```
File created:      5-Dec-2023 00:16:52  {WMEDLEY}<library>sketch>SKETCH-ELEMENTS.;1

     edit by:     rmk

  changes to:     (RECORDS LOCALCIRCLE CIRCLE LOCALELLIPSE ELLIPSE KNOTELT LOCALCURVE OPENCURVE CLOSEDCURVE
                         LOCALCLOSEDCURVE LOCALCLOSEDWIRE LOCALWIRE WIRE CLOSEDWIRE TEXT LOCALTEXT LOCALTEXTBOX
                         TEXTBOX BOX LOCALBOX ARC LOCALARC)

previous date:     4-Aug-2022 09:57:43  {WMEDLEY}<library>sketch>SKETCHELEMENTS.;2

  Read Table:     INTERLISP

     Package:     INTERLISP

      Format:      XCCS


(RPAQQ SKETCH-ELEMENTSCOMS
       (                                                                  ; contains the functions need to implement the sketch basic
                                                                          ; element types
        (FNS INIT.SKETCH.ELEMENTS CREATE.SKETCH.ELEMENT.TYPE SKETCH.ELEMENT.TYPEP SKETCH.ELEMENT.NAMEP
             \CURSOR.IN.MIDDLE.MENU)
        (COMS                                                             ; color and filling stuff
              (FNS SKETCHINCOLORP READ.COLOR.CHANGE)
              (INITVARS (SKETCHINCOLORFLG)
                     (FILLPOLYGONFLG T)
                     (FILLINGMODEFLG T))
              (INITVARS (SK.DEFAULT.BACKCOLOR)
                     (SK.DEFAULT.OPERATION))
              (GLOBALVARS SKETCHINCOLORFLG SK.DEFAULT.BACKCOLOR)
              (RECORDS SKFILLING)

              ;; fns included until system is fixed so that it is ok to call DSPCOLOR in a system without color loaded.  Should be removed after J
              ;; release.

              (FNS SK.CREATE.DEFAULT.FILLING SKFILLINGP SK.INSURE.FILLING SK.INSURE.COLOR)
              (FNS SK.TRANSLATE.MODE SK.CHANGE.FILLING.MODE READ.FILLING.MODE))
        (COMS (FNS SKETCH.CREATE.CIRCLE CIRCLE.EXPANDFN CIRCLE.DRAWFN \CIRCLE.DRAWFN1 CIRCLE.INPUTFN
                   SK.UPDATE.CIRCLE.AFTER.CHANGE SK.READ.CIRCLE.POINT SK.SHOW.CIRCLE CIRCLE.INSIDEFN
                   CIRCLE.REGIONFN CIRCLE.GLOBALREGIONFN CIRCLE.TRANSLATE CIRCLE.READCHANGEFN CIRCLE.TRANSFORMFN
                   CIRCLE.TRANSLATEPTS SK.CIRCLE.CREATE SET.CIRCLE.SCALE SK.BRUSH.READCHANGE)
              (FNS SK.INSURE.BRUSH SK.INSURE.DASHING)
              (RECORDS BRUSH)
              (DECLARE%: DONTCOPY (RECORDS LOCALCIRCLE CIRCLE))
              (UGLYVARS CIRCLEICON)
              (CURSORS CIRCLE.CENTER CIRCLE.EDGE)
              (INITVARS [SK.DEFAULT.BRUSH (CONS 'ROUND (CONS 1 (CONS 'BLACK NIL]
                                                                          ; Original was (create BRUSH BRUSHSHAPE _ 'ROUND
                                                                          ; BRUSHSIZE _ 1 BRUSHCOLOR _ 'BLACK).
                                                                          ; Changed by yabu.fx, for SUNLOADUP without DWIM.
                     (SK.DEFAULT.DASHING)
                     (SK.DEFAULT.TEXTURE))
              (GLOBALVARS SK.DEFAULT.BRUSH SK.DEFAULT.DASHING SK.DEFAULT.TEXTURE))
        (COMS (FNS SKETCH.CREATE.ELLIPSE ELLIPSE.EXPANDFN ELLIPSE.DRAWFN ELLIPSE.INPUTFN
                   SK.READ.ELLIPSE.MAJOR.PT SK.SHOW.ELLIPSE.MAJOR.RADIUS SK.READ.ELLIPSE.MINOR.PT
                   SK.SHOW.ELLIPSE.MINOR.RADIUS ELLIPSE.INSIDEFN ELLIPSE.CREATE SK.UPDATE.ELLIPSE.AFTER.CHANGE
                   ELLIPSE.REGIONFN ELLIPSE.GLOBALREGIONFN ELLIPSE.TRANSLATEFN ELLIPSE.TRANSFORMFN
                   ELLIPSE.TRANSLATEPTS MARK.SPOT DISTANCEBETWEEN SK.DISTANCE.TO SQUARE
                   COMPUTE.ELLIPSE.ORIENTATION SK.COMPUTE.ELLIPSE.MINOR.RADIUS.PT)
              (DECLARE%: DONTCOPY (RECORDS LOCALELLIPSE ELLIPSE))
              (UGLYVARS ELLIPSEICON)
              (CURSORS ELLIPSE.CENTER ELLIPSE.SEMI.MAJOR ELLIPSE.SEMI.MINOR))
        (COMS (FNS SKETCH.CREATE.OPEN.CURVE OPENCURVE.INPUTFN SK.CURVE.CREATE MAXXEXTENT MAXYEXTENT
                   KNOT.SET.SCALE.FIELD OPENCURVE.DRAWFN OPENCURVE.EXPANDFN OPENCURVE.READCHANGEFN
                   OPENCURVE.TRANSFORMFN OPENCURVE.TRANSLATEFN OPENCURVE.TRANSLATEPTSFN
                   SKETCH.CREATE.CLOSED.CURVE CLOSEDCURVE.DRAWFN CLOSEDCURVE.EXPANDFN CLOSEDCURVE.REGIONFN
                   CLOSEDCURVE.GLOBALREGIONFN READ.LIST.OF.POINTS CLOSEDCURVE.INPUTFN CLOSEDCURVE.READCHANGEFN
                   CLOSEDCURVE.TRANSFORMFN CLOSEDCURVE.TRANSLATEPTSFN INVISIBLEPARTP SHOWSKETCHPOINT
                   SHOWSKETCHXY KNOTS.REGIONFN OPENWIRE.GLOBALREGIONFN CURVE.REGIONFN OPENCURVE.GLOBALREGIONFN
                   KNOTS.TRANSLATEFN REGION.CONTAINING.PTS)
              (FNS CHANGE.ELTS.BRUSH.SIZE CHANGE.ELTS.BRUSH CHANGE.ELTS.BRUSH.SHAPE SK.CHANGE.BRUSH.SHAPE
                   SK.CHANGE.BRUSH.COLOR SK.CHANGE.BRUSH.SIZE SK.CHANGE.ANGLE SK.CHANGE.ARC.DIRECTION
                   SK.SET.DEFAULT.BRUSH.SIZE READSIZECHANGE)
              (FNS SK.CHANGE.ELEMENT.KNOTS)
              (FNS SK.INSURE.POINT.LIST SK.INSURE.POSITION)
              (DECLARE%: DONTCOPY (RECORDS KNOTELT LOCALCURVE OPENCURVE CLOSEDCURVE LOCALCLOSEDCURVE
                                         LOCALCLOSEDWIRE))
              (UGLYVARS OPENCURVEICON CLOSEDCURVEICON)
              (CURSORS CURVE.KNOT))
        (COMS (FNS SKETCH.CREATE.WIRE CLOSEDWIRE.EXPANDFN KNOTS.INSIDEFN OPEN.WIRE.DRAWFN WIRE.EXPANDFN
                   SK.UPDATE.WIRE.ELT.AFTER.CHANGE OPENWIRE.READCHANGEFN OPENWIRE.TRANSFORMFN
                   OPENWIRE.TRANSLATEFN OPENWIRE.TRANSLATEPTSFN WIRE.INPUTFN SK.READ.WIRE.POINTS
                   SK.READ.POINTS.WITH.FEEDBACK OPENWIRE.FEEDBACKFN CLOSEDWIRE.FEEDBACKFN CLOSEDWIRE.REGIONFN
                   CLOSEDWIRE.GLOBALREGIONFN SK.WIRE.CREATE WIRE.ADD.POINT.TO.END READ.ARROW.CHANGE
                   CHANGE.ELTS.ARROWHEADS)
              (FNS SKETCH.CREATE.CLOSED.WIRE CLOSED.WIRE.INPUTFN CLOSED.WIRE.DRAWFN CLOSEDWIRE.READCHANGEFN
                   CLOSEDWIRE.TRANSFORMFN CLOSEDWIRE.TRANSLATEPTSFN)
              (FNS SK.EXPAND.ARROWHEADS SK.COMPUTE.ARC.ARROWHEAD.POINTS ARC.ARROWHEAD.POINTS
                   SET.ARC.ARROWHEAD.POINTS SET.OPENCURVE.ARROWHEAD.POINTS SK.COMPUTE.CURVE.ARROWHEAD.POINTS
                   SET.WIRE.ARROWHEAD.POINTS SK.COMPUTE.WIRE.ARROWHEAD.POINTS SK.EXPAND.ARROWHEAD CHANGED.ARROW
```

```
          SK.CHANGE.ARROWHEAD SK.CHANGE.ARROWHEAD1 SK.CREATE.ARROWHEAD SK.ARROWHEAD.CREATE
          SK.ARROWHEAD.END.TEST READ.ARROWHEAD.END ARROW.HEAD.POSITIONS ARROWHEAD.POINTS.LIST
          CURVE.ARROWHEAD.POINTS LEFT.MOST.IS.BEGINP WIRE.ARROWHEAD.POINTS DRAWARROWHEADS
          \SK.DRAW.TRIANGLE.ARROWHEAD \SK.ENDPT.OF.ARROW \SK.ADJUST.FOR.ARROWHEADS
          SK.SET.ARROWHEAD.LENGTH SK.SET.ARROWHEAD.ANGLE SK.SET.ARROWHEAD.TYPE SK.SET.LINE.ARROWHEAD
          SK.UPDATE.ARROWHEAD.FORMAT SK.SET.LINE.LENGTH.MODE)
     (FNS SK.INSURE.ARROWHEADS SK.ARROWHEADP)
     (DECLARE%: DONTCOPY (RECORDS LOCALWIRE WIRE CLOSEDWIRE LOCALCLOSEDWIRE))
     (RECORDS ARROWHEAD)
     (UGLYVARS VSHAPE.ARROWHEAD.BITMAP TRIANGLE.ARROWHEAD.BITMAP SOLIDTRIANGLE.ARROWHEAD.BITMAP
            CURVEDV.ARROWHEAD.BITMAP)
     (UGLYVARS WIREICON CLOSEDWIREICON)
     (INITVARS (SK.ARROWHEAD.ANGLE.INCREMENT 5)
            (SK.ARROWHEAD.LENGTH.INCREMENT 2))
     (ADDVARS (SK.ARROWHEAD.TYPES LINE CLOSEDLINE CURVE SOLID))
     (INITVARS (SK.DEFAULT.ARROW.LENGTH 8)
            (SK.DEFAULT.ARROW.TYPE 'CURVE)
            (SK.DEFAULT.ARROW.ANGLE 18.0))
     (GLOBALVARS SK.DEFAULT.ARROW.LENGTH SK.DEFAULT.ARROW.TYPE SK.DEFAULT.ARROW.ANGLE
            SK.ARROWHEAD.TYPES)
     (INITVARS (SK.ARROW.END.MENU)
            (SK.ARROW.EDIT.MENU)))
(COMS                                                           ; stuff to support the text element type.
     (FNS SKETCH.CREATE.TEXT TEXT.CHANGEFN TEXT.READCHANGEFN \SK.READ.FONT.SIZE1
          SK.TEXT.ELT.WITH.SAME.FIELDS SK.READFONTFAMILY CLOSE.PROMPT.WINDOW TEXT.DRAWFN TEXT.DRAWFN1
          TEXT.INSIDEFN TEXT.EXPANDFN SK.TEXT.LINE.REGIONS TEXT.UPDATE.GLOBAL.REGIONS REL.MOVE.REGION
          LTEXT.LINE.REGIONS TEXT.INPUTFN READ.TEXT TEXT.POSITION.AND.CREATE CREATE.TEXT.ELEMENT
          SK.UPDATE.TEXT.AFTER.CHANGE SK.TEXT.FROM.TEXTBOX TEXT.SET.GLOBAL.REGIONS TEXT.REGIONFN
          TEXT.GLOBALREGIONFN TEXT.TRANSLATEFN TEXT.TRANSFORMFN TEXT.TRANSLATEPTSFN TEXT.UPDATEFN
          SK.CHANGE.TEXT TEXT.SET.SCALES BREAK.AT.CARRIAGE.RETURNS)
     (FNS ADD.KNOWN.SKETCH.FONT SK.PICK.FONT SK.CHOOSE.TEXT.FONT SK.NEXTSIZEFONT
          SK.DECREASING.FONT.LIST SK.GUESS.FONTSAVAILABLE)
     (INITVARS (\KNOWN.SKETCH.FONTSIZES))
     (GLOBALVARS \KNOWN.SKETCH.FONTSIZES)
     (DECLARE%: DONTCOPY (RECORDS TEXT LOCALTEXT))
     (FNS SK.SET.FONT SK.SET.TEXT.FONT SK.SET.TEXT.SIZE SK.SET.TEXT.HORIZ.ALIGN SK.READFONTSIZE
          SK.COLLECT.FONT.SIZES SK.SET.TEXT.VERT.ALIGN SK.SET.TEXT.LOOKS SK.SET.DEFAULT.TEXT.FACE)
     (FNS CREATE.SKETCH.TERMTABLE)
     (FNS SK.FONT.LIST SK.INSURE.FONT SK.INSURE.STYLE SK.INSURE.TEXT)
     (VARS INDICATE.TEXT.SHADE)
     [INITVARS (SK.DEFAULT.FONT)
            (SK.DEFAULT.TEXT.ALIGNMENT '(CENTER BASELINE]
     (INITVARS \FONTSONFILE)
     (ADDVARS (SK.HORIZONTAL.STYLES LEFT RIGHT CENTER)
            (SK.VERTICAL.STYLES TOP CENTER BASELINE BOTTOM))
     (VARS (SKETCH.TERMTABLE (CREATE.SKETCH.TERMTABLE)))
     (GLOBALVARS SKETCH.TERMTABLE SK.DEFAULT.TEXT.ALIGNMENT INDICATE.TEXT.SHADE \FONTSONFILE
            SK.HORIZONTAL.STYLES SK.VERTICAL.STYLES))
(COMS                                                           ; stuff for supporting the TEXTBOX sketch element.
     (FNS SKETCH.CREATE.TEXTBOX SK.COMPUTE.TEXTBOX.REGION.FOR.STRING SK.BREAK.INTO.LINES SK.BRUSH.SIZE
          SK.TEXTBOX.CREATE SK.TEXTBOX.CREATE1 SK.UPDATE.TEXTBOX.AFTER.CHANGE
          SK.TEXTBOX.POSITION.IN.BOX TEXTBOX.CHANGEFN TEXTBOX.DRAWFN SK.TEXTURE.AROUND.REGIONS
          ALL.EMPTY.REGIONS TEXTBOX.EXPANDFN TEXTBOX.INPUTFN TEXTBOX.INSIDEFN TEXTBOX.REGIONFN
          TEXTBOX.GLOBALREGIONFN TEXTBOX.SET.GLOBAL.REGIONS TEXTBOX.TRANSLATEFN TEXTBOX.TRANSLATEPTSFN
          TEXTBOX.TRANSFORMFN TEXTBOX.UPDATEFN TEXTBOX.READCHANGEFN SK.TEXTBOX.TEXT.POSITION
          SK.TEXTBOX.FROM.TEXT ADD.EOLS)
     (DECLARE%: DONTCOPY (RECORDS LOCALTEXTBOX TEXTBOX))
     (COMS                                                      ; stuff to handle default alignment for text boxes
           (FNS SK.SET.TEXTBOX.VERT.ALIGN SK.SET.TEXTBOX.HORIZ.ALIGN)
           (VARS TEXTBOXICON)
           [INITVARS (SK.DEFAULT.TEXTBOX.ALIGNMENT '(CENTER CENTER]
           (GLOBALVARS SK.DEFAULT.TEXTBOX.ALIGNMENT)))
(COMS                                                           ; functions to implement the box sketch element.
     (FNS SKETCH.CREATE.BOX SK.BOX.DRAWFN BOX.DRAWFN1 KNOTS.OF.REGION SK.DRAWAREABOX SK.DRAWBOX
          SK.BOX.EXPANDFN SK.BOX.GETREGIONFN BOX.SET.SCALES SK.BOX.INPUTFN SK.BOX.CREATE
          SK.UPDATE.BOX.AFTER.CHANGE SK.BOX.INSIDEFN SK.BOX.REGIONFN SK.BOX.GLOBALREGIONFN
          SK.BOX.READCHANGEFN SK.CHANGE.FILLING SK.CHANGE.FILLING.COLOR SK.BOX.TRANSLATEFN
          SK.BOX.TRANSFORMFN SK.BOX.TRANSLATEPTSFN UNSCALE.REGION.TO.GRID INCREASEREGION
          INSUREREGIONSIZE EXPANDREGION REGION.FROM.COORDINATES)
     (DECLARE%: DONTCOPY (RECORDS BOX LOCALBOX))
     (UGLYVARS BOXICON))
(COMS                                                           ; fns for the arc sketch element type
     (FNS SKETCH.CREATE.ARC ARC.DRAWFN ARC.EXPANDFN ARC.INPUTFN SK.INVERT.CIRCLE
          SK.READ.ARC.ANGLE.POINT SK.SHOW.ARC ARC.CREATE SK.UPDATE.ARC.AFTER.CHANGE ARC.MOVEFN
          ARC.TRANSLATEPTS ARC.INSIDEFN ARC.REGIONFN ARC.GLOBALREGIONFN ARC.TRANSLATE ARC.TRANSFORMFN
          ARC.READCHANGEFN)
     (FNS SK.COMPUTE.ARC.ANGLE.PT SK.COMPUTE.ARC.ANGLE.PT.FROM.ANGLE SK.COMPUTE.ARC.PTS
          SK.SET.ARC.DIRECTION SK.SET.ARC.DIRECTION.CW SK.SET.ARC.DIRECTION.CCW
          SK.COMPUTE.SLOPE.OF.LINE SK.CREATE.ARC.USING SET.ARC.SCALES)
     (FNS SK.INSURE.DIRECTION)
     (INITVARS (SK.NUMBER.OF.POINTS.IN.ARC 8))
     (GLOBALVARS SK.NUMBER.OF.POINTS.IN.ARC)
     (DECLARE%: DONTCOPY (RECORDS ARC LOCALARC))
     (CURSORS ARC.RADIUS.CURSOR ARC.ANGLE.CURSOR CW.ARC.ANGLE.CURSOR CW.ARC.RADIUS.CURSOR)
     (UGLYVARS ARCICON))
(COMS                                                           ; property getting and setting stuff
     (FNS GETSKETCHELEMENTPROP \SK.GET.ARC.ANGLEPT \GETSKETCHELEMENTPROP1 \SK.GET.BRUSH \SK.GET.FILLING
                  \SK.GET.ARROWHEADS \SK.GET.FONT \SK.GET.JUSTIFICATION \SK.GET.DIRECTION \SK.GET.DASHING
                  PUTSKETCHELEMENTPROP \SK.PUT.FILLING ADDSKETCHELEMENTPROP REMOVESKETCHELEMENTPROP
                  \SK.PUT.FONT \SK.PUT.JUSTIFICATION \SK.PUT.DIRECTION \SK.PUT.DASHING \SK.PUT.BRUSH
                  \SK.PUT.ARROWHEADS SK.COPY.ELEMENT.PROPERTY.LIST SKETCH.UPDATE SKETCH.UPDATE1
                  \SKELT.GET.SCALE \SKELT.PUT.SCALE \SKELT.PUT.DATA SK.REPLACE.TEXT.IN.ELEMENT \SKELT.GET.DATA
                  \SK.GET.1STCONTROLPT \SK.PUT.1STCONTROLPT \SK.GET.2NDCONTROLPT \SK.PUT.2NDCONTROLPT
                  \SK.GET.3RDCONTROLPT \SK.PUT.3RDCONTROLPT)
              (FNS LOWERLEFTCORNER UPPERRIGHTCORNER))))
```

;; contains the functions need to implement the sketch basic element types

```
(DEFINEQ

(INIT.SKETCH.ELEMENTS
  [LAMBDA NIL                                              ; Edited 23-Jul-90 15:38 by matsuda
                                                           (* sets up the initial sketch element types.)

           (* put the datatype for the element on the property list of the name and use the name in the instances.)

    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'CIRCLE))
        (CREATE.SKETCH.ELEMENT.TYPE 'CIRCLE CIRCLEICON "Adds a circle to the figure." (FUNCTION CIRCLE.DRAWFN)
               (FUNCTION CIRCLE.EXPANDFN)
               'OBSOLETE
               (FUNCTION SK.ELEMENTS.CHANGEFN)
               (FUNCTION CIRCLE.INPUTFN)
               (FUNCTION CIRCLE.INSIDEFN)
               (FUNCTION CIRCLE.REGIONFN)
               (FUNCTION CIRCLE.TRANSLATE)
               NIL
               (FUNCTION CIRCLE.READCHANGEFN)
               (FUNCTION CIRCLE.TRANSFORMFN)
               (FUNCTION CIRCLE.TRANSLATEPTS)
               (FUNCTION CIRCLE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'ELLIPSE))
        (CREATE.SKETCH.ELEMENT.TYPE 'ELLIPSE ELLIPSEICON "Adds an ellipse to the figure."
               (FUNCTION ELLIPSE.DRAWFN)
               (FUNCTION ELLIPSE.EXPANDFN)
               'OBSOLETE
               (FUNCTION SK.ELEMENTS.CHANGEFN)
               (FUNCTION ELLIPSE.INPUTFN)
               (FUNCTION ELLIPSE.INSIDEFN)
               (FUNCTION ELLIPSE.REGIONFN)
               (FUNCTION ELLIPSE.TRANSLATEFN)
               NIL
               (FUNCTION SK.BRUSH.READCHANGE)
               (FUNCTION ELLIPSE.TRANSFORMFN)
               (FUNCTION ELLIPSE.TRANSLATEPTS)
               (FUNCTION ELLIPSE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'ARC))
        (CREATE.SKETCH.ELEMENT.TYPE 'ARC ARCICON "Adds an arc to the figure." (FUNCTION ARC.DRAWFN)
               (FUNCTION ARC.EXPANDFN)
               'OBSOLETE
               (FUNCTION SK.ELEMENTS.CHANGEFN)
               (FUNCTION ARC.INPUTFN)
               (FUNCTION ARC.INSIDEFN)
               (FUNCTION ARC.REGIONFN)
               (FUNCTION ARC.TRANSLATE)
               NIL
               (FUNCTION ARC.READCHANGEFN)
               (FUNCTION ARC.TRANSFORMFN)
               (FUNCTION ARC.TRANSLATEPTS)
               (FUNCTION ARC.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'OPENCURVE))
        (CREATE.SKETCH.ELEMENT.TYPE 'OPENCURVE OPENCURVEICON "Adds a curve by accepting points the curve goes
               through." (FUNCTION OPENCURVE.DRAWFN)
               (FUNCTION OPENCURVE.EXPANDFN)
               'OBSOLETE
               (FUNCTION SK.ELEMENTS.CHANGEFN)
               (FUNCTION OPENCURVE.INPUTFN)
               (FUNCTION KNOTS.INSIDEFN)
               (FUNCTION CURVE.REGIONFN)
               (FUNCTION OPENCURVE.TRANSLATEFN)
               NIL
               (FUNCTION OPENCURVE.READCHANGEFN)
               (FUNCTION OPENCURVE.TRANSFORMFN)
               (FUNCTION OPENCURVE.TRANSLATEPTSFN)
               (FUNCTION OPENCURVE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'CLOSEDCURVE))
        (CREATE.SKETCH.ELEMENT.TYPE 'CLOSEDCURVE CLOSEDCURVEICON "Adds a closed curve by accepting points that
               it goes though." (FUNCTION CLOSEDCURVE.DRAWFN)
               (FUNCTION CLOSEDCURVE.EXPANDFN)
               'OBSOLETE
```

```
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION CLOSEDCURVE.INPUTFN)
              (FUNCTION KNOTS.INSIDEFN)
              (FUNCTION CLOSEDCURVE.REGIONFN)
              (FUNCTION KNOTS.TRANSLATEFN)
              NIL
              (FUNCTION CLOSEDCURVE.READCHANGEFN)
              (FUNCTION CLOSEDCURVE.TRANSFORMFN)
              (FUNCTION CLOSEDCURVE.TRANSLATEPTSFN)
              (FUNCTION CLOSEDCURVE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'WIRE))
        (CREATE.SKETCH.ELEMENT.TYPE 'WIRE WIREICON "Adds a series of lines by accepting points."
              (FUNCTION OPEN.WIRE.DRAWFN)
              (FUNCTION WIRE.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION WIRE.INPUTFN)
              (FUNCTION KNOTS.INSIDEFN)
              (FUNCTION KNOTS.REGIONFN)
              (FUNCTION OPENWIRE.TRANSLATEFN)
              NIL
              (FUNCTION OPENCURVE.READCHANGEFN)
              (FUNCTION OPENWIRE.TRANSFORMFN)
              (FUNCTION OPENWIRE.TRANSLATEPTSFN)
              (FUNCTION OPENWIRE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'CLOSEDWIRE))
        (CREATE.SKETCH.ELEMENT.TYPE 'CLOSEDWIRE CLOSEDWIREICON "Adds a closed polygon by accepting the
              corners." (FUNCTION CLOSED.WIRE.DRAWFN)
              (FUNCTION CLOSEDWIRE.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION CLOSED.WIRE.INPUTFN)
              (FUNCTION KNOTS.INSIDEFN)
              (FUNCTION CLOSEDWIRE.REGIONFN)
              (FUNCTION KNOTS.TRANSLATEFN)
              NIL
              (FUNCTION CLOSEDWIRE.READCHANGEFN)
              (FUNCTION CLOSEDWIRE.TRANSFORMFN)
              (FUNCTION CLOSEDWIRE.TRANSLATEPTSFN)
              (FUNCTION CLOSEDWIRE.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'TEXT))
        (CREATE.SKETCH.ELEMENT.TYPE 'TEXT NIL "text is added by pointing to its position and typing."
              (FUNCTION TEXT.DRAWFN)
              (FUNCTION TEXT.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION TEXT.INPUTFN)
              (FUNCTION TEXT.INSIDEFN)
              (FUNCTION TEXT.REGIONFN)
              (FUNCTION TEXT.TRANSLATEFN)
              (FUNCTION TEXT.UPDATEFN)
              (FUNCTION TEXT.READCHANGEFN)
              (FUNCTION TEXT.TRANSFORMFN)
              (FUNCTION TEXT.TRANSLATEPTSFN)
              (FUNCTION TEXT.GLOBALREGIONFN]
    [COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'BOX))
        (CREATE.SKETCH.ELEMENT.TYPE 'BOX BOXICON "Adds a box by accepting two corners." (FUNCTION SK.BOX.DRAWFN
                                                                                               )
              (FUNCTION SK.BOX.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION SK.BOX.INPUTFN)
              (FUNCTION SK.BOX.INSIDEFN)
              (FUNCTION SK.BOX.REGIONFN)
              (FUNCTION SK.BOX.TRANSLATEFN)
              NIL
              (FUNCTION SK.BOX.READCHANGEFN)
              (FUNCTION SK.BOX.TRANSFORMFN)
              (FUNCTION SK.BOX.TRANSLATEPTSFN)
              (FUNCTION SK.BOX.GLOBALREGIONFN]
    (COND
       ((NOT (SKETCH.ELEMENT.TYPEP 'TEXTBOX))
        (CREATE.SKETCH.ELEMENT.TYPE 'TEXTBOX TEXTBOXICON "Adds a box into which text can be typed."
              (FUNCTION TEXTBOX.DRAWFN)
              (FUNCTION TEXTBOX.EXPANDFN)
              'OBSOLETE
              (FUNCTION SK.ELEMENTS.CHANGEFN)
              (FUNCTION TEXTBOX.INPUTFN)
              (FUNCTION TEXTBOX.INSIDEFN)
              (FUNCTION TEXTBOX.REGIONFN)
              (FUNCTION TEXTBOX.TRANSLATEFN)
              (FUNCTION TEXTBOX.UPDATEFN)
              (FUNCTION TEXTBOX.READCHANGEFN)
```

```
                (FUNCTION TEXTBOX.TRANSFORMFN)
                (FUNCTION TEXTBOX.TRANSLATEPTSFN)
                (FUNCTION TEXTBOX.GLOBALREGIONFN])
```

(**CREATE.SKETCH.ELEMENT.TYPE**

```
  [LAMBDA (SKETCHTYPE LABEL DOCSTR DRAWFN EXPANDFN OBSOLETE CHANGEFN INPUTFN INSIDEFN REGIONFN TRANSLATEFN
                 UPDATEFN READCHANGEFN TRANSFORMFN TRANSLATEPTSFN GLOBALREGIONFN)
                                                                   (* rrb "18-Oct-85 17:18")
                                                                   (* creates a new sketch element type.)
    (COND
       ((AND OBSOLETE (NEQ OBSOLETE 'OBSOLETE))
        (printout T OBSOLETE " will never be called. CREATE.SKETCH.ELEMENT.TYPE")))
    (SETQ SKETCH.ELEMENT.TYPES
     (CONS (PUTPROP SKETCHTYPE 'SKETCHTYPE
                  (create SKETCHTYPE
                         LABEL _ LABEL
                         DOCSTR _ DOCSTR
                         DRAWFN _ DRAWFN
                         EXPANDFN _ EXPANDFN
                         CHANGEFN _ CHANGEFN
                         INPUTFN _ INPUTFN
                         INSIDEFN _ INSIDEFN
                         REGIONFN _ REGIONFN
                         TRANSLATEFN _ TRANSLATEFN
                         UPDATEFN _ UPDATEFN
                         READCHANGEFN _ READCHANGEFN
                         TRANSFORMFN _ TRANSFORMFN
                         TRANSLATEPTSFN _ TRANSLATEPTSFN
                         GLOBALREGIONFN _ GLOBALREGIONFN))
          SKETCH.ELEMENT.TYPES))
    (OR (MEMB SKETCHTYPE SKETCH.ELEMENT.TYPE.NAMES)
        (SETQ SKETCH.ELEMENT.TYPE.NAMES (CONS SKETCHTYPE SKETCH.ELEMENT.TYPE.NAMES)))
    SKETCHTYPE])
```

(**SKETCH.ELEMENT.TYPEP**

```
  [LAMBDA (SKETCHTYPE)                                             (* rrb "28-Dec-84 15:39")
                                                                   (* is SKETCHTYPE a sketch element type?)
    (AND (MEMB SKETCHTYPE SKETCH.ELEMENT.TYPE.NAMES)
         (GETPROP SKETCHTYPE 'SKETCHTYPE])
```

(**SKETCH.ELEMENT.NAMEP**

```
  [LAMBDA (X)                                                      (* rrb "18-MAR-83 11:53")
                                                                   (* is X a sketch element type name?)
    (FMEMB X SKETCH.ELEMENT.TYPE.NAMES])
```

(\CURSOR.IN.MIDDLE.MENU

```
  [LAMBDA (MENU)                                                   (* rrb " 6-Nov-85 09:46")
                                                                   (* brings up the menu so that the cursor is in the middle.)
    (MENU MENU (create POSITION
                      XCOORD _ (DIFFERENCE LASTMOUSEX (QUOTIENT (fetch (MENU IMAGEWIDTH) of MENU)
                                                                2))
                      YCOORD _ (DIFFERENCE LASTMOUSEY (QUOTIENT (fetch (MENU IMAGEHEIGHT) of MENU)
                                                                2])
)
```

;; color and filling stuff

```
(DEFINEQ
```

(**SKETCHINCOLORP**

```
  [LAMBDA NIL                                                      (* rrb "12-Jul-85 10:11")
                                                                   (* hook to determine if sketch should allow color.)
    SKETCHINCOLORFLG])
```

(**READ.COLOR.CHANGE**

```
  [LAMBDA (MSG ALLOWNONEFLG CURRENTCOLOR)                          (* rrb "29-Oct-85 12:30")
                                                                   (* reads a color from the user and returns it)
    (READCOLOR1 MSG ALLOWNONEFLG CURRENTCOLOR])
)
```

(RPAQ? **SKETCHINCOLORFLG**)

(RPAQ? **FILLPOLYGONFLG** T)

(RPAQ? **FILLINGMODEFLG** T)

(RPAQ? **SK.DEFAULT.BACKCOLOR**)

(RPAQ? **SK.DEFAULT.OPERATION**)

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCHINCOLORFLG SK.DEFAULT.BACKCOLOR)
)

(DECLARE%: EVAL@COMPILE

(RECORD SKFILLING (FILLING.TEXTURE FILLING.COLOR FILLING.OPERATION))
)
```

;; fns included until system is fixed so that it is ok to call DSPCOLOR in a system without color loaded. Should be removed after J release.

```
(DEFINEQ

(SK.CREATE.DEFAULT.FILLING
  [LAMBDA NIL                                                   (* rrb "21-Feb-86 11:22")
    (create SKFILLING
           FILLING.TEXTURE _ SK.DEFAULT.TEXTURE
           FILLING.COLOR _ SK.DEFAULT.BACKCOLOR
           FILLING.OPERATION _ SK.DEFAULT.OPERATION])

(SKFILLINGP
  [LAMBDA (FILLING)                                             (* rrb "21-Feb-86 11:20")
                                                                (* determines if FILLING is a legal sketch filling.)
    (COND
       ((AND (LISTP FILLING)
             (TEXTUREP (fetch (SKFILLING FILLING.TEXTURE) of FILLING))
             (NULL (CDDDR FILLING)))                            (* should also check if (fetch (SKFILLING FILLING.COLOR)) is
                                                                a color and that (SKFILLING FILLING.OPERATION) is an
                                                                operation.)
        FILLING])

(SK.INSURE.FILLING
  [LAMBDA (FILLING SKW)                                         (* rrb "16-Oct-85 15:47")
                                                                (* converts several possible legal filling specifications into a
                                                                sketch filling)
    (COND
       ((SKFILLINGP FILLING))
       (T (PROG [(DEFAULTFILLING (COND
                                    [(WINDOWP SKW)
                                     (fetch (SKETCHCONTEXT SKETCHFILLING) of (WINDOWPROP SKW 'SKETCHCONTEXT]
                                    (T (SK.CREATE.DEFAULT.FILLING]
                 (RETURN (COND
                            ((NULL FILLING)
                             DEFAULTFILLING)
                            ((TEXTUREP FILLING)
                             (create SKFILLING using DEFAULTFILLING FILLING.TEXTURE _ FILLING))
                            ((\POSSIBLECOLOR FILLING)

          (* note that small numbers can be either a texture or a color. This algorithm will make them be a texture.)

                             (create SKFILLING using DEFAULTFILLING FILLING.COLOR _ FILLING))
                            (T                                  (* should be a check here for a color too.)
                               (\ILLEGAL.ARG FILLING])

(SK.INSURE.COLOR
  [LAMBDA (COLOR)                                               (* rrb "16-Oct-85 18:05")
                                                                (* checks the validity of a color argument.)
    (COND
       ((NULL COLOR)
        NIL)
       ((\POSSIBLECOLOR COLOR))
       (T (\ILLEGAL.ARG COLOR])
)

(DEFINEQ

(SK.TRANSLATE.MODE
  [LAMBDA (OPERATION STREAM)                                    (* rrb "10-Mar-86 17:20")
                                                                (* picks the best operation for a filling.)
    (COND
       ((EQ (DSPOPERATION NIL STREAM)
            'ERASE)                                             (* drawing should do its best job of erasing the current image)
        (SELECTQ OPERATION
             (INVERT 'INVERT)
             (ERASE

          (* don't know what to do because we don't know what bits were removed but this at least lets the user know something
          happened wrt this element.)

                    'PAINT)
             'ERASE))
       (T OPERATION])
```

```
(SK.CHANGE.FILLING.MODE
  [LAMBDA (ELTWITHFILLING HOW SKW)                                    (* rrb " 3-Mar-86 14:36")
                                                                      (* changes the texture in the element ELTWITHFILLING.)
    (PROG (GFILLEDELT MODE FILLING NEWFILLING TYPE NEWELT)
          (RETURN (COND
                     ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHFILLING))
                            '(BOX TEXTBOX CLOSEDWIRE CIRCLE))     (* only works for things that have a filling, for now just boxes and
                      polygons)
                      (SETQ GFILLEDELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHFILLING))
                      [SETQ MODE (fetch (SKFILLING FILLING.OPERATION) of (SETQ FILLING
                                                                          (SELECTQ TYPE
                                                                              (BOX (fetch (BOX BOXFILLING)
                                                                                      of GFILLEDELT))
                                                                              (TEXTBOX (fetch (TEXTBOX
                                                                                               TEXTBOXFILLING
                                                                                               )
                                                                                          of GFILLEDELT))
                                                                              (CIRCLE (fetch (CIRCLE CIRCLEFILLING
                                                                                              )
                                                                                         of GFILLEDELT))
                                                                              (CLOSEDWIRE (fetch (CLOSEDWIRE
                                                                                                  CLOSEDWIREFILLING
                                                                                                  )
                                                                                             of GFILLEDELT))
                                                                              (SHOULDNT]
                      (COND
                         ((NOT (EQUAL HOW MODE))                      (* new filling mode)
                          (SETQ NEWFILLING (create SKFILLING using FILLING FILLING.OPERATION _ HOW))
                          (SETQ NEWELT (SELECTQ TYPE
                                           (BOX (create BOX using GFILLEDELT BOXFILLING _ NEWFILLING))
                                           (TEXTBOX (create TEXTBOX using GFILLEDELT TEXTBOXFILLING _ NEWFILLING)
)
                                           (CLOSEDWIRE (create CLOSEDWIRE using GFILLEDELT CLOSEDWIREFILLING _
                                                                                  NEWFILLING))
                                           (CIRCLE (create CIRCLE using GFILLEDELT CIRCLEFILLING _ NEWFILLING))
                                           (SHOULDNT)))
                          (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHFILLING)
                                                 INDIVIDUALGLOBALPART _ NEWELT)
                                 OLDELT _ ELTWITHFILLING
                                 PROPERTY _ 'FILLING
                                 NEWVALUE _ NEWFILLING
                                 OLDVALUE _ FILLING])

(READ.FILLING.MODE
  [LAMBDA NIL                                                         (* rrb " 3-Mar-86 14:30")
                                                                      (* reads a filling mode from the user.)
    (\CURSOR.IN.MIDDLE.MENU (create MENU
                                   CENTERFLG _ T
                                   TITLE _ "How should the filling merge with the covered figures?"
                                   MENUROWS _ 1
                                   ITEMS _ '((REPLACE 'REPLACE "the filling completely covers anything under
                                                      it.")
                                             (PAINT 'PAINT "the black parts of the filling cover but the white
                                                    parts show through.")
                                             (ERASE 'ERASE "the black parts of the filling are erased.")
                                             (INVERT 'INVERT "the black parts of the filling are inverted."])

)

(DEFINEQ

(SKETCH.CREATE.CIRCLE
  [LAMBDA (CENTERPT RADIUSPT BRUSH DASHING FILLING SCALE)             (* rrb "11-Dec-85 10:43")
                                                                      (* creates a sketch circle element.)
    (SK.CIRCLE.CREATE (SK.INSURE.POSITION CENTERPT)
           (COND
              [(NUMBERP RADIUSPT)
               (create POSITION
                      XCOORD _ (PLUS (fetch (POSITION XCOORD) of CENTERPT)
                                     RADIUSPT)
                      YCOORD _ (PLUS (fetch (POSITION YCOORD) of CENTERPT]
              (T (SK.INSURE.POSITION RADIUSPT)))
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.FILLING FILLING])

(CIRCLE.EXPANDFN
  [LAMBDA (GCIRCLE SCALE)                                             (* rrb " 7-Dec-85 20:45")
```

```
            (* returns a screen elt that has a circle screen element calculated from the global part.)

    (PROG (CENTER RADIUSPT BRUSH (INDGCIRCLE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCIRCLE)))

            (* check to make sure there is an initial scale field. This change was introduced on Apr 27 and can be taken out the release
            after Jazz It can also be taken out of the other expand fns as well.)

          [COND
             ((fetch (CIRCLE CIRCLEINITSCALE) of INDGCIRCLE))
             (T                                                        (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GCIRCLE
                   with (SETQ INDGCIRCLE (create CIRCLE using INDGCIRCLE CIRCLEINITSCALE _ 1.0]
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALCIRCLE
                                            CENTERPOSITION _ (SETQ CENTER (SK.SCALE.POSITION.INTO.VIEWER
                                                                          (fetch (CIRCLE CENTERLATLON)
                                                                             of INDGCIRCLE)
                                                                          SCALE))
                                            RADIUSPOSITION _ (SETQ RADIUSPT (SK.SCALE.POSITION.INTO.VIEWER
                                                                            (fetch (CIRCLE RADIUSLATLON)
                                                                               of INDGCIRCLE)
                                                                            SCALE))
                                            RADIUS _ (DISTANCEBETWEEN CENTER RADIUSPT)
                                            LOCALCIRCLEBRUSH _
                                            (SCALE.BRUSH (COND
                                                            ([NOT (NUMBERP (SETQ BRUSH (fetch (CIRCLE BRUSH)
                                                                                          of INDGCIRCLE]
                                                                       (* new format, old format had brush width only.)
                                                             BRUSH)
                                                            (T [replace (CIRCLE BRUSH) of INDGCIRCLE
                                                                  with (SETQ BRUSH
                                                                        (create BRUSH
                                                                               BRUSHSIZE _ BRUSH
                                                                               BRUSHSHAPE _ 'ROUND]
                                                               BRUSH))
                                                   (fetch (CIRCLE CIRCLEINITSCALE) of INDGCIRCLE)
                                                   SCALE)
                                            LOCALCIRCLEFILLING _ (APPEND (fetch (CIRCLE CIRCLEFILLING)
                                                                           of INDGCIRCLE))
                                            LOCALCIRCLEDASHING _ (fetch (CIRCLE DASHING) of INDGCIRCLE))
                         GLOBALPART _ GCIRCLE])
```

(CIRCLE.DRAWFN

```
  [LAMBDA (CIRCLEELT WINDOW REGION)                            (* rrb "20-Jun-86 17:08")
                                                               (* draws a circle from a circle element.)
    (PROG ((GCIRCLE (fetch (SCREENELT INDIVIDUALGLOBALPART) of CIRCLEELT))
           (LCIRCLE (fetch (SCREENELT LOCALPART) of CIRCLEELT))
           CPOS DASHING FILLING)
          (SETQ CPOS (fetch (LOCALCIRCLE CENTERPOSITION) of LCIRCLE))
          (SETQ DASHING (fetch (LOCALCIRCLE LOCALCIRCLEDASHING) of LCIRCLE))
          (SETQ FILLING (fetch (LOCALCIRCLE LOCALCIRCLEFILLING) of LCIRCLE))
          (COND
             ((fetch (SKFILLING FILLING.COLOR) of FILLING)

          (* if the circle is filled with a color call FILLCIRCLE with both the texture and the color.
          This allows iris to do its thing before textures and colors are merged.)

              (DSPOPERATION (PROG1 (DSPOPERATION (fetch (SKFILLING FILLING.OPERATION) of FILLING)
                                          WINDOW)
                                   (FILLCIRCLE (fetch (POSITION XCOORD) of CPOS)
                                          (fetch (POSITION YCOORD) of CPOS)
                                          (fetch (LOCALCIRCLE RADIUS) of LCIRCLE)
                                          FILLING WINDOW))
                     WINDOW))
             ((fetch (SKFILLING FILLING.TEXTURE) of FILLING)          (* if the circle is filled with texture, call FILLCIRCLE.)
              (DSPOPERATION (PROG1 (DSPOPERATION (fetch (SKFILLING FILLING.OPERATION) of FILLING)
                                          WINDOW)
                                   (FILLCIRCLE (fetch (POSITION XCOORD) of CPOS)
                                          (fetch (POSITION YCOORD) of CPOS)
                                          (fetch (LOCALCIRCLE RADIUS) of LCIRCLE)
                                          (COND
                                             ((EQ (DSPOPERATION NIL WINDOW)
                                                  'ERASE)              (* use black in case the window moved because of texture to
                                                                       window alignment bug.)
                                              BLACKSHADE)
                                             (T (fetch (SKFILLING FILLING.TEXTURE) of FILLING)))
                                          WINDOW))
                     WINDOW)))
          (RETURN (\CIRCLE.DRAWFN1 CPOS (fetch (LOCALCIRCLE RADIUSPOSITION) of LCIRCLE)
                         (fetch (LOCALCIRCLE RADIUS) of LCIRCLE)
                         (fetch (LOCALCIRCLE LOCALCIRCLEBRUSH) of LCIRCLE)
                         DASHING WINDOW])
```

(\CIRCLE.DRAWFN1

```
  [LAMBDA (CENTERPT RADIUSPT RADIUS BRUSH DASHING WINDOW)           ; Edited 17-Apr-90 17:24 by matsuda
                                                                    (* draws a circle for sketch from some information.
                                                                    Calls by CIRCLE.DRAWFN and ARC.DRAWFN)
    (COND
       (DASHING                                                     (* draw it with the arc drawing code which does dashing.)
              (DRAWCURVE (SK.COMPUTE.ARC.PTS CENTERPT RADIUSPT
                                [COND
                                   [(LESSP (FETCH (POSITION XCOORD) OF CENTERPT)
                                           (FETCH (POSITION XCOORD) OF RADIUSPT))
                                    (PTPLUS RADIUSPT (CONSTANT (create POSITION
                                                                       XCOORD _ 0
                                                                       YCOORD _ -1]
                                   [(GREATERP (FETCH (POSITION XCOORD) OF CENTERPT)
                                           (FETCH (POSITION XCOORD) OF RADIUSPT))
                                    (PTPLUS RADIUSPT (CONSTANT (create POSITION
                                                                       XCOORD _ 0
                                                                       YCOORD _ 1]
                                   [(LESSP (FETCH (POSITION YCOORD) OF CENTERPT)
                                           (FETCH (POSITION YCOORD) OF RADIUSPT))
                                    (PTPLUS RADIUSPT (CONSTANT (create POSITION
                                                                       XCOORD _ 1
                                                                       YCOORD _ 0]
                                   (T (PTPLUS RADIUSPT (CONSTANT (create POSITION
                                                                         XCOORD _ -1
                                                                         YCOORD _ 0]
                                NIL)
                     T BRUSH DASHING WINDOW))
       (T (DRAWCIRCLE (fetch (POSITION XCOORD) of CENTERPT)
                 (fetch (POSITION YCOORD) of CENTERPT)
                 RADIUS BRUSH DASHING WINDOW])
```

(**CIRCLE.INPUTFN**

```
  [LAMBDA (WINDOW)                                                  (* rrb "20-May-86 10:49")

            (* reads a two points from the user and returns a circle element that it represents.)

    (PROG [CENTERPT RADIUSPT (SKETCHCONTEXT (WINDOWPROP WINDOW 'SKETCHCONTEXT]
          (STATUSPRINT WINDOW "
                 " "Indicate center of circle")
          (COND
             ((NOT (SETQ CENTERPT (SK.READ.POINT.WITH.FEEDBACK WINDOW CIRCLE.CENTER NIL NIL NIL NIL
                                         SKETCH.USE.POSITION.PAD)))
              (CLOSEPROMPTWINDOW WINDOW)
              (RETURN NIL)))
          (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTERPT)
                 NIL WINDOW)
          (STATUSPRINT WINDOW "
                 " "Indicate a point of the circumference of the circle")
          (SETQ RADIUSPT (SK.READ.CIRCLE.POINT WINDOW (fetch (INPUTPT INPUT.POSITION) of CENTERPT)
                                CIRCLE.EDGE))                       (* erase center mark)
          (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTERPT)
                 NIL WINDOW)
          (CLOSEPROMPTWINDOW WINDOW)
          (OR RADIUSPT (RETURN NIL))
          (SETQ CENTERPT (SK.MAP.INPUT.PT.TO.GLOBAL CENTERPT WINDOW))
          (SETQ RADIUSPT (SK.MAP.INPUT.PT.TO.GLOBAL RADIUSPT WINDOW))
          (RETURN (SK.CIRCLE.CREATE CENTERPT RADIUSPT (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKETCHCONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHDASHING) of SKETCHCONTEXT)
                         (SK.INPUT.SCALE WINDOW)
                         (fetch (SKETCHCONTEXT SKETCHFILLING) of SKETCHCONTEXT])
```

(**SK.UPDATE.CIRCLE.AFTER.CHANGE**

```
  [LAMBDA (GCIRELT)                                                 (* rrb " 7-Dec-85 19:50")
                                                                    (* updates the dependent fields of a circle element when a field
    changes.)
    (replace (CIRCLE CIRCLEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCIRELT) with NIL])
```

(**SK.READ.CIRCLE.POINT**

```
  [LAMBDA (WINDOW CENTERPT CURSOR)                                  (* rrb "20-May-86 10:46")

            (* reads a point from the user prompting them with a circle that follows the cursor)

    (SK.READ.POINT.WITH.FEEDBACK WINDOW CURSOR (AND SKETCH.VERBOSE.FEEDBACK (FUNCTION SK.SHOW.CIRCLE))
           CENTERPT
           'MIDDLE NIL SKETCH.USE.POSITION.PAD])
```

(**SK.SHOW.CIRCLE**

```
  [LAMBDA (X Y WINDOW CENTERPT)                                     (* rrb "15-Nov-85 14:18")

            (* xors a circle to X Y from CENTERPT in a window. Used as the feedback function for reading the radius point for circles.)
                                                                    (* Mark the point too.)
    (SHOWSKETCHXY X Y WINDOW)
    (PROG ((CENTERX (fetch (POSITION XCOORD) of CENTERPT))
           (CENTERY (fetch (POSITION YCOORD) of CENTERPT)))
          (DRAWCIRCLE CENTERX CENTERY (SK.DISTANCE.TO CENTERX CENTERY X Y)
                 1 NIL WINDOW])

(CIRCLE.INSIDEFN
  [LAMBDA (GCIRCLE WREG)                                      (* rrb "20-Jan-87 14:44")
                                                              (* determines if the global circle GCIRCLE is inside of WREG.)
    (REGIONSINTERSECTP WREG (CIRCLE.GLOBALREGIONFN GCIRCLE])

(CIRCLE.REGIONFN
  [LAMBDA (CIRCSCRELT)                                        (* rrb " 3-Oct-85 17:12")
                                                              (* returns the region occuppied by a circle.)
    (PROG ((LOCALCIRCLE (fetch (SCREENELT LOCALPART) of CIRCSCRELT))
           RADIUS)
          (SETQ RADIUS (IPLUS (FIX (ADD1 (fetch (LOCALCIRCLE RADIUS) of LOCALCIRCLE)))
                              (LRSH [ADD1 (MAX 1 (fetch (BRUSH BRUSHSIZE) of (fetch (LOCALCIRCLE LOCALCIRCLEBRUSH)
                                                                                 of LOCALCIRCLE]
                                    1)))
          (RETURN (CREATEREGION (IDIFFERENCE (fetch (POSITION XCOORD) of (SETQ LOCALCIRCLE (fetch (LOCALCIRCLE
                                                                                                   CENTERPOSITION)
                                                                                            of LOCALCIRCLE)))
                                       RADIUS)
                         (IDIFFERENCE (fetch (POSITION YCOORD) of LOCALCIRCLE)
                                RADIUS)
                         (SETQ RADIUS (ITIMES RADIUS 2))
                         RADIUS])

(CIRCLE.GLOBALREGIONFN
  [LAMBDA (GCIRELT)                                           (* rrb "18-Oct-85 16:32")
                                                              (* returns the global region occupied by a global circle element.)
    (OR (fetch (CIRCLE CIRCLEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCIRELT))
        (PROG ((INDVCIRCLE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCIRELT))
               RADIUS CENTER REGION)

          (* make the radius be too large by the amount of the brush to catch round off, i.e.
          it should be half the brush size.)

