))
       (RECORD COMMONGLOBALPART (MINSCALE MAXSCALE SKELEMENTPROPLIST)))

(RECORD COMMONGLOBALPART (MINSCALE MAXSCALE SKELEMENTPROPLIST))

(RECORD INDIVIDUALGLOBALPART (GTYPE . RESTOFGLOBALPART))

(RECORD LOCALPART (HOTSPOTS LOCALHOTREGION . OTHERLOCALINFO))

[RECORD SKETCH (ALLSKETCHPROPS . SKETCHTCELL)
       [RECORD ALLSKETCHPROPS (SKETCHKEY SKETCHNAME . SKETCHPROPS)
              (CREATE (LIST 'SKETCH NIL 'VERSION SKETCH.VERSION 'PRIRANGE (CONS 0 0]
       [RECORD SKETCHTCELL (SKETCHELTS)
              (CREATE (CONS SKETCHELTS (LAST SKETCHELTS]
       (TYPE? (AND (LISTP DATUM)
                   (LISTP (CAR DATUM))
                   (EQ (CAAR DATUM)
                       'SKETCH]

(DATATYPE SKETCHTYPE (LABEL                                    (* the label if it is non-NIL will be used in the sketch menu.)
                            DOCSTR                             (* if put in the menu, this is the help string for its item.)
                            DRAWFN EXPANDFN obsolete CHANGEFN INPUTFN INSIDEFN REGIONFN TRANSLATEFN UPDATEFN
                            READCHANGEFN TRANSFORMFN           (* fn to transform the control points of an element.
                                                               takes args Gelt Tranfn trandata.)
                            TRANSLATEPTSFN

       (* fn to move some but not all points of a screen element. Takes args%: LocalSelectedPts GlobalDeltaToTranslate
       ScreenElt SketchWindow)

                            GLOBALREGIONFN

       (* takes a GLOBAL element and returns the global region it occupies.
       Note%: this is the only fn that takes a global rather that a local element.)

                            ))

(RECORD SKETCHCONTEXT (SKETCHBRUSH SKETCHFONT SKETCHTEXTALIGNMENT SKETCHARROWHEAD SKETCHDASHING
                             SKETCHUSEARROWHEAD SKETCHTEXTBOXALIGNMENT SKETCHFILLING SKETCHLINEMODE
                             SKETCHCHARCDIRECTION SKETCHMOVEMODE SKETCHINPUTSCALE SKETCHDRAWINGMODE))
)

(/DECLAREDATATYPE 'SKETCHTYPE
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '30)
)

(ADDTOVAR BackgroundMenuCommands
          (Sketch '(SKETCHW.CREATE NIL NIL (GETREGION)
                          NIL NIL T T)
                 "Opens a sketch window for use."
                 (SUBITEMS ("Page sized sketch" '(EDITSLIDE NIL)
                                 "Opens a sketch window the size of a page.")
                        ("Landscaped sketch" '(EDITSLIDE NIL T)
                               "Opens a sketch window the size of a landscaped page.")
                        ("Sketch, from a file" '(SKETCH.FROM.A.FILE)
                               "Reads a file name and opens a sketch window onto the sketch it contains."))))

(RPAQQ BackgroundMenu NIL)

(FILESLOAD SKETCH-OPS SKETCH-ELEMENTS SKETCH-EDIT SKETCH-OBJ SKETCH-BMELT)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       SKETCH-OPS SKETCH-ELEMENTS SKETCH-OBJ SKETCH-EDIT)
)
```

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (FROM LOADUPS)
       EXPORTS.ALL)
)
```

;; recompute the sketch element types because loading SKETCH clobbers the previous ones.

```
(INIT.BITMAP.ELEMENT)

(INIT.SKETCH.ELEMENTS)
```

(**INIT.GROUP.ELEMENT**)

;; version checking stuff

```
(DECLARE%: EVAL@COMPILE

(RPAQQ SKETCH.VERSION 3)

(CONSTANTS (SKETCH.VERSION 3))
)

(DEFINEQ
```