              [SETQ RADIUS (PLUS (DISTANCEBETWEEN (SETQ CENTER (fetch (CIRCLE CENTERLATLON) of INDVCIRCLE))
                                        (fetch (CIRCLE RADIUSLATLON) of INDVCIRCLE))
                                 (fetch (BRUSH BRUSHSIZE) of (fetch (CIRCLE BRUSH) of INDVCIRCLE]
              (SETQ REGION (CREATEREGION (DIFFERENCE (fetch (POSITION XCOORD) of CENTER)
                                                  RADIUS)
                                  (DIFFERENCE (fetch (POSITION YCOORD) of CENTER)
                                         RADIUS)
                                  (SETQ RADIUS (TIMES RADIUS 2))
                                  RADIUS))
              (replace (CIRCLE CIRCLEREGION) of INDVCIRCLE with REGION)
              (RETURN REGION])

(CIRCLE.TRANSLATE
  [LAMBDA (CIRCLESKELT DELTAPOS)                              (* rrb "18-Oct-85 11:00")
                                                              (* returns a changed global circle element which has the circle
                                                              translated by DELTAPOS.)
    (PROG ((GCIRCLE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of CIRCLESKELT)))
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of CIRCLESKELT))
                         INDIVIDUALGLOBALPART _ (create CIRCLE using GCIRCLE CENTERLATLON _
                                                                     (PTPLUS (fetch (CIRCLE CENTERLATLON)
                                                                                of GCIRCLE)
                                                                            DELTAPOS)
                                                                    RADIUSLATLON _ (PTPLUS (fetch (CIRCLE
                                                                                                     RADIUSLATLON
                                                                                                     )
                                                                                              of GCIRCLE)
                                                                                          DELTAPOS)
                                                                    CIRCLEREGION _ NIL])

(CIRCLE.READCHANGEFN
  [LAMBDA (SKW SCRNELTS)                                      ; Edited 23-Jul-90 15:30 by matsuda

          (* the users has selected SCRNELT to be changed this function reads a specification of how the circle elements should
          change.)

    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU
                                           (create MENU
                                                  CENTERFLG _ T
                                                  TITLE _ "Which aspect?"
                                                  ITEMS _ (APPEND (COND
                                                                     [(SKETCHINCOLORP)
                                                                      '(("Brush color" 'BRUSHCOLOR "changes the
                                                                                color of the outline")
```

```
                                                                        ("Filling color" 'FILLINGCOLOR
                                                                               "changes the color of the
                                                                               filling"]
                                                                     (T NIL))
                                                                 [COND
                                                                    (FILLPOLYGONFLG '((Filling 'FILLING
                                                                                             "allows changing of
                                                                                             the filling texture
                                                                                             of the box."]
                                                                 [COND
                                                                    (FILLINGMODEFLG '(("Filling mode"
                                                                                       'FILLINGMODE "changes how
                                                                                       the filling effects the
                                                                                       figures it covers."]
                                                                 '((Shape 'SHAPE "changes the shape of the
                                                                           brush")
                                                                   (Size 'SIZE "changes the size of the brush")
                                                                   (Dashing 'DASHING "changes the dashing of the
                                                                           line."]
                         (SIZE (READSIZECHANGE "Change size how?" T))
                         (FILLING (READ.FILLING.CHANGE))
                         (FILLINGMODE (READ.FILLING.MODE))
                         (DASHING (READ.DASHING.CHANGE))
                         (SHAPE (READBRUSHSHAPE))
                         (BRUSHCOLOR [READ.COLOR.CHANGE "Change outline color how?" NIL
                                         (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                        (fetch (SCREENELT GLOBALPART)
                                                                           of (CAR SCRNELTS))
                                                                        'BRUSH])
                         (FILLINGCOLOR [READ.COLOR.CHANGE "Change filling color how?" T
                                         (fetch (SKFILLING FILLING.COLOR) of (GETSKETCHELEMENTPROP
                                                                               (fetch (SCREENELT GLOBALPART)
                                                                                  of (CAR SCRNELTS))
                                                                               'FILLING])
                         NIL))
            (RETURN (AND HOW (LIST ASPECT HOW])
```

(CIRCLE.TRANSFORMFN

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "18-Oct-85 11:04")

            (* returns a copy of the global element that has had each of its control points transformed by transformfn.
            TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (create GLOBALPART using GELT INDIVIDUALGLOBALPART _ (create CIRCLE
                                                                           using INDVPART CENTERLATLON _
                                                                                 (SK.TRANSFORM.POINT
                                                                                  (fetch (CIRCLE CENTERLATLON)
                                                                                     of INDVPART)
                                                                                  TRANSFORMFN TRANSFORMDATA)
                                                                                 RADIUSLATLON _
                                                                                 (SK.TRANSFORM.POINT
                                                                                  (fetch (CIRCLE RADIUSLATLON)
                                                                                     of INDVPART)
                                                                                  TRANSFORMFN TRANSFORMDATA)
                                                                                 BRUSH _ (SK.TRANSFORM.BRUSH
                                                                                          (fetch (CIRCLE BRUSH)
                                                                                             of INDVPART)
                                                                                          SCALEFACTOR)
                                                                                 CIRCLEREGION _ NIL])
```

(CIRCLE.TRANSLATEPTS

```
  [LAMBDA (CIRCLESPEC SELPTS GLOBALDELTA WINDOW)                (* rrb " 9-Aug-85 09:55")
                                                                (* returns a changed global circle element which has the part
                                                                SELPOS moved to NEWPOS.)
    (PROG ((LCIRCLE (fetch (SCREENELT LOCALPART) of CIRCLESPEC))
           (GCIRCLE (fetch (SCREENELT INDIVIDUALGLOBALPART) of CIRCLESPEC)))
          (RETURN (SK.CIRCLE.CREATE (COND
                                       ((MEMBER (fetch (LOCALCIRCLE CENTERPOSITION) of LCIRCLE)
                                                SELPTS)         (* move the center)
                                        (PTPLUS (fetch (CIRCLE CENTERLATLON) of GCIRCLE)
                                                GLOBALDELTA))
                                       (T (fetch (CIRCLE CENTERLATLON) of GCIRCLE)))
                         (COND
                            ((MEMBER (fetch (LOCALCIRCLE RADIUSPOSITION) of LCIRCLE)
                                     SELPTS)                    (* move the radius point.)
                             (PTPLUS (fetch (CIRCLE RADIUSLATLON) of GCIRCLE)
                                     GLOBALDELTA))
                            (T (fetch (CIRCLE RADIUSLATLON) of GCIRCLE)))
                         (fetch (CIRCLE BRUSH) of GCIRCLE)
                         (fetch (CIRCLE DASHING) of GCIRCLE)
                         (fetch (CIRCLE CIRCLEINITSCALE) of GCIRCLE)
                         (fetch (CIRCLE CIRCLEFILLING) of GCIRCLE])
```

```
(SK.CIRCLE.CREATE
  [LAMBDA (CENTERPT RADIUSPT BRUSH DASHING INITSCALE FILLING)    (* rrb "18-Oct-85 11:01")
                                                                 (* creates a sketch element)
                                                                 (* region is a cache that will be filled if needed.)

    (SET.CIRCLE.SCALE (create GLOBALPART
                              INDIVIDUALGLOBALPART _
                              (create CIRCLE
                                      CENTERLATLON _ CENTERPT
                                      RADIUSLATLON _ RADIUSPT
                                      BRUSH _ BRUSH
                                      DASHING _ DASHING
                                      CIRCLEINITSCALE _ INITSCALE
                                      CIRCLEFILLING _ FILLING
                                      CIRCLEREGION _ NIL])

(SET.CIRCLE.SCALE
  [LAMBDA (GCIRCELT)                                             (* rrb " 7-Feb-85 12:22")

          (* sets the scale fields in a circle. Sets scale so that it goes from radius 1 to 3000.0)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCIRCELT))
           RAD)
          (SETQ RAD (DISTANCEBETWEEN (fetch (CIRCLE CENTERLATLON) of INDVPART)
                           (fetch (CIRCLE RADIUSLATLON) of INDVPART)))
          (replace (GLOBALPART COMMONGLOBALPART) of GCIRCELT with (create COMMONGLOBALPART
                                                                          MAXSCALE _ RAD
                                                                          MINSCALE _ (QUOTIENT RAD 3000.0)))
          (RETURN GCIRCELT])

(SK.BRUSH.READCHANGE
  [LAMBDA (SKW SCRNELTS)                                         (* rrb " 6-Nov-85 09:49")
                                                                 (* changefn for curves)
    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                          CENTERFLG _ T
                                                                          TITLE _ "select aspect of brush to
                                                                          change"
                                                                          ITEMS _
                                                                          (APPEND
                                                                           (COND
                                                                              [(SKETCHINCOLORP)
                                                                               '(("Color" 'BRUSHCOLOR "changes
                                                                                       the color of the brush"]
                                                                              (T NIL))
                                                                           '((Shape 'SHAPE "changes the shape of
                                                                                    the brush")
                                                                             (Size 'SIZE "changes the size of the
                                                                                   brush")
                                                                             (Dashing 'DASHING "changes the
                                                                                    dashing of the line."]
                             (SIZE (READSIZECHANGE "Change size how?"))
                             (SHAPE (READBRUSHSHAPE))
                             (DASHING (READ.DASHING.CHANGE))
                             (BRUSHCOLOR [READ.COLOR.CHANGE "Change brush color how?" NIL
                                                (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                              (fetch (SCREENELT GLOBALPART)
                                                                                 of (CAR SCRNELTS))
                                                                             'BRUSH])
                             NIL))
          (RETURN (AND HOW (LIST ASPECT HOW])
)

(DEFINEQ

(SK.INSURE.BRUSH
  [LAMBDA (BRUSH)                                                (* rrb "16-Oct-85 15:37")
                                                                 (* coerces BRUSH into a brush. Errors if it won't go.)
    (COND
       ((BRUSHP BRUSH))
       ((NUMBERP BRUSH)
        (create BRUSH
               BRUSHSIZE _ BRUSH))
       ((NULL BRUSH)
        SK.DEFAULT.BRUSH)
       (T (\ILLEGAL.ARG BRUSH])

(SK.INSURE.DASHING
  [LAMBDA (DASHING)                                              (* rrb "16-Oct-85 17:04")
                                                                 (* checks the validity of a dashing argument.
                                                                 NIL is ok and means no dashing.)

    (AND DASHING (OR (DASHINGP DASHING)
                     (\ILLEGAL.ARG DASHING])
```

```
)

(DECLARE%: EVAL@COMPILE

(RECORD BRUSH (BRUSHSHAPE BRUSHSIZE BRUSHCOLOR)
       BRUSHSHAPE _ 'ROUND BRUSHSIZE _ 1)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD LOCALCIRCLE ((CENTERPOSITION RADIUSPOSITION)
                     LOCALHOTREGION RADIUS LOCALCIRCLEBRUSH LOCALCIRCLEFILLING LOCALCIRCLEDASHING))

(TYPERECORD CIRCLE (CENTERLATLON RADIUSLATLON BRUSH DASHING CIRCLEINITSCALE CIRCLEFILLING CIRCLEREGION))
)
)

(READVARS-FROM-STRINGS '(CIRCLEICON)
       "({(READBITMAP)(20 12
       %"@AOH@@@@%"
       %"@COL@@@@%"
       %"@G@N@@@@%"
       %"@F@F@@@@%"
       %"@N@G@@@@%"
       %"@L@C@@@@%"
       %"@L@C@@@@%"
       %"@N@G@@@@%"
       %"@F@F@@@@%"
       %"@G@N@@@@%"
       %"@COL@@@@%"
       %"@AOH@@@@%")})
       ")

(RPAQ CIRCLE.CENTER (CURSORCREATE '
                                 'NIL 7 7))

(RPAQ CIRCLE.EDGE (CURSORCREATE '
                               'NIL 15 7))

(RPAQ? SK.DEFAULT.BRUSH (CONS 'ROUND (CONS 1 (CONS 'BLACK NIL))))

(RPAQ? SK.DEFAULT.DASHING )

(RPAQ? SK.DEFAULT.TEXTURE )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SK.DEFAULT.BRUSH SK.DEFAULT.DASHING SK.DEFAULT.TEXTURE)
)

(DEFINEQ

(SKETCH.CREATE.ELLIPSE
  [LAMBDA (CENTERPT ORIENTATIONPT OTHERRADIUSPT BRUSH DASHING WILLBEFILLING SCALE)
                                                                      (* rrb "16-Oct-85 17:05")
                                                                      (* creates a sketch ellipse element.)
    (ELLIPSE.CREATE (SK.INSURE.POSITION CENTERPT)
           (SK.INSURE.POSITION ORIENTATIONPT)
           (SK.INSURE.POSITION OTHERRADIUSPT)
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0])

(ELLIPSE.EXPANDFN
  [LAMBDA (GELLIPSE SCALE)                                            (* rrb " 7-Dec-85 20:40")

            (* returns a screen elt that has a ellipse screen element calculated from the global part.)

    (PROG (CENTER MINRAD MAJRAD BRUSH (INDGELLIPSE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELLIPSE)))
          [COND
             ((fetch (ELLIPSE ELLIPSEINITSCALE) of INDGELLIPSE))
             (T                                                       (* old format didn't have an initial scale, create one and default it
                to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GELLIPSE
                   with (SETQ INDGELLIPSE (create ELLIPSE using INDGELLIPSE ELLIPSEINITSCALE _ 1.0 ELLIPSEREGION _
                                                                   NIL]
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALELLIPSE
                                            ELLIPSECENTER _ (SETQ CENTER (SK.SCALE.POSITION.INTO.VIEWER
                                                                          (fetch (ELLIPSE ELLIPSECENTERLATLON)
                                                                             of INDGELLIPSE)
```

```
                                                                                    SCALE))
                                                     MINORRADIUSPOSITION _ (SETQ MINRAD (SK.SCALE.POSITION.INTO.VIEWER
                                                                                          (fetch (ELLIPSE SEMIMINORLATLON)
                                                                                             of INDGELLIPSE)
                                                                                          SCALE))
                                                     MAJORRADIUSPOSITION _ (SETQ MAJRAD (SK.SCALE.POSITION.INTO.VIEWER
                                                                                          (fetch (ELLIPSE SEMIMAJORLATLON)
                                                                                             of INDGELLIPSE)
                                                                                          SCALE))
                                                     SEMIMINORRADIUS _ (DISTANCEBETWEEN CENTER MINRAD)
                                                     SEMIMAJORRADIUS _ (DISTANCEBETWEEN CENTER MAJRAD)
                                                     LOCALELLIPSEBRUSH _
                                                     (SCALE.BRUSH (COND
                                                                     ([NOT (NUMBERP (SETQ BRUSH (fetch (ELLIPSE BRUSH)
                                                                                                   of INDGELLIPSE]
                                                                          (* new format, old format had brush width only.)
                                                                      BRUSH)
                                                                     (T [replace (ELLIPSE BRUSH) of INDGELLIPSE
                                                                           with (SETQ BRUSH
                                                                                 (create BRUSH
                                                                                        BRUSHSIZE _ BRUSH
                                                                                        BRUSHSHAPE _ 'ROUND]
                                                                        BRUSH))
                                                             (fetch (ELLIPSE ELLIPSEINITSCALE) of INDGELLIPSE)
                                                             SCALE)
                                                     LOCALELLIPSEDASHING _ (fetch (ELLIPSE DASHING) of INDGELLIPSE))
                                  GLOBALPART _ GELLIPSE])
```

(ELLIPSE.DRAWFN

```
  [LAMBDA (ELLIPSEELT WINDOW REGION)                                       (* rrb " 7-Dec-85 20:40")
                                                                           (* draws a ellipse from a circle element.)
    (PROG ((GELLIPSE (fetch (SCREENELT INDIVIDUALGLOBALPART) of ELLIPSEELT))
           (LELLIPSE (fetch (SCREENELT LOCALPART) of ELLIPSEELT))
           CPOS DASHING ORIENTATION)
          (SETQ CPOS (fetch (LOCALELLIPSE ELLIPSECENTER) of LELLIPSE))
          (SETQ DASHING (fetch (LOCALELLIPSE LOCALELLIPSEDASHING) of LELLIPSE))
          (SETQ ORIENTATION (fetch (ELLIPSE ORIENTATION) of GELLIPSE))
          (RETURN (COND
                     (DASHING                                              (* draw it with the curve drawing code which does dashing.)
                            (PROG ((SINOR (SIN ORIENTATION))
                                   (COSOR (COS ORIENTATION))
                                   (CENTERX (fetch (POSITION XCOORD) of CPOS))
                                   (CENTERY (fetch (POSITION YCOORD) of CPOS))
                                   (SEMIMINORRADIUS (fetch (LOCALELLIPSE SEMIMINORRADIUS) of LELLIPSE))
                                   (SEMIMAJORRADIUS (fetch (LOCALELLIPSE SEMIMAJORRADIUS) of LELLIPSE)))
                                  (DRAWCURVE [LIST (CREATEPOSITION (PLUS CENTERX (FTIMES COSOR SEMIMAJORRADIUS))
                                                          (PLUS CENTERY (FTIMES SINOR SEMIMAJORRADIUS)))
                                                   (CREATEPOSITION (DIFFERENCE CENTERX (FTIMES SINOR
                                                                                                   SEMIMINORRADIUS))
                                                          (PLUS CENTERY (FTIMES COSOR SEMIMINORRADIUS)))
                                                   (CREATEPOSITION (DIFFERENCE CENTERX (FTIMES COSOR
                                                                                                   SEMIMAJORRADIUS))
                                                          (DIFFERENCE CENTERY (FTIMES SINOR SEMIMAJORRADIUS)))
                                                   (CREATEPOSITION (PLUS CENTERX (FTIMES SINOR SEMIMINORRADIUS))
                                                          (DIFFERENCE CENTERY (FTIMES COSOR SEMIMINORRADIUS]
                                         T
                                         (fetch (LOCALELLIPSE LOCALELLIPSEBRUSH) of LELLIPSE)
                                         DASHING WINDOW)))
                     (T (DRAWELLIPSE (fetch (POSITION XCOORD) of CPOS)
                               (fetch (POSITION YCOORD) of CPOS)
                               (fetch (LOCALELLIPSE SEMIMINORRADIUS) of LELLIPSE)
                               (fetch (LOCALELLIPSE SEMIMAJORRADIUS) of LELLIPSE)
                               ORIENTATION
                               (fetch (LOCALELLIPSE LOCALELLIPSEBRUSH) of LELLIPSE)
                               DASHING WINDOW])
```

(ELLIPSE.INPUTFN

```
  [LAMBDA (WINDOW)                                                         (* rrb "21-May-86 16:13")

          (* reads three points from the user and returns the ellipse figure element that it represents.)

    (PROG (CENTER MAJRAD MINRAD)
          (STATUSPRINT WINDOW "
                 " "Indicate center of ellipse")
          (COND
             ((SETQ CENTER (SK.READ.POINT.WITH.FEEDBACK WINDOW ELLIPSE.CENTER NIL NIL NIL NIL
                                  SKETCH.USE.POSITION.PAD))
              (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                     NIL WINDOW))
             (T (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                 " "Indicate semi-major axis")
          (COND
             ((SETQ MAJRAD (SK.READ.ELLIPSE.MAJOR.PT WINDOW (fetch (INPUTPT INPUT.POSITION) of CENTER)))
              (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of MAJRAD)
                     NIL WINDOW))
             (T                                                  (* erase center pt on way out)
                (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                       NIL WINDOW)
                (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                 " "Indicate semi-minor axis")
          (SETQ MINRAD (SK.READ.ELLIPSE.MINOR.PT WINDOW (fetch (INPUTPT INPUT.POSITION) of CENTER)
                              (fetch (INPUTPT INPUT.POSITION) of MAJRAD)))
          (CLOSEPROMPTWINDOW WINDOW)                             (* erase the point marks.)
          (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of MAJRAD)
                 NIL WINDOW)
          (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                 NIL WINDOW)
          (OR MINRAD (RETURN NIL))
          (RETURN (ELLIPSE.CREATE (SK.MAP.INPUT.PT.TO.GLOBAL CENTER WINDOW)
                         (SK.MAP.INPUT.PT.TO.GLOBAL MINRAD WINDOW)
                         (SK.MAP.INPUT.PT.TO.GLOBAL MAJRAD WINDOW)
                         (fetch (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP WINDOW 'SKETCHCONTEXT))
                         (fetch (SKETCHCONTEXT SKETCHDASHING) of (WINDOWPROP WINDOW 'SKETCHCONTEXT))
                         (SK.INPUT.SCALE WINDOW])
```

(SK.READ.ELLIPSE.MAJOR.PT

```
  [LAMBDA (SKW CENTERPT)                                         (* rrb "20-May-86 10:47")

          (* reads a position from the user that will be the major radius point of an ellipse.)

    (SK.READ.POINT.WITH.FEEDBACK WINDOW ELLIPSE.SEMI.MAJOR (AND SKETCH.VERBOSE.FEEDBACK (FUNCTION
                                                                                   SK.SHOW.ELLIPSE.MAJOR.RADIUS
                                                                                         ))
           CENTERPT
           'MIDDLE NIL SKETCH.USE.POSITION.PAD])
```

(SK.SHOW.ELLIPSE.MAJOR.RADIUS

```
  [LAMBDA (X Y WINDOW CENTERPT)                                  (* rrb "14-Nov-85 16:46")

          (* xors a line from X Y to a point the opposite side of CENTERPT in a window.
          Used as the feedback function for reading the major radius point for ellipses.)
                                                                 (* Mark the point too.)
    (SHOWSKETCHXY X Y WINDOW)
    (DRAWLINE X Y (PLUS X (TIMES 2 (DIFFERENCE (fetch (POSITION XCOORD) of CENTERPT)
                                            X)))
           (PLUS Y (TIMES 2 (DIFFERENCE (fetch (POSITION YCOORD) of CENTERPT)
                                   Y)))
           1
           'INVERT WINDOW])
```

(SK.READ.ELLIPSE.MINOR.PT

```
  [LAMBDA (SKW CENTERPT MAJORPT)                                 (* rrb "20-May-86 10:47")

          (* reads a position from the user that will be the major radius point of an ellipse.)

    (SK.READ.POINT.WITH.FEEDBACK WINDOW ELLIPSE.SEMI.MINOR (AND SKETCH.VERBOSE.FEEDBACK (FUNCTION
                                                                                   SK.SHOW.ELLIPSE.MINOR.RADIUS
                                                                                         ))
           (LIST CENTERPT (DISTANCEBETWEEN CENTERPT MAJORPT)
                 (COMPUTE.ELLIPSE.ORIENTATION CENTERPT MAJORPT))
           'MIDDLE NIL SKETCH.USE.POSITION.PAD])
```

(SK.SHOW.ELLIPSE.MINOR.RADIUS

```
  [LAMBDA (X Y WINDOW ELLIPSEARGS)                               (* rrb "15-Nov-85 14:17")

          (* xors a line from X Y to a point the opposite side of CENTERPT in a window.
          Used as the feedback function for reading the major radius point for ellipses.)
                                                                 (* Mark the point too.)
    (SHOWSKETCHXY X Y WINDOW)
    (PROG ((CENTERX (CAR ELLIPSEARGS))
           CENTERY)
          (SETQ CENTERY (fetch (POSITION YCOORD) of CENTERX))
          (SETQ CENTERX (fetch (POSITION XCOORD) of CENTERX))
          (DRAWELLIPSE CENTERX CENTERY (SK.DISTANCE.TO CENTERX CENTERY X Y)
                 (CADR ELLIPSEARGS)
                 (CADDR ELLIPSEARGS)
                 1 NIL WINDOW])
```

(ELLIPSE.INSIDEFN

```
  [LAMBDA (GELLIPSE WREG)                                        (* rrb "20-Jan-87 14:45")
                                                                 (* determines if the global ellipse GELLIPSE is inside of WREG.)
    (REGIONSINTERSECTP WREG (ELLIPSE.GLOBALREGIONFN GELLIPSE])
(ELLIPSE.CREATE
  [LAMBDA (CENTERPT MINPT MAJPT BRUSH DASHING INITSCALE)       (* rrb "19-Jul-85 14:26")
                                                              (* creates a global ellipse element.)
    (PROG ((MAXRAD (MAX (DISTANCEBETWEEN CENTERPT MINPT)
                        (DISTANCEBETWEEN CENTERPT MAJPT)))
           ORIENTATION)
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (create COMMONGLOBALPART
                                                    MAXSCALE _ MAXRAD
                                                    MINSCALE _ (QUOTIENT MAXRAD 3000.0))
                         INDIVIDUALGLOBALPART _ (create ELLIPSE
                                                        ORIENTATION _ (SETQ ORIENTATION (
                                                                           COMPUTE.ELLIPSE.ORIENTATION
                                                                            CENTERPT MAJPT))
                                                        BRUSH _ BRUSH
                                                        DASHING _ DASHING
                                                        ELLIPSECENTERLATLON _ CENTERPT
                                                        SEMIMINORLATLON _ (SK.COMPUTE.ELLIPSE.MINOR.RADIUS.PT
                                                                           CENTERPT MAJPT MINPT ORIENTATION)
                                                        SEMIMAJORLATLON _ MAJPT
                                                        ELLIPSEINITSCALE _ INITSCALE])

(SK.UPDATE.ELLIPSE.AFTER.CHANGE
  [LAMBDA (GELLIPSEELT)                                       (* rrb " 7-Dec-85 19:54")
                                                              (* updates the dependent fields of an ellipse element when a
                                                              field changes.)
    (replace (ELLIPSE ELLIPSEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELLIPSEELT) with NIL])

(ELLIPSE.REGIONFN
  [LAMBDA (ELLIPSCRELT)                                       (* rrb " 3-Oct-85 17:10")
                                                              (* returns the region occuppied by an ellipse.)
    (PROG ((LOCALELLIPSE (fetch (SCREENELT LOCALPART) of ELLIPSCRELT))
           MAJORRADPT CENTERX CENTERY BRUSHADJ HALFWID HALFHGHT RADRATIO DELTAX DELTAY)
          (SETQ MAJORRADPT (fetch (LOCALELLIPSE MAJORRADIUSPOSITION) of LOCALELLIPSE))
          (SETQ CENTERY (fetch (LOCALELLIPSE ELLIPSECENTER) of LOCALELLIPSE))
          [SETQ RADRATIO (ABS (FQUOTIENT (fetch (LOCALELLIPSE SEMIMINORRADIUS) of LOCALELLIPSE)
                                     (fetch (LOCALELLIPSE SEMIMAJORRADIUS) of LOCALELLIPSE]
          [SETQ DELTAX (ABS (IDIFFERENCE (SETQ CENTERX (fetch (POSITION XCOORD) of CENTERY))
                                    (fetch (POSITION XCOORD) of MAJORRADPT]
          [SETQ DELTAY (ABS (IDIFFERENCE (SETQ CENTERY (fetch (POSITION YCOORD) of CENTERY))
                                    (fetch (POSITION YCOORD) of MAJORRADPT]
          (SETQ BRUSHADJ (LRSH (ADD1 (fetch (BRUSH BRUSHSIZE) of (fetch (LOCALELLIPSE LOCALELLIPSEBRUSH)
                                                                         of LOCALELLIPSE)))
                               1))
          (SETQ HALFWID (FIXR (PLUS DELTAX (FTIMES RADRATIO DELTAY)
                                    BRUSHADJ)))
          (SETQ HALFHGHT (FIXR (PLUS DELTAY (FTIMES RADRATIO DELTAX)
                                     BRUSHADJ)))

          (* use the rectangle that contains the rectangle made by the extreme points of the ellipse.
          This gets more than is called for when the orientation isn't 0 or 90.0)

          (RETURN (CREATEREGION (IDIFFERENCE CENTERX HALFWID)
                         (IDIFFERENCE CENTERY HALFHGHT)
                         (ITIMES HALFWID 2)
                         (ITIMES HALFHGHT 2])

(ELLIPSE.GLOBALREGIONFN
  [LAMBDA (GELELT)                                            (* rrb "20-Nov-85 16:09")
                                                              (* returns the global region occupied by a global ellipse element.)
    (OR (fetch (ELLIPSE ELLIPSEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELELT))
        (PROG ((INDVELLIPSE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELELT))
               CENTERPT HALFBOXSIZE MAXRAD REGION)
              (SETQ CENTERPT (fetch (ELLIPSE ELLIPSECENTERLATLON) of INDVELLIPSE))
              [SETQ MAXRAD (MAX (DISTANCEBETWEEN CENTERPT (fetch (ELLIPSE SEMIMAJORLATLON) of INDVELLIPSE))
                                (DISTANCEBETWEEN CENTERPT (fetch (ELLIPSE SEMIMINORLATLON) of INDVELLIPSE]
              [SETQ HALFBOXSIZE (PLUS MAXRAD (fetch (BRUSH BRUSHSIZE) of (fetch (ELLIPSE BRUSH) of INDVELLIPSE]

          (* use a square about the center as wide as the largest radius. This gets too much but is easy to calculate.)

              (SETQ REGION (CREATEREGION (DIFFERENCE (fetch (POSITION XCOORD) of CENTERPT)
                                                  HALFBOXSIZE)
                                  (DIFFERENCE (fetch (POSITION YCOORD) of CENTERPT)
                                         HALFBOXSIZE)
                                  (ITIMES HALFBOXSIZE 2)
                                  (ITIMES HALFBOXSIZE 2)))
              (replace (ELLIPSE ELLIPSEREGION) of INDVELLIPSE with REGION)
              (RETURN REGION])

(ELLIPSE.TRANSLATEFN
  [LAMBDA (SKELT DELTAPOS)                                    (* rrb "18-Oct-85 17:08")
                                                              (* returns a global ellipse element which has been translated by
    DELTAPOS.)
    (PROG ((GLOBALEL (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKELT)))
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of SKELT))
                         INDIVIDUALGLOBALPART _ (create ELLIPSE using GLOBALEL ORIENTATION _ (fetch (ELLIPSE
                                                                                                     ORIENTATION
                                                                                                      )
                                                                                               of GLOBALEL)
                                                                     ELLIPSECENTERLATLON _
                                                                     (PTPLUS (fetch (ELLIPSE ELLIPSECENTERLATLON)
                                                                                 of GLOBALEL)
                                                                            DELTAPOS)
                                                                     SEMIMINORLATLON _ (PTPLUS
                                                                                        (fetch (ELLIPSE
                                                                                                SEMIMINORLATLON
                                                                                                )
                                                                                           of GLOBALEL)
                                                                                        DELTAPOS)
                                                                     SEMIMAJORLATLON _ (PTPLUS
                                                                                        (fetch (ELLIPSE
                                                                                                SEMIMAJORLATLON
                                                                                                )
                                                                                           of GLOBALEL)
                                                                                        DELTAPOS)
                                                                     ELLIPSEREGION _ NIL])
```

(**ELLIPSE.TRANSFORMFN**

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "26-Apr-85 16:21")

          (* returns a copy of the global ellipse element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (ELLIPSE.CREATE (SK.TRANSFORM.POINT (fetch (ELLIPSE ELLIPSECENTERLATLON) of INDVPART)
                                         TRANSFORMFN TRANSFORMDATA)
                         (SK.TRANSFORM.POINT (fetch (ELLIPSE SEMIMINORLATLON) of INDVPART)
                                TRANSFORMFN TRANSFORMDATA)
                         (SK.TRANSFORM.POINT (fetch (ELLIPSE SEMIMAJORLATLON) of INDVPART)
                                TRANSFORMFN TRANSFORMDATA)
                         (SK.TRANSFORM.BRUSH (fetch (ELLIPSE BRUSH) of INDVPART)
                                SCALEFACTOR)
                         (fetch (ELLIPSE DASHING) of INDVPART)
                         (fetch (ELLIPSE ELLIPSEINITSCALE) of INDVPART])
```

(**ELLIPSE.TRANSLATEPTS**

```
  [LAMBDA (ELLIPSESPEC SELPTS GLOBALDELTA WINDOW)               (* rrb " 5-May-85 16:41")

          (* returns a new global ellipse element which has the points on SELPTS moved by a global distance.)

    (PROG ((LELLIPSE (fetch (SCREENELT LOCALPART) of ELLIPSESPEC))
           (GELLIPSE (fetch (SCREENELT INDIVIDUALGLOBALPART) of ELLIPSESPEC)))
          (RETURN (ELLIPSE.CREATE (COND
                                     ((MEMBER (fetch (LOCALELLIPSE ELLIPSECENTER) of LELLIPSE)
                                             SELPTS)                (* move the center)
                                      (PTPLUS (fetch (ELLIPSE ELLIPSECENTERLATLON) of GELLIPSE)
                                             GLOBALDELTA))
                                     (T (fetch (ELLIPSE ELLIPSECENTERLATLON) of GELLIPSE)))
                         (COND
                            ((MEMBER (fetch (LOCALELLIPSE MINORRADIUSPOSITION) of LELLIPSE)
                                    SELPTS)                         (* move the radius point.)
                             (PTPLUS (fetch (ELLIPSE SEMIMINORLATLON) of GELLIPSE)
                                    GLOBALDELTA))
                            (T (fetch (ELLIPSE SEMIMINORLATLON) of GELLIPSE)))
                         (COND
                            ((MEMBER (fetch (LOCALELLIPSE MAJORRADIUSPOSITION) of LELLIPSE)
                                    SELPTS)                         (* move the radius point.)
                             (PTPLUS (fetch (ELLIPSE SEMIMAJORLATLON) of GELLIPSE)
                                    GLOBALDELTA))
                            (T (fetch (ELLIPSE SEMIMAJORLATLON) of GELLIPSE)))
                         (fetch (ELLIPSE BRUSH) of GELLIPSE)
                         (fetch (ELLIPSE DASHING) of GELLIPSE)
                         (fetch (ELLIPSE ELLIPSEINITSCALE) of GELLIPSE])
```

(**MARK.SPOT**

```
  [LAMBDA (X/POSITION Y WINDOW)                                 (* rrb "14-JAN-83 15:40")
    (PROG [X WIDTH HEIGHT (COLORDS (WINDOWPROP WINDOW 'INCOLOR]
          (COND
             ((POSITIONP X/POSITION)
              (SETQ X (fetch (POSITION XCOORD) of X/POSITION))
              (SETQ Y (fetch (POSITION YCOORD) of X/POSITION)))
             (T (SETQ X X/POSITION)))
          (SETQ WIDTH (BITMAPWIDTH SPOTMARKER))
          (SETQ HEIGHT (BITMAPHEIGHT SPOTMARKER))
          (BITBLT (COND
                     [COLORDS (COND
                                 ((AND (BITMAPP COLORSPOTMARKER)
                                       (EQ (BITSPERPIXEL COLORSPOTMARKER)
                                           (COLORNUMBERBITSPERPIXEL)))
                                  COLORSPOTMARKER)
                                 (T (SETQ COLORSPOTMARKER (COLORIZEBITMAP SPOTMARKER 0 (MAXIMUMCOLOR)
                                                                 (COLORNUMBERBITSPERPIXEL]
                     (T SPOTMARKER))
                 0 0 (OR COLORDS WINDOW)
                 (IDIFFERENCE X (IQUOTIENT WIDTH 2))
                 (IDIFFERENCE Y (IQUOTIENT HEIGHT 2))
                 WIDTH HEIGHT 'INPUT 'INVERT])
```

(**DISTANCEBETWEEN**

```
  [LAMBDA (P1 P2)                                                (* rrb " 5-JAN-83 12:17")
                                                                 (* returns the distance between two points)
    (SQRT (PLUS (SQUARE (DIFFERENCE (fetch (POSITION XCOORD) of P1)
                                    (fetch (POSITION XCOORD) of P2)))
                (SQUARE (DIFFERENCE (fetch (POSITION YCOORD) of P1)
                                    (fetch (POSITION YCOORD) of P2])
```

(**SK.DISTANCE.TO**

```
  [LAMBDA (X1 Y1 X2 Y2)                                          (* rrb "15-Nov-85 14:17")
                                                                 (* returns the distance between two points)
    (SQRT (PLUS (SQUARE (DIFFERENCE X1 X2))
                (SQUARE (DIFFERENCE Y1 Y2])
```

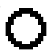
(**SQUARE**

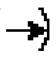
```
  [LAMBDA (X)
    (TIMES X X])
```

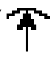
(**COMPUTE.ELLIPSE.ORIENTATION**

```
  [LAMBDA (CENTERPT MAJRADPT)                                    (* rrb "19-Oct-85 12:44")

          (* computes the orientation of an ellipse from its center point and its major radius point.)

    (PROG [(DELTAX (IDIFFERENCE (fetch (POSITION XCOORD) of MAJRADPT)
                          (fetch (POSITION XCOORD) of CENTERPT]
          (RETURN (COND
                     ((ZEROP DELTAX)
                      90.0)
                     (T (ARCTAN2 (IDIFFERENCE (fetch (POSITION YCOORD) of MAJRADPT)
                                        (fetch (POSITION YCOORD) of CENTERPT))
                               DELTAX])
```

(**SK.COMPUTE.ELLIPSE.MINOR.RADIUS.PT**

```
  [LAMBDA (CENTER MAJORRADPT MINORPT ORIENTATION)                (* rrb "19-Jul-85 14:23")

          (* computes the point that is on the minor radius of an ellipse about CENTER with major radius and axis determined by
          MAJORRADPT and minor radius determines by MINORPT.)

    (PROG ((SINOR (SIN ORIENTATION))
           (COSOR (COS ORIENTATION))
           (SEMIMINORRADIUS (DISTANCEBETWEEN CENTER MINORPT))
           (SEMIMAJORRADIUS (DISTANCEBETWEEN CENTER MAJORRADPT)))
          (RETURN (CREATEPOSITION (DIFFERENCE (fetch (POSITION XCOORD) of CENTER)
                                         (FTIMES SINOR SEMIMINORRADIUS))
                         (PLUS (fetch (POSITION YCOORD) of CENTER)
                               (FTIMES COSOR SEMIMINORRADIUS])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD LOCALELLIPSE ((ELLIPSECENTER MINORRADIUSPOSITION MAJORRADIUSPOSITION)
                      LOCALHOTREGION SEMIMINORRADIUS SEMIMAJORRADIUS LOCALELLIPSEBRUSH LOCALELLIPSEDASHING
                      LOCALELLIPSEFILLING))

(TYPERECORD ELLIPSE (ELLIPSECENTERLATLON SEMIMINORLATLON SEMIMAJORLATLON ORIENTATION BRUSH DASHING
                           ELLIPSEINITSCALE ELLIPSEFILLING ELLIPSEREGION))
)
)

(READVARS-FROM-STRINGS '(ELLIPSEICON)
       "({(READBITMAP)(20 12
       %"@COL@@@@%"
       %"AOOOH@@@%"
       %"CN@GL@@@%"
       %"G@@@N@@@%"
       %"N@@@G@@@%"
       %"L@@@C@@@%"
       %"L@@@C@@@%"
       %"N@@@G@@@%"
       %"G@@@N@@@%"
       %"CN@GL@@@%"
       %"AOOOH@@@%"
       %"@COL@@@@%")})
       ")

(RPAQ ELLIPSE.CENTER (CURSORCREATE '
```


```
                                   'NIL 7 7))

(RPAQ ELLIPSE.SEMI.MAJOR (CURSORCREATE '
```


```
                                       'NIL 15 7))

(RPAQ ELLIPSE.SEMI.MINOR (CURSORCREATE '
```


```
                                       'NIL 7 15))

(DEFINEQ

(SKETCH.CREATE.OPEN.CURVE
  [LAMBDA (POINTS BRUSH DASHING ARROWHEADS SCALE)                  (* rrb "16-Oct-85 17:14")
                                                                   (* creates a sketch open curve element.)

    (SK.CURVE.CREATE (SK.INSURE.POINT.LIST POINTS)
           NIL
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.ARROWHEADS ARROWHEADS])

(OPENCURVE.INPUTFN
  [LAMBDA (W)                                                      (* rrb "19-Mar-86 17:40")
                                                                   (* reads a spline {series of points} from the user.)
    (PROG ((SKCONTEXT (WINDOWPROP W 'SKETCHCONTEXT))
           KNOTS)
          (RETURN (SK.CURVE.CREATE (SETQ KNOTS (for PT in (READ.LIST.OF.POINTS W T) collect (
                                                                                       SK.MAP.INPUT.PT.TO.GLOBAL
                                                                                          PT W)))
                         NIL
                         (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKCONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHDASHING) of SKCONTEXT)
                         (SK.INPUT.SCALE W)
                         (SK.ARROWHEAD.CREATE W KNOTS])

(SK.CURVE.CREATE
  [LAMBDA (GKNOTS CLOSED BRUSH DASHING INITSCALE ARROWHEADS)       (* rrb "19-Mar-86 17:40")
                                                                   (* creates a sketch element representing a curve.)
    (AND GKNOTS
         (KNOT.SET.SCALE.FIELD (create GLOBALPART
                                      INDIVIDUALGLOBALPART _
                                      (COND
                                         (CLOSED (create CLOSEDCURVE
                                                        LATLONKNOTS _ GKNOTS
                                                        BRUSH _ BRUSH
                                                        DASHING _ DASHING
                                                        CLOSEDCURVEINITSCALE _ INITSCALE))
                                         (T (SET.OPENCURVE.ARROWHEAD.POINTS (create OPENCURVE
                                                                                   LATLONKNOTS _ GKNOTS
                                                                                   BRUSH _ BRUSH
                                                                                   DASHING _ DASHING
                                                                                   OPENCURVEINITSCALE _
                                                                                   INITSCALE
                                                                                   CURVEARROWHEADS _
                                                                                   ARROWHEADS])

(MAXXEXTENT
  [LAMBDA (PTS)                                                    (* rrb " 1-APR-83 17:24")
                                                                   (* returns the maximum width between any two points on pts)
    (COND
       ((NULL PTS)
        0)
       (T (PROG ((XMIN (fetch (POSITION XCOORD) of (CAR PTS)))
                 XMAX)
                (SETQ XMAX XMIN)
                [for PT in (CDR PTS) do (COND
                                           ((GREATERP (SETQ PT (fetch (POSITION XCOORD) of PT))
                                                    XMAX)
                                            (SETQ XMAX PT)))
                                        (COND
                                           ((GREATERP XMIN PT)
                                    (SETQ XMIN PT]
                (RETURN (DIFFERENCE XMAX XMIN])

(MAXYEXTENT
  [LAMBDA (PTS)                                               (* rrb " 1-APR-83 17:24")
                                                              (* returns the maximum height between any two points on pts)

    (COND
       ((NULL PTS)
        0)
       (T (PROG ((YMIN (fetch (POSITION YCOORD) of (CAR PTS)))
                 YMAX)
                (SETQ YMAX YMIN)
                [for PT in (CDR PTS) do (COND
                                           ((GREATERP (SETQ PT (fetch (POSITION YCOORD) of PT))
                                                    YMAX)
                                            (SETQ YMAX PT)))
                                        (COND
                                           ((GREATERP YMIN PT)
                                            (SETQ YMIN PT]
                (RETURN (DIFFERENCE YMAX YMIN])

(KNOT.SET.SCALE.FIELD
  [LAMBDA (GKNOTELT)                                          (* rrb "31-Jan-85 18:22")
                                                              (* updates the scale field after a change in the knots of a knotted
    element.)
    (PROG [(PTS (fetch (KNOTELT LATLONKNOTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GKNOTELT]
          (replace (GLOBALPART MINSCALE) of GKNOTELT with 0.0)       (* show it as long as it is two points wide or high.)
          (replace (GLOBALPART MAXSCALE) of GKNOTELT with (FQUOTIENT (MAX 8 (MAXXEXTENT PTS)
                                                                          (MAXYEXTENT PTS))
                                                             2.0))
          (RETURN GKNOTELT])

(OPENCURVE.DRAWFN
  [LAMBDA (CURVEELT WINDOW REGION)                            (* rrb " 6-May-86 17:42")
                                                              (* draws a curve figure element.)
    (PROG ((GCURVE (fetch (SCREENELT INDIVIDUALGLOBALPART) of CURVEELT))
           (LCURVE (fetch (SCREENELT LOCALPART) of CURVEELT))
           BRUSH LOCALPTS LOCALARROWPTS GARROWSPECS)
          (AND REGION (NOT (REGIONSINTERSECTP REGION (SK.ITEM.REGION CURVEELT)))
               (RETURN))
          (SETQ GARROWSPECS (fetch (OPENCURVE CURVEARROWHEADS) of GCURVE))
          (SETQ LOCALARROWPTS (fetch (LOCALCURVE ARROWHEADPTS) of LCURVE))
          (SETQ LOCALPTS (\SK.ADJUST.FOR.ARROWHEADS (fetch (LOCALCURVE KNOTS) of LCURVE)
                                LOCALARROWPTS GARROWSPECS WINDOW))
          (DRAWCURVE LOCALPTS NIL (SETQ BRUSH (fetch (LOCALCURVE LOCALCURVEBRUSH) of LCURVE))
                 (fetch (LOCALCURVE LOCALCURVEDASHING) of LCURVE)
                 WINDOW)
          (DRAWARROWHEADS GARROWSPECS LOCALARROWPTS WINDOW BRUSH])

(OPENCURVE.EXPANDFN
  [LAMBDA (GELT SCALE)                                        (* rrb " 2-May-86 10:50")

          (* returns a local record which has the LATLONKNOTS field of the global element GELT translated into window coordinats.
          Used for open curves)

    (PROG ((INDGELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           LOCALKNOTS TMP)
          [COND
             ((fetch (OPENCURVE OPENCURVEINITSCALE) of INDGELT))
             (T                                               (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GELT
                   with (SETQ INDGELT (create OPENCURVE using INDGELT OPENCURVEINITSCALE _ 1.0 OPENCURVEREGION _
                                                              NIL]
          (COND
             ((AND (fetch (OPENCURVE CURVEARROWHEADS) of INDGELT)
                   (NOT (fetch (OPENCURVE OPENCURVEARROWHEADPOINTS) of INDGELT)))
                                                              (* old form didn't have global points, update it)
              (SET.OPENCURVE.ARROWHEAD.POINTS INDGELT)))
          (SETQ LOCALKNOTS (for LATLONPT in (fetch (OPENCURVE LATLONKNOTS) of INDGELT)
                              collect (SK.SCALE.POSITION.INTO.VIEWER LATLONPT SCALE)))
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALCURVE
                                            KNOTS _ LOCALKNOTS
                                            ARROWHEADPTS _ (SK.EXPAND.ARROWHEADS (fetch (OPENCURVE
                                                                                         OPENCURVEARROWHEADPOINTS
                                                                                                )
                                                                                      of INDGELT)
                                                                  SCALE)
                                            LOCALCURVEBRUSH _
                                            (SCALE.BRUSH (COND
                                                            ([NOT (NUMBERP (SETQ TMP (fetch (OPENCURVE BRUSH)
                                                                                          of INDGELT]
                                                              (* new format, old format had brush width only.)
```

```
                                                                           TMP)
                                                                          (T [replace (OPENCURVE BRUSH) of INDGELT
                                                                                with (SETQ TMP
                                                                                      (create BRUSH
                                                                                             BRUSHSIZE _ TMP
                                                                                             BRUSHSHAPE _ 'ROUND]
                                                                             TMP))
                                                               (fetch (OPENCURVE OPENCURVEINITSCALE) of INDGELT)
                                                               SCALE)
                                                       LOCALCURVEDASHING _ (fetch (OPENCURVE DASHING) of INDGELT))
                                  GLOBALPART _ GELT])
```

(OPENCURVE.READCHANGEFN

```
  [LAMBDA (SKW SCRNELTS)                                                       (* rrb "17-Dec-85 16:22")
                                                                               (* changefn for curves)
    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                           CENTERFLG _ T
                                                                           TITLE _ "Which aspect?"
                                                                           ITEMS _
                                                                           (APPEND
                                                                            (COND
                                                                               [(SKETCHINCOLORP)
                                                                                '((Color 'BRUSHCOLOR "Changes the
                                                                                          color of the curve."]
                                                                               (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
                                                                                    brush")
                                                                              (Dashing 'DASHING "changes the
                                                                                     dashing of the line.")
                                                                              ("Add point" 'ADDPOINT "adds a point
                                                                                     to the curve."]
                                  (SIZE (READSIZECHANGE "Change size how?"))
                                  (SHAPE (READBRUSHSHAPE))
                                  (ARROW (READ.ARROW.CHANGE SCRNELTS))
                                  (DASHING (READ.DASHING.CHANGE))
                                  (BRUSHCOLOR [READ.COLOR.CHANGE "Change curve color how?" NIL
                                                     (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                                   (fetch (SCREENELT GLOBALPART)
                                                                                      of (CAR SCRNELTS))
                                                                                   'BRUSH])
                                  (ADDPOINT (READ.POINT.TO.ADD (CAR SCRNELTS)
                                                   SKW))
                                  NIL))
          (RETURN (AND HOW (LIST ASPECT HOW])
```

(OPENCURVE.TRANSFORMFN

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)                         (* rrb "19-Mar-86 17:40")

          (* returns a copy of the global OPENCURVE element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (KNOT.SET.SCALE.FIELD (create GLOBALPART using GELT INDIVIDUALGLOBALPART _
                                                                       (SET.OPENCURVE.ARROWHEAD.POINTS
                                                                        (create OPENCURVE
                                                                           using INDVPART LATLONKNOTS _
                                                                                 (SK.TRANSFORM.POINT.LIST
                                                                                  (fetch (OPENCURVE LATLONKNOTS)
                                                                                     of INDVPART)
                                                                                  TRANSFORMFN TRANSFORMDATA)
                                                                                 BRUSH _ (SK.TRANSFORM.BRUSH
                                                                                          (fetch (OPENCURVE BRUSH)
                                                                                             of INDVPART)
                                                                                          SCALEFACTOR)
                                                                                 CURVEARROWHEADS _
                                                                                 (SK.TRANSFORM.ARROWHEADS
                                                                                  (fetch (OPENCURVE CURVEARROWHEADS)
                                                                                     of INDVPART)
                                                                                  SCALEFACTOR)
                                                                                 OPENCURVEREGION _ NIL])
```

(OPENCURVE.TRANSLATEFN

```
  [LAMBDA (OCELT DELTAPOS)                                                     (* rrb "20-Mar-86 15:09")
                                                                               (* translates an opencurve element)
    (PROG ((NEWOCELT (KNOTS.TRANSLATEFN OCELT DELTAPOS)))
          (SET.OPENCURVE.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWOCELT))
          (RETURN NEWOCELT])
```

```
(OPENCURVE.TRANSLATEPTSFN
  [LAMBDA (KNOTELT SELPTS GDELTA WINDOW)                          (* rrb " 5-May-85 17:49")

            (* returns a curve element which has the knots that are members of SELPTS translated by the global amount GDELTA.)

    (PROG ((GKNOTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of KNOTELT)))
          (RETURN (SK.CURVE.CREATE (for PT in (fetch (LOCALCURVE KNOTS) of (fetch (SCREENELT LOCALPART)
                                                                                of KNOTELT))
                                      as LATLONPT in (fetch LATLONKNOTS of GKNOTELT)
                                      collect (COND
                                                 ((MEMBER PT SELPTS)
                                                  (PTPLUS LATLONPT GDELTA))
                                                 (T LATLONPT)))
                         NIL
                         (fetch (OPENCURVE BRUSH) of GKNOTELT)
                         (fetch (OPENCURVE DASHING) of GKNOTELT)
                         (fetch (OPENCURVE OPENCURVEINITSCALE) of GKNOTELT)
                         (fetch (OPENCURVE CURVEARROWHEADS) of GKNOTELT])

(SKETCH.CREATE.CLOSED.CURVE
  [LAMBDA (POINTS BRUSH DASHING WILLBEFILLING SCALE)              (* rrb "16-Oct-85 17:15")
                                                                  (* creates a sketch closed curve element.)
    (SK.CURVE.CREATE (SK.INSURE.POINT.LIST POINTS)
           T
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0])

(CLOSEDCURVE.DRAWFN
  [LAMBDA (CURVEELT WINDOW REGION)                                (* rrb " 7-Dec-85 20:45")
                                                                  (* draws a curve figure element.)
    (PROG ((LCURVE (fetch (SCREENELT LOCALPART) of CURVEELT)))    (* make sure this curve might be in the REGION of interest.)
          (AND REGION (NOT (REGIONSINTERSECTP REGION (SK.ITEM.REGION CURVEELT)))
               (RETURN))
          (DRAWCURVE (fetch (LOCALCLOSEDCURVE LOCALCLOSEDCURVEKNOTS) of LCURVE)
                 T
                 (fetch (LOCALCLOSEDCURVE LOCALCLOSEDCURVEBRUSH) of LCURVE)
                 (fetch (LOCALCLOSEDCURVE LOCALCLOSEDCURVEDASHING) of LCURVE)
                 WINDOW])

(CLOSEDCURVE.EXPANDFN
  [LAMBDA (GELT SCALE)                                            (* rrb " 7-Dec-85 20:45")

            (* returns a local record which has the LATLONKNOTS field of the global element GELT translated into window coordinats.
            Used for curves and wires.)

    (PROG ((INDVKNOTELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           BRSH)
          [COND
             ((fetch (CLOSEDCURVE CLOSEDCURVEINITSCALE) of INDVKNOTELT))
             (T                                                   (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GELT
                   with (SETQ INDVKNOTELT (create CLOSEDCURVE using INDVKNOTELT CLOSEDCURVEINITSCALE _ 1.0
                                                                  CLOSEDCURVEREGION _ NIL]
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALCLOSEDCURVE
                                            LOCALCLOSEDCURVEKNOTS _ (for LATLONPT
                                                                       in (fetch LATLONKNOTS of INDVKNOTELT)
                                                                       collect (SK.SCALE.POSITION.INTO.VIEWER
                                                                                 LATLONPT SCALE))
                                            LOCALCLOSEDCURVEBRUSH _
                                            (SCALE.BRUSH (COND
                                                            ([NOT (NUMBERP (SETQ BRSH (fetch (CLOSEDCURVE BRUSH)
                                                                                         of INDVKNOTELT]
                                                                  (* new format, old format had brush width only.)
                                                             BRSH)
                                                            (T [replace (CLOSEDCURVE BRUSH) of INDVKNOTELT
                                                                  with (SETQ BRSH
                                                                        (create BRUSH
                                                                               BRUSHSIZE _ BRSH
                                                                               BRUSHSHAPE _ 'ROUND]
                                                               BRSH))
                                                   (fetch (CLOSEDCURVE CLOSEDCURVEINITSCALE) of INDVKNOTELT)
                                                   SCALE)
                                            LOCALCLOSEDCURVEFILLING _ (APPEND (fetch (CLOSEDCURVE
                                                                                            CLOSEDCURVEFILLING)
                                                                                 of INDVKNOTELT))
                                            LOCALCLOSEDCURVEDASHING _ (fetch (CLOSEDCURVE DASHING) of INDVKNOTELT)
                                            )
                         GLOBALPART _ GELT])
```

```
(CLOSEDCURVE.REGIONFN
  [LAMBDA (KNOTSCRELT)                                    (* rrb " 2-Dec-85 20:40")
                                                          (* returns the region occuppied by a list of knots which represent
    a curve.)

          (* uses the heuristic that the region containing the curve is not more than 20% larger than the knots.
          This was determined empirically on several curves.)

    (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (fetch (SCREENELT HOTSPOTS) of KNOTSCRELT))
                           1.4)
           (IQUOTIENT [ADD1 (SK.BRUSH.SIZE (fetch (LOCALCLOSEDCURVE LOCALCLOSEDCURVEBRUSH)
                                             of (fetch (SCREENELT LOCALPART) of KNOTSCRELT]
                  2])

(CLOSEDCURVE.GLOBALREGIONFN
  [LAMBDA (GCLOSEDCURVEELT)                               (* rrb "18-Oct-85 16:37")
                                                          (* returns the global region occupied by a global closed curve
                                                          element.)
    (OR (fetch (CLOSEDCURVE CLOSEDCURVEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCLOSEDCURVEELT))
        (PROG ((INDVCLOSEDCURVE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCLOSEDCURVEELT))
               REGION)

          (* uses the heuristic that the region containing the curve is not more than 40% larger than the knots.
          This was determined empirically on several curves.)

              [SETQ REGION (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (fetch (CLOSEDCURVE
                                                                                        LATLONKNOTS)
                                                                                  of INDVCLOSEDCURVE))
                                                  1.4)
                                  (SK.BRUSH.SIZE (fetch (CLOSEDCURVE BRUSH) of INDVCLOSEDCURVE]
              (replace (CLOSEDCURVE CLOSEDCURVEREGION) of INDVCLOSEDCURVE with REGION)
              (RETURN REGION])

(READ.LIST.OF.POINTS
  [LAMBDA (W ALLOWDUPS?)                                  (* rrb "10-Jun-86 15:43")
                                                          (* reads a spline {series of points} from the user.)
    (PROG (PT PTS ERRSTAT)
          (STATUSPRINT W "
               " "Enter the points the curve goes through using the left button.
               Click outside the window to stop.")
      LP  (COND
             ((AND [SETQ ERRSTAT (ERSETQ (SETQ PT (SK.READ.POINT.WITH.FEEDBACK W POINTREADINGCURSOR NIL NIL NIL
                                                        NIL (AND SKETCH.USE.POSITION.PAD 'MULTIPLE]
                   PT)                                    (* add the point to the list and mark it.)
              [COND
                 ([OR ALLOWDUPS? (NOT (EQUAL (fetch (INPUTPT INPUT.POSITION) of (CAR (LAST PTS)))
                                             (fetch (INPUTPT INPUT.POSITION) of PT]
                  (SHOWSKETCHPOINT (fetch (INPUTPT INPUT.POSITION) of PT)
                         W PTS)
                  (SETQ PTS (NCONC1 PTS PT]
              (GO LP)))                                   (* erase point markers.)
          (for PTTAIL on PTS do (SHOWSKETCHPOINT (fetch (INPUTPT INPUT.POSITION) of (CAR PTTAIL))
                                       W
                                       (CDR PTTAIL)))
          (CLOSEPROMPTWINDOW W)
          (CLRPROMPT)
          (COND
             (ERRSTAT                                     (* no error.)
                    (RETURN PTS))
             (T                                           (* had an error, pass it on)
                (ERROR!])

(CLOSEDCURVE.INPUTFN
  [LAMBDA (W)                                             (* rrb " 4-Sep-85 15:49")
                                                          (* reads a spline {series of points} from the user.)
    (SK.CURVE.CREATE (for PT in (READ.LIST.OF.POINTS W T) collect (SK.MAP.INPUT.PT.TO.GLOBAL PT W))
           T
           (fetch (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP W 'SKETCHCONTEXT))
           (fetch (SKETCHCONTEXT SKETCHDASHING) of (WINDOWPROP W 'SKETCHCONTEXT))
           (SK.INPUT.SCALE W])

(CLOSEDCURVE.READCHANGEFN
  [LAMBDA (SKW SCRNELTS)                                  (* rrb "20-Nov-85 11:09")
                                                          (* changefn for curves)
    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                         CENTERFLG _ T
                                                                         TITLE _ "select aspect of brush to
                                                                         change"
                                                                         ITEMS _
                                                                         (APPEND
                                                                          (COND
                                                                             [(SKETCHINCOLORP)
                                                                         '(("Color" 'BRUSHCOLOR "changes
                                                                                   the color of the brush"]
                                                                          (T NIL))
                                                                      '((Shape 'SHAPE "changes the shape of
                                                                              the brush")
                                                                        (Size 'SIZE "changes the size of the
                                                                             brush")
                                                                        (Dashing 'DASHING "changes the
                                                                              dashing of the line.")
                                                                        ("Add point" 'ADDPOINT "adds a point
                                                                              to the curve."]
                          (SIZE (READSIZECHANGE "Change size how?"))
                          (SHAPE (READBRUSHSHAPE))
                          (DASHING (READ.DASHING.CHANGE))
                          (BRUSHCOLOR [READ.COLOR.CHANGE "Change brush color how?" NIL
                                          (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                        (fetch (SCREENELT GLOBALPART)
                                                                           of (CAR SCRNELTS))
                                                                        'BRUSH])
                          (ADDPOINT (READ.POINT.TO.ADD (CAR SCRNELTS)
                                           SKW))
                          NIL))
              (RETURN (AND HOW (LIST ASPECT HOW])
```

(CLOSEDCURVE.TRANSFORMFN

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "18-Oct-85 16:52")

              (* returns a copy of the global CLOSEDCURVE element that has had each of its control points transformed by transformfn.
              TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (KNOT.SET.SCALE.FIELD (create GLOBALPART using GELT INDIVIDUALGLOBALPART _
                                                                  (create CLOSEDCURVE
                                                                     using INDVPART LATLONKNOTS _
                                                                           (SK.TRANSFORM.POINT.LIST
                                                                            (fetch (CLOSEDCURVE LATLONKNOTS)
                                                                               of INDVPART)
                                                                            TRANSFORMFN TRANSFORMDATA)
                                                                           BRUSH _ (SK.TRANSFORM.BRUSH
                                                                                    (fetch (CLOSEDCURVE BRUSH)
                                                                                       of INDVPART)
                                                                                    SCALEFACTOR)
                                                                           CLOSEDCURVEREGION _ NIL])
```

(CLOSEDCURVE.TRANSLATEPTSFN

```
  [LAMBDA (KNOTELT SELPTS GDELTA WINDOW)                        (* rrb " 5-May-85 18:35")

              (* returns a closed curve element which has the knots that are members of SELPTS translated by the global amount
              GDELTA.)

    (PROG ((GKNOTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of KNOTELT)))
          (RETURN (SK.CURVE.CREATE (for PT in (fetch (LOCALCURVE KNOTS) of (fetch (SCREENELT LOCALPART)
                                                                                   of KNOTELT))
                                      as LATLONPT in (fetch LATLONKNOTS of GKNOTELT)
                                      collect (COND
                                                 ((MEMBER PT SELPTS)
                                                  (PTPLUS LATLONPT GDELTA))
                                                 (T LATLONPT)))
                          T
                          (fetch (CLOSEDCURVE BRUSH) of GKNOTELT)
                          (fetch (CLOSEDCURVE DASHING) of GKNOTELT)
                          (fetch (CLOSEDCURVE CLOSEDCURVEINITSCALE) of GKNOTELT)
                          NIL])
```

(INVISIBLEPARTP

```
  [LAMBDA (WINDOW POINT)                                        (* rrb "30-NOV-82 17:25")
                                                                (* determines if POINT is in the visible part of a window.)
    (INSIDE? (DSPCLIPPINGREGION NIL WINDOW)
           (fetch (POSITION XCOORD) of POINT)
           (fetch (POSITION YCOORD) of POINT])
```

(SHOWSKETCHPOINT

```
  [LAMBDA (NEWPT W PTS)                                         (* rrb "12-May-85 18:50")

              (* puts down the marker for a new point unless it is already a member of points.)

    (OR (MEMBER NEWPT PTS)
        (MARKPOINT NEWPT W SPOTMARKER])
```

(SHOWSKETCHXY

```
  [LAMBDA (X Y WINDOW)                                          (* rrb " 2-Oct-85 09:58")
                                                                (* puts down a marker for a point at position X,Y)
```

```
    (BITBLT SPOTMARKER NIL NIL WINDOW (IDIFFERENCE X (LRSH (fetch (BITMAP BITMAPWIDTH) of SPOTMARKER)
                                                            1))
           (IDIFFERENCE Y (LRSH (fetch (BITMAP BITMAPHEIGHT) of SPOTMARKER)
                                1))
           NIL NIL 'INPUT 'INVERT])
```

```
(KNOTS.REGIONFN
  [LAMBDA (KNOTSCRELT)                                      (* rrb "29-May-85 21:17")
                                                            (* returns the region occuppied by a list of knots)

          (* increase by half the brush size plus 2 This has the nice property of insuring that the region always has both height and
          width.)

    (INCREASEREGION (REGION.CONTAINING.PTS (fetch (SCREENELT HOTSPOTS) of KNOTSCRELT))
           (IPLUS 3 (QUOTIENT (fetch (BRUSH BRUSHSIZE) of (fetch (LOCALWIRE LOCALOPENWIREBRUSH)
                                                              of (fetch (SCREENELT LOCALPART) of KNOTSCRELT)))
                           2])

(OPENWIRE.GLOBALREGIONFN
  [LAMBDA (GOPENWIREELT)                                    (* rrb "23-Oct-85 11:30")
                                                            (* returns the global region occupied by a global open curve
                                                            element.)
    (OR (fetch (WIRE OPENWIREREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GOPENWIREELT))
        (PROG ((INDVOPENWIRE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GOPENWIREELT))
               REGION)
              [SETQ REGION (INCREASEREGION (REGION.CONTAINING.PTS (fetch (WIRE LATLONKNOTS) of INDVOPENWIRE))
                                  (SK.BRUSH.SIZE (fetch (WIRE BRUSH) of INDVOPENWIRE]
              (replace (WIRE OPENWIREREGION) of INDVOPENWIRE with REGION)
              (RETURN REGION])

(CURVE.REGIONFN
  [LAMBDA (OPENCURVESCRELT)                                 (* rrb "18-Oct-85 16:36")
                                                            (* returns the region occuppied by a list of knots which represent
    a curve.)

          (* uses the heuristic that the region containing the curve is not more than 40% larger than the knots.
          This was determined empirically on several curves.)

    (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (fetch (SCREENELT HOTSPOTS) of OPENCURVESCRELT))
                           1.4)
           (IQUOTIENT [ADD1 (SK.BRUSH.SIZE (fetch (LOCALCURVE LOCALCURVEBRUSH) of (fetch (SCREENELT LOCALPART)
                                                                                       of OPENCURVESCRELT]
                  2])

(OPENCURVE.GLOBALREGIONFN
  [LAMBDA (GOPENCURVEELT)                                   (* rrb "18-Oct-85 16:36")
                                                            (* returns the global region occupied by a global open curve
                                                            element.)
    (OR (fetch (OPENCURVE OPENCURVEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GOPENCURVEELT))
        (PROG ((INDVOPENCURVE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GOPENCURVEELT))
               REGION)

          (* uses the heuristic that the region containing the curve is not more than 40% larger than the knots.
          This was determined empirically on several curves.)

              [SETQ REGION (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (fetch (OPENCURVE LATLONKNOTS)
                                                                                   of INDVOPENCURVE))
                                                  1.4)
                                  (SK.BRUSH.SIZE (fetch (OPENCURVE BRUSH) of INDVOPENCURVE]
              (replace (OPENCURVE OPENCURVEREGION) of INDVOPENCURVE with REGION)
              (RETURN REGION])

(KNOTS.TRANSLATEFN
  [LAMBDA (SKELT DELTAPOS)                                  (* rrb " 4-Apr-86 11:31")

          (* replaces the knots field of the global part of a screen element with knots that have been translated DELTAPOS.)

    (PROG [(GKNOTELT (APPEND (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKELT]
          (replace (KNOTELT LATLONKNOTS) of GKNOTELT with (for PT in (fetch (KNOTELT LATLONKNOTS) of GKNOTELT)
                                                             collect (PTPLUS PT DELTAPOS)))
                                                            (* clear the region cache.)
          (replace (KNOTELT KNOTREGION) of GKNOTELT with NIL)
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of SKELT))
                         INDIVIDUALGLOBALPART _ GKNOTELT])

(REGION.CONTAINING.PTS
  [LAMBDA (PTLST)                                           (* rrb " 7-Sep-84 11:26")
                                                            (* returns the region that contains all of the points on PTLST.)
    (AND PTLST (PROG ((XMIN (fetch (POSITION XCOORD) of (CAR PTLST)))
                      (XMAX (fetch (POSITION XCOORD) of (CAR PTLST)))
```

```
                        (YMIN (fetch (POSITION YCOORD) of (CAR PTLST)))
                        (YMAX (fetch (POSITION YCOORD) of (CAR PTLST)))
                        TMP)
                       [for PT in (CDR PTLST) do (COND
                                                    ((GREATERP (SETQ TMP (fetch (POSITION XCOORD) of PT))
                                                             XMAX)
                                                     (SETQ XMAX TMP))
                                                    ((GREATERP XMIN TMP)
                                                     (SETQ XMIN TMP)))
                                                 (COND
                                                    ((GREATERP (SETQ TMP (fetch (POSITION YCOORD) of PT))
                                                             YMAX)
                                                     (SETQ YMAX TMP))
                                                    ((GREATERP YMIN TMP)
                                                     (SETQ YMIN TMP]
                       (RETURN (CREATEREGION XMIN YMIN (DIFFERENCE XMAX XMIN)
                                      (DIFFERENCE YMAX YMIN])

)

(DEFINEQ

(CHANGE.ELTS.BRUSH.SIZE
  [LAMBDA (HOWTOCHANGE ELTSWITHBRUSH SKW)                        (* rrb "10-Jan-85 14:00")

          (* * function that prompts for how the line thickness should change and changes it for all elements in ELTSWITHBRUSH
          that have a brush size or thickness.)
                                                                 (* knows about the various types of sketch elements types and
                                                                 shouldn't.)
    (AND HOWTOCHANGE (for LINEDELT in ELTSWITHBRUSH collect (SK.CHANGE.BRUSH.SIZE LINEDELT HOWTOCHANGE SKW])

(CHANGE.ELTS.BRUSH
  [LAMBDA (CURVELTS SKW HOW)                                     (* rrb " 4-Jan-85 14:55")
                                                                 (* changefn for curves Actually makes the change.)
    (SELECTQ (CAR HOW)
         (SIZE (CHANGE.ELTS.BRUSH.SIZE (CADR HOW)
                      CURVELTS SKW))
         (SHAPE (CHANGE.ELTS.BRUSH.SHAPE (CADR HOW)
                       CURVELTS SKW))
         NIL])

(CHANGE.ELTS.BRUSH.SHAPE
  [LAMBDA (NEWSHAPE CURVELTS SKW)                                (* rrb "10-Jan-85 16:49")

          (* changes the brush shape of a list of curve elements. Knows about the various sketch element types and shouldn't need
          to.)

    (AND NEWSHAPE (for CURVELT in CURVELTS collect (SK.CHANGE.BRUSH.SHAPE CURVELT NEWSHAPE SKW])

(SK.CHANGE.BRUSH.SHAPE
  [LAMBDA (ELTWITHBRUSH HOW SKW)                                 (* rrb "10-Mar-86 16:07")
                                                                 (* changes the brush shape in the element ELTWITHBRUSH.)
    (PROG (GCURVELT BRUSH TYPE NEWELT NEWBRUSH)
          (RETURN (COND
                     ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHBRUSH))
                            '(CLOSEDCURVE OPENCURVE ELLIPSE CIRCLE ARC CLOSEDWIRE WIRE))
                                                                 (* only works for things of curve type.)
                      (SETQ GCURVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHBRUSH))
                      (SETQ BRUSH (SELECTQ TYPE
                                      (CIRCLE (fetch (CIRCLE BRUSH) of GCURVELT))
                                      (ARC (fetch (ARC ARCBRUSH) of GCURVELT))
                                      (ELLIPSE (fetch (ELLIPSE BRUSH) of GCURVELT))
                                      (WIRE (fetch (WIRE BRUSH) of GCURVELT))
                                      (CLOSEDWIRE (fetch (CLOSEDWIRE BRUSH) of GCURVELT))
                                      (fetch (OPENCURVE BRUSH) of GCURVELT)))
                      (COND
                         ((NEQ HOW (fetch (BRUSH BRUSHSHAPE) of BRUSH))
                                                                 (* new brush shape)
                          (SETQ NEWBRUSH (create BRUSH using BRUSH BRUSHSHAPE _ HOW))
                          (SETQ NEWELT (SELECTQ TYPE
                                           (CLOSEDCURVE (create CLOSEDCURVE using GCURVELT BRUSH _ NEWBRUSH))
                                           (OPENCURVE (create OPENCURVE using GCURVELT BRUSH _ NEWBRUSH))
                                           (CIRCLE (create CIRCLE using GCURVELT BRUSH _ NEWBRUSH))
                                           (ARC (create ARC using GCURVELT ARCBRUSH _ NEWBRUSH))
                                           (ELLIPSE (create ELLIPSE using GCURVELT BRUSH _ NEWBRUSH))
                                           (WIRE (create WIRE using GCURVELT BRUSH _ NEWBRUSH))
                                           (CLOSEDWIRE (create CLOSEDWIRE using GCURVELT BRUSH _ NEWBRUSH))
                                           (SHOULDNT)))
                          (create SKHISTORYCHANGESPEC
                                 OLDELT _ ELTWITHBRUSH
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHBRUSH)
                                                 INDIVIDUALGLOBALPART _ NEWELT)
```

```
                                PROPERTY _ 'BRUSH
                                NEWVALUE _ NEWBRUSH
                                OLDVALUE _ BRUSH])
```

(**SK.CHANGE.BRUSH.COLOR**

```
  [LAMBDA (ELTWITHLINE COLOR SKW)                                   (* rrb " 8-Jan-86 17:25")
                                                                    (* changes the brush color of ELTWITHLINE if it has a brush)
                                                                    (* knows about the various types of sketch elements types and
                                                                    shouldn't.)
    (PROG ((GLINELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHLINE))
           TYPE BRUSH NEWBRUSH 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.)
                                                                    (* the brush is stored in the different place for all element types.)
              (SETQ BRUSH (SELECTQ TYPE
                                (CIRCLE (fetch (CIRCLE BRUSH) of GLINELT))
                                (ELLIPSE (fetch (ELLIPSE BRUSH) of GLINELT))
                                (TEXTBOX (fetch (TEXTBOX TEXTBOXBRUSH) of GLINELT))
                                (CLOSEDCURVE (fetch (CLOSEDCURVE BRUSH) of GLINELT))
                                (BOX (fetch (BOX BRUSH) of GLINELT))
                                (ARC (fetch (ARC ARCBRUSH) of GLINELT))
                                (fetch (OPENCURVE BRUSH) of GLINELT)))
              (COND
                 ((NOT (EQUAL COLOR (fetch (BRUSH BRUSHCOLOR) of BRUSH)))
                  (SETQ NEWBRUSH (create BRUSH using BRUSH BRUSHCOLOR _ COLOR))
                  (SETQ NEWELT (SELECTQ TYPE
                                     (WIRE (create WIRE using GLINELT BRUSH _ NEWBRUSH))
                                     (BOX (create BOX using GLINELT BRUSH _ NEWBRUSH))
                                     (ARC (create ARC using GLINELT ARCBRUSH _ NEWBRUSH))
                                     (TEXTBOX (create TEXTBOX using GLINELT TEXTBOXBRUSH _ NEWBRUSH TEXTCOLOR _
                                                     COLOR))
                                     (CLOSEDWIRE (create CLOSEDWIRE using GLINELT BRUSH _ NEWBRUSH))
                                     (CLOSEDCURVE (create CLOSEDCURVE using GLINELT BRUSH _ NEWBRUSH))
                                     (OPENCURVE (create OPENCURVE using GLINELT BRUSH _ NEWBRUSH))
                                     (CIRCLE (create CIRCLE using GLINELT BRUSH _ NEWBRUSH))
                                     (ELLIPSE (create ELLIPSE using GLINELT BRUSH _ NEWBRUSH))
                                     (SHOULDNT)))
                  (RETURN (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHLINE)
                                                 INDIVIDUALGLOBALPART _ NEWELT)
                                 OLDELT _ ELTWITHLINE
                                 PROPERTY _ 'BRUSH
                                 NEWVALUE _ NEWBRUSH
                                 OLDVALUE _ BRUSH]
             ((EQ TYPE 'TEXT)                                       (* change the color of text too.)
              (COND
                 ((NOT (EQUAL COLOR (fetch (TEXT TEXTCOLOR) of GLINELT)))
                  (RETURN (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHLINE)
                                                 INDIVIDUALGLOBALPART _ (create TEXT using GLINELT TEXTCOLOR _
                                                                               COLOR))
                                 OLDELT _ ELTWITHLINE
                                 PROPERTY _ 'TEXTCOLOR
                                 NEWVALUE _ COLOR
                                 OLDVALUE _ (fetch (TEXT TEXTCOLOR) of GLINELT])
```

(**SK.CHANGE.BRUSH.SIZE**

```
  [LAMBDA (ELTWITHLINE HOW SKW)                                     (* rrb "10-Jan-86 13:57")

          (* changes the line size of ELTWITHLINE if it has a brush size or thickness and returns a change event.)
                                                                    (* knows about the various types of sketch elements types and
                                                                    shouldn't.)
    (PROG (SIZE GLINELT TYPE BRUSH NEWBRUSH NEWELT)
          (COND
             ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHLINE))
                    '(WIRE BOX CLOSEDWIRE CLOSEDCURVE OPENCURVE CIRCLE ELLIPSE TEXTBOX ARC))
              (SETQ GLINELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHLINE))
              (SETQ BRUSH (SELECTQ TYPE
                                (CIRCLE (fetch (CIRCLE BRUSH) of GLINELT))
                                (ELLIPSE (fetch (ELLIPSE BRUSH) of GLINELT))
                                (TEXTBOX (fetch (TEXTBOX TEXTBOXBRUSH) of GLINELT))
                                (CLOSEDCURVE (fetch (CLOSEDCURVE BRUSH) of GLINELT))
                                (BOX (fetch (BOX BRUSH) of GLINELT))
                                (ARC (fetch (ARC ARCBRUSH) of GLINELT))
                                (fetch (OPENCURVE BRUSH) of GLINELT)))

          (* the change to the brush size must take into account the current scale and the scale at which the brush was entered.)

              (COND
                 ((GEQ [SETQ SIZE (COND
                                    ((NUMBERP HOW)
                                     HOW)
                                    (T (SELECTQ HOW
                                           (SMALLER (FQUOTIENT (fetch (BRUSH BRUSHSIZE) of BRUSH)
                                                           2.0))
                                           (FTIMES (fetch (BRUSH BRUSHSIZE) of BRUSH)
                                                  2.0]
                       0)                                    (* don't let the brush size go negative.)
                  (SETQ NEWBRUSH (create BRUSH using BRUSH BRUSHSIZE _ SIZE))
                  (SETQ NEWELT (SELECTQ TYPE
                                   (WIRE (create WIRE using GLINELT BRUSH _ NEWBRUSH OPENWIREREGION _ NIL))
                                   (BOX (create BOX using GLINELT BRUSH _ NEWBRUSH))
                                   (ARC (create ARC using GLINELT ARCBRUSH _ NEWBRUSH ARCREGION _ NIL))
                                   (TEXTBOX

            (* since this may change the location of characters in the box, clear the selection.
            Probably should happen somewhere else.)

                                           (SKED.CLEAR.SELECTION SKW)
                                           (create TEXTBOX using GLINELT TEXTBOXBRUSH _ NEWBRUSH))
                                   (CLOSEDWIRE (create CLOSEDWIRE using GLINELT BRUSH _ NEWBRUSH CLOSEDWIREREGION
                                                                        _ NIL))
                                   (CLOSEDCURVE (create CLOSEDCURVE using GLINELT BRUSH _ NEWBRUSH
                                                                          CLOSEDCURVEREGION _ NIL))
                                   (OPENCURVE (create OPENCURVE using GLINELT BRUSH _ NEWBRUSH OPENCURVEREGION _
                                                                      NIL))
                                   (CIRCLE (create CIRCLE using GLINELT BRUSH _ NEWBRUSH CIRCLEREGION _ NIL))
                                   (ELLIPSE (create ELLIPSE using GLINELT BRUSH _ NEWBRUSH ELLIPSEREGION _ NIL))
                                   (SHOULDNT)))
                  (RETURN (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                  COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                        of ELTWITHLINE)
                                                  INDIVIDUALGLOBALPART _ NEWELT)
                                 OLDELT _ ELTWITHLINE
                                 PROPERTY _ 'BRUSH
                                 NEWVALUE _ NEWBRUSH
                                 OLDVALUE _ BRUSH])
```

(SK.CHANGE.ANGLE

```
  [LAMBDA (ELTWITHARC HOW SKW)                               (* rrb "20-Jun-86 17:18")
                                                             (* changes the arc size of ELTWITHARC if it is an arc element)
    (PROG (GARCLT ARMANGLE RADIUS CENTERPT RADIUSPT CENTERX NEWANGLEPT CENTERY)
          (COND
             ((EQ (fetch (GLOBALPART GTYPE) of ELTWITHARC)
                  'ARC)                                      (* only works for things of arc type.)
              (SETQ GARCLT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHARC))
              (SETQ CENTERPT (fetch (ARC ARCCENTERPT) of GARCLT))
              (SETQ CENTERX (fetch (POSITION XCOORD) of CENTERPT))
              (SETQ CENTERY (fetch (POSITION YCOORD) of CENTERPT))
              (SETQ RADIUSPT (fetch (ARC ARCRADIUSPT) of GARCLT))
              [SETQ ARMANGLE (COND
                                ((fetch (ARC ARCDIRECTION) of GARCLT)
                                                             (* clockwise direction)
                                 (DIFFERENCE (SK.COMPUTE.SLOPE.OF.LINE CENTERPT RADIUSPT)
                                        HOW))
                                (T                           (* positive direction)
                                   (PLUS (SK.COMPUTE.SLOPE.OF.LINE CENTERPT RADIUSPT)
                                         HOW]
              (SETQ RADIUS (DISTANCEBETWEEN CENTERPT RADIUSPT))
                                                             (* calculate a position on the circle the right number of degrees
                                                             out.)
              [SETQ NEWANGLEPT (COND
                                  ((OR (GEQ ARMANGLE 360.0)
                                       (LEQ ARMANGLE -360.0))     (* mark greater than 360 by T)
                                   T)
                                  (T (create POSITION
                                            XCOORD _ [FIXR (PLUS CENTERX (TIMES RADIUS (COS ARMANGLE]
                                            YCOORD _ (FIXR (PLUS CENTERY (TIMES RADIUS (SIN ARMANGLE]
              (RETURN (create SKHISTORYCHANGESPEC
                             NEWELT _ (create GLOBALPART
                                              COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART) of ELTWITHARC
                                                                   )
                                              INDIVIDUALGLOBALPART _
                                              (SET.ARC.ARROWHEAD.POINTS (create ARC
                                                                          using GARCLT ARCANGLEPT _ NEWANGLEPT
                                                                                ARCREGION _ NIL)))
                             OLDELT _ ELTWITHARC
                             PROPERTY _ '3RDCONTROLPT
                             NEWVALUE _ NEWANGLEPT
                             OLDVALUE _ (fetch (ARC ARCRADIUSPT) of GARCLT])
```

(SK.CHANGE.ARC.DIRECTION

```
  [LAMBDA (ELTWITHARC HOW SKW)                               (* rrb "19-Mar-86 17:16")
```

```
                                                          (* changes the direction around the circle that the arc element
                                                          goes.)
    (PROG (GARCLT NOWDIRECTION)
          (COND
             ((EQ (fetch (GLOBALPART GTYPE) of ELTWITHARC)
                  'ARC)                                   (* only works for things of arc type.)
              (SETQ GARCLT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHARC))
              (SETQ NOWDIRECTION (fetch (ARC ARCDIRECTION) of GARCLT))
              (COND
                 ((OR (AND (EQ HOW 'CLOCKWISE)
                           (NULL NOWDIRECTION))
                      (AND (EQ HOW 'COUNTERCLOCKWISE)
                           NOWDIRECTION))                 (* spec calls for one direction and it is currently going the other.)
                  (RETURN (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHARC)
                                                 INDIVIDUALGLOBALPART _ (SET.ARC.ARROWHEAD.POINTS
                                                                         (create ARC using GARCLT ARCDIRECTION _
                                                                                           (NOT NOWDIRECTION)
                                                                                           ARCREGION _ NIL)))
                                 OLDELT _ ELTWITHARC
                                 PROPERTY _ 'DIRECTION
                                 NEWVALUE _ HOW
                                 OLDVALUE _ (COND
                                               (NOWDIRECTION 'CLOCKWISE)
                                               (T 'COUNTERCLOCKWISE])
```

(SK.SET.DEFAULT.BRUSH.SIZE

```
  [LAMBDA (NEWBRUSHSIZE SKW)                              (* rrb "12-Jan-85 10:13")
                                                          (* sets the default brush size to NEWBRUSHSIZE)
    (AND (NUMBERP NEWBRUSHSIZE)
         (replace (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP SKW 'SKETCHCONTEXT)
            with (create BRUSH using (fetch (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP SKW 'SKETCHCONTEXT))
                                     BRUSHSIZE _ NEWBRUSHSIZE])
```

(READSIZECHANGE

```
  [LAMBDA (MENUTITLE ALLOWZEROFLG)                        (* rrb "14-May-86 19:26")
                                                          (* interacts to get whether a line size should be increased or
                                                          decreased.)
    (PROG [(NEWVALUE (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                     TITLE _ MENUTITLE
                                                     ITEMS _ '(("smaller line" 'SMALLER "decreases the line
                                                                       thickness by 1.")
                                                               ("LARGER LINE" 'LARGER "increases the line
                                                                       thickness by 1.")
                                                               ("Set line size" 'SETSIZE "sets the line thickness
                                                                       to an entered value."))
                                                     CENTERFLG _ T]
          (RETURN (COND
                     ((EQ NEWVALUE 'SETSIZE)
                      (SETQ NEWVALUE (RNUMBER "Enter the new line thickness." NIL NIL NIL T T T T))
                      (COND
                         ((AND (NULL ALLOWZEROFLG)
                               (EQ NEWVALUE 0))
                          NIL)
                         ((GREATERP 0 NEWVALUE)           (* don't allow negative values)
                          (MINUS NEWVALUE))
                         (T NEWVALUE)))
                     (T NEWVALUE])
)

(DEFINEQ
```

(SK.CHANGE.ELEMENT.KNOTS

```
  [LAMBDA (ELTWITHKNOTS NEWKNOTS)                         (* rrb "19-Mar-86 17:50")
                                                          (* changes the knots in the element ELTWITHKNOTS)
    (PROG ((GCURVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHKNOTS))
           NEWELT)
          (SETQ NEWELT (SELECTQ (fetch (INDIVIDUALGLOBALPART GTYPE) of GCURVELT)
                           (CLOSEDCURVE (create CLOSEDCURVE using GCURVELT LATLONKNOTS _ NEWKNOTS))
                           (OPENCURVE (SET.OPENCURVE.ARROWHEAD.POINTS (create OPENCURVE
                                                                        using GCURVELT LATLONKNOTS _
                                                                              NEWKNOTS)))
                           (WIRE (SET.WIRE.ARROWHEAD.POINTS (create WIRE using GCURVELT LATLONKNOTS _ NEWKNOTS))
)
                           (CLOSEDWIRE (create CLOSEDWIRE using GCURVELT LATLONKNOTS _ NEWKNOTS))
                           (RETURN)))
          (RETURN (KNOT.SET.SCALE.FIELD (create GLOBALPART
                                               COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                     of ELTWITHKNOTS)
                                               INDIVIDUALGLOBALPART _ NEWELT])

)
```

```
(DEFINEQ

(SK.INSURE.POINT.LIST
  [LAMBDA (POINTLST)                                                        (* rrb "16-Oct-85 17:01")
                                                                            (* makes sure POINTLST is a list of positions.)
    (COND
       ((LISTP POINTLST)
        (AND (EVERY POINTLST (FUNCTION SK.INSURE.POSITION))
             POINTLST))
       (T (\ILLEGAL.ARG POINTLST])

(SK.INSURE.POSITION
  [LAMBDA (POSITION)                                                        (* rrb "16-Oct-85 17:02")
    (OR (POSITIONP POSITION)
        (\ILLEGAL.ARG POSITION])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(TYPERECORD KNOTELT (LATLONKNOTS BRUSH DASHING NIL NIL KNOTREGION))

(RECORD LOCALCURVE (KNOTS LOCALHOTREGION ARROWHEADPTS LOCALCURVEBRUSH LOCALCURVEDASHING))

(TYPERECORD OPENCURVE (LATLONKNOTS BRUSH DASHING CURVEARROWHEADS OPENCURVEINITSCALE OPENCURVEREGION
                             OPENCURVEARROWHEADPOINTS))

(TYPERECORD CLOSEDCURVE (LATLONKNOTS BRUSH DASHING CLOSEDCURVEINITSCALE CLOSEDCURVEFILLING CLOSEDCURVEREGION))

(RECORD LOCALCLOSEDCURVE (LOCALCLOSEDCURVEKNOTS LOCALCLOSEDCURVEHOTREGION LOCALCLOSEDCURVEBRUSH
                                LOCALCLOSEDCURVEFILLING LOCALCLOSEDCURVEDASHING))

(RECORD LOCALCLOSEDWIRE (KNOTS LOCALHOTREGION LOCALCLOSEDWIREBRUSH LOCALCLOSEDWIREFILLING))
)
)

(READVARS-FROM-STRINGS '(OPENCURVEICON CLOSEDCURVEICON)
       "({(READBITMAP)(20 12
       %"@@@@@@@@%"
       %"@L@@@@@@%"
       %"@L@@F@@@%"
       %"AL@@O@@@%"
       %"AH@@G@@@%"
       %"CH@@C@@@%"
       %"CH@@C@@@%"
       %"CH@@G@@@%"
       %"AN@@N@@@%"
       %"@OCLN@@@%"
       %"@COOL@@@%"
       %"@@NCH@@@%")}  {(READBITMAP)(20 12
       %"@@C@@@@@%"
       %"ALGO@@@@%"
       %"CNLOL@@@%"
       %"GCLAN@@@%"
       %"FAAHF@@@%"
       %"L@CLC@@@%"
       %"N@CFC@@@%"
       %"F@FFG@@@%"
       %"C@FGF@@@%"
       %"CLFCL@@@%"
       %"AON@H@@@%"
       %"@GL@@@@@%")})
       ")

(RPAQ CURVE.KNOT (CURSORCREATE '
                         'NIL 0 8))

(DEFINEQ

(SKETCH.CREATE.WIRE
  [LAMBDA (POINTS BRUSH DASHING ARROWHEADS SCALE)                           (* rrb "16-Oct-85 17:05")
                                                                            (* creates a sketch wire element.)
    (SK.WIRE.CREATE (SK.INSURE.POINT.LIST POINTS)
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           NIL
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.ARROWHEADS ARROWHEADS)
           NIL])
```

```
(CLOSEDWIRE.EXPANDFN
  [LAMBDA (GELT SCALE)                                             (* rrb " 2-Dec-85 20:42")

            (* returns a local record which has the LATLONKNOTS field of the global element GELT translated into window coordinats.
            Used for closed wires.)

    (PROG ((INDVKNOTELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           BRSH)
          [COND
             ((fetch (CLOSEDWIRE CLOSEDWIREINITSCALE) of INDVKNOTELT))
             (T                                                    (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GELT
                   with (SETQ INDVKNOTELT (create CLOSEDWIRE using INDVKNOTELT CLOSEDWIREINITSCALE _ 1.0
                                                                   CLOSEDWIREREGION _ NIL]
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALCLOSEDWIRE
                                            KNOTS _ (for LATLONPT in (fetch LATLONKNOTS of INDVKNOTELT)
                                                       collect (SK.SCALE.POSITION.INTO.VIEWER LATLONPT SCALE))
                                            LOCALCLOSEDWIREBRUSH _
                                            (SCALE.BRUSH (COND
                                                            ([NOT (NUMBERP (SETQ BRSH (fetch (CLOSEDWIRE BRUSH)
                                                                                         of INDVKNOTELT]
                                                                   (* new format, old format had brush width only.)
                                                             BRSH)
                                                            (T [replace (CLOSEDWIRE BRUSH) of INDVKNOTELT
                                                                  with (SETQ BRSH
                                                                             (create BRUSH
                                                                                    BRUSHSIZE _ BRSH
                                                                                    BRUSHSHAPE _ 'ROUND]
                                                               BRSH))
                                                   (fetch (CLOSEDWIRE CLOSEDWIREINITSCALE) of INDVKNOTELT)
                                                   SCALE)
                                            LOCALCLOSEDWIREFILLING _ (APPEND (fetch (CLOSEDWIRE CLOSEDWIREFILLING
                                                                                           )
                                                                                of INDVKNOTELT)))
                         GLOBALPART _ GELT])

(KNOTS.INSIDEFN
  [LAMBDA (KNOTELT WREG)                                           (* rrb "21-Jan-87 09:37")
                                                                   (* determines if the global curve GCURVE is inside of WREG.)

            (* this should be broken down between wires and curves but isn't here so it can be loaded as a patch.)

    (SELECTQ (fetch (GLOBALPART GTYPE) of KNOTELT)
         (WIRE (REGIONSINTERSECTP WREG (OPENWIRE.GLOBALREGIONFN KNOTELT)))
         (CLOSEDWIRE (REGIONSINTERSECTP WREG (CLOSEDWIRE.GLOBALREGIONFN KNOTELT)))
         (CLOSEDCURVE (REGIONSINTERSECTP WREG (CLOSEDCURVE.GLOBALREGIONFN KNOTELT)))
         (REGIONSINTERSECTP WREG (OPENCURVE.GLOBALREGIONFN KNOTELT])

(OPEN.WIRE.DRAWFN
  [LAMBDA (OPENWIREELT WIN REG OPERATION)                          (* rrb " 7-Dec-85 20:11")
                                                                   (* draws an open wire element.)
    (WB.DRAWLINE OPENWIREELT WIN REG OPERATION NIL (fetch (LOCALWIRE LOCALWIREDASHING) of (fetch (SCREENELT
                                                                                                   LOCALPART)
                                                                                             of OPENWIREELT))
           (fetch (LOCALWIRE LOCALOPENWIREBRUSH) of (fetch (SCREENELT LOCALPART) of OPENWIREELT])

(WIRE.EXPANDFN
  [LAMBDA (GELT SCALE)                                             (* rrb " 2-May-86 10:50")

            (* returns a local record which has the LATLONKNOTS field of the global element GELT translated into window coordinats.
            Used for wires.)

    (PROG ((INDGELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           LOCALKNOTS TMP)
          [COND
             ((fetch (WIRE OPENWIREINITSCALE) of INDGELT))
             (T                                                    (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GELT
                   with (SETQ INDGELT (create WIRE using INDGELT OPENWIREINITSCALE _ 1.0 OPENWIREREGION _ NIL]
          (COND
             ((AND (fetch (WIRE WIREARROWHEADS) of INDGELT)
                   (NOT (fetch (WIRE OPENWIREARROWHEADPOINTS) of INDGELT)))
                                                                   (* old form didn't have global points, update it)
              (SET.WIRE.ARROWHEAD.POINTS INDGELT)))
          (SETQ LOCALKNOTS (for LATLONPT in (fetch (WIRE LATLONKNOTS) of INDGELT) collect (
                                                                                       SK.SCALE.POSITION.INTO.VIEWER
                                                                                          LATLONPT SCALE)))
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALWIRE
                                            KNOTS _ LOCALKNOTS
                                            ARROWHEADPTS _ (SK.EXPAND.ARROWHEADS (fetch (WIRE
                                                                                         OPENWIREARROWHEADPOINTS
                                                                                                )
                                                                                    of INDGELT)
                                                                      SCALE)
                                                LOCALOPENWIREBRUSH _
                                                (SCALE.BRUSH (COND
                                                               ([NOT (NUMBERP (SETQ TMP (fetch (WIRE BRUSH)
                                                                                                of INDGELT]
                                                                    (* new format, old format had brush width only.)
                                                                TMP)
                                                               (T [replace (WIRE BRUSH) of INDGELT
                                                                     with (SETQ TMP
                                                                                (create BRUSH
                                                                                        BRUSHSIZE _ TMP
                                                                                        BRUSHSHAPE _ 'ROUND]
                                                                  TMP))
                                                      (fetch (WIRE OPENWIREINITSCALE) of INDGELT)
                                                      SCALE)
                                                LOCALWIREDASHING _ (fetch (WIRE OPENWIREDASHING) of INDGELT))
                              GLOBALPART _ GELT])
```

(**SK.UPDATE.WIRE.ELT.AFTER.CHANGE**

```
  [LAMBDA (GWIRELT)                                                   (* rrb "11-Dec-85 11:27")
                                                                      (* updates the dependent fields of a wire element after one of
                                                                      the fields changes.)
                                                                      (* clear the region cache)
    (replace (OPENCURVE OPENCURVEREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GWIRELT) with NIL)
    (KNOT.SET.SCALE.FIELD GWIRELT])
```

(**OPENWIRE.READCHANGEFN**

```
  [LAMBDA (SKW WIREELTS)                                              (* rrb "17-Dec-85 16:22")

          (* * change function for line elements.)