(SK.CHECK.SKETCH.VERSION

```
  [LAMBDA (SKETCH)                                          ; Edited 21-Oct-92 18:40 by sybalsky:mv:envos
    ;; makes sure the sketch is the correct version.  If not, it tries to update it.  Returns SKETCH.
    (COND
       ((EQ (LISTGET (fetch (SKETCH SKETCHPROPS) of SKETCH)
                  'VERSION)
            SKETCH.VERSION)
        SKETCH)
       (T (SK.INSURE.RECORD.LENGTH (fetch (SKETCH SKETCHELTS) of SKETCH))
          ;; this is basically a PUTSKETCHPROP expanded in line to avoid coersions which can cause loops.
          [PROG (PLIST)
                (SETQ PLIST (fetch (SKETCH SKETCHPROPS) of SKETCH))
                (COND
                   ((SETQ PLIST (fetch (SKETCH SKETCHPROPS) of SKETCH))
                    (LISTPUT PLIST 'VERSION SKETCH.VERSION))
                   (T (replace (SKETCH SKETCHPROPS) of SKETCH with (LIST 'VERSION SKETCH.VERSION]
          SKETCH])
```

(SK.INSURE.RECORD.LENGTH

```
  [LAMBDA (SKETCHELTS)                                      ; Edited 21-Oct-92 18:35 by sybalsky:mv:envos
    ;; makes sure the elements have the proper number of fields.
    (bind INDPART TYPE NFIELDS for ELT in SKETCHELTS
       do (SETQ INDPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELT))
          (SETQ TYPE (fetch (INDIVIDUALGLOBALPART GTYPE) of INDPART))
          (COND
             ([OR (SETQ NFIELDS (SK.RECORD.LENGTH TYPE))
                  (AND (RECLOOK TYPE)
                       (SETQ SKETCH.RECORD.LENGTHS (NCONC1 SKETCH.RECORD.LENGTHS
                                                           (LIST TYPE (SETQ NFIELDS
                                                                            (LENGTH (EVAL (LIST 'CREATE TYPE]
              (SK.INSURE.HAS.LENGTH INDPART NFIELDS TYPE)))
          ;; if it's not a record, either it's an unknown sketch element type or its declaration wasn't copied to the compiled file.  In either case, assume
          ;; it has the correct number of fields.
          (COND
             ((EQ TYPE 'GROUP)                                  ; recurse thru the subelements too.
              (SK.INSURE.RECORD.LENGTH (fetch (GROUP LISTOFGLOBALELTS) of INDPART])
```

(SK.INSURE.HAS.LENGTH

```
  [LAMBDA (LIST N TYPE)                                     ; Edited 21-Oct-92 18:36 by sybalsky:mv:envos
    ;; makes sure LIST is at least N long.  If not, it creates a record of type TYPE and nconcs the enough fields from the end to make it be N long.
    (OR (EQLENGTH LIST N)
        (NCONC LIST (COND
                       [(RECLOOK TYPE)
                        (NTH (EVAL (LIST 'CREATE TYPE))
                             (ADD1 (LENGTH LIST]
                       (T                                   ; no record, add NILs and hope.
                          (for I from (ADD1 (LENGTH LIST)) to N collect NIL])
```

(SK.RECORD.LENGTH

```
  [LAMBDA (SKETCHRECORDTYPE)                                (* rrb "20-Mar-86 14:11")
    (CADR (ASSOC SKETCHRECORDTYPE SKETCH.RECORD.LENGTHS])
```

```
(SK.SET.RECORD.LENGTHS
  [LAMBDA NIL                                                    (* rrb "18-Oct-85 15:35")
                                                                 (* sets up a variable that contains the lengths of the sketch
                                                                 element records.)
    (SETQ SKETCH.RECORD.LENGTHS (SK.SET.RECORD.LENGTHS.MACRO])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS SK.SET.RECORD.LENGTHS.MACRO MACRO [ARGS (CONS 'LIST (for X in SKETCH.ELEMENT.TYPE.NAMES
                                                                  collect (LIST 'LIST (KWOTE X)
                                                                                (LIST 'LENGTH
                                                                                      (LIST 'CREATE X])
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCH.RECORD.LENGTHS)
)

(SK.SET.RECORD.LENGTHS)

;; to correct for a bug in the file package that marks LOADCOMPed file as changed

(UNMARKASCHANGED 'SKETCH 'FILE)

(UNMARKASCHANGED 'SKETCH-ELEMENTS 'FILE)

(UNMARKASCHANGED 'SKETCH-OPS 'FILE)

(UNMARKASCHANGED 'SKETCH-EDIT 'FILE)

(UNMARKASCHANGED 'SKETCH-OBJ 'FILE)

;; add sketch as option to file browser edit command

(DEFINEQ

(SK.ADD.EDIT.COMMAND.TO.FILE.BROWSER
  [LAMBDA NIL                                                    ; Edited 12-Feb-88 16:49 by rrb
                                                                 (* adds sketch as an option to the file browser edit command.)
    (AND (BOUNDP 'FB.MENU.ITEMS)
         (PROG [(PTRX (for MITEM in FB.MENU.ITEMS when (STRING-EQUAL (CAR MITEM)
                                                                     "Edit")
                         do (RETURN MITEM]
               (SETQ PTRX (ASSOC 'SUBITEMS PTRX))
               (for SUBI in PTRX when (STRING-EQUAL (CAR SUBI)
                                                    "Sketch")
                  do (RETURN) finally (NCONC1 PTRX (LIST '"Sketch" '(FB.EDITCOMMAND SKETCH)
                                                         "Calls the Sketch editor on selected files"])

)

(SK.ADD.EDIT.COMMAND.TO.FILE.BROWSER)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA SK.UNIONREGIONS SKETCH.CREATE)
)
```