    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                          CENTERFLG _ T
                                                                          TITLE _ "Which aspect?"
                                                                          ITEMS _
                                                                          (APPEND
                                                                           (COND
                                                                              [(SKETCHINCOLORP)
                                                                               '(("Brush color" 'BRUSHCOLOR
                                                                                        "changes the color of the
                                                                                        outline"]
                                                                              (T NIL))
                                                                           '((Arrowheads 'ARROW "allows changing
                                                                                    of arrow head
                                                                                    characteristics.")
                                                                             (Size 'SIZE "changes the size of the
                                                                                   brush")
                                                                             (Dashing 'DASHING "changes the
                                                                                    dashing of the line."]
                             (SIZE (READSIZECHANGE "Change size how?"))
                             (ARROW (READ.ARROW.CHANGE WIREELTS))
                             (DASHING (READ.DASHING.CHANGE))
                             (BRUSHCOLOR [READ.COLOR.CHANGE "Change line color how?" NIL
                                                (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                               (fetch (SCREENELT GLOBALPART)
                                                                                  of (CAR WIREELTS))
                                                                              'BRUSH])
                             NIL))
          (RETURN (AND HOW (LIST ASPECT HOW])
```

(**OPENWIRE.TRANSFORMFN**

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)                (* rrb "19-Mar-86 17:51")

          (* returns a copy of the global WIRE element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (KNOT.SET.SCALE.FIELD (create GLOBALPART using GELT INDIVIDUALGLOBALPART _
                                                                 (SET.WIRE.ARROWHEAD.POINTS
                                                                  (create WIRE using INDVPART LATLONKNOTS _
                                                                                  (SK.TRANSFORM.POINT.LIST
                                                                                   (fetch (WIRE LATLONKNOTS)
                                                                                      of INDVPART)
                                                                                   TRANSFORMFN TRANSFORMDATA)
                                                                                  BRUSH _
                                                                                  (SK.TRANSFORM.BRUSH
                                                                                   (fetch (WIRE BRUSH)
                                                                                      of INDVPART)
                                                                                   SCALEFACTOR)
                                                                                  WIREARROWHEADS _
                                                                                  (SK.TRANSFORM.ARROWHEADS
                                                                                   (fetch (WIRE WIREARROWHEADS)
                                                                                        of INDVPART)
                                                                                     SCALEFACTOR)
                                                                                    OPENWIREREGION _ NIL])

(OPENWIRE.TRANSLATEFN
  [LAMBDA (WIREELT DELTAPOS)                                        (* rrb "20-Mar-86 15:08")
                                                                    (* translates an open wire element)
    (PROG ((NEWWIREELT (KNOTS.TRANSLATEFN WIREELT DELTAPOS)))
          (SET.WIRE.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWWIREELT))
          (RETURN NEWWIREELT])

(OPENWIRE.TRANSLATEPTSFN
  [LAMBDA (KNOTELT SELPTS GDELTA WINDOW)                            (* rrb "26-Sep-85 17:45")

          (* returns an open wire element which has the knots that are members of SELPTS translated by the global amount
          GDELTA.)

    (PROG ((GKNOTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of KNOTELT)))
          (RETURN (SK.WIRE.CREATE (for PT in (fetch (LOCALWIRE KNOTS) of (fetch (SCREENELT LOCALPART) of KNOTELT))
                                     as LATLONPT in (fetch (WIRE LATLONKNOTS) of GKNOTELT)
                                     collect (COND
                                                ((MEMBER PT SELPTS)
                                                 (PTPLUS LATLONPT GDELTA))
                                                (T LATLONPT)))
                         (fetch (WIRE BRUSH) of GKNOTELT)
                         (fetch (WIRE OPENWIREDASHING) of GKNOTELT)
                         NIL
                         (fetch (WIRE OPENWIREINITSCALE) of GKNOTELT)
                         (fetch (WIRE WIREARROWHEADS) of GKNOTELT])

(WIRE.INPUTFN
  [LAMBDA (W GPTLIST CLOSEDFLG BRUSH DEFSCALE DASHING FILLING)    (* rrb "15-Nov-85 11:39")

          (* creates a wire {a series of straight lines through a list of points} from a list of points passed in or a list that is read from the
          user via mouse.)

    (PROG ((SKCONTEXT (WINDOWPROP W 'SKETCHCONTEXT))
           KNOTS)
          (RETURN (SK.WIRE.CREATE [SETQ KNOTS (OR GPTLIST (for PT in (SK.READ.WIRE.POINTS W CLOSEDFLG)
                                                                     collect (SK.MAP.INPUT.PT.TO.GLOBAL PT W]
                         (COND
                            ((NUMBERP BRUSH)

          (* called with a number from the sketch stream drawline operation.
          Make it a round brush.)

                             (create BRUSH
                                    BRUSHSIZE _ BRUSH
                                    BRUSHSHAPE _ 'ROUND))
                            (T (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKCONTEXT)))
                         (OR (DASHINGP DASHING)
                             (fetch (SKETCHCONTEXT SKETCHDASHING) of SKCONTEXT))
                         CLOSEDFLG
                         (OR (NUMBERP DEFSCALE)
                             (SK.INPUT.SCALE W))
                         (SK.ARROWHEAD.CREATE W KNOTS)
                         FILLING])

(SK.READ.WIRE.POINTS
  [LAMBDA (SKW CLOSEDFLG)                                           (* rrb "12-May-86 18:31")
                                                                    (* reads a list of points for a wire.)
    (SK.READ.POINTS.WITH.FEEDBACK SKW NIL (AND SKETCH.VERBOSE.FEEDBACK (COND
                                                                           (CLOSEDFLG (FUNCTION
                                                                                       CLOSEDWIRE.FEEDBACKFN))
                                                                           (T (FUNCTION OPENWIRE.FEEDBACKFN])

(SK.READ.POINTS.WITH.FEEDBACK
  [LAMBDA (W ALLOWDUPS? FEEDBACKFN)                                 (* rrb "10-Jun-86 15:44")
                                                                    (* reads a {series of points} from the user.)
    (PROG (PT PTS ERRSTAT)
          (STATUSPRINT W "
                " "Enter the points the curve goes through using the left button.
                Click outside the window to stop.")
      LP  (COND
             ((AND [SETQ ERRSTAT (ERSETQ (SETQ PT (SK.READ.POINT.WITH.FEEDBACK W POINTREADINGCURSOR FEEDBACKFN
                                                         PTS 'MIDDLE NIL (AND SKETCH.USE.POSITION.PAD
                                                                              'MULTIPLE]
                   PT)                                              (* add the point to the list and mark it.)
              [COND
                 ([OR ALLOWDUPS? (NOT (EQUAL (fetch (INPUTPT INPUT.POSITION) of (CAR (LAST PTS)))
                                             (fetch (INPUTPT INPUT.POSITION) of PT]
                  (SHOWSKETCHPOINT (fetch (INPUTPT INPUT.POSITION) of PT)
                   W PTS)

        (* draw the line so it will remain displayed while the user adds other points.
        This will not close it.)

             (AND PTS (DRAWBETWEEN (fetch (INPUTPT INPUT.POSITION) of (CAR (LAST PTS)))
                          (fetch (INPUTPT INPUT.POSITION) of PT)
                          1
                          'INVERT W))
             (SETQ PTS (NCONC1 PTS PT]
           (GO LP)))                                         (* erase point markers.)
        (for PTTAIL on PTS do (SHOWSKETCHPOINT (fetch (INPUTPT INPUT.POSITION) of (CAR PTTAIL))
                                 W
                                 (CDR PTTAIL))               (* erase line)
                          (AND (CDR PTTAIL)
                               (DRAWBETWEEN (fetch (INPUTPT INPUT.POSITION) of (CAR PTTAIL))
                                      (fetch (INPUTPT INPUT.POSITION) of (CADR PTTAIL))
                                      1
                                      'INVERT W)))
        (CLRPROMPT)
        (CLOSEPROMPTWINDOW W)
        (COND
           (ERRSTAT                                          (* no error.)
                (RETURN PTS))
           (T                                                (* had an error, pass it on)
              (ERROR!])

(OPENWIRE.FEEDBACKFN
  [LAMBDA (X Y WINDOW PREVPTS)                               (* rrb "15-Nov-85 11:32")
                                                             (* provides the rubberbanding feedback for the user inputting a
                                                             point for an open wire.)
    (SHOWSKETCHXY X Y WINDOW)
    (AND PREVPTS (PROG (LASTPT)
                      (RETURN (DRAWLINE [fetch (POSITION XCOORD) of (SETQ LASTPT (fetch (INPUTPT INPUT.POSITION)
                                                                                  of (CAR (LAST PREVPTS]
                                     (fetch (POSITION YCOORD) of LASTPT)
                                     X Y 1 'INVERT WINDOW])

(CLOSEDWIRE.FEEDBACKFN
  [LAMBDA (X Y WINDOW PREVPTS)                               (* rrb "15-Nov-85 11:31")
                                                             (* provides the rubberbanding feedback for the user inputting a
                                                             point for an open wire.)
    (SHOWSKETCHXY X Y WINDOW)                                (* draw from the first pt to the new pt)
    (PROG (ENDPT)
         (AND PREVPTS (DRAWLINE [fetch (POSITION XCOORD) of (SETQ ENDPT (fetch (INPUTPT INPUT.POSITION)
                                                                         of (CAR PREVPTS]
                             (fetch (POSITION YCOORD) of ENDPT)
                             X Y 1 'INVERT WINDOW))          (* draw from the last pt to the new pt)
         (AND (CDR PREVPTS)
              (DRAWLINE [fetch (POSITION XCOORD) of (SETQ ENDPT (fetch (INPUTPT INPUT.POSITION)
                                                                 of (CAR (LAST PREVPTS]
                     (fetch (POSITION YCOORD) of ENDPT)
                     X Y 1 'INVERT WINDOW])

(CLOSEDWIRE.REGIONFN
  [LAMBDA (KNOTSCRELT)                                       (* rrb " 2-Jun-85 17:15")
                                                             (* returns the region occuppied by a closed wire)

        (* increase by half the brush size plus 2 This has the nice property of insuring that the region always has both height and
        width.)

    (INCREASEREGION (REGION.CONTAINING.PTS (fetch (SCREENELT HOTSPOTS) of KNOTSCRELT))
           (IPLUS 3 (QUOTIENT (fetch (BRUSH BRUSHSIZE) of (fetch (LOCALCLOSEDWIRE LOCALCLOSEDWIREBRUSH)
                                                           of (fetch (SCREENELT LOCALPART) of KNOTSCRELT)))
                           2])

(CLOSEDWIRE.GLOBALREGIONFN
  [LAMBDA (GCLOSEDWIREELT)                                   (* rrb "23-Oct-85 11:30")
                                                             (* returns the global region occupied by a global closed curve
                                                             element.)
    (OR (fetch (CLOSEDWIRE CLOSEDWIREREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCLOSEDWIREELT))
        (PROG ((INDVCLOSEDWIRE (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GCLOSEDWIREELT))
               REGION)
              [SETQ REGION (INCREASEREGION (REGION.CONTAINING.PTS (fetch (CLOSEDWIRE LATLONKNOTS) of
                                                                                                   INDVCLOSEDWIRE
                                                                   ))
                                  (SK.BRUSH.SIZE (fetch (CLOSEDWIRE BRUSH) of INDVCLOSEDWIRE]
              (replace (CLOSEDWIRE CLOSEDWIREREGION) of INDVCLOSEDWIRE with REGION)
              (RETURN REGION])

(SK.WIRE.CREATE
  [LAMBDA (KNOTS BRUSH DASHING CLOSED SCALE ARROWHEADS FILLING)   (* rrb "19-Mar-86 17:51")
```

```
                                                                    (* creates a wire sketch element.)
    (AND KNOTS
         (KNOT.SET.SCALE.FIELD (create GLOBALPART
                                          INDIVIDUALGLOBALPART _
                                          (COND
                                             (CLOSED (create CLOSEDWIRE
                                                            LATLONKNOTS _ KNOTS
                                                            BRUSH _ BRUSH
                                                            CLOSEDWIREDASHING _ DASHING
                                                            CLOSEDWIREINITSCALE _ SCALE
                                                            CLOSEDWIREFILLING _ FILLING))
                                             (T (SET.WIRE.ARROWHEAD.POINTS (create WIRE
                                                                                    LATLONKNOTS _ KNOTS
                                                                                    BRUSH _ BRUSH
                                                                                    WIREARROWHEADS _ ARROWHEADS
                                                                                    OPENWIREDASHING _ DASHING
                                                                                    OPENWIREINITSCALE _ SCALE])
```

(WIRE.ADD.POINT.TO.END

```
  [LAMBDA (WIREELT PT SKW)                                          (* rrb "11-Jul-85 11:26")
                                                                    (* adds a point onto the end of a wire element.)
    (PROG ((NEWPOS (SK.MAP.INPUT.PT.TO.GLOBAL PT SKW))
           KNOTS GWIREELT)
          (SETQ GWIREELT (fetch (SCREENELT GLOBALPART) of WIREELT))
          (SETQ KNOTS (fetch LATLONKNOTS of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GWIREELT)))
          (RETURN (COND
                     ((EQUAL NEWPOS (CAR (LAST KNOTS)))             (* don't add duplicate points)
                      WIREELT)
                     (T                                             (* add point at the end.)
                        (SK.UPDATE.ELEMENT GWIREELT (WIRE.INPUTFN SKW (APPEND KNOTS (CONS NEWPOS))
                                                           NIL)
                               SKW])
```

(READ.ARROW.CHANGE

```
  [LAMBDA (SCRELTS SKW)
    (DECLARE (GLOBALVARS SK.ARROW.EDIT.MENU))                       (* rrb "17-Dec-85 17:09")

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

    (OR (type? MENU SK.ARROW.EDIT.MENU)
        (SETQ SK.ARROW.EDIT.MENU (create MENU
                                         TITLE _ "specify change"
                                         ITEMS _ (APPEND '((Add% Arrow 'ADD "Adds an arrow head.")
                                                           ("Remove Arrow" 'DELETE "Removes the arrow head.")
                                                           ("Same as First" 'SAME "Makes all of the arrowheads be
                                                                  the same as the first one selected.")
                                                           (Wider 'WIDER "Makes the angle of the head wider.")
                                                           (Narrower 'NARROWER "Makes the angle of the head
                                                                  smaller.")
                                                           (Larger 'LARGER "Makes the arrow head larger.")
                                                           (Smaller 'SMALLER "Makes the arrow head smaller."))
                                                        (LIST (LIST VSHAPE.ARROWHEAD.BITMAP ''OPEN "Makes the
                                                                    head be the side lines only.")
                                                              (LIST CURVEDV.ARROWHEAD.BITMAP ''OPENCURVE
                                                                    "Makes the arrowhead have curved side
                                                                    lines.")
                                                              (LIST TRIANGLE.ARROWHEAD.BITMAP ''CLOSED "Makes the
                                                                    head be two sides and a base.")
                                                              (LIST SOLIDTRIANGLE.ARROWHEAD.BITMAP
                                                                    ''SOLID "makes a solid triangular
                                                                    arrowhead.")))
                                         CENTERFLG _ T)))
    (PROG (HOW)
          (RETURN (LIST (OR (READ.ARROWHEAD.END)
                            (RETURN))
                        (COND
                           ((EQ (SETQ HOW (\CURSOR.IN.MIDDLE.MENU SK.ARROW.EDIT.MENU))
                                'SAME)                              (* if the user chooses SAME, determine the characteristics.)
                            (OR (bind NOWARROWS INDGELT for ELT in SCRELTS
                                   do (SETQ INDGELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of ELT))
                                      [COND
                                         ((SETQ NOWARROWS (SELECTQ (fetch (INDIVIDUALGLOBALPART GTYPE)
                                                                          of INDGELT)
                                                              (OPENCURVE (fetch (OPENCURVE CURVEARROWHEADS)
                                                                                of INDGELT))
                                                              (ARC (fetch (ARC ARCARROWHEADS) of INDGELT))
                                                              (WIRE (fetch (WIRE WIREARROWHEADS) of INDGELT))
                                                              NIL))
                                          (COND
                                             [(CAR NOWARROWS)
                                              (RETURN (CONS 'SAME (CAR NOWARROWS]
                                             ((CADR NOWARROWS)
                                              (RETURN (CONS 'SAME (CADR NOWARROWS]
                                   finally (STATUSPRINT SKW "None of the selected elements have arrowheads."))
                                (RETURN)))
```

```
                              (HOW)
                              (T (RETURN])

(CHANGE.ELTS.ARROWHEADS
  [LAMBDA (CHANGESPEC ELTSWITHARROWS SKW)                          (* rrb "10-Jan-85 16:58")

               (* * function that changes the arrow characteristics for all elements in ELTSWITHARROWS that can have arrows.)

    (AND CHANGESPEC (for ARROWELT in ELTSWITHARROWS collect (SK.CHANGE.ARROWHEADS ARROWELT CHANGESPEC SKW])
)

(DEFINEQ

(SKETCH.CREATE.CLOSED.WIRE
  [LAMBDA (POINTS BRUSH DASHING FILLING SCALE)                     (* rrb "16-Oct-85 17:12")
                                                                   (* creates a sketch closed wire element.)
    (SK.WIRE.CREATE (SK.INSURE.POINT.LIST POINTS)
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           T
           (OR (NUMBERP SCALE)
               1.0)
           NIL
           (SK.INSURE.FILLING FILLING])

(CLOSED.WIRE.INPUTFN
  [LAMBDA (W PTLIST)                                               (* rrb "13-Dec-84 10:10")

               (* creates a closed wire {a series of straight lines through a list of points} from a list of points passed in or a list that is read
               from the user via mouse.)

    (WIRE.INPUTFN W PTLIST T])

(CLOSED.WIRE.DRAWFN
  [LAMBDA (CLOSEDWIREELT WIN REG OPERATION)                        ; Edited  3-Mar-87 10:09 by rrb
                                                                   (* draws a closed wire element.)
    (PROG ((GINDVELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of CLOSEDWIREELT))
           (LOCALPART (fetch (SCREENELT LOCALPART) of CLOSEDWIREELT))
           VARX)
          (SETQ VARX (fetch (LOCALCLOSEDWIRE LOCALCLOSEDWIREFILLING) of LOCALPART))
          [COND
             ((OR (fetch (SKFILLING FILLING.TEXTURE) of VARX)
                  (fetch (SKFILLING FILLING.COLOR) of VARX))       (* if there isn't any filling, don't fill.)
              (FILLPOLYGON (fetch (LOCALCLOSEDWIRE KNOTS) of LOCALPART)
                     [COND
                        (SKETCHINCOLORFLG VARX)
                        ((fetch (SKFILLING FILLING.TEXTURE) of VARX))
                        (T                                         (* simulate color)
                           (TEXTUREOFCOLOR (fetch (SKFILLING FILLING.COLOR) of VARX]
                     WIN
                     (COND
                        ((EQ (DSPOPERATION NIL WIN)
                             'ERASE)                               (* if the stream is erasing, erase.)
                         'ERASE)
                        (T                                         (* otherwise use the element's mode.)
                           (fetch (SKFILLING FILLING.OPERATION) of VARX]
          (OR (EQ (fetch (BRUSH BRUSHSIZE) of (SETQ VARX (fetch (LOCALCLOSEDWIRE LOCALCLOSEDWIREBRUSH)
                                                            of LOCALPART)))
                  0)
              (WB.DRAWLINE CLOSEDWIREELT WIN REG OPERATION T (fetch (CLOSEDWIRE CLOSEDWIREDASHING) of GINDVELT)
                     VARX])

(CLOSEDWIRE.READCHANGEFN
  [LAMBDA (SKW SCRNELTS)                                           (* rrb " 5-Mar-86 13:35")

               (* the users has selected SCRNELT to be changed this function reads a specification of how the closed wire elements
               should change.)

    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU
                                           (create MENU
                                                  CENTERFLG _ T
                                                  TITLE _ "Which aspect?"
                                                  ITEMS _ (APPEND (COND
                                                                     [(SKETCHINCOLORP)
                                                                      '(("Brush color" 'BRUSHCOLOR "changes the
                                                                              color of the outline")
                                                                        ("Filling color" 'FILLINGCOLOR
                                                                              "changes the color of the
                                                                              filling"]
                                                                     (T NIL))
                                                                 [COND
```

```
                                                                   (FILLPOLYGONFLG '((Filling 'FILLING
                                                                                         "allows changing of
                                                                                         the filling texture
                                                                                         of the box."]
                                                                [COND
                                                                   (FILLINGMODEFLG '(("Filling mode"
                                                                                   'FILLINGMODE "changes how
                                                                                   the filling effects the
                                                                                   figures it covers."]
                                                                '((Shape 'SHAPE "changes the shape of the
                                                                         brush")
                                                                  (Size 'SIZE "changes the size of the brush")
                                                                  (Dashing 'DASHING "changes the dashing of the
                                                                         line.")
                                                                  ("Add point" 'ADDPOINT "adds a point to the
                                                                         curve."]
                            (SIZE (READSIZECHANGE "Change size how?" T))
                            (FILLING (READ.FILLING.CHANGE))
                            (FILLINGMODE (READ.FILLING.MODE))
                            (DASHING (READ.DASHING.CHANGE))
                            (SHAPE (READBRUSHSHAPE))
                            (BRUSHCOLOR [READ.COLOR.CHANGE "Change outline color how?" NIL
                                               (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                              (fetch (SCREENELT GLOBALPART)
                                                                                 of (CAR SCRNELTS))
                                                                              'BRUSH])
                            (ADDPOINT (READ.POINT.TO.ADD (CAR SCRNELTS)
                                             SKW))
                            (FILLINGCOLOR [READ.COLOR.CHANGE "Change filling color how?" T
                                                 (fetch (SKFILLING FILLING.COLOR) of (GETSKETCHELEMENTPROP
                                                                                      (fetch (SCREENELT GLOBALPART)
                                                                                         of (CAR SCRNELTS))
                                                                                      'FILLING])
                            NIL))
              (RETURN (AND HOW (LIST ASPECT HOW])
```

(**CLOSEDWIRE.TRANSFORMFN**
```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "18-Oct-85 16:46")
```

(* returns a copy of the global CLOSEDWIRE element that has had each of its control points transformed by transformfn. TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

```
    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (KNOT.SET.SCALE.FIELD (create GLOBALPART using GELT INDIVIDUALGLOBALPART _
                                                                  (create CLOSEDWIRE
                                                                     using INDVPART LATLONKNOTS _
                                                                           (SK.TRANSFORM.POINT.LIST
                                                                            (fetch (CLOSEDWIRE LATLONKNOTS)
                                                                               of INDVPART)
                                                                            TRANSFORMFN TRANSFORMDATA)
                                                                           BRUSH _ (SK.TRANSFORM.BRUSH
                                                                                    (fetch (CLOSEDWIRE BRUSH)
                                                                                       of INDVPART)
                                                                                    SCALEFACTOR)
                                                                           CLOSEDWIREREGION _ NIL])
```

(**CLOSEDWIRE.TRANSLATEPTSFN**
```
  [LAMBDA (KNOTELT SELPTS GDELTA WINDOW)                         (* rrb "27-Sep-85 18:58")
```

(* returns a closed wire element which has the knots that are members of SELPTS translated by the global amount GDELTA.)

```
    (PROG ((GKNOTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of KNOTELT)))
          (RETURN (SK.WIRE.CREATE (for PT in (fetch (LOCALCLOSEDWIRE KNOTS) of (fetch (SCREENELT LOCALPART)
                                                                                  of KNOTELT))
                                       as LATLONPT in (fetch (CLOSEDWIRE LATLONKNOTS) of GKNOTELT)
                                       collect (COND
                                                  ((MEMBER PT SELPTS)
                                                   (PTPLUS LATLONPT GDELTA))
                                                  (T LATLONPT)))
                         (fetch (CLOSEDWIRE BRUSH) of GKNOTELT)
                         (fetch (CLOSEDWIRE CLOSEDWIREDASHING) of GKNOTELT)
                         T
                         (fetch (CLOSEDWIRE CLOSEDWIREINITSCALE) of GKNOTELT)
                         NIL
                         (fetch (CLOSEDWIRE CLOSEDWIREFILLING) of GKNOTELT])
)

(DEFINEQ
```

(**SK.EXPAND.ARROWHEADS**
```
  [LAMBDA (GARROWHEADPOINTS SCALE)                               (* rrb " 2-May-86 10:50")
```
(* returns a list of local arrowhead points from the list of global arrowhead points.)

```
    (for ARROWHEAD in GARROWHEADPOINTS collect (SK.EXPAND.ARROWHEAD ARROWHEAD SCALE])

(SK.COMPUTE.ARC.ARROWHEAD.POINTS
  [LAMBDA (ARROWSPEC CENTERPT RADPT ARCANGLEPT DIRECTION)          (* rrb "19-Mar-86 17:09")

            (* returns a list of global arrowhead points from the specs and points that define an arc.)

    (PROG (SPEC)
          (OR ARROWSPEC (RETURN NIL))                              (* format keeps arrow specs as (FIRST LAST T)%.)
          (RETURN (LIST (AND (SETQ SPEC (CAR ARROWSPEC))
                             (ARC.ARROWHEAD.POINTS CENTERPT RADPT DIRECTION (fetch (ARROWHEAD ARROWANGLE)
                                                                                of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC)))
                        (AND (SETQ SPEC (CADR ARROWSPEC))
                             (ARC.ARROWHEAD.POINTS CENTERPT ARCANGLEPT (NOT DIRECTION)
                                    (fetch (ARROWHEAD ARROWANGLE) of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC])

(ARC.ARROWHEAD.POINTS
  [LAMBDA (CENTERPT ENDPT CLOCKWISEFLG HEAD.ANGLE HEAD.LENGTH HEAD.TYPE)
                                                                   (* rrb "20-Mar-86 09:12")

            (* returns a list of arrowhead points for an arc. If CLOCKWISEFLG is T, it is to go on the clockwise direction.)

    (ARROWHEAD.POINTS.LIST ENDPT HEAD.ANGLE HEAD.LENGTH (TIMES (COND
                                                                  (CLOCKWISEFLG -1)
                                                                  (T 1))
                                                               (DIFFERENCE (fetch (POSITION YCOORD) of ENDPT)
                                                                      (fetch (POSITION YCOORD) of CENTERPT)))
           (TIMES (COND
                     (CLOCKWISEFLG 1)
                     (T -1))
                  (DIFFERENCE (fetch (POSITION XCOORD) of ENDPT)
                         (fetch (POSITION XCOORD) of CENTERPT)))
           HEAD.TYPE])

(SET.ARC.ARROWHEAD.POINTS
  [LAMBDA (INDVDARCELT)                                            (* rrb "20-Jun-86 13:56")

            (* * updates the global arrowhead points field of an element.)

    (PROG ((ARROWSPECS (fetch (ARC ARCARROWHEADS) of INDVDARCELT)))
          [COND
             (ARROWSPECS (SK.INSURE.HAS.LENGTH INDVDARCELT (SK.RECORD.LENGTH 'ARC)
                                'ARC)
                    (replace (ARC ARCARROWHEADPOINTS) of INDVDARCELT with (SK.COMPUTE.ARC.ARROWHEAD.POINTS
                                                                           ARROWSPECS
                                                                           (fetch (ARC ARCCENTERPT) of INDVDARCELT)
                                                                           (fetch (ARC ARCRADIUSPT) of INDVDARCELT)
                                                                           (\SK.GET.ARC.ANGLEPT INDVDARCELT)
                                                                           (fetch (ARC ARCDIRECTION) of INDVDARCELT]
          (RETURN INDVDARCELT])

(SET.OPENCURVE.ARROWHEAD.POINTS
  [LAMBDA (INDVOPENCURVEELT)                                       (* rrb "20-Mar-86 14:30")

            (* * updates the global arrowhead points field of an element.)

    (PROG ((ARROWSPECS (fetch (OPENCURVE CURVEARROWHEADS) of INDVOPENCURVEELT)))
          [COND
             (ARROWSPECS (SK.INSURE.HAS.LENGTH INDVOPENCURVEELT (SK.RECORD.LENGTH 'OPENCURVE)
                                'OPENCURVE)
                    (replace (OPENCURVE OPENCURVEARROWHEADPOINTS) of INDVOPENCURVEELT
                       with (SK.COMPUTE.CURVE.ARROWHEAD.POINTS ARROWSPECS (fetch (OPENCURVE LATLONKNOTS)
                                                                             of INDVOPENCURVEELT]
          (RETURN INDVOPENCURVEELT])

(SK.COMPUTE.CURVE.ARROWHEAD.POINTS
  [LAMBDA (ARROWSPEC KNOTS)                                        (* rrb "19-Mar-86 17:32")

            (* returns a list of global arrowhead points from the specs and points that define an curve.)

    (PROG (SPEC)
          (OR ARROWSPEC (RETURN NIL))                              (* format keeps arrow specs as (FIRST LAST T)%.)
          (RETURN (LIST (AND (SETQ SPEC (CAR ARROWSPEC))
                             (CURVE.ARROWHEAD.POINTS KNOTS T (fetch (ARROWHEAD ARROWANGLE) of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC)))
                        (AND (SETQ SPEC (CADR ARROWSPEC))
                             (CURVE.ARROWHEAD.POINTS KNOTS NIL (fetch (ARROWHEAD ARROWANGLE) of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC])
```

(SET.WIRE.ARROWHEAD.POINTS

```
  [LAMBDA (INDVWIREELT)                                         (* rrb "20-Mar-86 14:31")

            (* * updates the global arrowhead points field of an element.)

    (PROG ((ARROWSPECS (fetch (WIRE WIREARROWHEADS) of INDVWIREELT)))
          [COND
             (ARROWSPECS (SK.INSURE.HAS.LENGTH INDVWIREELT (SK.RECORD.LENGTH 'WIRE)
                                'WIRE)
                    (replace (WIRE OPENWIREARROWHEADPOINTS) of INDVWIREELT with (
                                                                        SK.COMPUTE.WIRE.ARROWHEAD.POINTS
                                                                         ARROWSPECS
                                                                         (fetch (WIRE LATLONKNOTS)
                                                                            of INDVWIREELT]
          (RETURN INDVWIREELT])
```

(SK.COMPUTE.WIRE.ARROWHEAD.POINTS

```
  [LAMBDA (ARROWSPEC KNOTS)                                     (* rrb "19-Mar-86 17:46")

            (* returns a list of global arrowhead points from the specs and points that define an curve.)

    (PROG (SPEC)
          (OR ARROWSPEC (RETURN NIL))                           (* format keeps arrow specs as (FIRST LAST T)%.)
          (RETURN (LIST (AND (SETQ SPEC (CAR ARROWSPEC))
                             (WIRE.ARROWHEAD.POINTS KNOTS T (fetch (ARROWHEAD ARROWANGLE) of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC)))
                        (AND (SETQ SPEC (CADR ARROWSPEC))
                             (WIRE.ARROWHEAD.POINTS KNOTS NIL (fetch (ARROWHEAD ARROWANGLE) of SPEC)
                                    (fetch (ARROWHEAD ARROWLENGTH) of SPEC)
                                    (fetch (ARROWHEAD ARROWTYPE) of SPEC])
```

(SK.EXPAND.ARROWHEAD

```
  [LAMBDA (ARROWHEAD SCALE)                                     (* rrb "11-Jul-86 15:54")

            (* expands an arrowhead to a given scale. The format of Arrowhead points is
            (HEADPT ONESIDEENDPT OTHERSIDEENDPT) or (HEADPT
            (SIDE1PT1 SIDE1PT2) (SIDE2PT1 SIDE2PT2)))

    (AND ARROWHEAD (CONS (SK.SCALE.POSITION.INTO.VIEWER (CAR ARROWHEAD)
                                SCALE)
                         (COND
                            ((POSITIONP (CADR ARROWHEAD))
                             (for PT in (CDR ARROWHEAD) collect (SK.SCALE.POSITION.INTO.VIEWER PT SCALE)))
                            (T                                  (* form is (HEADPT (SIDE1PT1 SIDE1PT2)
                                                                (SIDE2PT1 SIDE2PT2)))
                               (for PTLST in (CDR ARROWHEAD) collect (for PT in PTLST collect (
                                                                             SK.SCALE.POSITION.INTO.VIEWER
                                                                                  PT SCALE])
```

(CHANGED.ARROW

```
  [LAMBDA (ARROW HOWTOCHANGE SCALE DEFARROW)                    (* rrb "17-Dec-85 17:04")

            (* * returns an arrow that has been changed according to the spec HOWTOCHANGE.)

    (COND
       ((EQ HOWTOCHANGE 'ADD)                                   (* if there already is one, leave it alone.)
        (OR ARROW (SK.CREATE.ARROWHEAD DEFARROW SCALE)))
       ((OR (EQ HOWTOCHANGE 'DELETE)
            (NULL ARROW))
        NIL)
       ((EQ (CAR HOWTOCHANGE)
            'SAME)                                              (* make it the same as the one given.)
        (APPEND (CDR HOWTOCHANGE)))
       (T (SELECTQ HOWTOCHANGE
              (WIDER (create ARROWHEAD using ARROW ARROWANGLE _ (PLUS SK.ARROWHEAD.ANGLE.INCREMENT
                                                                      (fetch (ARROWHEAD ARROWANGLE) of ARROW))))
              (NARROWER (create ARROWHEAD using ARROW ARROWANGLE _ (DIFFERENCE (fetch (ARROWHEAD ARROWANGLE)
                                                                                  of ARROW)
                                                                           SK.ARROWHEAD.ANGLE.INCREMENT)))
              (LARGER (create ARROWHEAD using ARROW ARROWLENGTH _ (PLUS (TIMES SK.ARROWHEAD.LENGTH.INCREMENT
                                                                                SCALE)
                                                                         (fetch (ARROWHEAD ARROWLENGTH)
                                                                            of ARROW))))
              (SMALLER (create ARROWHEAD using ARROW ARROWLENGTH _ (MAX (DIFFERENCE (fetch (ARROWHEAD ARROWLENGTH)
                                                                                          of ARROW)
                                                                                   (TIMES
                                                                                        SK.ARROWHEAD.LENGTH.INCREMENT
                                                                                          SCALE))
                                                                        SCALE)))
```

```
               (OPEN (create ARROWHEAD using ARROW ARROWTYPE _ 'LINE))
               (CLOSED (create ARROWHEAD using ARROW ARROWTYPE _ 'CLOSEDLINE))
               (SOLID (create ARROWHEAD using ARROW ARROWTYPE _ 'SOLID))
               (OPENCURVE (create ARROWHEAD using ARROW ARROWTYPE _ 'CURVE))
               ARROW])
```

(SK.CHANGE.ARROWHEAD

```
  [LAMBDA (ARROWELT HOW SKW)                                 (* rrb " 1-May-86 16:27")

            (* changes the arrow heads of an element and returns the new element if any actually occurred.)

    (SK.CHANGE.ARROWHEAD1 ARROWELT (CAR HOW)
           (CADR HOW)
           (SK.INPUT.SCALE SKW)
           (fetch (SKETCHCONTEXT SKETCHARROWHEAD) of (WINDOWPROP SKW 'SKETCHCONTEXT])
```

(SK.CHANGE.ARROWHEAD1

```
  [LAMBDA (GARROWELT WHICHEND HOWTOCHANGE SCALE DEFAULTARROWHEAD)
                                                             (* rrb "20-Jun-86 13:57")
    (PROG (INDGARROWELT NEWARROWS NOWARROWS CHANGEDFLG TYPE KNOTS)
          (RETURN (COND
                     ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of GARROWELT))
                            '(WIRE OPENCURVE ARC))           (* only works for things of wire type.)
                      (SETQ INDGARROWELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARROWELT))
                      [SETQ NOWARROWS (OR (SELECTQ TYPE
                                              (OPENCURVE (fetch (OPENCURVE CURVEARROWHEADS) of INDGARROWELT))
                                              (ARC (fetch (ARC ARCARROWHEADS) of INDGARROWELT))
                                              (fetch (WIRE WIREARROWHEADS) of INDGARROWELT))
                                          '(NIL NIL T]
                      (SETQ KNOTS (SELECTQ TYPE
                                      (ARC                   (* calculate the knots for the left most test)
                                           (LIST (fetch (ARC ARCRADIUSPT) of INDGARROWELT)
                                                 (\SK.GET.ARC.ANGLEPT INDGARROWELT)))
                                      (fetch LATLONKNOTS of INDGARROWELT)))
                                                             (* the brush is stored in the same place for all element types.)
                      (SETQ NEWARROWS (bind NEWARROW for ARROW in NOWARROWS as END
                                         in '(FIRST LAST) collect (SETQ NEWARROW (COND
                                                                                    ((SK.ARROWHEAD.END.TEST
                                                                                      WHICHEND END KNOTS)
                                                             (* change the spec)
                                                                                     (CHANGED.ARROW ARROW
                                                                                      HOWTOCHANGE SCALE
                                                                                      DEFAULTARROWHEAD))
                                                                                    (T ARROW)))
                                                                 (COND
                                                                    ((NOT (EQUAL NEWARROW ARROW))
                                                             (* keep track of whether or not any arrow was changed.)
                                                                     (SETQ CHANGEDFLG T)))
                                                                 NEWARROW))
                      (AND CHANGEDFLG (create SKHISTORYCHANGESPEC
                                             NEWELT _
                                             (create GLOBALPART
                                                    COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                          of GARROWELT)
                                                    INDIVIDUALGLOBALPART _
                                                    (SELECTQ TYPE
                                                        (WIRE (SET.WIRE.ARROWHEAD.POINTS (create WIRE
                                                                                           using INDGARROWELT

                                                                                                 WIREARROWHEADS
                                                                                                  _ NEWARROWS))
)
                                                        (ARC (SET.ARC.ARROWHEAD.POINTS (create ARC
                                                                                         using INDGARROWELT
                                                                                               ARCARROWHEADS _
                                                                                               NEWARROWS)))
                                                        (OPENCURVE (SET.OPENCURVE.ARROWHEAD.POINTS
                                                                    (create OPENCURVE using INDGARROWELT
                                                                                            CURVEARROWHEADS _
                                                                                            NEWARROWS)))
                                                        (SHOULDNT)))
                                             OLDELT _ GARROWELT
                                             PROPERTY _ 'ARROWHEADS
                                             NEWVALUE _ NEWARROWS
                                             OLDVALUE _ NOWARROWS])
```

(SK.CREATE.ARROWHEAD

```
  [LAMBDA (DEFAULTARROWHEAD SCALE)                           (* rrb " 5-May-85 17:39")
                                                             (* creates a new arrowhead which is the default
                                                             DEFAULTARROWHEAD scaled to SCALE.)
    (create ARROWHEAD using DEFAULTARROWHEAD ARROWLENGTH _ (TIMES (fetch (ARROWHEAD ARROWLENGTH) of
                                                                                                 DEFAULTARROWHEAD
                                                                    )
                                                                  SCALE])
```

```
(SK.ARROWHEAD.CREATE
  [LAMBDA (SKW KNOTS)                                         (* rrb " 2-May-86 11:11")
                                                              (* creates the arrowhead specs that go with a global element
                                                              from the current context.)
    (PROG ((SKCONTEXT (WINDOWPROP SKW 'SKETCHCONTEXT))
           ARROWHEADWHERE)
          (SETQ ARROWHEADWHERE (fetch (SKETCHCONTEXT SKETCHUSEARROWHEAD) of SKCONTEXT))
          (RETURN (COND
                     ([NOT (MEMB ARROWHEADWHERE '(NIL NEITHER]    (* compute the arrowheads)
                                                              (* T is indicator of new format.)
                      (NCONC1 [for END in '(FIRST LAST) collect (COND
                                                             ((SK.ARROWHEAD.END.TEST ARROWHEADWHERE END
                                                                      KNOTS)
                                                              (* change the spec)
                                                              (SK.CREATE.ARROWHEAD (fetch (SKETCHCONTEXT
                                                                                           SKETCHARROWHEAD)
                                                                                      of SKCONTEXT)
                                                                     (SK.INPUT.SCALE SKW]
                             T])

(SK.ARROWHEAD.END.TEST
  [LAMBDA (WHICHENDS END KNOTS)                               (* rrb " 5-May-85 17:36")

          (* predicate which determines it END which is one of FIRST or LAST matches with WHICHENDS which is one of
          (FIRST LAST BOTH RIGHT LEFT) on the series of points KNOTS.)

    (OR (EQ WHICHENDS END)
        (SELECTQ WHICHENDS
            (BOTH T)
            (LEFT                                             (* determine if the specified end is END)
                  [COND
                     ((LEFT.MOST.IS.BEGINP KNOTS)
                      (EQ END 'FIRST))
                     ((EQ END 'LAST])
            (RIGHT [COND
                      ((LEFT.MOST.IS.BEGINP KNOTS)
                       (EQ END 'LAST))
                      ((EQ END 'FIRST])
            NIL])

(READ.ARROWHEAD.END
  [LAMBDA NIL                                                 (* rrb " 6-Nov-85 09:46")

          (* reads a specification of which end of a line or curve to put an arrowhead on.)

    (\CURSOR.IN.MIDDLE.MENU (COND
                               ((type? MENU SK.ARROW.END.MENU)
                                SK.ARROW.END.MENU)
                               (T (SETQ SK.ARROW.END.MENU (create MENU
                                                              TITLE _ "Which end?"
                                                              ITEMS _ '((|Left     | 'LEFT "changes will
                                                                                affect the left (or upper) end
                                                                                of the line.")
                                                                        (|    Right| 'RIGHT "changes will
                                                                                affect the right (or lower)
                                                                                end of the line.")
                                                                        (Both 'BOTH "changes will affect both
                                                                               ends of the line.")
                                                                        (First 'FIRST "changes will affect
                                                                                the end whose point was placed
                                                                                first.")
                                                                        (Last 'LAST "changes will affect the
                                                                               end placed last."))
                                                              CENTERFLG _ T])

(ARROW.HEAD.POSITIONS
  [LAMBDA (TAIL.POSITION HEAD.POSITION HEAD.ANGLE HEAD.LENGTH)   (* edited%: "16-MAR-83 11:56")
    (PROG (X0 Y0 X1 Y1 DX DY COS.THETA LL SIN.THETA COS.RHO SIN.RHO XP1 YP1 XP2 YP2)
          (SETQ X0 (fetch (POSITION XCOORD) of TAIL.POSITION))
          (SETQ Y0 (fetch (POSITION YCOORD) of TAIL.POSITION))
          (SETQ X1 (fetch (POSITION XCOORD) of HEAD.POSITION))
          (SETQ Y1 (fetch (POSITION YCOORD) of HEAD.POSITION))
          (SETQ DX (IDIFFERENCE X1 X0))
          (SETQ DY (IDIFFERENCE Y1 Y0))
          [SETQ LL (SQRT (PLUS (TIMES DX DX)
                               (TIMES DY DY]
          (SETQ COS.RHO (QUOTIENT DX LL))
          (SETQ SIN.RHO (QUOTIENT DY LL))
          (SETQ COS.THETA (COS HEAD.ANGLE))
          (SETQ SIN.THETA (SIN HEAD.ANGLE))
          [SETQ XP1 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES COS.RHO COS.THETA)
                                                   (TIMES SIN.RHO SIN.THETA]
          [SETQ YP1 (TIMES HEAD.LENGTH (PLUS (TIMES SIN.RHO COS.THETA)
                                             (TIMES SIN.THETA COS.RHO]
          [SETQ XP2 (TIMES HEAD.LENGTH (PLUS (TIMES COS.RHO COS.THETA)
                                             (TIMES SIN.RHO SIN.THETA]
          [SETQ YP2 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES SIN.RHO COS.THETA)
                                              (TIMES SIN.THETA COS.RHO]
          (RETURN (CONS (create POSITION
                                XCOORD _ (IDIFFERENCE X1 (FIX XP1))
                                YCOORD _ (IDIFFERENCE Y1 (FIX YP1)))
                        (create POSITION
                                XCOORD _ (IDIFFERENCE X1 (FIX XP2))
                                YCOORD _ (IDIFFERENCE Y1 (FIX YP2])
```

(ARROWHEAD.POINTS.LIST

```
  [LAMBDA (HEAD.POSITION HEAD.ANGLE HEAD.LENGTH DX DY HEAD.TYPE)   (* rrb " 1-May-86 16:15")
```

(* * returns a list of end points for an arrowhead ending on HEAD.POSITION with a slope of DX DY with an angle of HEAD.ANGLE and a length of HEAD.LENGTH If HEAD.TYPE is LINE or CLOSEDLINE, the result is a list of (HEADPT ONESIDEENDPT OTHERSIDEENDPT) If HEAD.TYPE is CURVE, the result is (HEADPT (SIDE1PT1 SIDE1PT2) (SIDE2PT1 SIDE2PT2)))

```
    (PROG (X1 Y1 COS.THETA LL SIN.THETA COS.RHO SIN.RHO XP1 YP1 XP2 YP2 ENDPT1 ENDPT2)
          (SETQ X1 (fetch (POSITION XCOORD) of HEAD.POSITION))
          (SETQ Y1 (fetch (POSITION YCOORD) of HEAD.POSITION))
          [SETQ LL (SQRT (PLUS (TIMES DX DX)
                               (TIMES DY DY]
          (SETQ COS.RHO (QUOTIENT DX LL))
          (SETQ SIN.RHO (QUOTIENT DY LL))
          (SETQ COS.THETA (COS HEAD.ANGLE))
          (SETQ SIN.THETA (SIN HEAD.ANGLE))
          [SETQ XP1 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES COS.RHO COS.THETA)
                                              (TIMES SIN.RHO SIN.THETA]
          [SETQ YP1 (TIMES HEAD.LENGTH (PLUS (TIMES SIN.RHO COS.THETA)
                                             (TIMES SIN.THETA COS.RHO]
          [SETQ XP2 (TIMES HEAD.LENGTH (PLUS (TIMES COS.RHO COS.THETA)
                                             (TIMES SIN.RHO SIN.THETA]
          [SETQ YP2 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES SIN.RHO COS.THETA)
                                              (TIMES SIN.THETA COS.RHO]
          (SETQ ENDPT1 (create POSITION
                               XCOORD _ (DIFFERENCE X1 XP1)
                               YCOORD _ (DIFFERENCE Y1 YP1)))
          (SETQ ENDPT2 (create POSITION
                               XCOORD _ (DIFFERENCE X1 XP2)
                               YCOORD _ (DIFFERENCE Y1 YP2)))
          (RETURN (SELECTQ HEAD.TYPE
                      ((LINE CLOSEDLINE SOLID)
                           (LIST HEAD.POSITION ENDPT1 ENDPT2))
                      (CURVE
```

(* redo calculations with half the angle and half the length for a midpoint in the curve.)

```
                             (SETQ HEAD.ANGLE (QUOTIENT HEAD.ANGLE 1.5))
                             (SETQ HEAD.LENGTH (QUOTIENT HEAD.LENGTH 2.0))
                             (SETQ COS.THETA (COS HEAD.ANGLE))
                             (SETQ SIN.THETA (SIN HEAD.ANGLE))
                             [SETQ XP1 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES COS.RHO COS.THETA)
                                                                 (TIMES SIN.RHO SIN.THETA]
                             [SETQ YP1 (TIMES HEAD.LENGTH (PLUS (TIMES SIN.RHO COS.THETA)
                                                                (TIMES SIN.THETA COS.RHO]
                             [SETQ XP2 (TIMES HEAD.LENGTH (PLUS (TIMES COS.RHO COS.THETA)
                                                                (TIMES SIN.RHO SIN.THETA]
                             [SETQ YP2 (TIMES HEAD.LENGTH (DIFFERENCE (TIMES SIN.RHO COS.THETA)
                                                                 (TIMES SIN.THETA COS.RHO]
                             (LIST HEAD.POSITION (LIST (create POSITION
                                                               XCOORD _ (FIXR (DIFFERENCE X1 XP1))
                                                               YCOORD _ (FIXR (DIFFERENCE Y1 YP1)))
                                                       ENDPT1)
                                   (LIST (create POSITION
                                                 XCOORD _ (FIXR (DIFFERENCE X1 XP2))
                                                 YCOORD _ (FIXR (DIFFERENCE Y1 YP2)))
                                         ENDPT2)))
                      NIL])
```

(CURVE.ARROWHEAD.POINTS

```
  [LAMBDA (KNOTS BEGFLG HEAD.ANGLE HEAD.LENGTH HEAD.TYPE)         (* rrb "19-Mar-86 17:32")
```

(* returns a list of arrowhead points for a curve. If BEGFLG is T, it is to go on the first end.)

```
    (PROG [(SLOPE (\CURVESLOPE KNOTS (NOT BEGFLG]
          (RETURN (ARROWHEAD.POINTS.LIST [COND
                                             (BEGFLG (CAR KNOTS))
                                             (T (CAR (LAST KNOTS]
                       HEAD.ANGLE HEAD.LENGTH (COND
                                                   (BEGFLG (MINUS (CAR SLOPE)))
                                                   (T (CAR SLOPE)))
```

```
                          (COND
                             (BEGFLG (MINUS (CDR SLOPE)))
                             (T (CDR SLOPE)))
                          HEAD.TYPE])

(LEFT.MOST.IS.BEGINP
  [LAMBDA (KNOTLST)                                      (* rrb "30-Nov-84 16:55")

             (* * returns T if the beginning of the curve thru KNOTLST is to the left of its end.)

    (COND
       ((NULL (CDR (LISTP KNOTLST)))
        (ERROR KNOTLST "should have at least two elements."))
       (T (PROG ((FIRST (CAR KNOTLST))
                 (LAST (CAR (LAST KNOTLST)))
                 FIRSTX LASTX)
                (RETURN (OR (GREATERP (SETQ LASTX (fetch (POSITION XCOORD) of LAST))
                                      (SETQ FIRSTX (fetch (POSITION XCOORD) of FIRST)))
                            (AND (EQP LASTX FIRSTX)
                                 (GREATERP (fetch (POSITION YCOORD) of FIRST)
                                        (fetch (POSITION YCOORD) of LAST])

(WIRE.ARROWHEAD.POINTS
  [LAMBDA (KNOTS FIRSTFLG HEAD.ANGLE HEAD.LENGTH HEAD.TYPE)      (* rrb "19-Mar-86 17:46")

             (* returns a list of arrowhead points for a wire. If FIRSTFLG is T, it is to go on the first end.)

    (PROG (HEADPT TAILPT)
          (COND
             (FIRSTFLG (SETQ HEADPT (CAR KNOTS))
                    (SETQ TAILPT (CADR KNOTS)))
             ((CDR KNOTS)
              (for KNOTTAIL on KNOTS when (NULL (CDDR KNOTTAIL)) do (SETQ TAILPT (CAR KNOTTAIL))
                                                                   (SETQ HEADPT (CADR KNOTTAIL))
                                                                   (RETURN)))
             (T                                          (* only one point, don't put on an arrowhead.)
                (RETURN)))
          (RETURN (ARROWHEAD.POINTS.LIST HEADPT HEAD.ANGLE HEAD.LENGTH (COND
                                                                           (TAILPT (DIFFERENCE
                                                                                    (fetch (POSITION XCOORD)
                                                                                       of HEADPT)
                                                                                    (fetch (POSITION XCOORD)
                                                                                       of TAILPT)))
                                                                           (T 1))
                          (COND
                             (TAILPT (DIFFERENCE (fetch (POSITION YCOORD) of HEADPT)
                                            (fetch (POSITION YCOORD) of TAILPT)))
                             (T 0))
                          HEAD.TYPE])

(DRAWARROWHEADS
  [LAMBDA (ARROWSPECS ARROWPTS WINDOW SIZE OPERATION)           (* rrb " 6-May-86 18:19")

             (* * draws the arrowhead from the specs in ARROWSPECS and the points in ARROWPTS)

             (* PTS may be NIL in the case where an arrowhead was added to a closed knot element that only has one point.)

    (bind ARROWTYPE for SPEC in ARROWSPECS as PTS in ARROWPTS when (AND SPEC PTS)
       do (SELECTQ (SETQ ARROWTYPE (fetch (ARROWHEAD ARROWTYPE) of SPEC))
              (CURVE                                     (* curve type. ARROWPTS format is
                                                         (headPt (side1pt1 side1pt2) (side2pt1 side2pt2)))
                     (DRAWCURVE (CONS (CAR PTS)
                                      (CADR PTS))
                            NIL SIZE NIL WINDOW)
                     (DRAWCURVE (CONS (CAR PTS)
                                      (CADDR PTS))
                            NIL SIZE NIL WINDOW))
              (SOLID                                     (* solid triangle)
                     (COND
                        ((IMAGESTREAMTYPEP WINDOW 'PRESS)   (* PRESS doesn't implement filled areas.)
                         (\SK.DRAW.TRIANGLE.ARROWHEAD PTS SIZE WINDOW T))
                        (T (COND
                              ((OR (WINDOWP WINDOW)
                                   (IMAGESTREAMTYPEP WINDOW 'DISPLAY))
                                                         (* DISPLAY code doesn't fill out the entire area.)
                               (\SK.DRAW.TRIANGLE.ARROWHEAD PTS SIZE WINDOW T)))
                           (FILLPOLYGON PTS BLACKSHADE WINDOW))))
              (LINE                                      (* straight line form of arrow.)
                    (\SK.DRAW.TRIANGLE.ARROWHEAD PTS SIZE WINDOW NIL))
              (CLOSEDLINE                                (* triangle form of arrow.)
                          (\SK.DRAW.TRIANGLE.ARROWHEAD PTS SIZE WINDOW T))
              NIL])
```

```
(\SK.DRAW.TRIANGLE.ARROWHEAD
  [LAMBDA (ARROWHEADPTS BRUSH STREAM CLOSED?)                  (* rrb " 6-May-86 18:15")
                                                               (* draws a triangle form arrowhead.)
                                                               (* could be replaced with a drawpolygon call if this were
                                                               implemented in everybody.)
    (COND
       ((OR [NOT (OR (WINDOWP STREAM)
                     (IMAGESTREAMTYPEP STREAM 'DISPLAY]
            (EQ (SK.BRUSH.SIZE BRUSH)
                1))

             (* call draw line instead because draw curve is off by 1 and makes arrowheads look bad.)

        (DRAWBETWEEN (CAR ARROWHEADPTS)
               (CADR ARROWHEADPTS)
               (SK.BRUSH.SIZE BRUSH)
               NIL STREAM)
        (DRAWBETWEEN (CAR ARROWHEADPTS)
               (CADDR ARROWHEADPTS)
               (SK.BRUSH.SIZE BRUSH)
               NIL STREAM)
        (AND CLOSED? (DRAWBETWEEN (CADR ARROWHEADPTS)
                            (CADDR ARROWHEADPTS)
                            (SK.BRUSH.SIZE BRUSH)
                            NIL STREAM)))
       (T                                                      (* use curve drawing because the end pts of the lines look better)
          (DRAWCURVE (LIST (CAR ARROWHEADPTS)
                           (CADR ARROWHEADPTS))
                 NIL BRUSH NIL STREAM)
          (DRAWCURVE (LIST (CAR ARROWHEADPTS)
                           (CADDR ARROWHEADPTS))
                 NIL BRUSH NIL STREAM)
          (AND CLOSED? (DRAWCURVE (LIST (CADR ARROWHEADPTS)
                                        (CADDR ARROWHEADPTS))
                              NIL BRUSH NIL STREAM])

(\SK.ENDPT.OF.ARROW
  [LAMBDA (LOCALARROWHEADPTS)                                  (* rrb " 2-May-86 10:58")

             (* returns the point inside an arrowhead that the last point of the line should hit.)

    (PROG ((LASTPT (CADDR LOCALARROWHEADPTS)))                 (* make it |1/4| of the way from the base mid point to the tip.)
          (RETURN (create POSITION
                         XCOORD _ (QUOTIENT (PLUS (fetch (POSITION XCOORD) of (CAR LOCALARROWHEADPTS))
                                                  (TIMES (QUOTIENT (PLUS (fetch (POSITION XCOORD)
                                                                            of (CADR LOCALARROWHEADPTS))
                                                                         (fetch (POSITION XCOORD) of LASTPT))
                                                                  2)
                                                         3))
                                         4)
                         YCOORD _ (QUOTIENT (PLUS (fetch (POSITION YCOORD) of (CAR LOCALARROWHEADPTS))
                                                  (TIMES (QUOTIENT (PLUS (fetch (POSITION YCOORD)
                                                                            of (CADR LOCALARROWHEADPTS))
                                                                         (fetch (POSITION YCOORD) of LASTPT))
                                                                  2)
                                                         3))
                                         4])

(\SK.ADJUST.FOR.ARROWHEADS
  [LAMBDA (LOCALKNOTS LOCALARROWPTSLST GARROWHEADSPECS STREAM)  (* rrb " 6-May-86 17:43")

             (* returns a list of the knots that LOCALKNOTS should really be drawn through.
             This is different when the arrowhead is solid because wide lines will make the arrow look funny if they are run out all the way
             to the end.)

    [COND
       ((IMAGESTREAMTYPEP STREAM 'PRESS)                       (* PRESS doesn't implement filled areas.)
        LOCALKNOTS)
       (T (PROG (LASTFIXED X)
                (SETQ LASTFIXED (COND
                                   ((AND (CADR LOCALARROWPTSLST)
                                         (EQ (fetch (ARROWHEAD ARROWTYPE) of (CADR GARROWHEADSPECS))
                                             'SOLID))
                                    (RPLACA (LAST (SETQ X (APPEND LOCALKNOTS)))
                                           (\SK.ENDPT.OF.ARROW (CADR LOCALARROWPTSLST)))
                                    X)
                                   (T LOCALKNOTS)))
                (RETURN (COND
                           ((AND (CAR LOCALARROWPTSLST)
                                 (EQ (fetch (ARROWHEAD ARROWTYPE) of (CAR GARROWHEADSPECS))
                                     'SOLID))
                            (CONS (\SK.ENDPT.OF.ARROW (CAR LOCALARROWPTSLST))
                                  (CDR LASTFIXED)))
                           (T LASTFIXED]
    (PROG (LASTFIXED X)
          (SETQ LASTFIXED (COND
                             ((AND (CADR LOCALARROWPTSLST)
                                   (EQ (fetch (ARROWHEAD ARROWTYPE) of (CADR GARROWHEADSPECS))
                                       'SOLID))
                              (RPLACA (LAST (SETQ X (APPEND LOCALKNOTS)))
                                     (\SK.ENDPT.OF.ARROW (CADR LOCALARROWPTSLST)))
                              X)
                             (T LOCALKNOTS)))
          (RETURN (COND
                     ((AND (CAR LOCALARROWPTSLST)
                           (EQ (fetch (ARROWHEAD ARROWTYPE) of (CAR GARROWHEADSPECS))
                               'SOLID))
                      (CONS (\SK.ENDPT.OF.ARROW (CAR LOCALARROWPTSLST))
                            (CDR LASTFIXED)))
                     (T LASTFIXED])
```