FUNCTION INDEX

ADD.ELEMENT.TO.SKETCH 33
ADD.SKETCH.TO.VIEWER 8
ADD.SKETCH.VIEWER 33
ALL.SKETCH.VIEWERS 33
CHANGEABLEFIELDITEMS 30
CREATE.DEFAULT.SKETCH.CONTEXT 24
CREATE.SKETCHW.COMMANDMENU 20
DELFROMGROUPELT 64
DELFROMTCONC 38
DISPLAY.POSITION.READER.TOTAL 89
EDITSKETCH 6
EDITSLIDE 6
ELT.INSIDE.REGION? 33
ELT.INSIDE.SKETCHWP 66
ELT.INSIDE.SKWP 34
FILENAMELESSVERSION 9
GET.BITMAP.POSITION 85
GETSKETCHPROP 22
GETSKETCHWREGION 10
GETWREGION 85
GLOBAL.KNOT.FROM.LOCAL 31
GROUP.DRAWFN 58
GROUP.EXPANDFN 58
GROUP.GLOBALREGIONFN 58
GROUP.INSIDEFN 58
GROUP.READCHANGEFN 59
GROUP.REGIONFN 58
GROUP.TRANSFORMFN 59
GROUP.TRANSLATEFN 58
INIT.GROUP.ELEMENT 57
INSPECT.SKETCH 66
INSURE.SKETCH 65
LIGHTGRAYWINDOW 15
LOCALSPECS.FROM.VIEWER 65
MAKE.LOCAL.SKETCH 13
MAP.GLOBAL.POSITION.INTO.VIEWER 81
MAP.SKETCH.ELEMENTS.INTO.VIEWER 81
MAP.SKETCHSPEC.INTO.VIEWER 13
MAP.VIEWER.PT.INTO.GLOBAL 82
MAP.VIEWER.XY.INTO.GLOBAL 81
MAPCOLLECTSKETCHSPECS 66
MAPGLOBALSKETCHELEMENTS 67
MAPGLOBALSKETCHSPECS 66
MAPSKETCHSPECS 66
MAPSKETCHSPECSUNTIL 66
NEAREST.HOT.SPOT 84
POSITION.PAD.READER.HANDLER 90
POSITIONPAD.HELDFN 91
PUTSKETCHPROP 23
READ.AND.SAVE.NEW.DASHING 78
READ.AND.SAVE.NEW.FILLING 79
READ.DASHING.CHANGE 78
READ.FILLING.CHANGE 79
READ.FUNCTION 76
READ.NEW.DASHING 78
READ.POINT.TO.ADD 31
READANGLE 77
READARCDIRECTION 77
READBRUSHSHAPE 76
READBRUSHSIZE 77
READMOVEMODE 52
REGION.CENTER 60
REMOVE.ELEMENT.FROM.SKETCH 36
REMOVE.LAST 60
REMOVE.SKETCH.VIEWER 33
SCALE.FROM.SKW 34
SCREENELEMENTP 68
SK.ADD.COPY.OF.ELEMENTS 39
SK.ADD.EDIT.COMMAND.TO.FILE.BROWSER 96
SK.ADD.ELEMENT 10
SK.ADD.ELEMENTS 11
SK.ADD.ELEMENTS.TO.SKETCH 9
SK.ADD.ITEM 35
SK.ADD.ITEM.TO.MENU 22
SK.ADD.KNOT.TO.ELEMENT 32
SK.ADD.POINT 51
SK.ADD.PRIORITY.ELEMENT.TO.SKETCH 10
SK.ADD.PRIORITY.LOCAL.ELEMENT.TO.SKETCH 11
SK.ADD.PT.SELECTION 50
SK.ADD.SELECTION 67
SK.ADD.SPACES 16
SK.ADDELT.TO.WINDOW 34
SK.ALIGN.POINTS 52
SK.ALIGN.POINTS.BOTTOM 52
SK.ALIGN.POINTS.LEFT 52
SK.ALIGN.POINTS.RIGHT 52
SK.ALIGN.POINTS.TOP 52
SK.APPLY.CHANGE.COMMAND 30
SK.APPLY.CHANGE.COMMAND1 31
SK.APPLY.DEFAULT.MOVE 41
SK.APPLY.IMAGEOBJ.WHENDELETEDFN 16
SK.APPLY.MENU.COMMAND 11
SK.APPLY.SINGLE.CHANGEFN 29
SK.BRING.UP.POSITION.PAD 86
SK.BUILD.IMAGEOBJ 26
SK.BUTTONEVENT.MARK 26
SK.BUTTONEVENT.OVERP 27
SK.BUTTONEVENT.SAME.KEYS 27
SK.CACHE.DASHING 78
SK.CACHE.FILLING 79
SK.CALC.REGION.VIEWED 34
SK.CHANGE.DASHING 77
SK.CHANGE.ELT 28
SK.CHANGE.THING 28
SK.CHANGEFN 29
SK.CHECK.END.INITIAL.EDIT 37
SK.CHECK.IMAGEOBJ.WHENDELETEDFN 16
SK.CHECK.PRECHANGEFN 28
SK.CHECK.PREEDITFN 37
SK.CHECK.SKETCH.VERSION 95
SK.CHECK.WHENADDEDFN 11
SK.CHECK.WHENCHANGEDFN 28
SK.CHECK.WHENDELETEDFN 37
SK.CHECK.WHENGROUPEDFN 61
SK.CHECK.WHENPOINTDELETEDFN 38
SK.CHECK.WHENUNGROUPEDFN 62
SK.CLEAR.POPUP.MENU 22
SK.CONFIRM.DESTRUCTION 13
SK.CONTROL.POINT.NUMBER 54
SK.CONTROL.POINTS.IN.REGION 50
SK.COPY.BUTTONEVENTFN 24
SK.COPY.ELEMENTS 39
SK.COPY.ELT 38
SK.COPY.GLOBAL.ELEMENT 44
SK.COPY.INSERTFN 67
SK.COPY.ITEM 40
SK.CORRESPONDING.CONTROL.PT 54
SK.CREATE.GROUP1 55
SK.CREATE.STANDARD.MENU 22
SK.DASHING.LABEL 79
SK.DEFAULT.CHANGEFN 30
SK.DELETE.ELEMENT 36
SK.DELETE.ELEMENT.KNOT 37
SK.DELETE.ELEMENT1 11
SK.DELETE.ELEMENT2 36
SK.DELETE.ELT 38
SK.DELETE.ITEM 38
SK.DELETE.KNOT 36
SK.DO.ALIGN.POINTS 52
SK.DO.ALIGN.SETVALUE 54
SK.DO.AND.RECORD.CHANGES 30
SK.DO.CHANGESPEC1 29
SK.DO.CHANGESPECS 29
SK.DO.FREEZE 63
SK.DO.GROUP 61
SK.DO.MOVE.ELEMENT.POINTS 47
SK.DO.UNFREEZE 63
SK.DO.UNGROUP 61
SK.DRAWFIGURE 34
SK.DRAWFIGURE.IF 14
SK.DRAWFIGURE1 34
SK.DRAWFN 93
SK.ELEMENT.CHANGED1 64
SK.ELEMENT.GLOBAL.REGION 68
SK.ELEMENTS.CHANGEFN 31
SK.ELTS.BY.PRIORITY 10
SK.ELTS.CONTAINING.PTS 51
SK.ERASE.AND.DELETE.ITEM 36
SK.ERASE.ELT 38
SK.EVAL.AS.PROCESS 20
SK.EVAL.WITH.LOCK 21
SK.EVEN.SPACE.POINTS.IN.X 52
SK.EVEN.SPACE.POINTS.IN.Y 52
SK.EXPANDFN 93