(SK.SET.ARROWHEAD.LENGTH

```
  [LAMBDA (W)                                                (* rrb "14-May-86 19:27")
                                                             (* sets the size of the default arrowhead.)
    (PROG [NEWSIZE (NOWARROWHEAD (fetch (SKETCHCONTEXT SKETCHARROWHEAD) of (WINDOWPROP W 'SKETCHCONTEXT]
          (SETQ NEWSIZE (RNUMBER (CONCAT "New arrowhead size in screen pts.  Current arrowhead size is "
                                        (MKSTRING (fetch (ARROWHEAD ARROWLENGTH) of NOWARROWHEAD)))
                               NIL NIL NIL T T T))
          (RETURN (COND
                     ((OR (NULL NEWSIZE)
                          (IGEQ 0 NEWSIZE))
                      NIL)
                     (T (replace (SKETCHCONTEXT SKETCHARROWHEAD) of (WINDOWPROP W 'SKETCHCONTEXT)
                           with (create ARROWHEAD using NOWARROWHEAD ARROWLENGTH _ NEWSIZE])
```

(SK.SET.ARROWHEAD.ANGLE

```
  [LAMBDA (W)                                                (* rrb "14-May-86 19:27")
                                                             (* sets the angle of the default arrowhead.)
    (PROG [NEWSIZE (NOWARROWHEAD (fetch (SKETCHCONTEXT SKETCHARROWHEAD) of (WINDOWPROP W 'SKETCHCONTEXT]
          (SETQ NEWSIZE (RNUMBER (CONCAT "New head angle in degrees. Current arrowhead angle is "
                                        (MKSTRING (fetch (ARROWHEAD ARROWANGLE) of NOWARROWHEAD)))
                               NIL NIL NIL T T T))
          (RETURN (COND
                     ((OR (NULL NEWSIZE)
                          (IGEQ 0 NEWSIZE))
                      NIL)
                     (T (replace (SKETCHCONTEXT SKETCHARROWHEAD) of (WINDOWPROP W 'SKETCHCONTEXT)
                           with (create ARROWHEAD using NOWARROWHEAD ARROWANGLE _ NEWSIZE])
```

(SK.SET.ARROWHEAD.TYPE

```
  [LAMBDA (W VALUE)                                          (* rrb "19-Mar-86 10:25")
                                                             (* Sets the type of the default arrowhead)
    (PROG ([NEWSHAPE (COND
                        ((MEMB VALUE '(LINE CURVE CLOSEDLINE SOLID))
                         VALUE)
                        (T (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                          TITLE _ "Choose style"
                                                          ITEMS _ (LIST (LIST VSHAPE.ARROWHEAD.BITMAP
                                                                              ''LINE "arrowhead consists of two
                                                                              line segments.")
                                                                        (LIST CURVEDV.ARROWHEAD.BITMAP
                                                                              ''CURVE "arrowhead has curved side
                                                                              lines.")
                                                                        (LIST TRIANGLE.ARROWHEAD.BITMAP
                                                                              ''CLOSEDLINE "arrowhead consists
                                                                              of a triangle.")
                                                                        (LIST SOLIDTRIANGLE.ARROWHEAD.BITMAP
                                                                              ''SOLID "makes a solid triangular
                                                                              arrowhead."))
                                                          ITEMHEIGHT _ (PLUS 2 (BITMAPHEIGHT
                                                                                VSHAPE.ARROWHEAD.BITMAP))
                                                          CENTERFLG _ T]
           SKETCHCONTEXT)
          (RETURN (AND NEWSHAPE (replace (SKETCHCONTEXT SKETCHARROWHEAD) of (SETQ SKETCHCONTEXT
                                                                                (WINDOWPROP W 'SKETCHCONTEXT))
                                       with (create ARROWHEAD using (fetch (SKETCHCONTEXT SKETCHARROWHEAD) of
                                                                                                   SKETCHCONTEXT
                                                                           )
                                                                     ARROWTYPE _ NEWSHAPE])
```

(SK.SET.LINE.ARROWHEAD

```
  [LAMBDA (W NEWVALUE)                                       (* rrb " 6-Nov-85 09:50")
                                                             (* sets whether or not the default line has an arrowhead.)
    (PROG [(ARROWHEADEND (COND
                            ((MEMB NEWVALUE '(FIRST LAST BOTH NEITHER LEFT RIGHT))
                             NEWVALUE)
                            (T (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                              TITLE _ "Which end?"
                                                              ITEMS _ '((First 'FIRST "An arrowhead will be at
                                                                               the first point of any new lines
                                                                               or curves.")
                                                                        (Last 'LAST "An arrowhead will be at the
                                                                              last point of any new lines or
                                                                              curves.")
                                                                        (Both 'BOTH "Arrowheads will be both
                                                                              ends of any new lines or curves.")
                                                                        (Neither 'NEITHER "New lines will not
                                                                               have any arrowheads.")
                                                                        (|Left     | 'LEFT "An arrowhead will be
                                                                               at the leftmost end of any new
                                                                               lines or curves.")
                                                                        (|    Right| 'RIGHT "An arrowhead will
                                                                               be at the rightmost end of any
                                                                               new lines or curves."))
                                                              CENTERFLG _ T]
        (RETURN (AND ARROWHEADEND (replace (SKETCHCONTEXT SKETCHUSEARROWHEAD) of (WINDOWPROP W 'SKETCHCONTEXT)
                                     with ARROWHEADEND])
```

(SK.UPDATE.ARROWHEAD.FORMAT

```
  [LAMBDA (GELT)                                              (* rrb "25-Apr-85 10:28")
                                                              (* makes sure that the element GELT is in new format.)

        (* the fields of this are first arrowhead, last arrowhead and new format indicator.
        The old format had left arrowhead and right arrowhead.)

    (PROG ((INDGARROWELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           NOWARROWS)
          (SELECTQ (fetch (INDIVIDUALGLOBALPART GTYPE) of INDGARROWELT)
              (OPENCURVE [AND (SETQ NOWARROWS (fetch (OPENCURVE CURVEARROWHEADS) of INDGARROWELT))
                              (NULL (CDDR NOWARROWS))
                              (replace (OPENCURVE CURVEARROWHEADS) of INDGARROWELT
                                 with (COND
                                         ((LEFT.MOST.IS.BEGINP (fetch LATLONKNOTS of INDGARROWELT))
                                          (LIST (CAR NOWARROWS)
                                                (CADR NOWARROWS)
                                                T))
                                         (T (LIST (CADR NOWARROWS)
                                                  (CAR NOWARROWS)
                                                  T])
              (WIRE [AND (SETQ NOWARROWS (fetch (WIRE WIREARROWHEADS) of INDGARROWELT))
                         (NULL (CDDR NOWARROWS))
                         (replace (WIRE WIREARROWHEADS) of INDGARROWELT with (COND
                                                                                 ((LEFT.MOST.IS.BEGINP (fetch
                                                                                                         LATLONKNOTS
                                                                                                          of
                                                                                                        INDGARROWELT
                                                                                                             ))
                                                                                  (LIST (CAR NOWARROWS)
                                                                                        (CADR NOWARROWS)
                                                                                        T))
                                                                                 (T (LIST (CADR NOWARROWS)
                                                                                          (CAR NOWARROWS)
                                                                                          T])
              NIL)
          (RETURN GELT])
```

(SK.SET.LINE.LENGTH.MODE

```
  [LAMBDA (W VAL?)                                            (* rrb " 6-Nov-85 09:51")

        (* sets whether the lines drawn with the middle button connect e.g the next segment begins where the last one left off or
        whether it takes two clicks to get a single segment line.)

    (PROG [(LINEMODE (COND
                        ((MEMBER VAL? '(YES NO))
                         VAL?)
                        (T (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                          TITLE _ "Connect middle button lines?"
                                                          ITEMS _ '((Yes 'YES "The lines drawn with the middle
                                                                           button will pick up where the last one
                                                                           left off.")
                                                                    (No 'NO "Sets the default so that two middle
                                                                            clicks make a line."))
                                                          CENTERFLG _ T]
          (RETURN (AND LINEMODE (replace (SKETCHCONTEXT SKETCHLINEMODE) of (WINDOWPROP W 'SKETCHCONTEXT)
                                   with (EQ LINEMODE 'NO])
)

(DEFINEQ
```

(SK.INSURE.ARROWHEADS

```
  [LAMBDA (ARROWHEADSPECS)                                    ; Edited  8-Jan-87 19:46 by rrb
                                                              (* makes sure ARROWHEADSPECS is a legal list of two
                                                              arrowhead specifications.)
```

```
            (* slap a T on the end of it so it will be recognized as the new format.)

    (COND
       ((NULL ARROWHEADSPECS)
        NIL)
       ((SK.ARROWHEADP ARROWHEADSPECS)                                  (* the user passed in only one spec, make it be the end.)
        (LIST NIL ARROWHEADSPECS T))
       ((APPEND [for SPEC in ARROWHEADSPECS
                   collect (COND
                              ((NULL SPEC)
                               NIL)
                              ((SK.ARROWHEADP SPEC))
                              ((EQ SPEC T)
                               (create ARROWHEAD
                                      ARROWTYPE _ SK.DEFAULT.ARROW.TYPE
                                      ARROWANGLE _ SK.DEFAULT.ARROW.ANGLE
                                      ARROWLENGTH _ SK.DEFAULT.ARROW.LENGTH))
                              (T (\ILLEGAL.ARG ARROWHEADSPECS]
               '(T])

(SK.ARROWHEADP
  [LAMBDA (ARROWHEAD)                                                   (* rrb "16-Oct-85 16:24")
                                                                        (* determines if ARROWHEAD is a legal arrowhead
                                                                        specification.)

    (AND (EQLENGTH ARROWHEAD 3)
         (MEMB (fetch (ARROWHEAD ARROWTYPE) of ARROWHEAD)
               SK.ARROWHEAD.TYPES)
         (NUMBERP (fetch (ARROWHEAD ARROWANGLE) of ARROWHEAD))
         (NUMBERP (fetch (ARROWHEAD ARROWLENGTH) of ARROWHEAD))
         ARROWHEAD])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD LOCALWIRE (KNOTS LOCALHOTREGION ARROWHEADPTS LOCALOPENWIREBRUSH LOCALWIREDASHING))

(TYPERECORD WIRE (LATLONKNOTS BRUSH WIREARROWHEADS OPENWIREDASHING OPENWIREINITSCALE OPENWIREREGION
                        OPENWIREARROWHEADPOINTS))

(TYPERECORD CLOSEDWIRE (LATLONKNOTS BRUSH CLOSEDWIREDASHING CLOSEDWIREINITSCALE CLOSEDWIREFILLING
                              CLOSEDWIREREGION))

(RECORD LOCALCLOSEDWIRE (KNOTS LOCALHOTREGION LOCALCLOSEDWIREBRUSH LOCALCLOSEDWIREFILLING))
)
)

(DECLARE%: EVAL@COMPILE

(RECORD ARROWHEAD (ARROWTYPE ARROWANGLE ARROWLENGTH))
)

(READVARS-FROM-STRINGS '(VSHAPE.ARROWHEAD.BITMAP TRIANGLE.ARROWHEAD.BITMAP SOLIDTRIANGLE.ARROWHEAD.BITMAP
                               CURVEDV.ARROWHEAD.BITMAP)
       "({(READBITMAP)(24 18
       %"@@@@@@@@%"
       %"@@L@@@@@%"
       %"@@C@@@@@%"
       %"@@@L@@@@%"
       %"@@@C@@@@%"
       %"@@@@L@@@%"
       %"@@@@C@@@%"
       %"@@@@@L@@%"
       %"@@@@@B@@%"
       %"OOOOOO@@%"
       %"@@@@@B@@%"
       %"@@@@@L@@%"
       %"@@@@C@@@%"
       %"@@@@L@@@%"
       %"@@@C@@@@%"
       %"@@@L@@@@%"
       %"@@C@@@@@%"
       %"@@L@@@@@%")}  {(READBITMAP)(24 18
       %"@@@@@@@@%"
       %"@@L@@@@@%"
       %"@@K@@@@@%"
       %"@@HL@@@@%"
       %"@@HC@@@@%"
       %"@@H@L@@@%"
       %"@@H@C@@@%"
       %"@@H@@L@@%"
       %"@@H@@B@@%"
       %"OOOOOO@@%"
```

```
        %"@@H@@B@@%"
        %"@@H@@L@@%"
        %"@@H@C@@@%"
        %"@@H@L@@@%"
        %"@@HC@@@@%"
        %"@@HL@@@@%"
        %"@@K@@@@@%"
        %"@@L@@@@@%")}  {(READBITMAP)(24 18
        %"@@@@@@@@%"
        %"@@L@@@@@%"
        %"@@O@@@@@%"
        %"@@OL@@@@%"
        %"@@OO@@@@%"
        %"@@OOL@@@%"
        %"@@OOO@@@%"
        %"@@OOOL@@%"
        %"@@OOON@@%"
        %"OOOOOO@@%"
        %"@@OOON@@%"
        %"@@OOOL@@%"
        %"@@OOO@@@%"
        %"@@OOL@@@%"
        %"@@OO@@@@%"
        %"@@OL@@@@%"
        %"@@O@@@@@%"
        %"@@L@@@@@%")}  {(READBITMAP)(24 18
        %"@@@@@@@@%"
        %"@@@@@@@@%"
        %"@A@@@@@@%"
        %"@@H@@@@@%"
        %"@@D@@@@@%"
        %"@@C@@@@@%"
        %"@@@N@@@@%"
        %"@@@AL@@@%"
        %"@@@@CH@@%"
        %"OOOOOO@@%"
        %"@@@@CH@@%"
        %"@@@AL@@@%"
        %"@@@N@@@@%"
        %"@@C@@@@@%"
        %"@@D@@@@@%"
        %"@@H@@@@@%"
        %"@A@@@@@@%"
        %"@@@@@@@@%")})
        ")

(READVARS-FROM-STRINGS '(WIREICON CLOSEDWIREICON)
        "({(READBITMAP)(20 12
        %"@D@@@@@@%"
        %"@L@@@@@@%"
        %"AH@@@@@@%"
        %"C@GOL@@@%"
        %"F@OOL@@@%"
        %"L@L@L@@@%"
        %"LAH@L@@@%"
        %"FAHAH@@@%"
        %"CC@C@@@@%"
        %"AK@C@@@@%"
        %"@N@F@@@@%"
        %"@F@L@@@@%")}  {(READBITMAP)(20 12
        %"@G@GN@@@%"
        %"@OHON@@@%"
        %"AMMLN@@@%"
        %"CHOIL@@@%"
        %"G@GCH@@@%"
        %"N@@G@@@@%"
        %"G@@N@@@@%"
        %"CH@GH@@@%"
        %"AL@AN@@@%"
        %"@O@@F@@@%"
        %"@GOON@@@%"
        %"@COON@@@%")})
        ")
```

(RPAQ? **SK.ARROWHEAD.ANGLE.INCREMENT** 5)

(RPAQ? **SK.ARROWHEAD.LENGTH.INCREMENT** 2)

(ADDTOVAR **SK.ARROWHEAD.TYPES** LINE CLOSEDLINE CURVE SOLID)

(RPAQ? **SK.DEFAULT.ARROW.LENGTH** 8)

(RPAQ? **SK.DEFAULT.ARROW.TYPE** 'CURVE)

(RPAQ? **SK.DEFAULT.ARROW.ANGLE** 18.0)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

```
(GLOBALVARS SK.DEFAULT.ARROW.LENGTH SK.DEFAULT.ARROW.TYPE SK.DEFAULT.ARROW.ANGLE SK.ARROWHEAD.TYPES)
)

(RPAQ? SK.ARROW.END.MENU )

(RPAQ? SK.ARROW.EDIT.MENU )
```

;; stuff to support the text element type.

```
(DEFINEQ

(SKETCH.CREATE.TEXT
  [LAMBDA (STRING POSITION FONT JUSTIFICATION COLOR SCALE)        (* rrb " 4-Dec-85 20:51")
                                                                  (* creates a text element.)
    (CREATE.TEXT.ELEMENT (SK.INSURE.TEXT STRING)
           (SK.INSURE.POSITION POSITION)
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.STYLE JUSTIFICATION SK.DEFAULT.TEXT.ALIGNMENT)
           (SK.INSURE.FONT FONT)
           (SK.INSURE.COLOR COLOR])

(TEXT.CHANGEFN
  [LAMBDA (SCRNELTS SKW HOW)                                      (* rrb "10-Jan-85 16:58")
                                                                  (* the users has selected SCRNELT to be changed)
    (for ELTWITHTEXT inside SCRNELTS collect (SK.CHANGE.TEXT ELTWITHTEXT HOW SKW])

(TEXT.READCHANGEFN
  [LAMBDA (SKW SCRNELTS INTEXTBOXFLG)                             (* rrb " 3-Oct-86 15:26")

           (* the users has selected SCRNELT to be changed this function reads a specification of how the text elements should
           change.)

    (PROG ((COMMAND (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                    TITLE _ "Change text how?"
                                                    ITEMS _ [APPEND (COND
                                                                       [(SKETCHINCOLORP)
                                                                        '(("Color" 'BRUSHCOLOR "changes the color
                                                                                of the text"]
                                                                       (T NIL))
                                                                    [COND
                                                                       ((SCREENELEMENTP SCRNELTS)
                                                                        NIL)
                                                                       (T '(("look same" 'SAME "makes the font
                                                                                  characteristics the same as
                                                                                  those of the first selected
                                                                                  piece of text."]
                                                                    [COND
                                                                       ((AND (NULL INTEXTBOXFLG)
                                                                             (SKETCH.ELEMENT.TYPEP 'TEXTBOX))
                                                                        '(("box the text" 'BOX "makes the selected
                                                                                text into boxed text."]
                                                                    [COND
                                                                       ((DATATYPEP 'LOOKEDSTRING)
                                                                        '(("Fancy format" 'LOOKEDSTRING
                                                                                "changes to a form that can have
                                                                                complete character formatting."]
                                                                    '(("different font" 'NEWFONT "prompts for a
                                                                            new font family.")
                                                                      ("smaller font" 'SMALLER "Make the text
                                                                            smaller")
                                                                      ("LARGER FONT" 'LARGER "Make the text font
                                                                            larger.")
                                                                      ("set font size" 'SETSIZE "makes all fonts a
                                                                            prompted for size")
                                                                      ("set family & size" 'FAMILY&SIZE "allows
                                                                            changing both the family and the
                                                                            size")
                                                                      ("BOLD" 'BOLD "makes the text bold.")
                                                                      ("unbold" 'UNBOLD "removes the bold look of
                                                                            text.")
                                                                      ("italic" 'ITALIC "makes the text italic.")
                                                                      ("unitalic" 'UNITALIC "removes the italic
                                                                            look of text.")
                                                                      ("center justify" 'CENTER "centers the text
                                                                            about its location")
                                                                      ("left justify    " 'LEFT "left justifies
                                                                            the text to its location")
                                                                      ("    right justify" 'RIGHT "right justifies
                                                                            the text to its location.")
                                                                      ("top justify" 'TOP "makes the location be
                                                                            the top of the text.")
                                                                      ("bottom justify" 'BOTTOM "makes the
                                                                            location be the bottom of the text.")
```

```
                                                                              ("middle justify" 'MIDDLE "makes the control
                                                                                     point specify the mid-height of the
                                                                                     text.")
                                                                              ("baseline justify" 'BASELINE "makes the
                                                                                     control popint specify the baseline
                                                                                     of the text."]
                                                            CENTERFLG _ T)))
          FIRSTTEXTELT VAL)
         (OR COMMAND (RETURN))
         (SKED.CLEAR.SELECTION SKW)
         [SETQ VAL (SELECTQ COMMAND
                        (SETSIZE                                        (* read the new font size once)
                               (\SK.READ.FONT.SIZE1 SCRNELTS SKW))
                        (FAMILY&SIZE                                    (* gets both a font size and a family)
                                 (AND (SETQ VAL (SK.READFONTFAMILY SKW "New font family?"))
                                      (\SK.READ.FONT.SIZE1 SCRNELTS SKW VAL)))
                        (SAME                                           (* set the text characteristics from the first selection.)
                            (AND (SETQ FIRSTTEXTELT (for SCRNELT inside SCRNELTS
                                                       when (MEMB (fetch (SCREENELT GTYPE) of SCRNELT)
                                                                  '(TEXTBOX TEXT))
                                                       do (RETURN SCRNELT)))
                                 (fetch (SCREENELT GLOBALPART) of FIRSTTEXTELT)))
                        (NEWFONT                                        (* get a new font family)
                               (SK.READFONTFAMILY SKW "New font family?"))
                        (BRUSHCOLOR [READ.COLOR.CHANGE "Change text color how?" NIL
                                          (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                        (fetch (SCREENELT GLOBALPART)
                                                                           of (CAR SCRNELTS))
                                                                        'BRUSH])
                        (RETURN (LIST 'TEXT COMMAND]
         (RETURN (AND VAL (LIST COMMAND VAL])
```

(**\SK.READ.FONT.SIZE1**

```
  [LAMBDA (SELECTEDELTS SKETCHW NEWFAMILY)                              (* rrb "14-Jul-86 13:43")

                (* reads a font size from the user. If NEWFONT is NIL, use the one of the first selected element.)

    (PROG (FIRSTTEXTELT NEWSIZE NOWFONT NEWFONT)
         (OR (SETQ FIRSTTEXTELT (for SCRNELT inside SELECTEDELTS when (MEMB (fetch (SCREENELT GTYPE) of SCRNELT)
                                                                          '(TEXTBOX TEXT))
                                   do (RETURN SCRNELT)))
             (RETURN))
         (SETQ FIRSTTEXTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of FIRSTTEXTELT))
         (SETQ NOWFONT (fetch (TEXT FONT) of FIRSTTEXTELT))
         (STATUSPRINT SKETCHW "Size of " (COND
                                            ((SCREENELEMENTP SELECTEDELTS)
                                             "text")
                                            (T "first selected text"))
                " is "
                (FONTPROP NOWFONT 'SIZE))
         (SETQ NEWSIZE (SK.READFONTSIZE NIL [OR NEWFAMILY (SETQ NEWFAMILY (FONTPROP NOWFONT 'FAMILY]
                              SKETCHW))
         (RETURN (COND
                    ((NULL NEWSIZE)
                     (CLOSE.PROMPT.WINDOW SKETCHW)
                     NIL)
                    ((NULL (SETQ NEWFONT (FONTCREATE NEWFAMILY NEWSIZE (FONTPROP NOWFONT 'FACE)
                                                NIL NIL T)))
                     (STATUSPRINT SKETCHW NEWFAMILY NEWSIZE " not found.")
                     NIL)
                    (T (CLOSE.PROMPT.WINDOW SKETCHW)
                       (SK.FONTNAMELIST NEWFONT])
```

(**SK.TEXT.ELT.WITH.SAME.FIELDS**

```
  [LAMBDA (NEWONE ORGONE)                                               (* rrb "18-Jul-85 14:16")

                (* returns an element of the type of ORGONE whose text fields are the same as NEWONE.)

    (SELECTQ (fetch (INDIVIDUALGLOBALPART GTYPE) of ORGONE)
        (TEXT (create TEXT
                     LOCATIONLATLON _ (fetch (TEXT LOCATIONLATLON) of ORGONE)
                     LISTOFCHARACTERS _ (fetch (TEXT LISTOFCHARACTERS) of ORGONE)
                     INITIALSCALE _ (fetch (TEXT INITIALSCALE) of NEWONE)
                     TEXTSTYLE _ (fetch (TEXT TEXTSTYLE) of NEWONE)
                     FONT _ (fetch (TEXT FONT) of NEWONE)
                     LISTOFREGIONS _ (fetch (TEXT LISTOFREGIONS) of NEWONE)
                     TEXTCOLOR _ (fetch (TEXT TEXTCOLOR) of NEWONE)))
        (TEXTBOX (create TEXTBOX
                        TEXTBOXREGION _ (fetch (TEXTBOX TEXTBOXREGION) of ORGONE)
                        LISTOFCHARACTERS _ (fetch (TEXT LISTOFCHARACTERS) of ORGONE)
                        INITIALSCALE _ (fetch (TEXT INITIALSCALE) of NEWONE)
                        TEXTSTYLE _ (fetch (TEXT TEXTSTYLE) of NEWONE)
                        FONT _ (fetch (TEXT FONT) of NEWONE)
                        LISTOFREGIONS _ (fetch (TEXT LISTOFREGIONS) of NEWONE)
                        TEXTCOLOR _ (fetch (TEXT TEXTCOLOR) of NEWONE)
```

```
                       TEXTBOXBRUSH _ (fetch (TEXTBOX TEXTBOXBRUSH) of ORGONE)))
         NIL])
```

(**SK.READFONTFAMILY**

```
  [LAMBDA (SKW TITLE)                                             (* rrb "21-Nov-85 11:28")
                                                                  (* reads a font family name.)
    (PROG ([KNOWNFAMILIES (UNION (for X in \FONTSONFILE collect (CAR X))
                                 (for X in \FONTSINCORE collect (CAR X]
           FAMILY)                                                (* offers a menu of possible choices.)
          (COND
             ((AND KNOWNFAMILIES (NEQ (SETQ FAMILY (\CURSOR.IN.MIDDLE.MENU
                                                    (create MENU
                                                           ITEMS _ (APPEND '(("other" 'OTHER "prompts for a
                                                                                   family not on the menu."))
                                                                          KNOWNFAMILIES)
                                                           TITLE _ (OR TITLE "Choose font")
                                                           CENTERFLG _ T)))
                                      'OTHER))
              (RETURN FAMILY))
             (T                                                   (* grab the tty.)
                (TTY.PROCESS (THIS.PROCESS))
                (RETURN (CAR (ERSETQ (MKATOM (U-CASE (PROMPTFORWORD "New family: " NIL NIL (GETPROMPTWINDOW
                                                                                          SKW])
```

(**CLOSE.PROMPT.WINDOW**

```
  [LAMBDA (WINDOW)                                                (* rrb "28-Oct-84 14:14")
                                                                  (* gets rid of the prompt window.)
    (PROG (PRMPTWIN)
          (RETURN (COND
                     ((SETQ PRMPTWIN (GETPROMPTWINDOW WINDOW NIL NIL T))
                      (DETACHWINDOW PRMPTWIN)
                      (CLOSEW PRMPTWIN])
```

(**TEXT.DRAWFN**

```
  [LAMBDA (TEXTELT WINDOW)                                        (* rrb " 9-Aug-85 09:38")
                                                                  (* shows a text element)
    (TEXT.DRAWFN1 (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of (fetch (SCREENELT LOCALPART) of TEXTELT))
           (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of TEXTELT))
           (fetch (LOCALTEXT LOCALFONT) of (fetch (SCREENELT LOCALPART) of TEXTELT))
           (fetch (TEXT TEXTCOLOR) of (fetch (SCREENELT INDIVIDUALGLOBALPART) of TEXTELT))
           WINDOW])
```

(**TEXT.DRAWFN1**

```
  [LAMBDA (STRS REGIONS FONT COLOR SKWINDOW OPERATION)            ; Edited  3-Oct-89 13:48 by rmk:
                                                                  ; Edited  3-Oct-89 13:47 by rmk:
                                                                  (* rrb " 3-Mar-86 21:37")

    ;; draws the image of a list of string in their local regions on a sketch window.  It is broken out as a subfunction so that it can be called by the
    ;; update function also.

    (COND
       ((AND COLOR (SKETCHINCOLORP))
        (DSPCOLOR COLOR SKWINDOW)))
    (PROG (DESCENT OLDFONT)
          (COND
             ((NULL FONT)                                         ; text is too small or too large to be at this scale.
              (RETURN))
             ((FONTP FONT)                                        ; font was found.
              (SETQ OLDFONT (DSPFONT FONT SKWINDOW))              ; Install font, then refetch it from window/stream, in case there
                                                                  ; was device coercion, so descent will be right.
              (SETQ DESCENT (FONTPROP (DSPFONT NIL SKWINDOW)
                                   'DESCENT))
              (DSPOPERATION (PROG1 (DSPOPERATION OPERATION SKWINDOW)
                                   (RESETFORM (SETTERMTABLE SKETCH.TERMTABLE)
                                          (for REGION in REGIONS as CHARS in STRS
                                             do (MOVETO (fetch (REGION LEFT) of REGION)
                                                       (PLUS (fetch (REGION BOTTOM) of REGION)
                                                             DESCENT)
                                                       SKWINDOW)
                                                (PRIN3 CHARS SKWINDOW))))
                     SKWINDOW)                                    ; return to original font so that messages come out ok.
              (DSPFONT OLDFONT SKWINDOW))
             (T                                                   ; if no font, just gray in regions

;;; This code was left by RRB on the theory that hardcopy can't support bitblt, which I think is wrong--RMK.  (COND ((EQ (IMAGESTREAMTYPE
;;; SKWINDOW) 'DISPLAY) (for REGION in REGIONS do (BITBLT NIL NIL NIL SKWINDOW (fetch LEFT of REGION) (fetch BOTTOM of REGION)
;;; (fetch WIDTH of REGION) (IQUOTIENT (ADD1 (fetch HEIGHT of REGION)) 2) 'TEXTURE OPERATION INDICATE.TEXT.SHADE))) (T ; hardcopy
;;; can't support bitblt, draw a line instead. (bind MIDHGHT for REGION in REGIONS do (DRAWLINE (fetch LEFT of REGION) (SETQ MIDHGHT (PLUS
;;; (fetch BOTTOM of REGION) (IQUOTIENT (ADD1 (fetch HEIGHT of REGION)) 2))) (fetch RIGHT of REGION) MIDHGHT (fetch HEIGHT of REGION)
;;; OPERATION SKWINDOW))))

                (for REGION in REGIONS do (BITBLT NIL NIL NIL SKWINDOW (fetch LEFT of REGION)
                                                 (fetch BOTTOM of REGION)
                                                 (fetch WIDTH of REGION)
```

```
                                                  (IQUOTIENT (ADD1 (fetch HEIGHT of REGION))
                                                         2)
                                                  'TEXTURE OPERATION INDICATE.TEXT.SHADE])
```

(TEXT.INSIDEFN

```
  [LAMBDA (GTEXT WREG)                                          (* rrb " 5-AUG-83 16:54")
                                                                (* determines if the global text element is inside of WREG.)
    (for GREG in (fetch (TEXT LISTOFREGIONS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXT))
       when (REGIONSINTERSECTP GREG WREG) do (RETURN T])
```

(TEXT.EXPANDFN

```
  [LAMBDA (GTEXTPART SCALE STREAM)                              (* rrb "19-Mar-86 15:59")
                                                                (* creates a local text screen element from a global text
                                                                element.)
    (PROG ((GTEXT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTPART))
           TEXTPOS LOCALFONT STYLE LINEREGIONS)
          [COND
             ((NLISTP (SETQ STYLE (fetch (TEXT TEXTSTYLE) of GTEXT)))
                                                                (* old format had horizontal positioning only, now has vertical
                                                                too. Fill in old default.)
              (replace (TEXT TEXTSTYLE) of GTEXT with (SETQ STYLE '(CENTER CENTER]
          (SETQ LOCALFONT (SK.CHOOSE.TEXT.FONT GTEXT SCALE STREAM))
          [SETQ LINEREGIONS (SK.TEXT.LINE.REGIONS (fetch (TEXT LISTOFCHARACTERS) of GTEXT)
                                     (SETQ TEXTPOS (SK.SCALE.POSITION.INTO.VIEWER (fetch (TEXT LOCATIONLATLON)
                                                                                         of GTEXT)
                                                          SCALE))
                                     (fetch (TEXT LISTOFREGIONS) of GTEXT)
                                     LOCALFONT STYLE SCALE (COND
                                                              ((STREAMP STREAM))
                                                              (T (WINDOWPROP STREAM 'DSP]
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALTEXT
                                            DISPLAYPOSITION _ TEXTPOS
                                            LINEREGIONS _ LINEREGIONS
                                            LOCALFONT _ LOCALFONT
                                            LOCALLISTOFCHARACTERS _ (APPEND (fetch (TEXT LISTOFCHARACTERS)
                                                                                  of GTEXT)))
                         GLOBALPART _ GTEXTPART])
```

(SK.TEXT.LINE.REGIONS

```
  [LAMBDA (LISTOFTEXT TEXTPOS GREGIONS LOCALFONT STYLE SCALE IMAGESTREAM)
                                                                (* rrb "19-Mar-86 15:44")

          (* calculates the list of regions that each line of text in LISTOFTEXT will occupy.
          Used by both TEXT.EXPANDFN and TEXTBOX.EXPANDFN. Captures those things which are common to the two
          elements.)

    (COND
       [(FONTP LOCALFONT)
        (LTEXT.LINE.REGIONS LISTOFTEXT TEXTPOS (COND
                                                  ((IMAGESTREAMTYPEP IMAGESTREAM 'HARDCOPY)

          (* actually make the font be the font of the stream so that the stream can be passed to STRINGWIDTH to get hardcopy
          characteristics.)

                                                   (DSPFONT LOCALFONT IMAGESTREAM)
                                                   IMAGESTREAM)
                                                  (T LOCALFONT))
               STYLE
               (FIXR (TIMES (QUOTIENT (fetch (REGION HEIGHT) of (CAR GREGIONS))
                                   SCALE)
                            (LENGTH LISTOFTEXT]
       (T (for GREG in GREGIONS collect (CREATEREGION (FIXR (QUOTIENT (fetch (REGION LEFT) of GREG)
                                                                    SCALE))
                                               (FIXR (QUOTIENT (fetch (REGION BOTTOM) of GREG)
                                                            SCALE))
                                               (FIXR (QUOTIENT (fetch (REGION WIDTH) of GREG)
                                                            SCALE))
                                               1])
```

(TEXT.UPDATE.GLOBAL.REGIONS

```
  [LAMBDA (GTEXTELT NEWGPOS OLDGPOS)                            (* rrb "12-Sep-84 11:36")
                                                                (* updates the list of regions occupied by the text in the global
                                                                coordinate space.)
                                                                (* this is used to determine the extent of a text element in a
                                                                region.)
    (PROG ((XDIFF (DIFFERENCE (fetch (POSITION XCOORD) of NEWGPOS)
                         (fetch (POSITION XCOORD) of OLDGPOS)))
           (YDIFF (DIFFERENCE (fetch (POSITION YCOORD) of NEWGPOS)
                         (fetch (POSITION YCOORD) of OLDGPOS)))
           (INDTEXTGELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT)))
          (replace (TEXT LISTOFREGIONS) of INDTEXTGELT with (for GREG in (fetch (TEXT LISTOFREGIONS) of INDTEXTGELT)
                                                                  collect (REL.MOVE.REGION GREG XDIFF YDIFF)))
          (RETURN GTEXTELT])

(REL.MOVE.REGION
  [LAMBDA (REGION DELTAX DELTAY)                                    (* rrb "15-AUG-83 17:30")
                                                                    (* moves a region by an amount DELTAX DELTAY)
    (CREATEREGION (PLUS DELTAX (fetch (REGION LEFT) of REGION))
           (PLUS DELTAY (fetch (REGION BOTTOM) of REGION))
           (fetch (REGION WIDTH) of REGION)
           (fetch (REGION HEIGHT) of REGION])

(LTEXT.LINE.REGIONS
  [LAMBDA (LINES LPOSITION STREAMORFONT STYLE TOTALHEIGHT)          (* rrb " 4-Dec-85 11:51")

          (* returns the regions occupied by the lines of text LINES to format them in STYLE in font FONT at position LPOSITION.)

    (AND STREAMORFONT (PROG ((FONT (FONTCREATE STREAMORFONT))
                             (TEXTXPOS (fetch (POSITION XCOORD) of LPOSITION))
                             (TEXTYPOS (fetch (POSITION YCOORD) of LPOSITION))
                             HEIGHT HEIGHTOFLOCALTEXT LINEWIDTH)
                            [SETQ HEIGHT (COND
                                            ((STREAMP STREAMORFONT)
                                                                    (* use the line feed height because in hardcopy streams this is
                                                                    more correct.)
                                             (MINUS (DSPLINEFEED NIL STREAMORFONT)))
                                            (T (FONTPROP FONT 'HEIGHT]
                            (SETQ HEIGHTOFLOCALTEXT (TIMES HEIGHT (LENGTH LINES)))
                            (RETURN (for CHARS in LINES as Y
                                       from [PLUS TEXTYPOS (SELECTQ (CADR STYLE)
                                                                   (BASELINE

          (* vertically center the baseline. The baseline alignment should probably be independent of the top -
          bottom alignment eventually.)

                                                                             [DIFFERENCE (DIFFERENCE (QUOTIENT
                                                                                                      HEIGHTOFLOCALTEXT
                                                                                                               2.0)
                                                                                                  HEIGHT)
                                                                                    (MINUS (FONTPROP FONT
                                                                                                     'DESCENT])
                                                                   (CENTER (DIFFERENCE (QUOTIENT HEIGHTOFLOCALTEXT
                                                                                                  2.0)
                                                                                  HEIGHT))
                                                                   (TOP (DIFFERENCE 1 HEIGHT))
                                                                   (BOTTOM (DIFFERENCE HEIGHTOFLOCALTEXT HEIGHT))
                                                                   (ERROR "illegal vertical text style" (CADR STYLE]
                                       by (IMINUS HEIGHT)
                                       collect [SETQ LINEWIDTH (DIFFERENCE (STRINGWIDTH CHARS STREAMORFONT)
                                                                      (COND
                                                                         ((EQ (NTHCHARCODE CHARS -1)
                                                                              (CHARCODE CR))
                                                                          (CHARWIDTH (CHARCODE CR)
                                                                                 STREAMORFONT))
                                                                         (T 0]
                                             (CREATEREGION (SELECTQ (CAR STYLE)
                                                                (CENTER (DIFFERENCE TEXTXPOS (QUOTIENT LINEWIDTH
                                                                                                        2.0)))
                                                                (LEFT TEXTXPOS)
                                                                (DIFFERENCE TEXTXPOS LINEWIDTH))
                                                    Y LINEWIDTH HEIGHT])

(TEXT.INPUTFN
  [LAMBDA (WINDOW)                                                  (* rrb "12-Dec-84 11:44")

          (* reads text and a place to put it from the user and returns a TEXTELT that represents it.
          Can return NIL if the user positions it outside of the window.)

    (TEXT.POSITION.AND.CREATE (READ.TEXT "Text to be added: ")
           (fetch (SKETCHCONTEXT SKETCHFONT) of (WINDOWPROP WINDOW 'SKETCHCONTEXT))
           WINDOW "locate the text"])

(READ.TEXT
  [LAMBDA (PRMPT)                                                   (* rrb " 9-AUG-83 12:42")
    (PROG ([CLOSEWFLG (COND
                         ((EQ (TTYDISPLAYSTREAM)
                              \DEFAULTTTYDISPLAYSTREAM)
                          T)
                         ((AND (WFROMDS (TTYDISPLAYSTREAM))
                               (NOT (OPENWP (TTYDISPLAYSTREAM]
           LST)
          (AND PRMPT (PRIN1 PRMPT T))
          (SETQ LST (CONS (READ T)
                          (READLINE)))
          (AND CLOSEWFLG (CLOSEW (TTYDISPLAYSTREAM)))
          (RETURN (APPLY (FUNCTION CONCAT)
                         (CONS (CAR LST)
                               (for WORD in (CDR LST) join (LIST '%  WORD])
```

(**TEXT.POSITION.AND.CREATE**

```
  [LAMBDA (TEXT FONT WINDOW PROMPTMSG)                          (* rrb "16-Oct-85 18:29")

          (* gets a position for a piece of text from the user and returns a text element that represents it.
          The text location is the center position of the text.)
                                                                (* later this should change the cursor to the image being placed.)
    (PROG [P1 LOCATION DISPLAYPOSITION (SCALE (SK.INPUT.SCALE WINDOW))
              NEW.BITMAP DSP (WDTH (STRINGWIDTH TEXT FONT))
              (HGHT (FONTHEIGHT FONT))
              (TEXTALIGNMENT (fetch (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of (WINDOWPROP WINDOW 'SKETCHCONTEXT]
          (SETQ NEW.BITMAP (BITMAPCREATE WDTH HGHT))
          (SETQ DSP (DSPCREATE NEW.BITMAP))
          (DSPFONT FONT DSP)
          (MOVETO 0 (FONTDESCENT FONT)
                 DSP)
          (PRIN3 TEXT DSP)
          [SETQ P1 (GET.BITMAP.POSITION WINDOW NEW.BITMAP 'PAINT PROMPTMSG
                          (IMINUS (SELECTQ (CAR TEXTALIGNMENT)
                                      (CENTER (LRSH (ADD1 WDTH)
                                                    1))
                                      (LEFT 0)
                                      (SUB1 WDTH)))
                          (IMINUS (SELECTQ (CADR TEXTALIGNMENT)
                                      (BASELINE (FONTPROP FONT 'DESCENT))
                                      (CENTER (LRSH (ADD1 HGHT)
                                                    1))
                                      (TOP (SUB1 HGHT))
                                      0]                        (* scale range goes from 20 to 1.0 Use FONT as an initial.)
          (RETURN (AND P1 (CREATE.TEXT.ELEMENT (CONS TEXT)
                                 (SK.MAP.INPUT.PT.TO.GLOBAL P1 WINDOW)
                                 SCALE TEXTALIGNMENT FONT (fetch (BRUSH BRUSHCOLOR)
                                                              of (fetch (SKETCHCONTEXT SKETCHBRUSH)
                                                                    of (WINDOWPROP WINDOW 'SKETCHCONTEXT])
```

(**CREATE.TEXT.ELEMENT**

```
  [LAMBDA (STRLST GPOSITION SCALE JUSTIFICATION FONT COLOR)     (* rrb " 4-Dec-85 20:50")
                                                                (* creates a text element for a sketch)
    (SK.UPDATE.TEXT.AFTER.CHANGE (create GLOBALPART
                                         INDIVIDUALGLOBALPART _
                                         (create TEXT
                                                LOCATIONLATLON _ GPOSITION
                                                LISTOFCHARACTERS _ STRLST
                                                INITIALSCALE _ SCALE
                                                TEXTSTYLE _ JUSTIFICATION
                                                FONT _ FONT
                                                TEXTCOLOR _ COLOR])
```

(**SK.UPDATE.TEXT.AFTER.CHANGE**

```
  [LAMBDA (GTEXTELT)                                            (* rrb " 4-Dec-85 20:50")

          (* updates the dependent fields in a text element that has had its text field changed.)

    (TEXT.SET.GLOBAL.REGIONS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT))
    (TEXT.SET.SCALES GTEXTELT)
    GTEXTELT])
```

(**SK.TEXT.FROM.TEXTBOX**

```
  [LAMBDA (TEXTBOXELT SKW)                                      (* rrb "30-Sep-86 18:34")
                                                                (* returns change event spec with a textbox that replaces
                                                                GTEXTBOXELT.)
    (PROG ((INDTEXTBOXELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of TEXTBOXELT))
           TEXTSTYLE REGION NEWTEXTELT)
          (SETQ TEXTSTYLE (fetch (TEXTBOX TEXTSTYLE) of INDTEXTBOXELT))
          (SETQ REGION (APPLY (FUNCTION SK.UNIONREGIONS)
                              (fetch (TEXTBOX LISTOFREGIONS) of INDTEXTBOXELT)))
          (SETQ NEWTEXTELT (CREATE.TEXT.ELEMENT (ADD.EOLS (fetch (TEXTBOX LISTOFCHARACTERS) of INDTEXTBOXELT))
                                  (MAP.GLOBAL.PT.ONTO.GRID [create POSITION
                                                                   XCOORD _
                                                                   (SELECTQ (CAR TEXTSTYLE)
                                                                        (LEFT (fetch (REGION LEFT) of REGION))
                                                                        (RIGHT (fetch (REGION RIGHT) of REGION))
                                                                        (PLUS (fetch (REGION LEFT) of REGION)
                                                                              (QUOTIENT (fetch (REGION WIDTH)
                                                                                               of REGION)
                                                                                     2)))
                                                                   YCOORD _
                                                                   (SELECTQ (CADR TEXTSTYLE)
                                                                        (TOP (fetch (REGION TOP) of REGION))
                                                                        (BOTTOM (fetch (REGION BOTTOM) of REGION))
```

```
                                                                               (PLUS (fetch (REGION BOTTOM) of REGION)
                                                                                     (QUOTIENT (fetch (REGION HEIGHT)
                                                                                                  of REGION)
                                                                                            2]
                                              SKW)
                                         (fetch (TEXTBOX INITIALSCALE) of INDTEXTBOXELT)
                                         (COND
                                            ((EQ (CADR TEXTSTYLE)
                                                 'CENTER)

          (* make center into baseline because it looks better and because it is converted the other direction.)

                                             (LIST (CAR TEXTSTYLE)
                                                   'BASELINE))
                                            (T TEXTSTYLE))
                                         (fetch (TEXTBOX FONT) of INDTEXTBOXELT)
                                         (fetch (TEXTBOX TEXTCOLOR) of INDTEXTBOXELT)))
          (RETURN (create SKHISTORYCHANGESPEC
                         NEWELT _ NEWTEXTELT
                         OLDELT _ TEXTBOXELT
                         PROPERTY _ 'HASBOX
                         NEWVALUE _ NEWTEXTELT
                         OLDVALUE _ TEXTBOXELT])
```

(TEXT.SET.GLOBAL.REGIONS

```
  [LAMBDA (GTEXTELT)                                                    (* rrb "29-Jan-85 14:50")
                                                                        (* updates the list of regions occupied by the text in the global
                                                                        coordinate space.)
                                                                        (* this is used to determine the extent of a text element in a
                                                                        region.)
    (PROG ((SCALE (fetch (TEXT INITIALSCALE) of GTEXTELT)))
          (replace (TEXT LISTOFREGIONS) of GTEXTELT with (for LREG
                                                            in [LTEXT.LINE.REGIONS
                                                                (fetch (TEXT LISTOFCHARACTERS) of GTEXTELT)
                                                                (SK.SCALE.POSITION.INTO.VIEWER (fetch (TEXT
                                                                                                       LOCATIONLATLON
                                                                                                         )
                                                                                                  of GTEXTELT)
                                                                       SCALE)
                                                                (fetch (TEXT FONT) of GTEXTELT)
                                                                (fetch (TEXT TEXTSTYLE) of GTEXTELT)
                                                                (ITIMES (FONTHEIGHT (fetch (TEXT FONT) of GTEXTELT))
                                                                       (LENGTH (fetch (TEXT LISTOFCHARACTERS)
                                                                                  of GTEXTELT]
                                                            collect (UNSCALE.REGION LREG SCALE)))
          (RETURN GTEXTELT])
```