{MEDLEY}<library>sketch>SKETCH.;1

SK.FILLING.LABEL 79
SK.FIX.MENU 21
SK.FLASHREGION 57
SK.FREEZE.ELEMENTS 62
SK.FREEZE.ELTS 62
SK.FREEZE.UNDO 63
SK.GET.FROM.FILE 7
SK.GET.IMAGEOBJ.FROM.FILE 8
SK.GET.SELECTED.ELEMENT.STRUCTURE 54
SK.GET.VIEWER.POPUP.MENU 22
SK.GETGLOBALPOSITION 80
SK.GLOBAL.FROM.LOCAL.ELEMENTS 40
SK.GLOBAL.REGION.OF.GLOBAL.ELEMENTS 57
SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS 56
SK.GLOBAL.REGIONFN 69
SK.GROUP.CHANGEFN 32
SK.GROUP.CHANGEFN1 32
SK.GROUP.ELEMENTS 56
SK.GROUP.ELTS 55
SK.GROUP.UNDO 62
SK.HOTSPOTS.NOT.ON.LIST 51
SK.INCLUDE.FILE 7
SK.INIT.POSITION.NUMBER.PAD.MENU 88
SK.INPUT 93
SK.INPUT.SCALE 72
SK.INSERT.SKETCH 40
SK.INSIDE.REGION 66
SK.INSIDEFN 93
SK.INSURE.HAS.LENGTH 95
SK.INSURE.HAS.MENU 21
SK.INSURE.RECORD.LENGTH 95
SK.INSURE.SCALE 82
SK.ITEM.REGION 68
SK.LOCAL.ELT.FROM.GLOBALPART 66
SK.LOCAL.FROM.GLOBAL 34
SK.LOCAL.ITEMS.IN.REGION 68
SK.LOCAL.REGION.OF.LOCAL.ELEMENTS 56
SK.MAKE.ELEMENT.MOVE.ARG 44
SK.MAKE.ELEMENTS.MOVE.ARG 45
SK.MAKE.POINTS.AND.ELEMENTS.MOVE.ARG 45
SK.MARK.DIRTY 12
SK.MARK.UNDIRTY 12
SK.MENU.AND.RETURN.FIELD 12
SK.MOVE.ELEMENT.POINT 47
SK.MOVE.ELEMENTS 41
SK.MOVE.ELT 41
SK.MOVE.ELT.OR.PT 41
SK.MOVE.GROUP.CONTROL.PT 60
SK.MOVE.GROUP.ELEMENT.CONTROL.POINT 60
SK.MOVE.ITEM.POINTS 48
SK.MOVE.POINTS 47
SK.MOVE.THING 45
SK.NTH.CONTROL.POINT 54
SK.ORDER.ELEMENTS 11
SK.OUTPUT.FILE.NAME 6
SK.PAD.READER.POSITION 87
SK.PICKOUT.WHOLE.MOVE.ELEMENTS 80
SK.POPUP.SELECTIONFN 10
SK.POSITION.PAD.FROM.VIEWER 88
SK.POSITION.READER.REPAINTFN 88
SK.PUT.MARKS.UP 71
SK.PUT.ON.FILE 6
SK.READ.NEW.GROUP.CONTROL.PT 61
SK.READ.POINT.WITH.FEEDBACK 82
SK.READ.POSITION.PAD.HANDLER 88
SK.READCHANGEFN 30
SK.RECORD.LENGTH 95
SK.REGIONFN 68
SK.REMOVE.PT.SELECTION 51
SK.REMOVE.SELECTION 69
SK.RETURN.TTY 16
SK.SEL.AND.ALIGN.POINTS 52
SK.SEL.AND.CHANGE 28
SK.SEL.AND.COPY 39
SK.SEL.AND.DELETE 36
SK.SEL.AND.DELETE.KNOT 36
SK.SEL.AND.FREEZE 62
SK.SEL.AND.GROUP 55
SK.SEL.AND.MOVE 41
SK.SEL.AND.MOVE.CONTROL.PT 60
SK.SEL.AND.MOVE.POINTS 47
SK.SEL.AND.UNFREEZE 62
SK.SEL.AND.UNGROUP 56