(TEXT.REGIONFN

```
  [LAMBDA (SCRTEXTELT)                                                  (* rrb " 2-Oct-84 16:36")
                                                                        (* determines the local region covered by TEXTELT.)
    (PROG [REG (LINEREGIONS (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of SCRTEXTELT]
          (RETURN (COND
                     ((NULL LINEREGIONS)
                      NIL)
                     (T (SETQ REG (CAR LINEREGIONS))
                        (for LINEREG in (CDR LINEREGIONS) do (SETQ REG (UNIONREGIONS REG LINEREG)))
                        REG])
```

(TEXT.GLOBALREGIONFN

```
  [LAMBDA (GTEXTELT)                                                    (* rrb "18-Oct-85 16:43")
                                                                        (* returns the global region occupied by a global text element.)
    (PROG [REG (LINEREGIONS (fetch (TEXT LISTOFREGIONS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT]
          (RETURN (COND
                     ((NULL LINEREGIONS)
                      NIL)
                     (T (SETQ REG (CAR LINEREGIONS))
                        (for LINEREG in (CDR LINEREGIONS) do (SETQ REG (UNIONREGIONS REG LINEREG)))
                        REG])
```

(TEXT.TRANSLATEFN

```
  [LAMBDA (GTEXT DELTAPOS WINDOW)                                       (* rrb "28-Apr-85 18:45")
                                                                        (* moves a text figure element to a new position.)
    (PROG ((INDTEXTELT (COPY (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXT)))
           NEWGPOS NEWTEXTELT)                                          (* update the region positions.)
          (TEXT.UPDATE.GLOBAL.REGIONS (SETQ NEWTEXTELT (create GLOBALPART
                                                                COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART

                                                                                                   COMMONGLOBALPART
                                                                                                    )
                                                                                              of GTEXT))
                                                                INDIVIDUALGLOBALPART _ INDTEXTELT))
                 (SETQ NEWGPOS (PTPLUS DELTAPOS (fetch (TEXT LOCATIONLATLON) of INDTEXTELT)))
                 (fetch (TEXT LOCATIONLATLON) of INDTEXTELT))
          (replace (TEXT LOCATIONLATLON) of INDTEXTELT with NEWGPOS)
          (RETURN NEWTEXTELT])

(TEXT.TRANSFORMFN
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "16-Oct-85 18:30")

          (* returns a copy of the global TEXT element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.
          SCALEFACTOR is the amount the transformation scales by and is used to reset the size of the text.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (CREATE.TEXT.ELEMENT (fetch (TEXT LISTOFCHARACTERS) of INDVPART)
                         (SK.TRANSFORM.POINT (fetch (TEXT LOCATIONLATLON) of INDVPART)
                                TRANSFORMFN TRANSFORMDATA)
                         (FTIMES (fetch (TEXT INITIALSCALE) of INDVPART)
                                SCALEFACTOR)
                         (fetch (TEXT TEXTSTYLE) of INDVPART)
                         (fetch (TEXT FONT) of INDVPART)
                         (fetch (TEXT TEXTCOLOR) of INDVPART])

(TEXT.TRANSLATEPTSFN
  [LAMBDA (TEXTELT SELPTS GDELTA WINDOW)                        (* rrb " 5-May-85 18:05")
                                                               (* returns a text element that has its position translated.)

          (* shouldn't ever happen because a text element only has one control pt and its translatefn should get used.)

    (fetch (SCREENELT GLOBALPART) of TEXTELT])

(TEXT.UPDATEFN
  [LAMBDA (OLDLOCALELT NEWGELT SKETCHW)                         (* rrb "11-Jul-86 15:51")

          (* update function for text. Tries to repaint only the lines of text that have changed.)

    (PROG ((NEWTEXTELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT))
           (OLDTEXTELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of OLDLOCALELT))
           LOCALTEXT NEWSCREENELT)
          (COND
             ((AND (EQUAL (fetch (TEXT FONT) of NEWTEXTELT)
                          (fetch (TEXT FONT) of OLDTEXTELT))
                   (EQUAL (fetch (TEXT TEXTSTYLE) of NEWTEXTELT)
                          (fetch (TEXT TEXTSTYLE) of OLDTEXTELT))
                   (EQUAL (fetch (TEXT LOCATIONLATLON) of NEWTEXTELT)
                          (fetch (TEXT LOCATIONLATLON) of OLDTEXTELT))
                   (EQUAL (fetch (TEXT INITIALSCALE) of NEWTEXTELT)
                          (fetch (TEXT INITIALSCALE) of OLDTEXTELT))
                   (EQUAL (LENGTH (fetch (TEXT LISTOFCHARACTERS) of NEWTEXTELT))
                          (LENGTH (fetch (TEXT LISTOFCHARACTERS) of OLDTEXTELT)))
                   (EQUAL (fetch (TEXT TEXTCOLOR) of NEWTEXTELT)
                          (fetch (TEXT TEXTCOLOR) of OLDTEXTELT)))
                                                               (* if font, style or number of lines has changed, erase and
                                                               redraw.)
              (SETQ LOCALTEXT (fetch (SCREENELT LOCALPART) of OLDLOCALELT))
              (SETQ NEWSCREENELT (SK.ADD.ITEM NEWGELT SKETCHW))  (* update the screen display)
              [PROG ((NEWSTRS (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of (fetch (SCREENELT LOCALPART)
                                                                                of NEWSCREENELT)))
                     (OLDSTRS (fetch (LOCALTEXT LOCALLISTOFCHARACTERS) of LOCALTEXT))
                     (NEWLOCALREGIONS (fetch (LOCALTEXT LINEREGIONS) of (fetch (SCREENELT LOCALPART) of
                                                                                                 NEWSCREENELT
                                                                            )))
                     (OLDLOCALREGIONS (fetch (LOCALTEXT LINEREGIONS) of LOCALTEXT)))
                    (COND
                       ((NEQ (LENGTH NEWSTRS)
                             (LENGTH OLDSTRS))

          (* creating the new element caused the line filling routines to change the number of lines so the partial redrawing algorithms
          don't work and we have to redraw the whole element. Do this by erasing the old one then drawing the new one.)

                        (SK.ERASE.ELT OLDLOCALELT SKETCHW)
                        (SK.DRAWFIGURE NEWSCREENELT SKETCHW NIL (VIEWER.SCALE SKETCHW))
                        (RETURN NEWSCREENELT)))
                LP  (COND
                       ((OR NEWSTRS OLDSTRS)                   (* continue until both new and old are exhausted.)
                        [COND
                           ([NOT (AND (EQUAL (CAR NEWSTRS)
                                             (CAR OLDSTRS))
                                      (EQUAL (CAR NEWLOCALREGIONS)
                                             (CAR OLDLOCALREGIONS]
                                                               (* this line is the different, redraw it.)
                            (AND OLDLOCALREGIONS (DSPFILL (CAR OLDLOCALREGIONS)
                                                        BLACKSHADE
                                                        'ERASE SKETCHW))
                            (AND NEWSTRS (TEXT.DRAWFN1 (LIST (CAR NEWSTRS))
                                                (LIST (CAR NEWLOCALREGIONS))
                                                (fetch (LOCALTEXT LOCALFONT) of LOCALTEXT)
```

```
                                                  (fetch (TEXT TEXTCOLOR) of OLDTEXTELT)
                                                  SKETCHW]
                        (SETQ NEWSTRS (CDR NEWSTRS))
                        (SETQ OLDSTRS (CDR OLDSTRS))
                        (SETQ NEWLOCALREGIONS (CDR NEWLOCALREGIONS))
                        (SETQ OLDLOCALREGIONS (CDR OLDLOCALREGIONS))
                        (GO LP]
              (RETURN NEWSCREENELT])
```

(**SK.CHANGE.TEXT**

```
  [LAMBDA (ELTWITHTEXT HOW SKW)                                    ; Edited  7-Apr-87 13:41 by rrb
    (PROG ((COMMAND (CADR HOW))
           (PROPERTY 'FONT)
           NEWVALUE GINDTEXTELT NEWGTEXT OLDVALUE OLDFACE GTYPE)
          (OR HOW (RETURN))                                        (* take down the caret before any change.)
          (SKED.CLEAR.SELECTION SKW)
          (COND
             ((MEMB (SETQ GTYPE (fetch (GLOBALPART GTYPE) of ELTWITHTEXT))
                    '(TEXTBOX TEXT))
              (SETQ GINDTEXTELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHTEXT))

          (* set the old value to the old font. In the case where the thing that changes is the justification, this will get re-set)

              (SETQ OLDVALUE (fetch (TEXT FONT) of GINDTEXTELT))
              (SETQ NEWGTEXT (SELECTQ (CAR HOW)
                                 (TEXT (SELECTQ COMMAND
                                           ((SMALLER LARGER)       (* change the font)
                                                [COND
                                                   [[SETQ NEWVALUE (SK.NEXTSIZEFONT COMMAND
                                                                          (LIST (FONTPROP OLDVALUE 'FAMILY)
                                                                                (FONTPROP OLDVALUE 'SIZE]
                                                                   (* if there is an appropriate size font, use it.)
                                                    [SETQ NEWVALUE (LIST (FONTPROP NEWVALUE 'FAMILY)
                                                                         (FONTPROP NEWVALUE 'SIZE)
                                                                         (FONTPROP OLDVALUE 'FACE]
                                                    (COND
                                                       ((EQ GTYPE 'TEXT)
                                                        (create TEXT using GINDTEXTELT FONT _ NEWVALUE))
                                                       (T (create TEXTBOX using GINDTEXTELT FONT _ NEWVALUE]
                                                   (T              (* otherwise just scale the area some.)
                                                      (SETQ NEWVALUE (FTIMES (SETQ OLDVALUE (fetch (TEXT
                                                                                                    INITIALSCALE
                                                                                                         )
                                                                                               of GINDTEXTELT))
                                                                            (SELECTQ COMMAND
                                                                                (LARGER 1.4)
                                                                                0.7142858)))
                                                      (SETQ PROPERTY 'SCALE)
                                                      (COND
                                                         ((EQ GTYPE 'TEXT)
                                                          (create TEXT using GINDTEXTELT INITIALSCALE _ NEWVALUE)
                                                          )
                                                         (T (create TEXTBOX using GINDTEXTELT INITIALSCALE _
                                                                       NEWVALUE])
                                           ((CENTER LEFT RIGHT)
                                                                   (* change the horizontal justification)
                                                [SETQ NEWVALUE (LIST COMMAND (CADR (SETQ OLDVALUE
                                                                                    (fetch (TEXT TEXTSTYLE)
                                                                                       of GINDTEXTELT]
                                                (SETQ PROPERTY 'JUSTIFICATION)
                                                (COND
                                                   ((EQ GTYPE 'TEXT)
                                                    (create TEXT using GINDTEXTELT TEXTSTYLE _ NEWVALUE))
                                                   (T (create TEXTBOX using GINDTEXTELT TEXTSTYLE _ NEWVALUE))))
                                           ((TOP BOTTOM MIDDLE BASELINE)
                                                                   (* change the vertical justification)
                                                [SETQ NEWVALUE (LIST (CAR (SETQ OLDVALUE (fetch (TEXT TEXTSTYLE)
                                                                                            of GINDTEXTELT)))
                                                                     (COND
                                                                        ((EQ COMMAND 'MIDDLE)
                                                                         'CENTER)
                                                                        (T COMMAND]
                                                (SETQ PROPERTY 'JUSTIFICATION)
                                                (COND
                                                   ((EQ GTYPE 'TEXT)
                                                    (create TEXT using GINDTEXTELT TEXTSTYLE _ NEWVALUE))
                                                   (T (create TEXTBOX using GINDTEXTELT TEXTSTYLE _ NEWVALUE))))
                                           ((BOLD UNBOLD ITALIC UNITALIC)
                                                                   (* change the face)
                                                (SETQ OLDFACE (FONTPROP OLDVALUE 'FACE))
                                                [SETQ NEWVALUE (LIST (FONTPROP OLDVALUE 'FAMILY)
                                                                     (FONTPROP OLDVALUE 'SIZE)
                                                                     (LIST (SELECTQ COMMAND
                                                                               (BOLD 'BOLD)
                                                                               (UNBOLD 'MEDIUM)
                                                                               (CAR OLDFACE))
                                                                   (SELECTQ COMMAND
                                                                               (ITALIC 'ITALIC)
                                                                               (UNITALIC 'REGULAR)
                                                                               (CADR OLDFACE))
                                                                   (CADDR OLDFACE]
                                                (COND
                                                   ((EQ GTYPE 'TEXT)
                                                    (create TEXT using GINDTEXTELT FONT _ NEWVALUE))
                                                   (T (create TEXTBOX using GINDTEXTELT FONT _ NEWVALUE))))
                                           (BOX                (* if it is a text element, BOX it)
                                                [COND
                                                   ((EQ GTYPE 'TEXT)
                                                    (RETURN (SK.TEXTBOX.FROM.TEXT ELTWITHTEXT SKW])
                                           (UNBOX              (* if it is a text box, unbox it.)
                                                  [COND
                                                     ((EQ GTYPE 'TEXTBOX)
                                                      (RETURN (SK.TEXT.FROM.TEXTBOX ELTWITHTEXT SKW])
                                           (LOOKEDSTRING [COND
                                                            ((EQ GTYPE 'TEXT)
                                                             (RETURN (SK.LOOKEDSTRING.FROM.TEXT ELTWITHTEXT SKW
                                                                            ])
                                           (SHOULDNT)))
                                 (SETSIZE (SETQ NEWVALUE COMMAND)
                                          (COND
                                             [(EQ (FONTPROP NEWVALUE 'FAMILY)
                                                  (FONTPROP OLDVALUE 'FAMILY))
```

(* if the families are the same, change them, otherwise don't as it isn't known whether or not this family has the right size.)

```
                                     (COND
                                        [(EQ GTYPE 'TEXT)
                                         (create TEXT using GINDTEXTELT FONT _
                                                                  (LIST (FONTPROP OLDVALUE 'FAMILY)
                                                                        (FONTPROP NEWVALUE 'SIZE)
                                                                        (FONTPROP OLDVALUE 'FACE]
                                        (T (create TEXTBOX using GINDTEXTELT FONT _
                                                                       (LIST (FONTPROP OLDVALUE 'FAMILY)
                                                                             (FONTPROP NEWVALUE 'SIZE)
                                                                             (FONTPROP OLDVALUE 'FACE]
                                    (T (RETURN))))
                        (NEWFONT                       (* set the font family)
                                 [SETQ NEWVALUE (LIST COMMAND (FONTPROP OLDVALUE 'SIZE)
                                                      (FONTPROP OLDVALUE 'FACE]
                                 (COND
                                    ((NULL (FONTCREATE NEWVALUE NIL NIL NIL NIL T))
                                     (STATUSPRINT SKW "  Couldn't find " (CAR NEWVALUE)
                                            " in size "
                                            (CADR NEWVALUE))
                                     (RETURN)))
                                 (COND
                                    ((EQ GTYPE 'TEXT)
                                     (create TEXT using GINDTEXTELT FONT _ NEWVALUE))
                                    (T (create TEXTBOX using GINDTEXTELT FONT _ NEWVALUE))))
                        (FAMILY&SIZE                   (* set the font family and size)
                                     [SETQ NEWVALUE (LIST (CAR COMMAND)
                                                          (CADR COMMAND)
                                                          (FONTPROP (fetch (TEXT FONT) of GINDTEXTELT)
                                                                 'FACE]
                                     (COND
                                        ((EQ GTYPE 'TEXT)
                                         (create TEXT using GINDTEXTELT FONT _ NEWVALUE))
                                        (T (create TEXTBOX using GINDTEXTELT FONT _ NEWVALUE))))
                        (SAME                          (* set all of the font characteristics from the first selected one.)
```

(* set the variables to cause the right things to go into the change spec event.)

```
                              (SETQ OLDVALUE ELTWITHTEXT)
                              (SETQ PROPERTY 'LOOKSAME)
                              (SETQ NEWVALUE (SK.TEXT.ELT.WITH.SAME.FIELDS (fetch (GLOBALPART
                                                                                   INDIVIDUALGLOBALPART
                                                                                   )
                                                                              of COMMAND)
                                                    GINDTEXTELT)))
                        (SHOULDNT)))
        [SETQ NEWGTEXT (COND
                          [(EQ GTYPE 'TEXT)            (* adjust the scales at which this appears because font or scale
                           may have changed.)
                           (TEXT.SET.SCALES (create GLOBALPART
                                                    COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                          of ELTWITHTEXT)
                                                    INDIVIDUALGLOBALPART _ (TEXT.SET.GLOBAL.REGIONS
                                                                            NEWGTEXT]
                          (T
```

(* scaling for text boxes depends on the box size which can't change in this function.)

```
                             (create GLOBALPART
                                       COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART) of ELTWITHTEXT)
                                       INDIVIDUALGLOBALPART _ (TEXTBOX.SET.GLOBAL.REGIONS NEWGTEXT]
                (RETURN (create SKHISTORYCHANGESPEC
                                NEWELT _ NEWGTEXT
                                OLDELT _ ELTWITHTEXT
                                PROPERTY _ PROPERTY
                                NEWVALUE _ NEWVALUE
                                OLDVALUE _ OLDVALUE])
```

(TEXT.SET.SCALES

```
  [LAMBDA (GTEXTELT)                                                  (* rrb "12-May-85 16:29")

             (* sets the min and max scale properties of a global text element.
             Called after something about the text changes.)

    (PROG [(COMMONPART (fetch (GLOBALPART COMMONGLOBALPART) of GTEXTELT))
            (ORIGSCALE (fetch (TEXT INITIALSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT]
           (replace (COMMONGLOBALPART MINSCALE) of COMMONPART with (QUOTIENT ORIGSCALE 20.0))
           (replace (COMMONGLOBALPART MAXSCALE) of COMMONPART with (FTIMES (FONTHEIGHT (fetch (TEXT FONT)
                                                                                     of (fetch (GLOBALPART
                                                                                                INDIVIDUALGLOBALPART
                                                                                                     )
                                                                                           of GTEXTELT)))
                                                                            ORIGSCALE))
           (RETURN GTEXTELT])
```

(BREAK.AT.CARRIAGE.RETURNS

```
  [LAMBDA (STRING)                                                    (* rrb "16-Oct-85 18:24")
                                                                      (* returns a list of strings breaking string at carriage returns.)
    (PROG (STRLST (STR (OR (STRINGP STRING)
                           (MKSTRING STRING)))
                  (PREV 0)
                  (WHERE 0))
      LP  (COND
             ((SETQ WHERE (STRPOS "
                            " STR (ADD1 WHERE)))
              [SETQ STRLST (NCONC1 STRLST (COND
                                             ((EQ WHERE (ADD1 PREV))
                                              "")
                                             (T (SUBSTRING STR (ADD1 PREV)
                                                       (SUB1 WHERE]
              (SETQ PREV WHERE)
              (GO LP)))
          (RETURN (NCONC1 STRLST (OR (SUBSTRING STR (ADD1 PREV)
                                            -1)
                                     ""])
)

(DEFINEQ
```

(ADD.KNOWN.SKETCH.FONT

```
  [LAMBDA (FAMILY WID DEVICE FONT)                                    ; Edited 10-May-93 16:49 by rmk:
                                                                      ; Edited 21-Feb-89 15:06 by snow

    ;; add to the globally cached font list

    (DECLARE (GLOBALVARS \KNOWN.SKETCH.FONTSIZES))
    [LET ((CACHE (ASSOC FAMILY \KNOWN.SKETCH.FONTSIZES))
          (CACHED))
         (COND
            [(NULL CACHE)
             (if \KNOWN.SKETCH.FONTSIZES
                 then [NCONC1 \KNOWN.SKETCH.FONTSIZES (LIST FAMILY (LIST DEVICE (CONS WID FONT]
               else (SETQ \KNOWN.SKETCH.FONTSIZES (LIST (LIST FAMILY (LIST DEVICE (CONS WID FONT]
            (T (COND
                  ((SETQ CACHED (ASSOC DEVICE CACHE))
                   (NCONC1 CACHED (CONS WID FONT)))
                  (T (NCONC1 CACHE (CONS DEVICE (CONS WID FONT]
    FONT])
```

(SK.PICK.FONT

```
  [LAMBDA (WID STRING DEVICE DISPLAYGFONT)                            ; Edited 10-May-93 17:11 by rmk:
                                                                      ; Edited 22-Feb-89 07:53 by snow

    ;; returns the font in FAMILY that text should be printed in to have the text STRING fit into a region WID points wide

    (DECLARE (GLOBALVARS \KNOWN.SKETCH.FONTSIZES))
    (LET [STARTFONT FONTWIDTH SCALE CACHEDFONT SIZE (FAMILY (FONTPROP DISPLAYGFONT 'FAMILY]
         (IF [SETQ CACHEDFONT (ASSOC WID (ASSOC DEVICE (ASSOC FAMILY \KNOWN.SKETCH.FONTSIZES]
             THEN (CDR CACHEDFONT)
           ELSE (SETQ STARTFONT (FONTCOPY DISPLAYGFONT 'DEVICE DEVICE))
                NIL
                (SETQ SCALE (FONTPROP STARTFONT 'SCALE))
                (SETQ SIZE (FONTPROP STARTFONT 'SIZE))
                [SETQ FONTWIDTH (COND
                                                (SCALE ;; IF THERE IS A SCALE, YOU MUST SCALE THE WIDTH.
                                                       (FIXR (QUOTIENT (STRINGWIDTH STRING STARTFONT)
                                                                     SCALE)))
                                                (T (STRINGWIDTH STRING STARTFONT]
                          [SETQ CACHEDFONT
                           (IF (IGREATERP FONTWIDTH WID)
                               THEN ;; Font width was too big, try smaller fonts in decreasing size.
                                    [FOR FONT IN [CDR (FIND F ON [SORT (FONTSAVAILABLE FAMILY '* 'MRR 0 DEVICE T)
                                                                     (FUNCTION (LAMBDA (F1 F2)
                                                                                  (IGREATERP (CADR F1)
                                                                                         (CADR F2]
                                                           SUCHTHAT (EQ SIZE (CADR F]
                                       WHEN (ILESSP [SETQ FONTWIDTH (COND
                                                                        (SCALE ;; IF THERE IS A SCALE, YOU MUST SCALE THE
                                                                               ;; WIDTH.
                                                                               (FIXR (QUOTIENT (STRINGWIDTH STRING FONT)
                                                                                             SCALE)))
                                                                        (T (STRINGWIDTH STRING FONT]
                                                   WID)
                                       DO (RETURN (ADD.KNOWN.SKETCH.FONT FAMILY WID DEVICE FONT))
                                       FINALLY (RETURN (ADD.KNOWN.SKETCH.FONT FAMILY WID DEVICE
                                                                 (IF (GREATERP FONTWIDTH (TIMES 1.5 WID))
                                                                     THEN 'SHADE
                                                                   ELSEIF (OR FONT STARTFONT]
                             ELSEIF (IEQP FONTWIDTH WID)
                               THEN (ADD.KNOWN.SKETCH.FONT FAMILY WID DEVICE STARTFONT)
                             ELSE ;; FONT width was too small, try bigger fonts.
                                  (FOR FONT PREVFONT IN [CDR (FIND F ON [SORT (FONTSAVAILABLE FAMILY '* 'MRR 0 DEVICE T)
                                                                            (FUNCTION (LAMBDA (F1 F2)
                                                                                         (ILESSP (CADR F1)
                                                                                                (CADR F2]
                                                                  SUCHTHAT (EQ SIZE (CADR F]
                                     DO (IF (IGREATERP (COND
                                                          (SCALE ;; IF THERE IS A SCALE, YOU MUST SCALE THE WIDTH.
                                                                 (FIXR (QUOTIENT (STRINGWIDTH STRING FONT)
                                                                               SCALE)))
                                                          (T (STRINGWIDTH STRING FONT)))
                                                   WID)
                                            THEN (RETURN (ADD.KNOWN.SKETCH.FONT FAMILY WID DEVICE PREVFONT)))
                                          (SETQ PREVFONT FONT)
                                     FINALLY (RETURN (ADD.KNOWN.SKETCH.FONT FAMILY WID DEVICE (OR FONT PREVFONT
                                                                                                  STARTFONT]
                          (IF (FONTP CACHEDFONT)
                              THEN                                           ; Could be SHADE
                                   (FONTCOPY CACHEDFONT 'FACE (FONTPROP DISPLAYGFONT 'FACE))
                            ELSE CACHEDFONT])
```

(SK.CHOOSE.TEXT.FONT

```
  [LAMBDA (GTEXT SCALE VIEWER)                                         ; Edited 10-May-93 16:18 by rmk:
                                                                       ; Edited  1-Nov-91 16:56 by jds

    ;; returns the font that text in the individual global part of a text or textbox element GTEXT should be displayed in when shown in VIEWER.

    (PROG ([VIEWERFONTCACHE (OR (AND (WINDOWP VIEWER)
                                     (WINDOWPROP VIEWER 'PICKFONTCACHE))
                                (AND (STREAMP VIEWER)
                                     (STREAMPROP VIEWER 'PICKFONTCACHE]
           (GFONT (fetch (TEXT FONT) of GTEXT))
           LOCALFONT)
          [COND
             ((SETQ LOCALFONT (SASSOC GFONT VIEWERFONTCACHE))          ; look in the viewer's font cache.
              (RETURN (CDR LOCALFONT]
          (RETURN (PROG ((CANONICALTESTSTR "AWIaiw")
                         CANONICALWIDTH DEVICE DISPLAYGFONT)
                        [SETQ DEVICE (COND
                                        ((STREAMP VIEWER)
                                         (fetch (IMAGEOPS IMFONTCREATE) of (fetch (STREAM IMAGEOPS) of VIEWER)))
                                        (T 'DISPLAY]
                        [COND
                           ((EQUAL (TIMES SCALE (DSPSCALE NIL VIEWER))
                                   (fetch (TEXT INITIALSCALE) of GTEXT))

                            ;; special case scales being the same so there is not a large delay when first character is typed and to avoid font
                            ;; look up problems when hardcopying at scale 1

                            (SETQ LOCALFONT (FONTCREATE GFONT NIL NIL NIL DEVICE)))
                           (T ;; use a canonical string to determine the font size so that all strings of a given font at a given scale look the
                              ;; same.  If font is determined by the width of the particular string, two different string will appear in different
                              ;; fonts.  In particular, the string may change fonts as the user is typing into it.

                              ;; don't use the face information when determining string width because in some cases HELVETICA 10, the
                              ;; bold is smaller than the regular.

                              (SETQ DISPLAYGFONT (FONTCREATE GFONT NIL NIL NIL 'DISPLAY))
                              [SETQ CANONICALWIDTH (FIXR (QUOTIENT (TIMES [STRINGWIDTH
                                                                          CANONICALTESTSTR
                                                                          (LIST (FONTPROP DISPLAYGFONT
                                                                                          'FAMILY)
                                                                                (FONTPROP DISPLAYGFONT
                                                                                          'SIZE]
                                                                         (fetch (TEXT INITIALSCALE) of GTEXT))
                                                                  (TIMES SCALE (DSPSCALE NIL VIEWER]
                                                                  ; calculate the local font.
                              (SETQ LOCALFONT (SK.PICK.FONT CANONICALWIDTH CANONICALTESTSTR DEVICE DISPLAYGFONT]
                         [COND
                            ((WINDOWP VIEWER)
                             (WINDOWPROP VIEWER 'PICKFONTCACHE (CONS (CONS GFONT LOCALFONT)
                                                                     VIEWERFONTCACHE)))
                            ((STREAMP VIEWER)
                             (STREAMPROP VIEWER 'PICKFONTCACHE (CONS (CONS GFONT LOCALFONT)
                                                                     VIEWERFONTCACHE]
                         (RETURN LOCALFONT])
```

(SK.NEXTSIZEFONT

```
  [LAMBDA (WHICHDIR NOWFONT)                                      (* rrb "14-Jul-86 13:43")
                                                                  (* returns the next sized font either SMALLER or LARGER that
                                                                  on of size FONT.)
    (PROG [(NOWSIZE (FONTPROP NOWFONT 'HEIGHT))
           (DECREASEFONTLST (SK.DECREASING.FONT.LIST (CAR NOWFONT)
                                     'DISPLAY]
          (RETURN (COND
                     [(EQ WHICHDIR 'LARGER)
                      (COND
                         ((IGEQ NOWSIZE (FONTPROP (CAR DECREASEFONTLST)
                                               'HEIGHT))          (* nothing larger)
                          NIL)
                         (T (for FONTTAIL on DECREASEFONTLST when [AND (CDR FONTTAIL)
                                                                       (IGEQ NOWSIZE (FONTPROP (CADR FONTTAIL)
                                                                                           'HEIGHT]
                               do (RETURN (SK.FONTNAMELIST (CAR FONTTAIL]
                     (T (for FONT in DECREASEFONTLST when (LESSP (FONTPROP FONT 'HEIGHT)
                                                                  NOWSIZE)
                           do (RETURN (SK.FONTNAMELIST FONT])
```

(SK.DECREASING.FONT.LIST

```
  [LAMBDA (FAMILY DEVICETYPE)                                     ; Edited 12-Oct-92 12:39 by sybalsky:mv:envos

    ;; returns a list of fonts of family FAMILY which work on device DEVICETYPE

    [COND
       ((NULL FAMILY)
        (SETQ FAMILY 'MODERN]

    ;; convert to families that exist on the known devices.

;;; NOTE: this is a very bad way to convert the family.  It HARDCODES in the conversions for PRESS and INTERPRESS  and does nothing for new
;;; device types.  I have added the conversion for POSTSCRIPT that does things a little cleaner, but it should really look at a property of the device
;;; (fontconversions or some such animal.)  --was 2/19/89

    (LET ((CONVERSION))
         [COND
            [(EQ DEVICETYPE 'PRESS)
             (COND
                ((EQ FAMILY 'MODERN)
                 (SETQ FAMILY 'HELVETICA))
                ((EQ FAMILY 'CLASSIC)
                 (SETQ FAMILY 'TIMESROMAN))
                ((EQ FAMILY 'TERMINAL)
                 (SETQ FAMILY 'GACHA]
            [(EQ DEVICETYPE 'INTERPRESS)
             (COND
                ((EQ FAMILY 'HELVETICA)
                 (SETQ FAMILY 'MODERN))
                ((EQ FAMILY 'TIMESROMAN)
                 (SETQ FAMILY 'CLASSIC))
                ((EQ FAMILY 'GACHA)
                 (SETQ FAMILY 'TERMINAL]
            ((EQ DEVICETYPE 'POSTSCRIPT)
             (if (SETQ CONVERSION (ASSOC FAMILY POSTSCRIPT.FONT.ALIST))
                 then
                      ;; convert the family here for postscript as well as the other well known devices.

                      (SETQ FAMILY (CDR CONVERSION]
         (for FONT in (SK.GUESS.FONTSAVAILABLE FAMILY DEVICETYPE) collect (FONTCOPY FONT 'DEVICE DEVICETYPE])
```

(SK.GUESS.FONTSAVAILABLE

```
  [LAMBDA (FAMILY HDCPYTYPE)                                      (* rrb " 9-Oct-85 16:10")
                                                                  (* returns a list of all fonts of a FAMILY in decreasing order.)
    (PROG (FILEFONTS CACHE DISPLAYFONTSIZES)
          (SETQ HDCPYTYPE (COND
                             ((NULL HDCPYTYPE)
                         (PRINTERTYPE))
                        ((NLISTP HDCPYTYPE)
                         HDCPYTYPE)
                        (T HDCPYTYPE)))                    (* cache the file fonts.)
           [COND
              [[SETQ FILEFONTS (ASSOC HDCPYTYPE (CDR (ASSOC FAMILY \FONTSONFILE]
                                                           (* note if a cache has been calculated.
                                                           Use it even if it is NIL)

           (* \FONTSONFILE seems to group things such as CLASSICTHIN under CLASSIC so make sure to remove anything that
           has the wrong family.)

               (SETQ FILEFONTS (SUBSET (CDR FILEFONTS)
                                     (FUNCTION (LAMBDA (X)
                                                 (EQ (CAR X)
                                                     FAMILY]
              (T (RESETFORM (CURSOR WAITINGCURSOR)
                        (SETQ FILEFONTS (FONTSAVAILABLE FAMILY '* '(MEDIUM REGULAR REGULAR)
                                               NIL HDCPYTYPE T))

           (* Since there is no way to determine the real sizes for PRESS fonts with size of 0 {meaning the widths scale}, guess that
           they are available in 10)

                        [COND
                           [(EQ HDCPYTYPE 'PRESS)                  (* make sure to look for anything that has a display font.)
                            (SETQ DISPLAYFONTSIZES (for FONT in (FONTSAVAILABLE FAMILY '* '(MEDIUM REGULAR REGULAR
                                                                                                     )
                                                                             NIL
                                                                             'DISPLAY)
                                                      collect (CADR FONT)))
                            (SETQ FILEFONTS
                             (for FONT in FILEFONTS
                                join (COND
                                         [(EQ (CADR FONT)
                                              0)
                                          (for SIZE
                                             in (UNION DISPLAYFONTSIZES
                                                       '(36 30 24 18 14 12 10 8 6))
                                             when (FONTCREATE (CAR FONT)
                                                          SIZE NIL NIL 'DISPLAY T)
                                             collect (CONS (CAR FONT)
                                                           (CONS SIZE (CDDR FONT]
                                         (T (CONS FONT]
                           ((EQ HDCPYTYPE 'DISPLAY)                (* patch around the bug in FONTSAVAILABLE.
                                                                   Remove after J release.)
                            (SETQ FILEFONTS (SUBSET FILEFONTS (FUNCTION (LAMBDA (FONT)
                                                                          (EQUAL (CADDR FONT)
                                                                                 '(MEDIUM REGULAR REGULAR]
                                                                   (* remove duplicates and sort)
                        [SETQ FILEFONTS (SORT (INTERSECTION FILEFONTS FILEFONTS)
                                              (FUNCTION (LAMBDA (A B)
                                                          (GREATERP (CADR A)
                                                                    (CADR B]
                        (COND
                           ((NULL (SETQ CACHE (ASSOC FAMILY \FONTSONFILE)))
                            (SETQ \FONTSONFILE (CONS (LIST FAMILY (CONS HDCPYTYPE FILEFONTS))
                                                     \FONTSONFILE)))
                           (T (NCONC1 CACHE (CONS HDCPYTYPE FILEFONTS]
                                                                   (* reget the fonts in core since they may have changed since
                                                                   last time.)
           (RETURN (SORT (UNION (FONTSAVAILABLE FAMILY '* NIL NIL HDCPYTYPE)
                                FILEFONTS)
                         (FUNCTION (LAMBDA (A B)
                                     (COND
                                        ((EQ (CADR A)
                                             (CADR B))             (* in case both TIMESROMAN and TIMESROMAND for
                                                                   example make it in.)
                                         (ALPHORDER (CADR A)
                                                (CADR B)))
                                        (T (GREATERP (CADR A)
                                                  (CADR B])

)

(RPAQ? \KNOWN.SKETCH.FONTSIZES )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \KNOWN.SKETCH.FONTSIZES)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(TYPERECORD TEXT (LOCATIONLATLON LISTOFCHARACTERS INITIALSCALE TEXTSTYLE FONT LISTOFREGIONS TEXTCOLOR))
```

```
(RECORD LOCALTEXT ((DISPLAYPOSITION)
                   LOCALHOTREGION LINEREGIONS LOCALFONT LOCALLISTOFCHARACTERS))
)
)

(DEFINEQ

(SK.SET.FONT
  [LAMBDA (W NEWFONT)                                          (* rrb " 2-Oct-85 14:55")

          (* sets the entire default font. Used when a sketch stream is created.
          or any of the defaults are changed. NEWFONT is a list of (FAMILY SIZE FACE))

    (COND
       (NEWFONT (COND
                   ((FONTCREATE NEWFONT NIL NIL NIL NIL T)

          (* clear the cache of looked up fonts. This provides the user a way of clearing the cache that shouldn't happen too much and
          is documented.)

                    (AND (FASSOC (CAR NEWFONT)
                                \FONTSONFILE)
                         (SETQ \FONTSONFILE (for BUCKET in \FONTSONFILE when (NEQ (CAR BUCKET)
                                                                                  (CAR NEWFONT))
                                               collect BUCKET)))
                    (replace (SKETCHCONTEXT SKETCHFONT) of (WINDOWPROP W 'SKETCHCONTEXT) with NEWFONT))
                   (T (STATUSPRINT W (CAR NEWFONT)
                             " "
                             (CADR NEWFONT)
                             " "
                             (SELECTQ (CAR (CADDR NEWFONT))
                                 (BOLD 'BOLD)
                                 "")
                             (SELECTQ (CADR (CADDR NEWFONT))
                                 (ITALIC 'ITALIC)
                                 "")
                             " not found"])

(SK.SET.TEXT.FONT
  [LAMBDA (W)                                                  (* rrb " 4-Oct-85 16:21")
                                                               (* sets the size of the default arrowhead.)
    (PROG [NEWFONT NOWFONT (SKCONTEXT (WINDOWPROP W 'SKETCHCONTEXT]
          (SETQ NEWFONT (SK.READFONTFAMILY W (PACK* "now: " (CAR (SETQ NOWFONT (fetch (SKETCHCONTEXT SKETCHFONT)
                                                                                 of SKCONTEXT)))
                                                   " "
                                                   (CADR NOWFONT)
                                                   ". New?")))
          (COND
             (NEWFONT (SK.SET.FONT W (LIST NEWFONT (CADR NOWFONT)
                                           (CADDR NOWFONT])

(SK.SET.TEXT.SIZE
  [LAMBDA (W)                                                  (* rrb " 2-Oct-85 14:36")
                                                               (* sets the size of the default font.)
    (PROG (NEWSIZE (SKCONTEXT (WINDOWPROP W 'SKETCHCONTEXT))
                 NOWFONT)
          (SETQ NOWFONT (fetch (SKETCHCONTEXT SKETCHFONT) of SKCONTEXT))
          (SETQ NEWSIZE (SK.READFONTSIZE NIL (FONTPROP NOWFONT 'FAMILY)
                               W))
          (COND
             (NEWSIZE (SK.SET.FONT W (LIST (CAR NOWFONT)
                                           NEWSIZE
                                           (CADDR NOWFONT])

(SK.SET.TEXT.HORIZ.ALIGN
  [LAMBDA (SKW NEWALIGN)                                       (* rrb " 6-Nov-85 09:51")

          (* * reads a new value for the horizontal justification)

    (PROG ([NEWJUST (COND
                       ((MEMB NEWALIGN '(CENTER LEFT RIGHT))
                        NEWALIGN)
                       (T (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                         ITEMS _ '((" Center " 'CENTER "New text will be
                                                                            centered around its position")
                                                                   ("Left     " 'LEFT "the left edge of the text
                                                                            will be at its position.")
                                                                   ("   Right" 'RIGHT "the right edge of the
                                                                            text will be at its position."]
           SKCONTEXT)
          (RETURN (AND NEWJUST (replace (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of (SETQ SKCONTEXT (WINDOWPROP
                                                                                                   SKW
                                                                                                   'SKETCHCONTEXT))
                                  with (CONS NEWJUST (CDR (fetch (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of SKCONTEXT]
)

(SK.READFONTSIZE
  [LAMBDA (TITLE FONTFAMILY SKW)                               (* rrb " 6-Nov-85 09:51")

              (* * gets a legal known font size from the user.)

              (* this should have MENUROWS _ 1 when title height bug in menu package gets fixed.)

    (PROG ((FONTSIZES (SK.COLLECT.FONT.SIZES FONTFAMILY))
           NEWSIZE)
          (COND
             ((NULL FONTSIZES)
              (GO MORE)))
          (SETQ NEWSIZE (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                         TITLE _ (COND
                                                                    (TITLE)
                                                                    (FONTFAMILY (CONCAT "new " FONTFAMILY "
                                                                                      size?"))
                                                                    (T "New font size?"))
                                                         ITEMS _ (CONS '(More 'MORE "will look on font directories
                                                                                to find more sizes.")
                                                                       FONTSIZES)
                                                         CENTERFLG _ T)))
          (COND
             ((NEQ NEWSIZE 'MORE)
              (RETURN NEWSIZE)))
      MORE
                                                                (* do longer search of files)
          (SETQ NEWSIZE (SK.COLLECT.FONT.SIZES FONTFAMILY T))
          (COND
             ((NULL NEWSIZE)                                    (* could not find any fonts of that family)
              (RETURN NIL))
             ((EQUAL NEWSIZE FONTSIZES)                         (* not new ones found)
              (STATUSPRINT SKW "
                     No more font sizes found.")))
          (RETURN (MENU (create MENU
                               TITLE _ (OR TITLE "New font size?")
                               ITEMS _ NEWSIZE
                               CENTERFLG _ T])

(SK.COLLECT.FONT.SIZES
  [LAMBDA (FAMILY FILESTOOFLG)                                 (* rrb " 2-Oct-85 10:43")

              (* collects all of the sizes that are known. If FAMILY is given, gets just those sizes.)

    (PROG (INCORESIZES FILESIZES)
          [COND
             [FAMILY (for TYPEBUCKET in (CDR (FASSOC FAMILY \FONTSONFILE))
                        do (for FFONT in (CDR TYPEBUCKET) do (OR (MEMB (CADR FFONT)
                                                                        INCORESIZES)
                                                                  (SETQ INCORESIZES (CONS (CADR FFONT)
                                                                                          INCORESIZES]
             (T                                                 (* look at all fonts)
                (for FAMILYBUCKET in \FONTSONFILE
                   do (for TYPEBUCKET in (CDR FAMILYBUCKET)
                         do (for FFONT in (CDR TYPEBUCKET) do (OR (MEMB (CADR FFONT)
                                                                         INCORESIZES)
                                                                   (SETQ INCORESIZES (CONS (CADR FFONT)
                                                                                           INCORESIZES]
          (RETURN (SORT (UNION INCORESIZES (COND
                                              [FILESTOOFLG      (* wants those on files too, Flip the cursor to note wait.)
                                               (RESETFORM (CURSOR WAITINGCURSOR)
                                                      (bind SIZES for FONT
                                                         in (FONTSAVAILABLE (OR FAMILY '*)
                                                                   '* NIL NIL 'DISPLAY T)
                                                         do (OR (MEMB (FONTPROP FONT 'SIZE)
                                                                      SIZES)
                                                                (SETQ SIZES (CONS (FONTPROP FONT 'SIZE)
                                                                                  SIZES)))
                                                         finally (RETURN SIZES]
                                              (T (bind SIZES for FONT in (FONTSAVAILABLE (OR FAMILY '*)
                                                                                '* NIL NIL 'DISPLAY FILESTOOFLG)
                                                    do (OR (MEMB (FONTPROP FONT 'SIZE)
                                                                 SIZES)
                                                           (SETQ SIZES (CONS (FONTPROP FONT 'SIZE)
                                                                             SIZES)))
                                                    finally (RETURN SIZES])

(SK.SET.TEXT.VERT.ALIGN
  [LAMBDA (SKW NEWALIGN)                                       (* rrb " 6-Nov-85 09:52")

              (* * reads a new value for the vertical justification)
    (PROG ([NEWJUST (COND
                       ((MEMB NEWALIGN '(TOP CENTER BASELINE BOTTOM))
                        NEWALIGN)
                       (T (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                          TITLE _ "New vertical alignment?"
                                                          ITEMS _ '(("Top" 'TOP "the top of new text's vertical
                                                                          extent will be at its position")
                                                                    ("Center" 'CENTER "New text's vertical extent
                                                                          will be centered around its position")
                                                                    ("Baseline" 'BASELINE "The baseline of new
                                                                          text will be at its position.")
                                                                    ("Bottom" 'BOTTOM "the bottom of new text's
                                                                          vertical extent will be at its
                                                                          position"))
                                                          CENTERFLG _ T]
           SKCONTEXT)
          (RETURN (AND NEWJUST (replace (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of (SETQ SKCONTEXT (WINDOWPROP
                                                                                                  SKW
                                                                                                  'SKETCHCONTEXT))
                                      with (LIST (CAR (fetch (SKETCHCONTEXT SKETCHTEXTALIGNMENT) of SKCONTEXT))
                                                 NEWJUST])

(SK.SET.TEXT.LOOKS
  [LAMBDA (W)                                                      (* rrb " 6-Nov-85 09:52")

          (* * reads a new value for the looks of default text)

    (SK.SET.DEFAULT.TEXT.FACE (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                              ITEMS _ '((regular '(MEDIUM REGULAR REGULAR)
                                                                                 "new text will be neither bold nor
                                                                                 italic.")
                                                                        (bold '(BOLD REGULAR REGULAR)
                                                                              "new text will be bold.")
                                                                        (italic '(MEDIUM ITALIC REGULAR)
                                                                                 "new text will be italic.")
                                                                        (bold/italic '(BOLD ITALIC REGULAR)
                                                                                 "new text will be bold and
                                                                                 italic."))
                                                              TITLE _ "New default look"
                                                              CENTERFLG _ T))
           W])

(SK.SET.DEFAULT.TEXT.FACE
  [LAMBDA (NEWDEFAULTFACE SKW)                                     (* rrb " 4-Oct-85 16:24")
                                                                   (* changes the default text face to NEWDEFAULTFACE.)
    (PROG [(NOWFONT (fetch (SKETCHCONTEXT SKETCHFONT) of (WINDOWPROP SKW 'SKETCHCONTEXT]
          (RETURN (AND NEWDEFAULTFACE (SK.SET.FONT SKW (LIST (CAR NOWFONT)
                                                             (CADR NOWFONT)
                                                             NEWDEFAULTFACE])
)

(DEFINEQ

(CREATE.SKETCH.TERMTABLE
  [LAMBDA NIL                                                      (* rrb " 2-Oct-85 10:40")
                                                                   (* returns a terminal table that has most characters printing as
                                                                   REAL)
                                                                   (* it is used by TEXT.DRAWFN1 to print strings in sketch.)
    (PROG ((TTBL (COPYTERMTABLE NIL)))
          (for I from 128 to 255 do (AND (EQ (ECHOCHAR I NIL TTBL)
                                            'INDICATE)
                                        (ECHOCHAR I 'REAL TTBL)))
          (RETURN TTBL])
)

(DEFINEQ

(SK.FONT.LIST
  [LAMBDA (FONTDESCRIPTOR)                                         (* rrb " 2-Oct-85 14:41")
                                                                   (* returns the font family, and size of a font descriptor)
    (LIST (FONTPROP FONTDESCRIPTOR 'FAMILY)
          (FONTPROP FONTDESCRIPTOR 'SIZE)
          (FONTPROP FONTDESCRIPTOR 'FACE])

(SK.INSURE.FONT
  [LAMBDA (FONT)                                                   (* rrb "16-Oct-85 17:46")
                                                                   (* checks the validity of a font argument for a sketch element.)
    (COND
       [(NULL FONT)
        (SK.FONT.LIST (OR (AND SK.DEFAULT.FONT (FONTCREATE SK.DEFAULT.FONT))
                          (DEFAULTFONT 'DISPLAY]
       ((FONTP FONT)
        (SK.FONT.LIST FONT))
       ((FONTCREATE FONT)
        (SK.FONT.LIST (FONTCREATE FONT)))
       (T (\ILLEGAL.ARG FONT])
```

```
(SK.INSURE.STYLE
  [LAMBDA (STYLE DEFAULT)                                    (* rrb "16-Oct-85 17:51")
                                                             (* checks the validity of a STYLE argument for a sketch element)
    (COND
       ((NULL STYLE)
        DEFAULT)
       ((AND (LISTP STYLE)
             (MEMB (CAR STYLE)
                   SK.HORIZONTAL.STYLES)
             (MEMB (CAR (LISTP (CDR STYLE)))
                   SK.VERTICAL.STYLES)
             (NULL (CDDR STYLE)))
        STYLE)
       (T (\ILLEGAL.ARG STYLE])

(SK.INSURE.TEXT
  [LAMBDA (TEXTTHING)                                        (* rrb " 4-Nov-85 18:53")
                                                             (* puts something in the form necessary for a text list of
                                                             characters.)
    (COND
       ((NLISTP TEXTTHING)
        (BREAK.AT.CARRIAGE.RETURNS TEXTTHING))
       (T (for X in TEXTTHING join (BREAK.AT.CARRIAGE.RETURNS X])
)

(RPAQQ INDICATE.TEXT.SHADE 23130)

(RPAQ? SK.DEFAULT.FONT )

(RPAQ? SK.DEFAULT.TEXT.ALIGNMENT '(CENTER BASELINE))

(RPAQ? \FONTSONFILE NIL)

(ADDTOVAR SK.HORIZONTAL.STYLES LEFT RIGHT CENTER)

(ADDTOVAR SK.VERTICAL.STYLES TOP CENTER BASELINE BOTTOM)

(RPAQ SKETCH.TERMTABLE (CREATE.SKETCH.TERMTABLE))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCH.TERMTABLE SK.DEFAULT.TEXT.ALIGNMENT INDICATE.TEXT.SHADE \FONTSONFILE SK.HORIZONTAL.STYLES
       SK.VERTICAL.STYLES)
)
```

;; stuff for supporting the TEXTBOX sketch element.

```
(DEFINEQ

(SKETCH.CREATE.TEXTBOX
  [LAMBDA (STRING REGION FONT JUSTIFICATION BOXBRUSH BOXDASHING FILLING TEXTCOLOR SCALE)
                                                             (* rrb " 6-Aug-86 17:06")
                                                             (* creates a sketch box element.)
    (PROG ((XBRUSH (SK.INSURE.BRUSH BOXBRUSH))
           [XTEXT (COND
                     ((NLISTP STRING)
                      (BREAK.AT.CARRIAGE.RETURNS STRING))
                     (T (for X in STRING join (BREAK.AT.CARRIAGE.RETURNS X]
           (XFONT (SK.INSURE.FONT FONT))
           (XJUSTIFICATION (SK.INSURE.STYLE JUSTIFICATION SK.DEFAULT.TEXTBOX.ALIGNMENT))
           XREGION)

          (* calculate the region the textbox is to have. This is complicated in the case where REGION is a position because all of the
          other parameters must be know to calculate the region.)

          [SETQ XREGION (COND
                           ((REGIONP REGION))
                           ((POSITIONP REGION)
                            (SK.COMPUTE.TEXTBOX.REGION.FOR.STRING REGION XTEXT XFONT XBRUSH XJUSTIFICATION))
                           (T (\ILLEGAL.ARG REGION]
          (RETURN (SK.TEXTBOX.CREATE1 XREGION XBRUSH XTEXT (OR (NUMBERP SCALE)
                                                                 1.0)
                         XJUSTIFICATION XFONT (SK.INSURE.DASHING BOXDASHING)
                         (SK.INSURE.FILLING FILLING)
                         (SK.INSURE.COLOR TEXTCOLOR])

(SK.COMPUTE.TEXTBOX.REGION.FOR.STRING
  [LAMBDA (POSITION STRLST FONT BRUSH JUSTIFICATION)        (* rrb "30-Jul-86 14:30")
```

```
                                                                                 (* returns the region of the box around STRLST whose control
                                                                                 point is POSITION.)
    (PROG ((TEXTWIDTH (bind NOWWIDTH (WIDTH _ 0) for STR in STRLST do (COND
                                                                         ((GREATERP (SETQ NOWWIDTH
                                                                                          (STRINGWIDTH STR FONT))
                                                                                     WIDTH)
                                                                          (SETQ WIDTH NOWWIDTH)))
                        finally (RETURN WIDTH)))
           (TEXTHEIGHT (TIMES (LENGTH STRLST)
                              (FONTHEIGHT FONT)))
           (MARGIN (SK.BRUSH.SIZE BRUSH)))                                (* leave two extra points for the width because it looks better.)
          (SETQ TEXTWIDTH (PLUS MARGIN MARGIN TEXTWIDTH 2))
          (SETQ TEXTHEIGHT (PLUS MARGIN MARGIN TEXTHEIGHT))
          (RETURN (CREATEREGION (DIFFERENCE (fetch (POSITION XCOORD) of POSITION)
                                       (SELECTQ (CAR JUSTIFICATION)
                                            (LEFT 0)
                                            (RIGHT TEXTWIDTH)
                                            (CENTER (QUOTIENT TEXTWIDTH 2.0))
                                            (SHOULDNT)))
                        (DIFFERENCE (fetch (POSITION YCOORD) of POSITION)
                               (SELECTQ (CADR JUSTIFICATION)
                                   (BASELINE (PLUS (QUOTIENT (DIFFERENCE TEXTHEIGHT (FONTHEIGHT FONT))
                                                          2.0)
                                                   (FONTPROP FONT 'DESCENT)))
                                   (TOP TEXTHEIGHT)
                                   (BOTTOM 0)
                                   (CENTER (QUOTIENT TEXTHEIGHT 2.0))
                                   (SHOULDNT)))
                        TEXTWIDTH TEXTHEIGHT])
```

(SK.BREAK.INTO.LINES

```
  [LAMBDA (STRLST FONT WIDTH)                                            (* rrb "14-Jun-85 18:04")

          (* returns a list of lines {as strings} of the text stored on STRLST broken so that as many words as possible fit on a line
          WIDTH wide.)

    (COND
       [(OR (FONTP FONT)
            (WINDOWP FONT))
        (PROG ((SPACEWIDTH (CHARWIDTH (CHARCODE % )
                                  FONT))
               (REMAINING WIDTH)
               THISLINE NEWLST PREVCHARCR)
              (for STR in STRLST
                 do (PROG ((BEGPTR 1)
                           (CHPTR 1)
                           (CHARSWID 0)
                           (LIMITPTR (ADD1 (NCHARS STR)))
                           CHCODE ENDPTR)
                     CHLP
                          (COND
                             ((EQ CHPTR LIMITPTR)                        (* ran out of characters.)
                              (COND
                                 ((EQ LIMITPTR 1)                        (* empty line, ignore it.)
                                  (RETURN))
                                 [(ILEQ CHARSWID REMAINING)              (* this whole thing fits.)
                                  (SETQ THISLINE (CONS [COND
                                                          ((EQ BEGPTR 1)
                                                                         (* save substring call.)
                                                           STR)
                                                          (T             (* put substring in.)
                                                             (SUBSTRING STR BEGPTR (SUB1 CHPTR]
                                                       (COND
                                                          (THISLINE
                                                                         (* put a space in)
                                                                          (CONS " " THISLINE]
                                 (ENDPTR

          (* found a word or words that will fit, put them on this line and finish this line.)

                                         (SETQ NEWLST (CONS [CONS (COND
                                                                     ((EQ ENDPTR 0)
                                                                         (* line began with a space and only it fit)
                                                                      " ")
                                                                     (T (SUBSTRING STR BEGPTR ENDPTR)))
                                                                  (COND
                                                                     (THISLINE
                                                                         (* put a space in)
                                                                              (CONS " " THISLINE]
                                                            NEWLST))
                                         (SETQ THISLINE (CONS (OR (SUBSTRING STR (PLUS ENDPTR 2)
                                                                         (SUB1 CHPTR))
                                                                  "")))
                                         (SETQ REMAINING WIDTH))
                                 (T                                      (* the remainder of this string goes on the next line.)
                                    (AND THISLINE (SETQ NEWLST (CONS THISLINE NEWLST)))
```

```
                               [SETQ THISLINE (CONS (COND
                                                       ((EQ BEGPTR 1)
                                                            (* save substring call.)
                                                        STR)
                                                       (T   (* put substring in.)
                                                          (SUBSTRING STR BEGPTR (SUB1 CHPTR]
                               (SETQ REMAINING WIDTH)))     (* decrement space remaining.)
                         (SETQ REMAINING (IDIFFERENCE REMAINING (IPLUS CHARSWID SPACEWIDTH)))
                         (RETURN)                           (* put the part of this line that didn't fit on the next line.)
                         )
                        ((EQ (CHARCODE % )
                             (SETQ CHCODE (NTHCHARCODE STR CHPTR)))
                                                            (* got to a space)
                         [COND
                            ((ILEQ CHARSWID REMAINING)      (* mark the end of something that we know fits.)
                                                            (* decrement space remaining.)
                             (SETQ REMAINING (DIFFERENCE REMAINING CHARSWID)))
                            (ENDPTR

(* found a word or words that will fit, put them on this line and finish this line.)

                                   (SETQ NEWLST (CONS [CONS (OR (SUBSTRING STR BEGPTR ENDPTR)
                                                                   "")
                                                             (COND
                                                                (THISLINE
                                                            (* put a space in)
                                                                         (CONS " " THISLINE]
                                                       NEWLST))
                                                            (* reset the pointers to note this beginning.)
                                   (SETQ THISLINE NIL)

(* ENDPTR is always just before a space, put the beginning at the character following the space.)

                                   (SETQ BEGPTR (PLUS ENDPTR 2))
                                   (SETQ REMAINING (DIFFERENCE WIDTH CHARSWID)))
                            (T                              (* the rest of the current string goes on the next line.)
                               (COND
                                  (THISLINE (SETQ NEWLST (CONS THISLINE NEWLST))
                                         (SETQ THISLINE NIL)))
                               (SETQ REMAINING (DIFFERENCE WIDTH CHARSWID]
                         (SETQ ENDPTR (SUB1 CHPTR))
                         (SETQ CHARSWID 0))
                        ((EQ CHCODE (CHARCODE EOL))         (* CR, end a line.)
                         [COND
                            ((GREATERP CHARSWID REMAINING)  (* the last word before the CR doesn't fit on this line.)
                             (COND
                                (ENDPTR                     (* put some of it on the previous line)
                                       (SETQ NEWLST (CONS [CONS (OR (SUBSTRING STR BEGPTR ENDPTR)
                                                                       "")
                                                                 (COND
                                                                    (THISLINE
                                                            (* put a space in)
                                                                             (CONS " " THISLINE]
                                                           NEWLST))
                                       (SETQ THISLINE NIL)
                                       (SETQ BEGPTR (PLUS ENDPTR 2)))
                                (T                          (* end the previous line and put this stuff on a new one.)
                                   (COND
                                      (THISLINE (SETQ NEWLST (CONS THISLINE NEWLST))
                                             (SETQ THISLINE NIL]
                         [SETQ THISLINE (CONS (COND
                                                 ((AND (EQ (ADD1 CHPTR)
                                                           LIMITPTR)
                                                       (EQ BEGPTR 1))
                                                            (* last character of str, save substring call.
                                                            for efficiency)
                                                  STR)
                                                 (T         (* put substring in.)
                                                    (SUBSTRING STR BEGPTR CHPTR)))
                                              (COND
                                                 (THISLINE  (* put a space in)
                                                        (CONS " " THISLINE]
                         (SETQ NEWLST (CONS THISLINE NEWLST))
                         (SETQ THISLINE NIL)
                         (SETQ CHARSWID 0)
                         (SETQ REMAINING WIDTH)
                         (COND
                            ((EQ (ADD1 CHPTR)
                                 LIMITPTR)
                             (SETQ PREVCHARCR T)
                             (RETURN))
                            (T (SETQ BEGPTR (ADD1 CHPTR))
                               (SETQ ENDPTR)))
                         (SETQ CHPTR (ADD1 CHPTR))
                         (GO CHLP)))
                     (SETQ CHARSWID (PLUS CHARSWID (CHARWIDTH CHCODE FONT)))
                     (SETQ CHPTR (ADD1 CHPTR))
```

```
                              (SETQ PREVCHARCR NIL)
                              (GO CHLP)))
                 (RETURN (for LINE in [REVERSE (COND
                                                  (THISLINE (CONS THISLINE NEWLSTD))
                                                  (NEWLST (COND
                                                             (PREVCHARCR

           (* if end of last line was a CR, put an empty line in so cursor shows there.)

                                                                    (CONS "" NEWLST))
                                                             (T NEWLST)))
                                                  (T (LIST ""]
                              collect (APPLY (FUNCTION CONCAT)
                                             (REVERSE LINE]
       (T                                                    (* if there isn't any font, it is probably SHADE.
                                                             Just leave the strings alone)

          STRLST])
```

```
(SK.BRUSH.SIZE
  [LAMBDA (SKBRUSH)                                          (* rrb "30-Dec-84 13:38")

           (* returns the size of a brush. This is used in places where the brush can be either an instance of the record BRUSH or a
           thickness.)

    (COND
       ((NUMBERP SKBRUSH))
       (T (fetch (BRUSH BRUSHSIZE) of SKBRUSH])
```

```
(SK.TEXTBOX.CREATE
  [LAMBDA (SKETCHREGION BRUSH SCALE WINDOW)                  (* rrb "16-Oct-85 17:59")

           (* * creates a sketch element from a region)

    (PROG [(CONTEXT (WINDOWPROP WINDOW 'SKETCHCONTEXT]
          (RETURN (SK.TEXTBOX.CREATE1 SKETCHREGION BRUSH (LIST "")
                         SCALE
                         (fetch (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of CONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHFONT) of CONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHDASHING) of CONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHFILLING) of CONTEXT)
                         (fetch (BRUSH BRUSHCOLOR) of (fetch (SKETCHCONTEXT SKETCHBRUSH) of CONTEXT])
```

```
(SK.TEXTBOX.CREATE1
  [LAMBDA (SKETCHREGION BRUSH LSTOFSTRS INITSCALE STYLE INITFONT DASHING FILLING TEXTCOLOR)
                                                             (* rrb " 4-Dec-85 20:45")
    (SK.UPDATE.TEXTBOX.AFTER.CHANGE (create GLOBALPART
                                           INDIVIDUALGLOBALPART _
                                           (create TEXTBOX
                                                  TEXTBOXREGION _ SKETCHREGION
                                                  LISTOFCHARACTERS _ LSTOFSTRS
                                                  INITIALSCALE _ INITSCALE
                                                  TEXTSTYLE _ STYLE
                                                  FONT _ INITFONT
                                                  TEXTCOLOR _ TEXTCOLOR
                                                  TEXTBOXBRUSH _ BRUSH
                                                  TEXTBOXDASHING _ DASHING
                                                  TEXTBOXFILLING _ FILLING])
```

```
(SK.UPDATE.TEXTBOX.AFTER.CHANGE
  [LAMBDA (GTEXTBOXELT)                                      (* rrb " 4-Dec-85 21:51")

           (* updates the dependent fields in a textbox element that has had its text field changed.)

    (PROG ((INDELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTBOXELT)))
          (TEXTBOX.SET.GLOBAL.REGIONS INDELT)
          (BOX.SET.SCALES (fetch (TEXTBOX TEXTBOXREGION) of INDELT)
                 GTEXTBOXELT)
          (RETURN GTEXTBOXELT])
```

```
(SK.TEXTBOX.POSITION.IN.BOX
  [LAMBDA (REGION STYLE FONT BRUSHWIDTH)                     (* rrb "31-Jul-86 15:43")

           (* returns the position that the text should be put at to have it look right within box REGION, sytle STYLE in font FONT)

    (create POSITION
           XCOORD _ (SELECTQ (CAR STYLE)
                         (LEFT (PLUS (fetch (REGION LEFT) of REGION)
                                     BRUSHWIDTH))
                         (RIGHT (DIFFERENCE (fetch (REGION RIGHT) of REGION)
                                       BRUSHWIDTH))
                         (CENTER (PLUS (fetch (REGION LEFT) of REGION)
                                       (QUOTIENT (fetch (REGION WIDTH) of REGION)
                                2.0)))
                    (SHOULDNT))
         YCOORD _ (SELECTQ (CADR STYLE)
                    (TOP (DIFFERENCE (fetch (REGION TOP) of REGION)
                               BRUSHWIDTH))
                    (BOTTOM (PLUS (fetch (REGION BOTTOM) of REGION)
                                BRUSHWIDTH))
                    (CENTER (PLUS (fetch (REGION BOTTOM) of REGION)
                                (QUOTIENT (fetch (REGION HEIGHT) of REGION)
                                    2.0)))
                    (BASELINE [PLUS (fetch (REGION BOTTOM) of REGION)
                                (PLUS (QUOTIENT (DIFFERENCE (fetch (REGION HEIGHT) of REGION)
                                                    (FONTPROP FONT 'HEIGHT))
                                          2.0)
                                    (FONTPROP FONT 'DESCENT])
                    (SHOULDNT])
```

(**TEXTBOX.CHANGEFN**

```
  [LAMBDA (SCRNELTS SKW HOW)                                   (* rrb " 6-Jan-85 19:03")
                                                               (* the users has selected SCRNELT to be changed)
    (SELECTQ (CAR HOW)
        (TEXT (TEXT.CHANGEFN SCRNELTS SKW HOW))
        (SIZE (CHANGE.ELTS.BRUSH.SIZE (CADR HOW)
                     SCRNELTS SKW))
        NIL])
```

(**TEXTBOX.DRAWFN**

```
  [LAMBDA (TEXTBOXELT WINDOW WINREG OPERATION)                 (* rrb " 3-Mar-86 21:38")
                                                               (* draws a text box element.)
    (PROG ((LOCALPART (fetch (SCREENELT LOCALPART) of TEXTBOXELT))
           FILLING BRUSH ELTOPERATION)
          (OR (NULL WINREG)
              (REGIONSINTERSECTP WINREG (fetch (LOCALTEXTBOX LOCALTEXTBOXREGION) of LOCALPART))
              (RETURN))
          (SETQ BRUSH (fetch (LOCALTEXTBOX LOCALTEXTBOXBRUSH) of LOCALPART))
          (SETQ FILLING (fetch (LOCALTEXTBOX LOCALTEXTBOXFILLING) of LOCALPART))
          (SETQ ELTOPERATION (fetch (SKFILLING FILLING.OPERATION) of FILLING))
                                                               (* just put texture where there won't be any text.)
          (SK.TEXTURE.AROUND.REGIONS (fetch (LOCALTEXTBOX LOCALTEXTBOXREGION) of LOCALPART)
                 (fetch (LOCALTEXTBOX LINEREGIONS) of LOCALPART)
                 (fetch (SKFILLING FILLING.TEXTURE) of FILLING)
                 WINDOW
                 (fetch (SKFILLING FILLING.COLOR) of FILLING)
                 ELTOPERATION
                 (fetch (BRUSH BRUSHSIZE) of BRUSH))
          (BOX.DRAWFN1 (fetch (LOCALTEXTBOX LOCALTEXTBOXREGION) of LOCALPART)
                 (fetch (BRUSH BRUSHSIZE) of BRUSH)
                 WINDOW WINREG ELTOPERATION (fetch (LOCALTEXTBOX LOCALTEXTBOXDASHING) of LOCALPART)
                 NIL
                 (fetch (BRUSH BRUSHCOLOR) of BRUSH))
          (TEXT.DRAWFN1 (fetch (LOCALTEXTBOX LOCALLISTOFCHARACTERS) of LOCALPART)
                 (fetch (LOCALTEXTBOX LINEREGIONS) of LOCALPART)
                 (fetch (LOCALTEXTBOX LOCALFONT) of LOCALPART)
                 (fetch (BRUSH BRUSHCOLOR) of BRUSH)
                 WINDOW ELTOPERATION])
```

(**SK.TEXTURE.AROUND.REGIONS**

```
  [LAMBDA (BOXREGION INREGIONS TEXTURE STREAM COLOR OPERATION BRUSHSIZE)
                                                               ; Edited 29-Sep-92 23:18 by jds

    ;; puts texture inside of a box but not in a collection of interior regions.  Assumes INREGIONS are in order from top to bottom and abut in the Y
    ;; direction.

    ;;  JDS 9/29/92 -- CHANGED TO AVOID DOING THIS WHEN TEXTURE IS NIL, THE MOST COMMON CASE IN TEXTBOXES.  This speeds up
    ;; PostScript printing something fierce.

    (AND TEXTURE (PROG (BOXLEFT BOXRIGHT BOXTOP BOXBOTTOM X Y (MARGIN (TIMES 2 (DSPSCALE NIL STREAM)))
                              (USEOP (SK.TRANSLATE.MODE OPERATION STREAM)))
                       [SETQ BOXLEFT (PLUS (fetch (REGION LEFT) of BOXREGION)
                                           (ADD1 (IQUOTIENT BRUSHSIZE 2]
                       [SETQ BOXBOTTOM (PLUS (fetch (REGION BOTTOM) of BOXREGION)
                                             (ADD1 (IQUOTIENT BRUSHSIZE 2]
                       (SETQ BOXTOP (DIFFERENCE (fetch (REGION TOP) of BOXREGION)
                                           (IQUOTIENT (ADD1 BRUSHSIZE)
                                                  2)))
                       (SETQ BOXRIGHT (DIFFERENCE (fetch (REGION RIGHT) of BOXREGION)
                                             (IQUOTIENT (ADD1 BRUSHSIZE)
                                                    2)))
                       (COND
                          ((OR (NULL INREGIONS)
                               (ALL.EMPTY.REGIONS INREGIONS))
                           (DSPFILL (CREATEREGION BOXLEFT BOXBOTTOM (ADD1 (DIFFERENCE BOXRIGHT BOXLEFT))
                                           (ADD1 (DIFFERENCE BOXTOP BOXBOTTOM)))
                                  TEXTURE USEOP STREAM)
                           (RETURN)))
                       (COND
                          ([GREATERP BOXTOP (SETQ X (fetch (REGION TOP) of (CAR INREGIONS]
                                                                ; fill area above the first region
                           (BLTSHADE TEXTURE STREAM BOXLEFT (ADD1 X)
                                  (ADD1 (DIFFERENCE BOXRIGHT BOXLEFT))
                                  (DIFFERENCE BOXTOP X)
                                  USEOP NIL COLOR)))
                       [for LEAVEREGION in INREGIONS
                          do (COND
                                ((ZEROP (fetch (REGION WIDTH) of LEAVEREGION))
                                                                ; this line doesn't have any characters, just fill all the way across.
                                 (BLTSHADE TEXTURE STREAM BOXLEFT (fetch (REGION BOTTOM) of LEAVEREGION)
                                        (ADD1 (DIFFERENCE BOXRIGHT BOXLEFT))
                                        (fetch (REGION HEIGHT) of LEAVEREGION)
                                        USEOP NIL COLOR))
                                (T                                  ; look for the part before and after the characters on this line.
                                   (COND
                                      ((GREATERP (SETQ X (DIFFERENCE (fetch (REGION LEFT) of LEAVEREGION)
                                                                MARGIN))
                                              BOXLEFT)              ; fill area to the left of this region
                                       (BLTSHADE TEXTURE STREAM BOXLEFT (fetch (REGION BOTTOM) of LEAVEREGION)
                                              (DIFFERENCE X BOXLEFT)
                                              (fetch (REGION HEIGHT) of LEAVEREGION)
                                              USEOP NIL COLOR)))
                                   (COND
                                      ((GREATERP BOXRIGHT (SETQ X (PLUS (fetch (REGION RIGHT) of LEAVEREGION)
                                                                        MARGIN)))
                                                                ; fill area to the right of this region
                                       (BLTSHADE TEXTURE STREAM (ADD1 X)
                                              (fetch (REGION BOTTOM) of LEAVEREGION)
                                              (DIFFERENCE BOXRIGHT X)
                                              (fetch (REGION HEIGHT) of LEAVEREGION)
                                              USEOP NIL COLOR]
                       (COND
                          ((GREATERP [SETQ X (fetch (REGION BOTTOM) of (CAR (LAST INREGIONS]
                                  BOXBOTTOM)                        ; fill area below the last region
                           (BLTSHADE TEXTURE STREAM BOXLEFT BOXBOTTOM (ADD1 (DIFFERENCE BOXRIGHT BOXLEFT))
                                  (DIFFERENCE X BOXBOTTOM)
                                  USEOP NIL COLOR])
```

(ALL.EMPTY.REGIONS

```
  [LAMBDA (REGIONLST)                                                   (* rrb " 3-Mar-86 20:42")
                                                                        (* returns T if REGIONLST contains nothing but empty regions.)
    (for REG in REGIONLST always (OR (ZEROP (fetch (REGION WIDTH) of REG))
                                     (ZEROP (fetch (REGION HEIGHT) of REG])
```

(TEXTBOX.EXPANDFN

```
  [LAMBDA (GTEXTBOXELT SCALE STREAM)                                    (* rrb "30-Jul-86 15:23")
                                                                        (* creates a local textbox screen element from a global text box
    element)
    (PROG ((GTEXTBOX (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTBOXELT))
           (CANONICALTESTSTR "AWIaiw")
           LREG TEXTPOS LOCALFONT STYLE IMAGESTREAM LINEREGIONS BRUSHWIDTH NEWLISTOFSTRS LOCALBRUSH)
                                                                        (* calculate the local brush)
          (SETQ LOCALBRUSH (SCALE.BRUSH (COND
                                           ([NOT (NUMBERP (SETQ LOCALBRUSH (fetch (TEXTBOX TEXTBOXBRUSH)
                                                                                of GTEXTBOX]
                                                                        (* new format, old format had brush width only.)
                                            LOCALBRUSH)
                                           (T [replace (TEXTBOX TEXTBOXBRUSH) of GTEXTBOX
                                                 with (SETQ LOCALBRUSH (create BRUSH
                                                                              BRUSHSIZE _ LOCALBRUSH
                                                                              BRUSHSHAPE _ 'ROUND]
                                              LOCALBRUSH))
                                    (fetch (TEXTBOX INITIALSCALE) of GTEXTBOX)
                                    SCALE))
          [COND
             ((TEXTUREP (fetch (TEXTBOX TEXTBOXFILLING) of GTEXTBOX))
                                                                        (* old format, update to new one which has a list of
                                                                        (texture color))
              (replace (TEXTBOX TEXTBOXFILLING) of GTEXTBOX with (create SKFILLING
                                                                        FILLING.TEXTURE _ (fetch (TEXTBOX
                                                                                                   TEXTBOXFILLING
                                                                                                     )
                                                                                             of GTEXTBOX)
                                                                        FILLING.COLOR _ NIL]
                                                                        (* calculate the local region for the text box.)
          (SETQ BRUSHWIDTH (ADD1 (QUOTIENT (fetch (BRUSH BRUSHSIZE) of LOCALBRUSH)
                                          2)))
          (SETQ LREG (SK.SCALE.REGION (fetch (TEXTBOX TEXTBOXREGION) of GTEXTBOX)
                            SCALE))                                     (* calculate the local font.)
          (SETQ LOCALFONT (SK.CHOOSE.TEXT.FONT GTEXTBOX SCALE STREAM))

          (* recalculate the line breaks for the particular stream given. This is necessary because the difference between display and
          hardcopy must be taken into account.)
          [SETQ IMAGESTREAM (COND
                                 ((STREAMP STREAM))
                                 (T (WINDOWPROP STREAM 'DSP]
          [SETQ NEWLISTOFSTRS (COND
                                   [(FONTP LOCALFONT)
                                    (SK.BREAK.INTO.LINES (fetch (TEXTBOX LISTOFCHARACTERS) of GTEXTBOX)
                                           (COND
                                              ((IMAGESTREAMTYPEP IMAGESTREAM 'HARDCOPY)
                                               IMAGESTREAM)
                                              (T LOCALFONT))
                                           (COND
                                              [(IMAGESTREAMTYPEP IMAGESTREAM 'HARDCOPY)
                                                                    (* do the split on the basis of the hardcopy font)
                                               (FIXR (TIMES (IDIFFERENCE (fetch (REGION WIDTH) of LREG)
                                                                   (ITIMES BRUSHWIDTH 2))
                                                            (PROGN

          (* the scale should be a parameter of the hardcopy font, maybe font widths scale.
          but for now assume widths are in micas.)

                                                                   MICASPERPT]
                                              (T (IDIFFERENCE (fetch (REGION WIDTH) of LREG)
                                                        (ITIMES BRUSHWIDTH 2]
                                   (T                               (* if not local font, leave line breaks alone.)
                                      (fetch (TEXTBOX LISTOFCHARACTERS) of GTEXTBOX]
          (SETQ STYLE (fetch (TEXTBOX TEXTSTYLE) of GTEXTBOX))
          (SETQ LINEREGIONS (SK.TEXT.LINE.REGIONS (OR NEWLISTOFSTRS '(""))
                                   (SK.TEXTBOX.POSITION.IN.BOX LREG STYLE (OR LOCALFONT (fetch (TEXTBOX FONT)
                                                                                             of GTEXTBOX))
                                          BRUSHWIDTH)
                                   (fetch (TEXTBOX LISTOFREGIONS) of GTEXTBOX)
                                   LOCALFONT STYLE SCALE IMAGESTREAM))
          (RETURN (create SCREENELT
                         LOCALPART _
                         (create LOCALTEXTBOX
                                TEXTBOXLL _ (create POSITION
                                                   XCOORD _ (fetch (REGION LEFT) of LREG)
                                                   YCOORD _ (fetch (REGION BOTTOM) of LREG))
                                TEXTBOXUR _ (create POSITION
                                                   XCOORD _ (fetch (REGION PRIGHT) of LREG)
                                                   YCOORD _ (fetch (REGION PTOP) of LREG))
                                LINEREGIONS _ LINEREGIONS
                                LOCALFONT _ LOCALFONT
                                LOCALTEXTBOXREGION _ LREG
                                LOCALLISTOFCHARACTERS _ NEWLISTOFSTRS
                                LOCALTEXTBOXBRUSH _ LOCALBRUSH
                                LOCALTEXTBOXFILLING _ (APPEND (fetch (TEXTBOX TEXTBOXFILLING) of GTEXTBOX))
                                LOCALTEXTBOXDASHING _ (fetch (TEXTBOX TEXTBOXDASHING) of GTEXTBOX))
                         GLOBALPART _ GTEXTBOXELT])
```

(**TEXTBOX.INPUTFN**

```
  [LAMBDA (W LREGION)                                               (* rrb "11-Jul-86 15:48")

          (* creates a box element for a sketch window. Prompts the user for one if none is given.)

    (PROG (LOCALREG)
          (COND
             ((REGIONP LREGION)
              (SETQ LOCALREG LREGION))
             [(NULL LREGION)
              (COND
                 [[SETQ LOCALREG (CAR (ERSETQ (GETWREGION W (FUNCTION SK.BOX.GETREGIONFN)
                                                     W]             (* WINDOWPROP will get exterior of window which should
                                                                    really be reduced to the interior.)
                                                                    (* make sure the last selected point wasn't outside.)
                  (COND
                     ((OR (NOT (SUBREGIONP (DSPCLIPPINGREGION NIL W)
                                      LOCALREG))
                          (AND (EQ (fetch (REGION WIDTH) of LOCALREG)
                                   0)
                               (EQ (fetch (REGION HEIGHT) of LOCALREG)
                                   0)))
                      (RETURN]
                 (T (RETURN]
             (T (\ILLEGAL.ARG LREGION)))
          (RETURN (SK.TEXTBOX.CREATE (UNSCALE.REGION.TO.GRID LOCALREG (VIEWER.SCALE W))
                         (fetch (SKETCHCONTEXT SKETCHBRUSH) of (WINDOWPROP W 'SKETCHCONTEXT))
                         (SK.INPUT.SCALE W)
                         W])
```

(**TEXTBOX.INSIDEFN**

```
  [LAMBDA (GTEXTBOX WREG)                                           (* rrb "30-Dec-84 17:23")
                                                                    (* determines if the global TEXTBOX GTEXTBOX is inside of
                                                                    WREG.)
    (REGIONSINTERSECTP (fetch (TEXTBOX TEXTBOXREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTBOX))
          WREG])

(TEXTBOX.REGIONFN
  [LAMBDA (TEXTBOXSCRELT)                                      (* rrb " 3-May-85 16:47")
                                                               (* returns the region occuppied by a box.)

          (* is increased by the brush size This has the nice property of insuring that the region always has both height and width.)

    (INCREASEREGION (fetch (LOCALTEXTBOX LOCALTEXTBOXREGION) of (fetch (SCREENELT LOCALPART) of TEXTBOXSCRELT))
           (SK.BRUSH.SIZE (fetch (TEXTBOX TEXTBOXBRUSH) of (fetch (SCREENELT INDIVIDUALGLOBALPART) of TEXTBOXSCRELT
                                                               ])

(TEXTBOX.GLOBALREGIONFN
  [LAMBDA (GTEXTBOXELT)                                        (* rrb "18-Oct-85 17:11")
                                                               (* returns the global region occupied by a global textbox
                                                               element.)
    (fetch (TEXTBOX TEXTBOXREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTBOXELT])

(TEXTBOX.SET.GLOBAL.REGIONS
  [LAMBDA (GTEXTBOXELT)                                        (* rrb "30-Jul-86 14:48")

          (* updates the list of characters and list of regions occupied by the textbox in the global coordinate space.)
                                                               (* this is used to determine the extent of a text element in a
                                                               region.)
    (PROG [(SCALE (fetch (TEXTBOX INITIALSCALE) of GTEXTBOXELT))
           (FONT (fetch (TEXTBOX FONT) of GTEXTBOXELT))
           (LISTOFSTRS (fetch (TEXTBOX LISTOFCHARACTERS) of GTEXTBOXELT))
           (TEXTSTYLE (fetch (TEXTBOX TEXTSTYLE) of GTEXTBOXELT))
           (REGION (fetch (TEXTBOX TEXTBOXREGION) of GTEXTBOXELT))
           (BRUSHWIDTH (SK.BRUSH.SIZE (fetch (TEXTBOX TEXTBOXBRUSH) of GTEXTBOXELT]
          (replace (TEXTBOX LISTOFREGIONS) of GTEXTBOXELT with (for LREG
                                                               in (LTEXT.LINE.REGIONS LISTOFSTRS
                                                                          (SK.TEXTBOX.POSITION.IN.BOX REGION
                                                                                 TEXTSTYLE FONT BRUSHWIDTH)
                                                                          FONT TEXTSTYLE (ITIMES (FONTHEIGHT
                                                                                                  FONT)
                                                                                                (LENGTH
                                                                                                   LISTOFSTRS
                                                                                                   )))
                                                               collect LREG))
          (RETURN GTEXTBOXELT])

(TEXTBOX.TRANSLATEFN
  [LAMBDA (SKELT DELTAPOS)                                     (* rrb "28-Apr-85 18:46")

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

    (PROG ((GTEXTBOXELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKELT))
           OLDREG NEWREG)
          (SETQ NEWREG (REL.MOVE.REGION (SETQ OLDREG (fetch (TEXTBOX TEXTBOXREGION) of GTEXTBOXELT))
                              (fetch (POSITION XCOORD) of DELTAPOS)
                              (fetch (POSITION YCOORD) of DELTAPOS)))
          (RETURN (TEXT.UPDATE.GLOBAL.REGIONS (create GLOBALPART
                                                     COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART
                                                                                              COMMONGLOBALPART)
                                                                                       of SKELT))
                                                     INDIVIDUALGLOBALPART _
                                                     (create TEXTBOX using GTEXTBOXELT TEXTBOXREGION _ NEWREG))
                         (create POSITION
                                XCOORD _ (fetch (REGION LEFT) of NEWREG)
                                YCOORD _ (fetch (REGION BOTTOM) of NEWREG))
                         (create POSITION
                                XCOORD _ (fetch (REGION LEFT) of OLDREG)
                                YCOORD _ (fetch (REGION BOTTOM) of OLDREG])

(TEXTBOX.TRANSLATEPTSFN
  [LAMBDA (TEXTBOXELT SELPTS GDELTA WINDOW)                    (* rrb "16-Oct-85 17:59")

          (* returns a closed wire element which has the knots that are members of SELPTS translated by the global amount
          GDELTA.)

    (PROG ((GTEXTBOXELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of TEXTBOXELT))
           OLDGLOBALREGION LLX LLY URX URY)
          (SETQ OLDGLOBALREGION (fetch (TEXTBOX TEXTBOXREGION) of GTEXTBOXELT))
          [COND
             [(MEMBER (fetch (LOCALTEXTBOX TEXTBOXLL) of (fetch (SCREENELT LOCALPART) of TEXTBOXELT))
                     SELPTS)                                   (* lower left point is moving.)
               (SETQ LLX (PLUS (fetch (REGION LEFT) of OLDGLOBALREGION)
                               (fetch (POSITION XCOORD) of GDELTA)))
               (SETQ LLY (PLUS (fetch (REGION BOTTOM) of OLDGLOBALREGION)
                               (fetch (POSITION YCOORD) of GDELTA]
             (T (SETQ LLX (fetch (REGION LEFT) of OLDGLOBALREGION))
              (SETQ LLY (fetch (REGION BOTTOM) of OLDGLOBALREGION]
          [COND
             [(MEMBER (fetch (LOCALTEXTBOX TEXTBOXUR) of (fetch (SCREENELT LOCALPART) of TEXTBOXELT))
                     SELPTS)                                         (* upper right point)
               (SETQ URX (PLUS (fetch (REGION PRIGHT) of OLDGLOBALREGION)
                               (fetch (POSITION XCOORD) of GDELTA)))
               (SETQ URY (PLUS (fetch (REGION PTOP) of OLDGLOBALREGION)
                               (fetch (POSITION YCOORD) of GDELTA]
             (T (SETQ URX (fetch (REGION PRIGHT) of OLDGLOBALREGION))
                (SETQ URY (fetch (REGION PTOP) of OLDGLOBALREGION]
          (RETURN (SK.TEXTBOX.CREATE1 (CREATEREGION (MIN LLX URX)
                                             (MIN LLY URY)
                                             (ABS (DIFFERENCE LLX URX))
                                             (ABS (DIFFERENCE LLY URY)))
                         (fetch (TEXTBOX TEXTBOXBRUSH) of GTEXTBOXELT)
                         (fetch (TEXTBOX LISTOFCHARACTERS) of GTEXTBOXELT)
                         (fetch (TEXTBOX INITIALSCALE) of GTEXTBOXELT)
                         (fetch (TEXTBOX TEXTSTYLE) of GTEXTBOXELT)
                         (fetch (TEXTBOX FONT) of GTEXTBOXELT)
                         (fetch (TEXTBOX TEXTBOXDASHING) of GTEXTBOXELT)
                         (fetch (TEXTBOX TEXTBOXFILLING) of GTEXTBOXELT)
                         (fetch (TEXTBOX TEXTCOLOR) of GTEXTBOXELT])
```

(TEXTBOX.TRANSFORMFN

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "16-Oct-85 17:59")

      (* returns a copy of the global TEXTBOX element that has had each of its control points transformed by transformfn.
      TRANSFORMDATA is arbitrary data that is passed to tranformfn.
      SCALEFACTOR is how much the transformation scales the figure and is used to determine the size of the font.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))

      (* transform the font by changing the scale according to how much the width of the box around the first line of text changes
      from the transformation.)

      (RETURN (SK.TEXTBOX.CREATE1 (SK.TRANSFORM.REGION (fetch (TEXTBOX TEXTBOXREGION) of INDVPART)
                                         TRANSFORMFN TRANSFORMDATA)
                     (fetch (TEXTBOX TEXTBOXBRUSH) of INDVPART)
                     (fetch (TEXTBOX LISTOFCHARACTERS) of INDVPART)
                     (FTIMES (fetch (TEXTBOX INITIALSCALE) of INDVPART)
                            SCALEFACTOR)
                     (fetch (TEXTBOX TEXTSTYLE) of INDVPART)
                     (fetch (TEXTBOX FONT) of INDVPART)
                     (fetch (TEXTBOX TEXTBOXDASHING) of INDVPART)
                     (fetch (TEXTBOX TEXTBOXFILLING) of INDVPART)
                     (fetch (TEXTBOX TEXTCOLOR) of INDVPART])
```

(TEXTBOX.UPDATEFN

```
  [LAMBDA (OLDLOCALELT NEWGELT SKETCHW)                         (* rrb " 5-Dec-85 18:02")

      (* update function for text inside of textboxes. Tries to repaint only the lines of text that have changed.)

      (* takes advantage of the fact that all relevant text fields are in the same place in TEXT and TEXTBOX records.)
                                                                (* if the box size has changed, reprint the whole thing anyway.)
    (PROG ((NEWTB (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT))
           (OLDTB (fetch (SCREENELT INDIVIDUALGLOBALPART) of OLDLOCALELT))
           (OLDLOCALTB (fetch (SCREENELT LOCALPART) of OLDLOCALELT)))
      (RETURN (COND
                 ((AND (EQUAL (fetch (TEXTBOX TEXTBOXBRUSH) of NEWTB)
                              (fetch (TEXTBOX TEXTBOXBRUSH) of OLDTB))
                       (EQUAL (fetch (TEXTBOX TEXTBOXDASHING) of NEWTB)
                              (fetch (LOCALTEXTBOX LOCALTEXTBOXDASHING) of OLDLOCALTB))
                       (EQUAL (fetch (TEXTBOX TEXTBOXFILLING) of NEWTB)
                              (fetch (LOCALTEXTBOX LOCALTEXTBOXFILLING) of OLDLOCALTB))
                       (EQUAL (fetch (TEXTBOX TEXTCOLOR) of NEWTB)
                              (fetch (TEXTBOX TEXTCOLOR) of OLDTB)))
                  (DSPOPERATION (PROG1 (DSPOPERATION 'REPLACE SKETCHW)
                                                                (* change to replace mode to erase background.)
                                       (SETQ NEWTB (TEXT.UPDATEFN OLDLOCALELT NEWGELT SKETCHW)))
                         SKETCHW)
                  NEWTB])
```

(TEXTBOX.READCHANGEFN

```
  [LAMBDA (SKW SCRNELTS)                                        (* rrb " 5-Mar-86 13:33")
                                                                (* reads how the user wants to change a textbox.)
    (PROG ((COMMAND (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                   TITLE _ "Change which part?"
                                                   ITEMS _ [APPEND (COND
                                                                      [(SKETCHINCOLORP)
                                                                       '(("Outline color" 'BRUSHCOLOR
                                                                                 "changes the color of the
                                                                                 outline")
                                                                         ("Filling color" 'FILLINGCOLOR
                                                                                 "changes the color of the
```

```
                                                                                          filling"]
                                                                               (T NIL))
                                                                          '(("The text" 'TEXT "allows changing the
                                                                                 properties of the text.")
                                                                            ("Box thickness" 'SIZE "changes the size of
                                                                                 the brush")
                                                                            (Dashing 'DASHING "changes the dashing of
                                                                                 the box.")
                                                                            ("Unbox the text" '(TEXT UNBOX)
                                                                                 "takes the text out of any selected
                                                                                 text boxes.")
                                                                            (Filling 'FILLING "allows changing of the
                                                                                 filling texture of the box."))
                                                                          (COND
                                                                             (FILLINGMODEFLG
                                                                              '(("Filling mode" 'FILLINGMODE "changes
                                                                                     how the filling effects the
                                                                                     figures it covers."]
                                                          CENTERFLG _ T)))
         HOW)
        (RETURN (SELECTQ COMMAND
                     (TEXT (TEXT.READCHANGEFN SKW SCRNELTS T))
                     (COND
                        ((LISTP COMMAND)
                         COMMAND)
                        ((SETQ HOW (SELECTQ COMMAND
                                        (FILLING (READ.FILLING.CHANGE))
                                        (FILLINGMODE (READ.FILLING.MODE))
                                        (SIZE (READSIZECHANGE "Change size how?" T))
                                        (DASHING (READ.DASHING.CHANGE))
                                        (BRUSHCOLOR [READ.COLOR.CHANGE "Change outline color how?" NIL
                                                           (fetch (BRUSH BRUSHCOLOR)
                                                              of (GETSKETCHELEMENTPROP (fetch (SCREENELT
                                                                                                GLOBALPART)
                                                                                          of (CAR SCRNELTS))
                                                                         'BRUSH])
                                        (FILLINGCOLOR [READ.COLOR.CHANGE "Change filling color how?" T
                                                             (fetch (SKFILLING FILLING.COLOR)
                                                                of (GETSKETCHELEMENTPROP (fetch (SCREENELT
                                                                                                  GLOBALPART)
                                                                                            of (CAR SCRNELTS)
                                                                                               )
                                                                           'FILLING])
                                        COMMAND))
                         (LIST COMMAND HOW])
```

```
(SK.TEXTBOX.TEXT.POSITION
  [LAMBDA (GTEXTBOXELT)                                              (* returns the position of the text in a text box element.)
    (create POSITION
           XCOORD _ (fetch (REGION LEFT) of (SETQ GTEXTBOXELT (fetch (TEXTBOX TEXTBOXREGION) of GTEXTBOXELT)))
           YCOORD _ (fetch (REGION TOP) of GTEXTBOXELT])
```

```
(SK.TEXTBOX.FROM.TEXT
  [LAMBDA (TEXTELT SKW)                                              (* rrb "30-Sep-86 18:34")
                                                                     (* returns a textbox that replaces GTEXTELT.)
    (PROG ((INDTEXTELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of TEXTELT))
           BRUSH STYLE CONTEXT NEWTEXTBOXELT)
          [SETQ BRUSH (fetch (SKETCHCONTEXT SKETCHBRUSH) of (SETQ CONTEXT (WINDOWPROP SKW 'SKETCHCONTEXT]
          (SETQ NEWTEXTBOXELT (SK.TEXTBOX.CREATE1 (INCREASEREGION (APPLY (FUNCTION SK.UNIONREGIONS)
                                                                         (fetch (TEXT LISTOFREGIONS)
                                                                            of INDTEXTELT))
                                                           (IQUOTIENT (ADD1 (SK.BRUSH.SIZE (fetch (BRUSH BRUSHSIZE
                                                                                                          )
                                                                                                 of BRUSH)))
                                                                  2))
                                              BRUSH
                                              (fetch (TEXT LISTOFCHARACTERS) of INDTEXTELT)
                                              (fetch (TEXT INITIALSCALE) of INDTEXTELT)
                                              (COND
                                                 ((EQ (CADR (SETQ STYLE (fetch (TEXT TEXTSTYLE) of INDTEXTELT)))
                                                      'BASELINE)             (* change from baseline to center because this usually looks
                                                                             better.)
                                                  (LIST (CAR STYLE)
                                                        'CENTER))
                                                 (T STYLE))
                                              (fetch (TEXT FONT) of INDTEXTELT)
                                              (fetch (SKETCHCONTEXT SKETCHDASHING) of CONTEXT)
                                              (fetch (SKETCHCONTEXT SKETCHFILLING) of CONTEXT)
                                              (fetch (BRUSH BRUSHCOLOR) of BRUSH)))
          (RETURN (create SKHISTORYCHANGESPEC
                         NEWELT _ NEWTEXTBOXELT
                         OLDELT _ TEXTELT
                         PROPERTY _ 'HASBOX
                         NEWVALUE _ NEWTEXTBOXELT
                         OLDVALUE _ TEXTELT])
```

```
(ADD.EOLS
  [LAMBDA (STRLST)                                               (* rrb "22-Jul-86 15:23")
                                                                 (* adds an eol to every string in STRLST that doesn't end in
                                                                 one.)
    (for STRTAIL on STRLST collect (COND
                                     ((EQ (CHARCODE EOL)
                                          (NTHCHARCODE (CAR STRTAIL)
                                                 -1))
                                      (CAR STRTAIL))
                                     ((CDR STRTAIL)              (* don't put a cr after the last line.)
                                      (CONCAT (CAR STRTAIL)
                                             "
                                             "))
                                     (T (CAR STRTAIL])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD LOCALTEXTBOX ((TEXTBOXLL TEXTBOXUR)
                      LOCALHOTREGION LINEREGIONS LOCALFONT LOCALLISTOFCHARACTERS LOCALTEXTBOXREGION
                      LOCALTEXTBOXBRUSH LOCALTEXTBOXFILLING LOCALTEXTBOXDASHING))

(TYPERECORD TEXTBOX (TEXTBOXREGION LISTOFCHARACTERS INITIALSCALE TEXTSTYLE FONT LISTOFREGIONS TEXTCOLOR
                           TEXTBOXBRUSH TEXTBOXDASHING TEXTBOXFILLING))
)
)

;; stuff to handle default alignment for text boxes

(DEFINEQ

(SK.SET.TEXTBOX.VERT.ALIGN
  [LAMBDA (SKW)                                                  (* rrb " 6-Nov-85 09:52")

          (* * reads a new value for the vertical justification default for text boxes)

    (PROG ((NEWJUST (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                    TITLE _ "New vertical alignment?"
                                                    ITEMS _ '(("Top" 'TOP "the top of new text's vertical extent
                                                                     will be at its position")
                                                              ("Center" 'CENTER "New text's vertical extent will
                                                                     be centered around its position")
                                                              ("Baseline" 'BASELINE "The baseline of new text
                                                                     will be at its position.")
                                                              ("Bottom" 'BOTTOM "the bottom of new text's
                                                                     vertical extent will be at its position"))
                                                    CENTERFLG _ T)))
                  SKCONTEXT)
          (RETURN (AND NEWJUST (replace (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of (SETQ SKCONTEXT
                                                                                   (WINDOWPROP SKW 'SKETCHCONTEXT)
                                                                                   )
                                  with (LIST (CAR (fetch (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of SKCONTEXT))
                                             NEWJUST])

(SK.SET.TEXTBOX.HORIZ.ALIGN
  [LAMBDA (SKW NEWALIGN)                                         (* rrb " 6-Nov-85 09:52")

          (* * reads a new value for the horizontal justification default for text boxes)

    (PROG ([NEWJUST (OR NEWALIGN (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                 ITEMS _ '((" Center " 'CENTER "New text will be
                                                                                 centered around its position")
                                                                           ("Left    " 'LEFT "the left edge of
                                                                                 the text will be at its
                                                                                 position.")
                                                                           ("   Right" 'RIGHT "the right edge of
                                                                                 the text will be at its
                                                                                 position."]
                  SKCONTEXT)
          (RETURN (AND NEWJUST (replace (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT) of (SETQ SKCONTEXT
                                                                                   (WINDOWPROP SKW 'SKETCHCONTEXT)
                                                                                   )
                                  with (CONS NEWJUST (CDR (fetch (SKETCHCONTEXT SKETCHTEXTBOXALIGNMENT)
                                                                  of SKCONTEXT])
)

(RPAQQ TEXTBOXICON TEXT BOX)

(RPAQ? SK.DEFAULT.TEXTBOX.ALIGNMENT '(CENTER CENTER))
```

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SK.DEFAULT.TEXTBOX.ALIGNMENT)
)
```

;; functions to implement the box sketch element.

```
(DEFINEQ

(SKETCH.CREATE.BOX
  [LAMBDA (REGION BRUSH DASHING FILLING SCALE)                    (* rrb "16-Oct-85 17:31")
                                                                  (* creates a sketch box element.)
    (SK.BOX.CREATE (OR (REGIONP REGION)
                       (\ILLEGAL.ARG REGION))
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.FILLING FILLING])

(SK.BOX.DRAWFN
  [LAMBDA (BOXELT WIN WINREG)                                     (* rrb "21-Feb-86 11:36")
                                                                  (* draws a box from its sketch element.)
    (PROG ((LOCALBOXELT (fetch (SCREENELT LOCALPART) of BOXELT))
           FILLING BRUSH)
          (SETQ FILLING (fetch (LOCALBOX LOCALBOXFILLING) of LOCALBOXELT))
          (RETURN (BOX.DRAWFN1 (fetch (LOCALBOX LOCALREGION) of LOCALBOXELT)
                         (fetch (BRUSH BRUSHSIZE) of (SETQ BRUSH (fetch (LOCALBOX LOCALBOXBRUSH) of LOCALBOXELT)))
                         WIN WINREG (fetch (SKFILLING FILLING.OPERATION) of FILLING)
                         (fetch (LOCALBOX LOCALBOXDASHING) of LOCALBOXELT)
                         (fetch (SKFILLING FILLING.TEXTURE) of FILLING)
                         (fetch (BRUSH BRUSHCOLOR) of BRUSH)
                         (fetch (SKFILLING FILLING.COLOR) of FILLING])

(BOX.DRAWFN1
  [LAMBDA (REG SIZE WIN WINREG OPERATION DASHING TEXTURE OUTLINECOLOR FILLINGCOLOR)
                                                                  (* rrb " 5-Mar-86 14:27")
                                                                  (* draws a box. Used by both box and text box elements.)
    (COND
       ((OR (NULL WINREG)
            (REGIONSINTERSECTP WINREG REG))
        (COND
           ((AND SKETCHINCOLORFLG (OR FILLINGCOLOR TEXTURE))      (* call the filling routine that does color.)
            (FILLPOLYGON (KNOTS.OF.REGION REG SIZE)
                   (create SKFILLING
                          FILLING.TEXTURE _ TEXTURE
                          FILLING.COLOR _ FILLINGCOLOR)
                   WIN))
           (TEXTURE (DSPFILL REG (COND
                                    ((EQ (DSPOPERATION NIL WIN)
                                         'ERASE)                   (* use black in case the window moved because of texture to
                                                                  window alignment bug.)
                                     BLACKSHADE)
                                    (T TEXTURE))
                           (SK.TRANSLATE.MODE OPERATION WIN)
                           WIN))
           (FILLINGCOLOR                                          (* if no texture, use the color.)
                  (DSPFILL REG (TEXTUREOFCOLOR FILLINGCOLOR)
                         OPERATION WIN)))

          (* code to fix white space bug in Interpress. It works but Masters are larger and the one I tried wouldn't print.
          (SELECTQ (IMAGESTREAMTYPE WIN) ((NIL DISPLAY PRESS)
          (* special case DISPLAY for speed and PRESS because rounded corners don't work for large brushes.)
          (SK.DRAWAREABOX (fetch (REGION LEFT) of REG) (fetch (REGION BOTTOM) of REG)
          (fetch (REGION WIDTH) of REG) (fetch (REGION HEIGHT) of REG) SIZE OPERATION WIN DASHING OUTLINECOLOR))
          (PROG ((LFT (fetch (REGION LEFT) of REG)) (BTM (fetch (REGION BOTTOM) of REG))
          (TOP (fetch (REGION TOP) of REG)) (RGHT (fetch (REGION RIGHT) of REG)))
          (DRAWCURVE (LIST (CREATEPOSITION LFT BTM) (CREATEPOSITION LFT TOP)
          (CREATEPOSITION RIGHT TOP) (CREATEPOSITION RIGHT BTM)) T
          (create BRUSH BRUSHSHAPE _ (QUOTE ROUND) BRUSHSIZE _ SIZE BRUSHCOLOR _ OUTLINECOLOR) DASHING
          WIN))))

        (SK.DRAWAREABOX (fetch (REGION LEFT) of REG)
               (fetch (REGION BOTTOM) of REG)
               (fetch (REGION WIDTH) of REG)
               (fetch (REGION HEIGHT) of REG)
               SIZE
               (SK.TRANSLATE.MODE OPERATION WIN)
               WIN DASHING OUTLINECOLOR])

(KNOTS.OF.REGION
  [LAMBDA (REGION BORDER)                                         (* rrb "18-Jul-85 09:49")
                                                                  (* returns the knots of the interior rectangle of a region.)
    (PROG (LFT BTM TP RGHT (HLFBORDER (FQUOTIENT BORDER 2.0)))
          (SETQ LFT (PLUS (fetch (REGION LEFT) of REGION)
                          HLFBORDER))
          (SETQ BTM (PLUS (fetch (REGION BOTTOM) of REGION)
                          HLFBORDER))
          (SETQ TP (DIFFERENCE (fetch (REGION TOP) of REGION)
                          HLFBORDER))
          (SETQ RGHT (DIFFERENCE (fetch (REGION RIGHT) of REGION)
                            HLFBORDER))
          (RETURN (LIST (create POSITION
                                XCOORD _ LFT
                                YCOORD _ BTM)
                        (create POSITION
                                XCOORD _ LFT
                                YCOORD _ TP)
                        (create POSITION
                                XCOORD _ RGHT
                                YCOORD _ TP)
                        (create POSITION
                                XCOORD _ RGHT
                                YCOORD _ BTM])
```