SK.SELECT.MULTIPLE.ITEMS 69
SK.SELECT.MULTIPLE.POINTS 49
SK.SET.FEEDBACK.ALWAYS 73
SK.SET.FEEDBACK.MODE 73
SK.SET.FEEDBACK.POINT 73
SK.SET.FEEDBACK.VERBOSE 73
SK.SET.INPUT.SCALE 73
SK.SET.INPUT.SCALE.CURRENT 73
SK.SET.INPUT.SCALE.VALUE 73
SK.SET.MOVE.MODE 51
SK.SET.MOVE.MODE.COMBINED 51
SK.SET.MOVE.MODE.ELEMENTS 51
SK.SET.MOVE.MODE.POINTS 51
SK.SET.POSITION 81
SK.SET.RECORD.LENGTHS 96
SK.SET.UP.MENUS 21
SK.SHOW.FIG.FROM.INFO 45
SK.SHRINK.ICONCREATE 74
SK.SKETCH.MENU 16
SK.TAKE.MARKS.DOWN 71
SK.TAKE.TTY 16
SK.TRACK.BITMAP1 85
SK.TRACK.IMAGE1 81
SK.TRANSFORMFN 93
SK.TRANSLATE.ELEMENT 44
SK.TRANSLATE.GLOBALPART 71
SK.TRANSLATE.ITEM 72
SK.TRANSLATE.POINTS 49
SK.TRANSLATEFN 72
SK.TRANSLATEPTSFN 49
SK.UNFREEZE.ELEMENTS 63
SK.UNFREEZE.ELT 62
SK.UNFREEZE.UNDO 63
SK.UNGROUP.ELEMENT 56
SK.UNGROUP.ELT 56
SK.UNGROUP.UNDO 62
SK.UNIONREGIONS 57
SK.UPDATE.ELEMENT 46
SK.UPDATE.ELEMENT1 46
SK.UPDATE.ELEMENTS 46
SK.UPDATE.EVENT.SELECTION 15
SK.UPDATE.GLOBAL.IMAGE.OBJECT.ELEMENT 65
SK.UPDATE.GROUP.AFTER.CHANGE 55
SK.UPDATE.REGION.VIEWED 35
SK.UPDATE.SKETCHCONTEXT 73
SK.UPDATEFN 93
SK.VIEWER.FROM.SKETCH.ARG 29
SK.WINDOW.TITLE 6
SKETCH 4
SKETCH.ADD.AND.DISPLAY 35
SKETCH.ADD.AND.DISPLAY1 35
SKETCH.ADD.ELEMENT 63
SKETCH.ALL.VIEWERS 33
SKETCH.CHANGE.ELEMENTS 28
SKETCH.COMMANDMENU 16
SKETCH.COMMANDMENU.ITEMS 16
SKETCH.COPY.ELEMENTS 44
SKETCH.CREATE 22
SKETCH.CREATE.GROUP 55
SKETCH.DELETE.ELEMENT 64
SKETCH.ELEMENT.CHANGED 64
SKETCH.ELEMENT.TYPE 64
SKETCH.ELEMENTS.OF.SKETCH 63
SKETCH.FROM.A.FILE 4
SKETCH.FROM.VIEWER 66
SKETCH.GET 8
SKETCH.GET.ELEMENTS 71
SKETCH.GET.POSITION 84
SKETCH.LIST.OF.ELEMENTS 63
SKETCH.MONITORLOCK 20
SKETCH.MOVE.ELEMENTS 43
SKETCH.PUT 7
SKETCH.REGION.OF.SKETCH 57
SKETCH.REGION.VIEWED 34
SKETCH.RESET 6
SKETCH.SET.A.DEFAULT 9
SKETCH.SET.BRUSH.SHAPE 12
SKETCH.SET.BRUSH.SIZE 12
SKETCH.TITLE 74
SKETCH.TO.VIEWER.POSITION 81
SKETCH.TO.VIEWER.REGION 82
SKETCH.TRACK.ELEMENTS 80
SKETCH.TRACK.IMAGE 81
SKETCH.VIEW.FROM.NAME 35
SKETCHW.ADD.INSTANCE 35
SKETCHW.CLOSEFN 12
SKETCHW.CREATE 4
SKETCHW.FIG.CHANGED 6
SKETCHW.OUTFN 13

SKETCHW.REOPENFN 13
SKETCHW.REPAINTFN 13
SKETCHW.REPAINTFN1 14
SKETCHW.RESHAPEFN 15
SKETCHW.SCROLLFN 14
SKETCHW.SELECTIONFN 20
TRANSLATE.SKETCH 72
UPDATE.ELEMENT.IN.SKETCH 45
VIEWER.BUCKET 33
VIEWER.TO.SKETCH.POSITION 82
VIEWER.TO.SKETCH.REGION 82
\CLOBBER.POSITION 84
\POSITION.PAD.ADD.DIGIT.MENU 91
\POSITION.READER.NUMBERPAD 92
\SKETCH.COPY.ELEMENT 44

RECORD INDEX

COMMONGLOBALPART 94
GLOBALPART 94
GROUP 61
INDIVIDUALGLOBALPART ...94
INPUTPT 86
LOCALGROUP 61
LOCALPART 94
SCREENELT 94
SKETCH 94
SKETCHCONTEXT 94
SKETCHTYPE 94
SKFIGUREIMAGE 72
SKHISTORYCHANGESPEC 33

VARIABLE INDEX

ALL.SKETCHES 92
BackgroundMenu 94
BackgroundMenuCommands 94
DEFAULT.VISIBLE.SCALE.FACTOR 92
INITIAL.SCALE 92
MINIMUM.VISIBLE.SCALE.FACTOR 92
SK.DASHING.PATTERNS 80
SK.FILLING.PATTERNS 80
SKETCH.ELEMENT.TYPE.NAMES 92
SKETCH.ELEMENT.TYPES 92
SKETCH.USE.POSITION.PAD 86
SKETCH.VERBOSE.FEEDBACK 73

MACRO INDEX

.DELETEKEYDOWNP. 28
.MOVEKEYDOWNP. 28
.SHIFTKEYDOWNP. 51
SK.SET.RECORD.LENGTHS.MACRO 96

PROPERTY INDEX

FREEZE 63
GROUP 62
SKETCH.CREATE 24
UNFREEZE 63
UNGROUP 62

CONSTANT INDEX

SK.NO.MOVE.DISTANCE 72
SKETCH.VERSION 95