(SK.DRAWAREABOX

```
  [LAMBDA (LEFT BOTTOM WIDTH HEIGHT BORDER OP W DASHING COLOR)    (* rrb "16-Sep-86 16:12")

          (* draws lines along the region. Copied from the function DRAWAREABOX in GRAPHER and changed to be the same as
          drawing lines between the corner points.)

    (COND
       [[OR DASHING (AND COLOR (NEQ COLOR 'BLACK]

          (* start a line at each corner so that the corners will have black on them.)

        (COND
           ((OR (IMAGESTREAMTYPEP W 'PRESS)
                (IMAGESTREAMTYPEP W 'INTERPRESS))                 (* both these use BUTT, overlap the lines)
            (PROG (BIG/HALF SM/HALF TOP RIGHT)
                  (SETQ BIG/HALF (LRSH (ADD1 BORDER)
                                       1))
                  (SETQ SM/HALF (DIFFERENCE BORDER BIG/HALF))
                  (SETQ TOP (PLUS BOTTOM HEIGHT))
                  (SETQ RIGHT (PLUS LEFT WIDTH))                  (* draw left edge)
                  (DRAWLINE LEFT (DIFFERENCE BOTTOM SM/HALF)
                         LEFT
                         (PLUS TOP BIG/HALF)
                         BORDER OP W COLOR DASHING)               (* draw top.)
                  (DRAWLINE (IDIFFERENCE LEFT SM/HALF)
                         TOP
                         (IPLUS RIGHT BIG/HALF)
                         TOP BORDER OP W COLOR DASHING)           (* draw right edge)
                  (DRAWLINE RIGHT (PLUS TOP BIG/HALF)
                         RIGHT
                         (DIFFERENCE BOTTOM SM/HALF)
                         BORDER OP W COLOR DASHING)               (* draw bottom)
                  (DRAWLINE (IPLUS RIGHT BIG/HALF)
                         BOTTOM
                         (IDIFFERENCE LEFT SM/HALF)
                         BOTTOM BORDER OP W COLOR DASHING)))
           (T (PROG (TOP RIGHT HALFBORDER)
                    (SETQ TOP (PLUS BOTTOM HEIGHT))
                    (SETQ RIGHT (PLUS LEFT WIDTH))                (* draw left edge)
                    (DRAWLINE LEFT BOTTOM LEFT TOP BORDER OP W COLOR DASHING)
                                                                  (* draw top)
                    (DRAWLINE LEFT TOP RIGHT TOP BORDER OP W COLOR DASHING)
                                                                  (* draw right edge)
                    (DRAWLINE RIGHT TOP RIGHT BOTTOM BORDER OP W COLOR DASHING)
                                                                  (* draw bottom)
                    (DRAWLINE RIGHT BOTTOM LEFT BOTTOM BORDER OP W COLOR DASHING]
       ((IMAGESTREAMTYPEP W 'PRESS)                               (* overlap the ends of the lines.)
        (PROG (BIG/HALF SM/HALF TOP HORIZLEFT HORIZRIGHT RIGHT)
              (SETQ BIG/HALF (LRSH (ADD1 BORDER)
                                   1))
              (SETQ SM/HALF (DIFFERENCE BORDER BIG/HALF))
              (SETQ TOP (PLUS BOTTOM HEIGHT))
              (SETQ RIGHT (PLUS LEFT WIDTH))                      (* draw left edge)
              (DRAWLINE LEFT (DIFFERENCE BOTTOM SM/HALF)
                     LEFT
                     (PLUS TOP BIG/HALF)
                     BORDER OP W COLOR DASHING)                   (* draw top.)
              (DRAWLINE (SETQ HORIZLEFT (IPLUS LEFT BIG/HALF))
                     TOP
                     (SETQ HORIZRIGHT (SUB1 (IDIFFERENCE RIGHT SM/HALF)))
                     TOP BORDER OP W COLOR DASHING)               (* draw right edge)
              (DRAWLINE RIGHT (DIFFERENCE BOTTOM SM/HALF)
                     RIGHT
                     (PLUS TOP BIG/HALF)
                    BORDER OP W COLOR DASHING)                  (* draw bottom)
             (DRAWLINE HORIZLEFT BOTTOM HORIZRIGHT BOTTOM BORDER OP W COLOR DASHING)))
       ((IMAGESTREAMTYPEP W 'INTERPRESS)

            (* kludge for interpress in koto because BLTSHADE rounds down so brushes of 1 don't show, Drawline is always BUTT and
            DRAWPOLYGON isn't implemented.)

        (PROG (BIG/HALF SM/HALF TOP HORIZLEFT HORIZRIGHT RIGHT)
              (SETQ BIG/HALF (LRSH (ADD1 BORDER)
                                   1))
              (SETQ SM/HALF (DIFFERENCE BORDER BIG/HALF))
              (SETQ TOP (PLUS BOTTOM HEIGHT))
              (SETQ RIGHT (PLUS LEFT WIDTH))                     (* draw left edge)
              (DRAWLINE LEFT (DIFFERENCE BOTTOM SM/HALF)
                     LEFT
                     (PLUS TOP BIG/HALF)
                     BORDER OP W COLOR DASHING)

            (* draw top. 9 is to fix an error on the 8044 which may be from rounding to its pixel size.)

              (DRAWLINE (SETQ HORIZLEFT (DIFFERENCE (IPLUS LEFT BIG/HALF)
                                               9))
                     TOP
                     (SETQ HORIZRIGHT (SUB1 (IDIFFERENCE RIGHT SM/HALF)))
                     TOP BORDER OP W COLOR DASHING)             (* draw right edge)
              (DRAWLINE RIGHT (DIFFERENCE BOTTOM SM/HALF)
                     RIGHT
                     (PLUS TOP BIG/HALF)
                     BORDER OP W COLOR DASHING)                  (* draw bottom)
              (DRAWLINE HORIZLEFT BOTTOM HORIZRIGHT BOTTOM BORDER OP W COLOR DASHING)))
       (T                                                        (* do other cases with bitblt)
          (PROG (BIG/HALF SM/HALF HORIZLEFT BOXBOTTOM SIDEWIDTH SIDEHEIGHT)
                (SETQ BIG/HALF (LRSH BORDER 1))
                (SETQ SM/HALF (SUB1 (DIFFERENCE BORDER BIG/HALF)))
                                                                 (* draw left edge)
                (BLTSHADE BLACKSHADE W (DIFFERENCE LEFT SM/HALF)
                       (SETQ BOXBOTTOM (DIFFERENCE BOTTOM SM/HALF))
                       BORDER
                       (SETQ SIDEHEIGHT (PLUS HEIGHT BORDER))
                       OP)                                       (* draw right edge)
                (BLTSHADE BLACKSHADE W (DIFFERENCE (PLUS LEFT WIDTH)
                                              SM/HALF)
                       BOXBOTTOM BORDER SIDEHEIGHT OP)           (* draw top)
                (BLTSHADE BLACKSHADE W (SETQ HORIZLEFT (ADD1 (PLUS LEFT BIG/HALF)))
                       (DIFFERENCE (PLUS BOTTOM HEIGHT)
                              SM/HALF)
                       (SETQ SIDEWIDTH (DIFFERENCE WIDTH BORDER))
                       BORDER OP)
                (BLTSHADE BLACKSHADE W HORIZLEFT BOXBOTTOM SIDEWIDTH BORDER OP])
```

(**SK.DRAWBOX**

```
  [LAMBDA (BOXLEFT BOXBOTTOM BOXWIDTH BOXHEIGHT BORDER OP W TEXTURE)
                                                                 (* rrb "14-Jul-86 13:51")
                                                                 (* draws lines inside the region.)
    (OR TEXTURE (SETQ TEXTURE BLACKSHADE))                       (* draw left edge)
    (BITBLT NIL NIL NIL W BOXLEFT BOXBOTTOM BORDER BOXHEIGHT 'TEXTURE OP TEXTURE)
                                                                 (* draw top)
    (BITBLT NIL NIL NIL W (PLUS BOXLEFT BORDER)
           (DIFFERENCE (PLUS BOXBOTTOM BOXHEIGHT)
                  BORDER)
           (DIFFERENCE BOXWIDTH (PLUS BORDER BORDER))
           BORDER
           'TEXTURE OP TEXTURE)                                  (* draw bottom)
    (BITBLT NIL NIL NIL W (PLUS BOXLEFT BORDER)
           BOXBOTTOM
           (DIFFERENCE BOXWIDTH (PLUS BORDER BORDER))
           BORDER
           'TEXTURE OP TEXTURE)                                  (* draw right edge)
    (BITBLT NIL NIL NIL W (DIFFERENCE (PLUS BOXLEFT BOXWIDTH)
                                 BORDER)
           BOXBOTTOM BORDER BOXHEIGHT 'TEXTURE OP TEXTURE])
```

(**SK.BOX.EXPANDFN**

```
  [LAMBDA (GBOX SCALE)                                           (* rrb "11-Jul-86 15:56")

            (* returns a local record which has the region field of the global element GELT translated into window coordinats.)
                                                                 (* for now only allow to move the left-bottom or right-top corner.)
    (PROG ((INDGELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GBOX))
           LREG)
          [COND
             ((fetch (BOX BOXINITSCALE) of INDGELT))
             (T                                                  (* old format didn't have an initial scale, default it to 1.0)
                (replace (GLOBALPART INDIVIDUALGLOBALPART) of GBOX with (SETQ INDGELT
                                                                          (create BOX using INDGELT BOXINITSCALE _
                                                                                              1.0]
          [COND
             ((TEXTUREP (fetch (BOX BOXFILLING) of INDGELT))        (* old format, update to new one which has a list of
                                                                      (texture color))
              (replace (BOX BOXFILLING) of INDGELT with (create SKFILLING
                                                                 FILLING.TEXTURE _ (fetch (BOX BOXFILLING)
                                                                                       of INDGELT))
                                                                 FILLING.COLOR _ NIL]
          (SETQ LREG (SK.SCALE.REGION (fetch (BOX GLOBALREGION) of INDGELT)
                            SCALE))
          (RETURN (create SCREENELT
                         LOCALPART _ (create LOCALBOX
                                            BOXLL _ (create POSITION
                                                           XCOORD _ (fetch (REGION LEFT) of LREG)
                                                           YCOORD _ (fetch (REGION BOTTOM) of LREG))
                                            BOXUR _ (create POSITION
                                                           XCOORD _ (fetch (REGION PRIGHT) of LREG)
                                                           YCOORD _ (fetch (REGION PTOP) of LREG))
                                            LOCALREGION _ LREG
                                            LOCALBOXBRUSH _
                                            (SCALE.BRUSH (COND
                                                             ([NOT (NUMBERP (SETQ LREG (fetch (BOX BRUSH)
                                                                                          of INDGELT]
                                                                  (* new format, old format had brush width only.)
                                                              LREG)
                                                             (T [replace (BOX BRUSH) of INDGELT
                                                                   with (SETQ LREG
                                                                         (create BRUSH
                                                                                BRUSHSIZE _ LREG
                                                                                BRUSHSHAPE _ 'ROUND]
                                                                LREG))
                                                   (fetch (BOX BOXINITSCALE) of INDGELT)
                                                   SCALE)
                                            LOCALBOXFILLING _ (APPEND (fetch (BOX BOXFILLING) of INDGELT))
                                            LOCALBOXDASHING _ (fetch (BOX BOXDASHING) of INDGELT))
                         GLOBALPART _ GBOX])
```

(SK.BOX.GETREGIONFN

```
  [LAMBDA (FIXPT MOVINGPT W)                                         (* rrb "12-May-86 18:38")

          (* getregion fn that generates an error if a point is clicked outside of window.
          Also puts things on the window grid.)

    (SKETCHW.UPDATE.LOCATORS W)
    (COND
       [MOVINGPT

          (* this test the fixed pt every time which is unnecessary but does allow us to catch button down.)

              (PROG [(REG (WINDOWPROP W 'REGION]
                    (RETURN (COND
                               ((INSIDEP REG FIXPT)
                                (COND
                                   ((INSIDEP REG MOVINGPT)
                                    (MAP.SCREEN.POSITION.ONTO.GRID MOVINGPT W (LASTMOUSESTATE MIDDLE)))
                                   (T                                (* if the cursor is outside, return the fixed point so the feedback
                                                                      box disappears.)
                                      FIXPT)))
                               (T (ERROR!]
       (T (MAP.SCREEN.POSITION.ONTO.GRID FIXPT W (LASTMOUSESTATE RIGHT])
```

(BOX.SET.SCALES

```
  [LAMBDA (GREG GBOXELT)                                             (* rrb " 7-Feb-85 12:30")
                                                                     (* updates the scale field after a change in the region of a box
                                                                      element.)

          (* removed the part of the scale that was limiting it to defaults. If it has to go back in, please leave a note as to why.)

    (PROG (WIDTH HEIGHT)
          (replace (GLOBALPART MINSCALE) of GBOXELT with (FQUOTIENT (MIN (SETQ WIDTH (fetch (REGION WIDTH)
                                                                                          of GREG))
                                                                          (SETQ HEIGHT (fetch (REGION HEIGHT)
                                                                                          of GREG)))
                                                                     1000.0))
          (replace (GLOBALPART MAXSCALE) of GBOXELT with (FQUOTIENT (MAX WIDTH HEIGHT)
                                                                     2.0))
          (RETURN GBOXELT])
```

(SK.BOX.INPUTFN

```
  [LAMBDA (W LREGION)                                                (* rrb "11-Jul-86 15:48")

          (* creates a box element for a sketch window. Prompts the user for one if none is given.)

    (PROG (LOCALREG SKCONTEXT)
          (COND
              ((REGIONP LREGION)
               (SETQ LOCALREG LREGION))
              [(NULL LREGION)
               (COND
                  [[SETQ LOCALREG (CAR (ERSETQ (GETWREGION W (FUNCTION SK.BOX.GETREGIONFN)
                                                     W]            (* WINDOWPROP will get exterior of window which should
                                                                   really be reduced to the interior.)
                                                                   (* make sure the last selected point wasn't outside.)
                   (COND
                      ((OR (NOT (SUBREGIONP (DSPCLIPPINGREGION NIL W)
                                            LOCALREG))
                           (AND (EQ (fetch (REGION WIDTH) of LOCALREG)
                                    0)
                                (EQ (fetch (REGION HEIGHT) of LOCALREG)
                                    0)))
                       (RETURN]
                  (T (RETURN]
              (T (\ILLEGAL.ARG LREGION)))
           (RETURN (SK.BOX.CREATE (UNSCALE.REGION.TO.GRID LOCALREG (VIEWER.SCALE W))
                          [fetch (SKETCHCONTEXT SKETCHBRUSH) of (SETQ SKCONTEXT (WINDOWPROP W 'SKETCHCONTEXT]
                          (fetch (SKETCHCONTEXT SKETCHDASHING) of SKCONTEXT)
                          (SK.INPUT.SCALE W)
                          (fetch (SKETCHCONTEXT SKETCHFILLING) of SKCONTEXT])
```

```
(SK.BOX.CREATE
  [LAMBDA (SKETCHREGION BRUSH DASHING INITSCALE FILLING)           (* rrb "12-Dec-85 14:33")

           (* * creates a sketch element from a region)

    (SK.UPDATE.BOX.AFTER.CHANGE (create GLOBALPART
                                        INDIVIDUALGLOBALPART _
                                        (create BOX
                                               GLOBALREGION _ SKETCHREGION
                                               BRUSH _ BRUSH
                                               BOXDASHING _ DASHING
                                               BOXINITSCALE _ INITSCALE
                                               BOXFILLING _ FILLING])
```

```
(SK.UPDATE.BOX.AFTER.CHANGE
  [LAMBDA (GBOXELT)                                                (* rrb "12-Dec-85 14:33")
                                                                   (* changes dependent fields after a box element changes.)
    (BOX.SET.SCALES (fetch (BOX GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GBOXELT))
           GBOXELT])
```

```
(SK.BOX.INSIDEFN
  [LAMBDA (GBOX WREG)                                              (* rrb " 5-AUG-83 16:04")
                                                                   (* determines if the global BOX GBOX is inside of WREG.)
    (REGIONSINTERSECTP (fetch (BOX GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GBOX))
           WREG])
```

```
(SK.BOX.REGIONFN
  [LAMBDA (BOXSCRLET)                                              (* rrb " 7-Dec-85 19:41")
                                                                   (* returns the region occupied by a box.)
    (INCREASEREGION (fetch (LOCALBOX LOCALREGION) of (fetch (SCREENELT LOCALPART) of BOXSCRLET))
           (fetch (BRUSH BRUSHSIZE) of (fetch (LOCALBOX LOCALBOXBRUSH) of (fetch (SCREENELT LOCALPART) of BOXSCRLET]
)
```

```
(SK.BOX.GLOBALREGIONFN
  [LAMBDA (GBOXELT)                                                ; Edited 20-Feb-87 16:20 by rrb
                                                                   (* returns the global region occupied by a global box element.)
    (INCREASEREGION (fetch (BOX GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GBOXELT))
           (QUOTIENT (fetch (BRUSH BRUSHSIZE) of (fetch (BOX BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                             of GBOXELT)))
                  2])
```

```
(SK.BOX.READCHANGEFN
  [LAMBDA (SKW SCRNELTS)                                           (* rrb " 5-Mar-86 13:35")

           (* the users has selected SCRNELT to be changed this function reads a specification of how the box elements should
           change.)

    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU
                                           (create MENU
                                                  CENTERFLG _ T
                                                  TITLE _ "Which aspect?"
                                                  ITEMS _ (APPEND (COND
                                                                     [(SKETCHINCOLORP)
                                                                      '(("Outline color" 'BRUSHCOLOR "changes
                                                                               the color of the outline")
                                                                        ("Filling color" 'FILLINGCOLOR
                                                                                   "changes the color of the
                                                                                   filling"]
                                                                         (T NIL))
                                                                     [COND
                                                                        (FILLINGMODEFLG '(("Filling mode"
                                                                                            'FILLINGMODE "changes how
                                                                                            the filling effects the
                                                                                            figures it covers."]
                                                                     '((Filling 'FILLING "allows changing of the
                                                                               filling texture of the box.")
                                                                       ("Outline size" 'SIZE "changes the size of
                                                                               the brush")
                                                                       ("Outline dashing" 'DASHING "changes the
                                                                               dashing of the line."]
                              (SIZE (READSIZECHANGE "Change size how?" T))
                              (FILLING (READ.FILLING.CHANGE))
                              (FILLINGMODE (READ.FILLING.MODE))
                              (DASHING (READ.DASHING.CHANGE))
                              (BRUSHCOLOR [READ.COLOR.CHANGE "Change outline color how?" NIL
                                                (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                              (fetch (SCREENELT GLOBALPART)
                                                                                 of (CAR SCRNELTS))
                                                                              'BRUSH])
                              (FILLINGCOLOR [READ.COLOR.CHANGE "Change filling color how?" T
                                                  (fetch (SKFILLING FILLING.COLOR) of (GETSKETCHELEMENTPROP
                                                                                       (fetch (SCREENELT GLOBALPART)
                                                                                          of (CAR SCRNELTS))
                                                                                       'FILLING])
                              NIL))
                (RETURN (AND HOW (LIST ASPECT HOW])
```

(SK.CHANGE.FILLING

```
  [LAMBDA (ELTWITHFILLING HOW SKW)                                   (* rrb " 9-Jan-86 16:57")
                                                                     (* changes the texture in the element ELTWITHFILLING.)
    (PROG (GFILLEDELT TEXTURE OLDFILLING NEWFILLING TYPE NEWELT)
          (AND (EQ HOW 'NONE)
               (SETQ HOW NIL))
          (RETURN (COND
                     ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHFILLING))
                            '(BOX TEXTBOX CLOSEDWIRE CIRCLE))        (* only works for things that have a filling, for now just boxes and
                      polygons)
                      (SETQ GFILLEDELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHFILLING))
                      [SETQ TEXTURE (fetch (SKFILLING FILLING.TEXTURE) of (SETQ OLDFILLING
                                                                           (SELECTQ TYPE
                                                                               (BOX (fetch (BOX BOXFILLING)
                                                                                       of GFILLEDELT))
                                                                               (TEXTBOX (fetch (TEXTBOX
                                                                                                  TEXTBOXFILLING
                                                                                                  )
                                                                                           of GFILLEDELT))
                                                                               (CLOSEDWIRE (fetch (CLOSEDWIRE

                                                                                                   CLOSEDWIREFILLING
                                                                                                     )
                                                                                              of GFILLEDELT))
                                                                               (CIRCLE (fetch (CIRCLE
                                                                                                 CIRCLEFILLING
                                                                                                 )
                                                                                          of GFILLEDELT))
                                                                               (SHOULDNT]
                      (COND
                         ((NOT (EQUAL HOW TEXTURE))                  (* new filling)
                          (SETQ NEWFILLING (create SKFILLING using OLDFILLING FILLING.TEXTURE _ HOW))
                          (SETQ NEWELT (SELECTQ TYPE
                                           (BOX (create BOX using GFILLEDELT BOXFILLING _ NEWFILLING))
                                           (TEXTBOX (create TEXTBOX using GFILLEDELT TEXTBOXFILLING _ NEWFILLING)
)
                                           (CLOSEDWIRE (create CLOSEDWIRE using GFILLEDELT CLOSEDWIREFILLING _
                                                                                  NEWFILLING))
                                           (CIRCLE (create CIRCLE using GFILLEDELT CIRCLEFILLING _ NEWFILLING))
                                           (SHOULDNT)))
                          (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHFILLING)
                                                 INDIVIDUALGLOBALPART _ NEWELT)
                                 OLDELT _ ELTWITHFILLING
                                 PROPERTY _ 'FILLING
                                 NEWVALUE _ NEWFILLING
                                 OLDVALUE _ OLDFILLING])
```

(SK.CHANGE.FILLING.COLOR

```
  [LAMBDA (ELTWITHFILLING HOW SKW)                                   (* rrb " 9-Jan-86 19:42")
                                                                     (* changes the texture in the element ELTWITHFILLING.)
    (PROG (GFILLEDELT COLOR FILLING NEWFILLING TYPE NEWELT)
          (AND (EQ HOW 'NONE)
               (SETQ HOW NIL))
          (RETURN (COND
                     ((MEMB (SETQ TYPE (fetch (GLOBALPART GTYPE) of ELTWITHFILLING))
                            '(BOX TEXTBOX CLOSEDWIRE CIRCLE))     (* only works for things that have a filling, for now just boxes and
                      polygons)
                      (SETQ GFILLEDELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of ELTWITHFILLING))
                      [SETQ COLOR (fetch (SKFILLING FILLING.COLOR) of (SETQ FILLING
                                                                        (SELECTQ TYPE
                                                                            (BOX (fetch (BOX BOXFILLING)
                                                                                    of GFILLEDELT))
                                                                            (TEXTBOX (fetch (TEXTBOX TEXTBOXFILLING
                                                                                                     )
                                                                                        of GFILLEDELT))
                                                                            (CIRCLE (fetch (CIRCLE CIRCLEFILLING)
                                                                                       of GFILLEDELT))
                                                                            (CLOSEDWIRE (fetch (CLOSEDWIRE
                                                                                                 CLOSEDWIREFILLING
                                                                                                     )
                                                                                           of GFILLEDELT))
                                                                            (SHOULDNT]
                      (COND
                         ((NOT (EQUAL HOW COLOR))                 (* new filling)
                          (SETQ NEWFILLING (create SKFILLING using FILLING FILLING.COLOR _ HOW))
                          (SETQ NEWELT (SELECTQ TYPE
                                           (BOX (create BOX using GFILLEDELT BOXFILLING _ NEWFILLING))
                                           (TEXTBOX (create TEXTBOX using GFILLEDELT TEXTBOXFILLING _ NEWFILLING)
)
                                           (CLOSEDWIRE (create CLOSEDWIRE using GFILLEDELT CLOSEDWIREFILLING _
                                                                                 NEWFILLING))
                                           (CIRCLE (create CIRCLE using GFILLEDELT CIRCLEFILLING _ NEWFILLING))
                                           (SHOULDNT)))
                          (create SKHISTORYCHANGESPEC
                                 NEWELT _ (create GLOBALPART
                                                 COMMONGLOBALPART _ (fetch (GLOBALPART COMMONGLOBALPART)
                                                                       of ELTWITHFILLING)
                                                 INDIVIDUALGLOBALPART _ NEWELT)
                                 OLDELT _ ELTWITHFILLING
                                 PROPERTY _ 'FILLING
                                 NEWVALUE _ NEWFILLING
                                 OLDVALUE _ FILLING])
```

(SK.BOX.TRANSLATEFN

```
  [LAMBDA (SKELT DELTAPOS)                                        (* rrb "28-Apr-85 18:46")

          (* * returns a curve element which has the box translated by DELTAPOS)

    (PROG ((GBOXELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of SKELT)))
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of SKELT))
                         INDIVIDUALGLOBALPART _ (create BOX using GBOXELT GLOBALREGION _
                                                                   (REL.MOVE.REGION (fetch (BOX GLOBALREGION)
                                                                                       of GBOXELT)
                                                                          (fetch (POSITION XCOORD) of DELTAPOS)
                                                                          (fetch (POSITION YCOORD) of DELTAPOS])
```

(SK.BOX.TRANSFORMFN

```
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)            (* rrb "12-Jul-85 17:16")

          (* returns a copy of the global BOX element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (SK.BOX.CREATE (SK.TRANSFORM.REGION (fetch (BOX GLOBALREGION) of INDVPART)
                                        TRANSFORMFN TRANSFORMDATA)
                         (SK.TRANSFORM.BRUSH (fetch (BOX BRUSH) of INDVPART)
                                SCALEFACTOR)
                         (fetch (BOX BOXDASHING) of INDVPART)
                         (fetch (BOX BOXINITSCALE) of INDVPART)
                         (fetch (BOX BOXFILLING) of INDVPART])
```

(SK.BOX.TRANSLATEPTSFN

```
  [LAMBDA (BOXELT SELPTS GDELTA WINDOW)                           (* rrb "12-Jul-85 17:55")

          (* returns a closed wire element which has the knots that are members of SELPTS translated by the global amount
          GDELTA.)

    (PROG ((GBOXELT (fetch (SCREENELT INDIVIDUALGLOBALPART) of BOXELT))
           OLDGLOBALREGION LLX LLY URX URY)
          (SETQ OLDGLOBALREGION (fetch (BOX GLOBALREGION) of GBOXELT))
          [COND
             [(MEMBER (fetch (LOCALBOX BOXLL) of (fetch (SCREENELT LOCALPART) of BOXELT))
                     SELPTS)                                      (* lower left point is moving.)
              (SETQ LLX (PLUS (fetch (REGION LEFT) of OLDGLOBALREGION)
                             (fetch (POSITION XCOORD) of GDELTA)))
              (SETQ LLY (PLUS (fetch (REGION BOTTOM) of OLDGLOBALREGION)
                             (fetch (POSITION YCOORD) of GDELTA]
             (T (SETQ LLX (fetch (REGION LEFT) of OLDGLOBALREGION))
                (SETQ LLY (fetch (REGION BOTTOM) of OLDGLOBALREGION]
          [COND
             [(MEMBER (fetch (LOCALBOX BOXUR) of (fetch (SCREENELT LOCALPART) of BOXELT))
                     SELPTS)                                          (* upper right point)
              (SETQ URX (PLUS (fetch (REGION PRIGHT) of OLDGLOBALREGION)
                             (fetch (POSITION XCOORD) of GDELTA)))
              (SETQ URY (PLUS (fetch (REGION PTOP) of OLDGLOBALREGION)
                             (fetch (POSITION YCOORD) of GDELTA]
             (T (SETQ URX (fetch (REGION PRIGHT) of OLDGLOBALREGION))
                (SETQ URY (fetch (REGION PTOP) of OLDGLOBALREGION]
          (RETURN (SK.BOX.CREATE (CREATEREGION (MIN LLX URX)
                                        (MIN LLY URY)
                                        (ABS (DIFFERENCE LLX URX))
                                        (ABS (DIFFERENCE LLY URY)))
                         (fetch (BOX BRUSH) of GBOXELT)
                         (fetch (BOX BOXDASHING) of GBOXELT)
                         (fetch (BOX BOXINITSCALE) of GBOXELT)
                         (fetch (BOX BOXFILLING) of GBOXELT])
```

(UNSCALE.REGION.TO.GRID

```
  [LAMBDA (REGION SCALE GRIDSIZE)                                      (* rrb "25-Oct-84 12:53")
                                                                       (* scales a region from a window region to the larger coordinate
   space.)
    (PROG [(LFT (TIMES SCALE (fetch (REGION LEFT) of REGION)))
           (BTM (TIMES SCALE (fetch (REGION BOTTOM) of REGION)))
           (WDTH (TIMES SCALE (fetch (REGION WIDTH) of REGION)))
           (HGHT (TIMES SCALE (fetch (REGION HEIGHT) of REGION]
          [COND
             (GRIDSIZE                                                 (* move X and Y to nearest point on the grid)
                    (SETQ LFT (NEAREST.ON.GRID LFT GRIDSIZE))
                    (SETQ BTM (NEAREST.ON.GRID BTM GRIDSIZE))
                    (SETQ WDTH (NEAREST.ON.GRID WDTH GRIDSIZE))
                    (SETQ HGHT (NEAREST.ON.GRID HGHT GRIDSIZE]
          (RETURN (CREATEREGION LFT BTM WDTH HGHT])
```

(INCREASEREGION

```
  [LAMBDA (REGION BYAMOUNT)                                            (* rrb " 9-Sep-84 19:58")

          (* * increases a region by a fixed amount in all directions.)

    (CREATEREGION (DIFFERENCE (fetch (REGION LEFT) of REGION)
                         BYAMOUNT)
           (DIFFERENCE (fetch (REGION BOTTOM) of REGION)
                  BYAMOUNT)
           (PLUS (fetch (REGION WIDTH) of REGION)
                 (TIMES BYAMOUNT 2))
           (PLUS (fetch (REGION HEIGHT) of REGION)
                 (TIMES BYAMOUNT 2])
```

(INSUREREGIONSIZE

```
  [LAMBDA (REGION MINSIZE)                                             (* rrb " 5-Dec-84 11:27")

          (* * makes sure the height and width of REGION are at least MINSIZE.)

    (PROG (X)
          (COND
             ((GREATERP MINSIZE (SETQ X (fetch (REGION WIDTH) of REGION)))
              (replace (REGION LEFT) of REGION with (DIFFERENCE (fetch (REGION LEFT) of REGION)
                                                           (QUOTIENT (DIFFERENCE MINSIZE X)
                                                                  2)))
              (replace (REGION WIDTH) of REGION with MINSIZE)))
          (COND
             ((GREATERP MINSIZE (SETQ X (fetch (REGION HEIGHT) of REGION)))
              (replace (REGION BOTTOM) of REGION with (DIFFERENCE (fetch (REGION BOTTOM) of REGION)
                                                             (QUOTIENT (DIFFERENCE MINSIZE X)
                                                                    2)))
              (replace (REGION HEIGHT) of REGION with MINSIZE)))
          (RETURN REGION])
```

(EXPANDREGION

```
  [LAMBDA (REGION BYFACTOR)                                            (* rrb "30-Nov-84 10:43")

          (* * expands a region by a factor.)

    (PROG ((WIDTH (fetch (REGION WIDTH) of REGION))
           (HEIGHT (fetch (REGION HEIGHT) of REGION))
          NEWWIDTH NEWHEIGHT)
          (SETQ NEWWIDTH (TIMES WIDTH BYFACTOR))
          (SETQ NEWHEIGHT (TIMES HEIGHT BYFACTOR))
          (RETURN (CREATEREGION (DIFFERENCE (fetch (REGION LEFT) of REGION)
                                      (QUOTIENT (IDIFFERENCE NEWWIDTH WIDTH)
                                             2))
                         (DIFFERENCE (fetch (REGION BOTTOM) of REGION)
                                (QUOTIENT (IDIFFERENCE NEWHEIGHT HEIGHT)
                                       2))
                         NEWWIDTH NEWHEIGHT])
```

(**REGION.FROM.COORDINATES**

```
  [LAMBDA (X1 Y1 X2 Y2)                                         (* rrb "11-Sep-84 16:27")

          (* * returns the region for which { X1 Y1 } and { X2 Y2} are the corners.)

    (CREATEREGION (MIN X1 X2)
           (MIN Y1 Y2)
           (ADD1 (ABS (IDIFFERENCE X2 X1)))
           (ADD1 (ABS (IDIFFERENCE Y2 Y1])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(TYPERECORD BOX (GLOBALREGION BRUSH BOXDASHING BOXINITSCALE BOXFILLING))

(RECORD LOCALBOX ((BOXLL BOXUR)
                  LOCALHOTREGION LOCALREGION LOCALBOXBRUSH LOCALBOXFILLING LOCALBOXDASHING))
)
)

(READVARS-FROM-STRINGS '(BOXICON)
       "({(READBITMAP)(20 12
       %"@@@@@@@@%"
       %"GOOON@@@%"
       %"GOOON@@@%"
       %"F@@@F@@@%"
       %"F@@@F@@@%"
       %"F@@@F@@@%"
       %"F@@@F@@@%"
       %"F@@@F@@@%"
       %"F@@@F@@@%"
       %"GOOON@@@%"
       %"GOOON@@@%"
       %"@@@@@@@@%")})
       ")
```

;; fns for the arc sketch element type

```
(DEFINEQ
```

(**SKETCH.CREATE.ARC**

```
  [LAMBDA (CENTERPT RADIUSPT ANGLEPT BRUSH DASHING ARROWHEADS DIRECTION SCALE)
                                                                (* rrb " 7-Jul-86 14:49")
                                                                (* creates a sketch arc element.)
    (ARC.CREATE (SK.INSURE.POSITION CENTERPT)
           (SK.INSURE.POSITION RADIUSPT)
           (COND
              ((NUMBERP ANGLEPT)
               (SK.COMPUTE.ARC.ANGLE.PT.FROM.ANGLE CENTERPT RADIUSPT ANGLEPT))
              (T (SK.INSURE.POSITION ANGLEPT)))
           (SK.INSURE.BRUSH BRUSH)
           (SK.INSURE.DASHING DASHING)
           (OR (NUMBERP SCALE)
               1.0)
           (SK.INSURE.ARROWHEADS ARROWHEADS)
           (SK.INSURE.DIRECTION DIRECTION])
```

(**ARC.DRAWFN**

```
  [LAMBDA (ARCELT WINDOW REGION)                                (* rrb "20-Jun-86 17:12")
                                                                (* draws a arc from a arc element.)
    (PROG ((GARC (fetch (SCREENELT INDIVIDUALGLOBALPART) of ARCELT))
           (LARC (fetch (SCREENELT LOCALPART) of ARCELT))
           BRUSH DASHING LOCALPTS LOCALARROWPTS GARROWSPECS)
          (AND REGION (NOT (REGIONSINTERSECTP REGION (SK.ITEM.REGION ARCELT)))
               (RETURN))
          (SETQ GARROWSPECS (fetch (ARC ARCARROWHEADS) of GARC))
          (SETQ LOCALARROWPTS (fetch (LOCALARC LOCALARCARROWHEADPTS) of LARC))
          (SETQ BRUSH (fetch (LOCALARC LOCALARCBRUSH) of LARC))
          (SETQ DASHING (fetch (LOCALARC LOCALARCDASHING) of LARC))
          (COND
             [(EQ T (fetch (ARC ARCANGLEPT) of GARC))            (* T means greater than 360)
              (PROG ((CPT (fetch (LOCALARC LOCALARCCENTERPT) of LARC))
                     (RPT (fetch (LOCALARC LOCALARCRADIUSPT) of LARC)))
                    (RETURN (\CIRCLE.DRAWFN1 CPT RPT (DISTANCEBETWEEN CPT RPT)
                                 BRUSH DASHING WINDOW]
             (T (SETQ LOCALPTS (\SK.ADJUST.FOR.ARROWHEADS (fetch (LOCALARC LOCALARCKNOTS) of LARC)
                                    LOCALARROWPTS GARROWSPECS WINDOW))
                                                                      (* draw the curve from the knots)
                (DRAWCURVE LOCALPTS NIL BRUSH DASHING WINDOW)))
          (DRAWARROWHEADS GARROWSPECS LOCALARROWPTS WINDOW BRUSH])
```

(**ARC.EXPANDFN**

```
  [LAMBDA (GARC SCALE)                                              (* rrb "20-Jun-86 13:58")

          (* returns a screen elt that has a arc screen element calculated from the global part.)

    (PROG ((INDGARC (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARC))
           CENTER RADIUSPT ANGLEPT LOCALKNOTS LOCALARROWHEADS)
          (SETQ CENTER (SK.SCALE.POSITION.INTO.VIEWER (fetch (ARC ARCCENTERPT) of INDGARC)
                              SCALE))
          (SETQ RADIUSPT (SK.SCALE.POSITION.INTO.VIEWER (fetch (ARC ARCRADIUSPT) of INDGARC)
                                SCALE))
          (SETQ ANGLEPT (SK.SCALE.POSITION.INTO.VIEWER (\SK.GET.ARC.ANGLEPT INDGARC)
                               SCALE))
          (SETQ LOCALKNOTS (SK.COMPUTE.ARC.PTS CENTER RADIUSPT ANGLEPT (fetch (ARC ARCDIRECTION) of INDGARC)))
          (COND
             ((AND (fetch (ARC ARCARROWHEADS) of INDGARC)
                   (NOT (fetch (ARC ARCARROWHEADPOINTS) of INDGARC)))

          (* check to make sure the global arrowhead points have been calculated.
          Old form didn't have them.)

              (SET.ARC.ARROWHEAD.POINTS INDGARC)))
          (SETQ LOCALARROWHEADS (SK.EXPAND.ARROWHEADS (fetch (ARC ARCARROWHEADPOINTS) of INDGARC)
                                       SCALE))
          (RETURN (create SCREENELT
                         LOCALPART _
                         (create LOCALARC
                                LOCALARCCENTERPT _ CENTER
                                LOCALARCRADIUSPT _ RADIUSPT
                                LOCALARCANGLEPT _ ANGLEPT
                                LOCALARCARROWHEADPTS _ LOCALARROWHEADS
                                LOCALARCBRUSH _ (SCALE.BRUSH (fetch (ARC ARCBRUSH) of INDGARC)
                                                       (fetch (ARC ARCINITSCALE) of INDGARC)
                                                       SCALE)
                                LOCALARCKNOTS _ LOCALKNOTS
                                LOCALARCDASHING _ (fetch (ARC ARCDASHING) of INDGARC))
                         GLOBALPART _ GARC])
```

(**ARC.INPUTFN**

```
  [LAMBDA (WINDOW)                                                  (* rrb "20-May-86 10:53")

          (* reads three points from the user and returns the arc figure element that it represents.)

    (PROG [CENTER RADPT ANGLEPT DIRECTION (SKCONTEXT (WINDOWPROP WINDOW 'SKETCHCONTEXT]
          (SETQ DIRECTION (fetch (SKETCHCONTEXT SKETCHARCDIRECTION) of SKCONTEXT))
          (STATUSPRINT WINDOW "
                 " "Indicate center of the arc")
          (COND
             ((SETQ CENTER (SK.READ.POINT.WITH.FEEDBACK WINDOW ELLIPSE.CENTER NIL NIL NIL NIL
                                  SKETCH.USE.POSITION.PAD))
              (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                     NIL WINDOW))
             (T (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                 " "Indicate end of the arc")
          (COND
             [(SETQ RADPT (SK.READ.CIRCLE.POINT WINDOW (fetch (INPUTPT INPUT.POSITION) of CENTER)
                                 (COND
                                    (DIRECTION                      (* use a cursor that shows the arc going in the correct direction.)
                                           CW.ARC.RADIUS.CURSOR)
                                    (T ARC.RADIUS.CURSOR]
             (T                                                     (* erase center pt on way out)
                (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                       NIL WINDOW)
                (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (COND
             ((NEQ SKETCH.VERBOSE.FEEDBACK 'ALWAYS)                 (* if feedback in medium mode, put up circle)
              (SK.INVERT.CIRCLE CENTER RADPT WINDOW))
             (T                                                     (* if feedback is in very verbose mode, just put up the radius pt.)
                (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of RADPT)
                       NIL WINDOW)))
          (STATUSPRINT WINDOW "
                 " "Indicate the angle of the arc")
          (SETQ ANGLEPT (SK.READ.ARC.ANGLE.POINT WINDOW (COND
                                                               (DIRECTION CW.ARC.ANGLE.CURSOR)
                                                               (T ARC.ANGLE.CURSOR))
```

```
                              (fetch (INPUTPT INPUT.POSITION) of CENTER)
                              (fetch (INPUTPT INPUT.POSITION) of RADPT)
                              DIRECTION))
          (CLOSEPROMPTWINDOW WINDOW)                               (* erase the point marks.)
          (COND
             ((NEQ SKETCH.VERBOSE.FEEDBACK 'ALWAYS)                (* if feedback in medium mode, put up circle)
              (SK.INVERT.CIRCLE CENTER RADPT WINDOW))
             (T                                                    (* if feedback is in very verbose mode, just put up the radius pt.)
                (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of RADPT)
                       NIL WINDOW)))
          (MARK.SPOT (fetch (INPUTPT INPUT.POSITION) of CENTER)
                 NIL WINDOW)
          (OR ANGLEPT (RETURN NIL))

          (* the list of knots passed to SK.ARROWHEAD.CREATE is only used to determine right and left so don't bother to create a
          good one. Actually this introduces a bug when the angle point is not on the same side of the radius point as the end of the
          arc is. should fix.)

          (RETURN (ARC.CREATE (SK.MAP.INPUT.PT.TO.GLOBAL CENTER WINDOW)
                         (SK.MAP.INPUT.PT.TO.GLOBAL RADPT WINDOW)
                         (SK.MAP.INPUT.PT.TO.GLOBAL ANGLEPT WINDOW)
                         (fetch (SKETCHCONTEXT SKETCHBRUSH) of SKCONTEXT)
                         (fetch (SKETCHCONTEXT SKETCHDASHING) of SKCONTEXT)
                         (SK.INPUT.SCALE WINDOW)
                         (SK.ARROWHEAD.CREATE WINDOW (LIST RADPT ANGLEPT))
                         DIRECTION])
```

(SK.INVERT.CIRCLE

```
  [LAMBDA (CENTERIPT RADIUSIPT SKW)                                (* rrb "18-Nov-85 14:36")

          (* draws a circle as feedback while the user in inputting the angle point of an arc.)

    (PROG ((PREVOP (DSPOPERATION 'INVERT SKW)))
          (RETURN (PROG1 (SK.SHOW.CIRCLE (fetch (POSITION XCOORD) of (fetch (INPUTPT INPUT.POSITION) of RADIUSIPT)
                                                )
                                (fetch (POSITION YCOORD) of (fetch (INPUTPT INPUT.POSITION) of RADIUSIPT))
                                SKW
                                (fetch (INPUTPT INPUT.POSITION) of CENTERIPT))
                         (DSPOPERATION PREVOP SKW])
```

(SK.READ.ARC.ANGLE.POINT

```
  [LAMBDA (WINDOW CURSOR CENTERPT RADIUSPT DIRECTION)              (* rrb "20-May-86 10:48")

          (* reads a point from the user prompting them with an arc that follows the cursor)

    (SK.READ.POINT.WITH.FEEDBACK WINDOW CURSOR (AND (EQ SKETCH.VERBOSE.FEEDBACK 'ALWAYS)
                                                    (FUNCTION SK.SHOW.ARC))
           (LIST CENTERPT RADIUSPT DIRECTION)
           'MIDDLE NIL SKETCH.USE.POSITION.PAD])
```

(SK.SHOW.ARC

```
  [LAMBDA (X Y WINDOW ARCARGS)                                     (* rrb "15-Nov-85 14:32")
                                                                   (* draws an arc as feedback for reading the angle point of an
                                                                   arc.)
                                                                   (* Mark the point too.)
    (SHOWSKETCHXY X Y WINDOW)
    (DRAWCURVE (SK.COMPUTE.ARC.PTS (CAR ARCARGS)
                      (CADR ARCARGS)
                      (create POSITION
                             XCOORD _ X
                             YCOORD _ Y)
                      (CADDR ARCARGS))
           NIL 1 NIL WINDOW])
```

(ARC.CREATE

```
  [LAMBDA (CENTERPT RADPT ANGLEPT BRUSH DASHING INITSCALE ARROWHEADS DIRECTION)
                                                                   (* rrb "19-Mar-86 17:19")
                                                                   (* creates a global arc element.)
    (PROG ((ARCANGLEPT (SK.COMPUTE.ARC.ANGLE.PT CENTERPT RADPT ANGLEPT)))
          (RETURN (SET.ARC.SCALES (create GLOBALPART
                                         INDIVIDUALGLOBALPART _
                                         (SET.ARC.ARROWHEAD.POINTS (create ARC
                                                                          ARCCENTERPT _ CENTERPT
                                                                          ARCRADIUSPT _ RADPT
                                                                          ARCBRUSH _ BRUSH
                                                                          ARCDASHING _ DASHING
                                                                          ARCINITSCALE _ INITSCALE
                                                                          ARCARROWHEADS _ ARROWHEADS
                                                                          ARCANGLEPT _ ARCANGLEPT
                                                                          ARCDIRECTION _ DIRECTION])
```

(SK.UPDATE.ARC.AFTER.CHANGE

```
  [LAMBDA (GARCELT)                                                  (* rrb " 7-Dec-85 19:52")
                                                                     (* updates the dependent fields of a arc element when a field
                                                                     changes.)
    (replace (ARC ARCREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARCELT) with NIL])
```

(ARC.MOVEFN

```
  [LAMBDA (ARCELT SELPOS NEWPOS WINDOW)                              (* rrb "15-Dec-86 15:19")
                                                                     (* returns a global arc element which has the part SELPOS
                                                                     moved to NEWPOS.)
    (PROG ((LOCALEL (fetch (SCREENELT LOCALPART) of ARCELT))
           (GLOBALEL (fetch (SCREENELT INDIVIDUALGLOBALPART) of ARCELT))
          CENTERPT ANGLEPT RADPT PTSCALE)
          (SETQ CENTERPT (fetch (ARC ARCCENTERPT) of GLOBALEL))
          (SETQ ANGLEPT (fetch (ARC ARCANGLEPT) of GLOBALEL))
          (SETQ RADPT (fetch (ARC ARCRADIUSPT) of GLOBALEL))         (* find the point that has moved and change it.)
          [COND
             ((EQUAL SELPOS (fetch (LOCALARC LOCALARCCENTERPT) of LOCALEL))
              (SETQ CENTERPT (SK.MAP.FROM.WINDOW.TO.GLOBAL.GRID NEWPOS WINDOW)))
             ((EQUAL SELPOS (fetch (LOCALARC LOCALARCANGLEPT) of LOCALEL))
              (SETQ ANGLEPT (SK.MAP.FROM.WINDOW.TO.GLOBAL.GRID NEWPOS WINDOW)))
             ((EQUAL SELPOS (fetch (LOCALARC LOCALARCRADIUSPT) of LOCALEL))
              (SETQ RADPT (SK.MAP.FROM.WINDOW.TO.GLOBAL.GRID NEWPOS WINDOW]

          (* return a new global elt because the orientation changes but is needed to erase the one that is already on the screen.)

          (RETURN (SK.CREATE.ARC.USING CENTERPT RADPT ANGLEPT (fetch (SCREENELT GLOBALPART) of ARCELT])
```

(ARC.TRANSLATEPTS

```
  [LAMBDA (ARCELT SELPTS GLOBALDELTA WINDOW)                         (* rrb "15-Dec-86 15:19")

          (* returns a new global arc element which has the points on SELPTS moved by a global distance.)

    (PROG ((LOCALEL (fetch (SCREENELT LOCALPART) of ARCELT))
           (GLOBALEL (fetch (SCREENELT INDIVIDUALGLOBALPART) of ARCELT))
          CENTERPT ANGLEPT RADPT PTSCALE)
          (SETQ CENTERPT (fetch (ARC ARCCENTERPT) of GLOBALEL))
          (SETQ ANGLEPT (fetch (ARC ARCANGLEPT) of GLOBALEL))
          (SETQ RADPT (fetch (ARC ARCRADIUSPT) of GLOBALEL))         (* find the point that has moved and change it.)
          [COND
             ((MEMBER (fetch (LOCALARC LOCALARCCENTERPT) of LOCALEL)
                     SELPTS)
              (SETQ CENTERPT (PTPLUS CENTERPT GLOBALDELTA]
          [COND
             ((MEMBER (fetch (LOCALARC LOCALARCRADIUSPT) of LOCALEL)
                     SELPTS)
              (SETQ RADPT (PTPLUS RADPT GLOBALDELTA]
          [COND
             ((MEMBER (fetch (LOCALARC LOCALARCANGLEPT) of LOCALEL)
                     SELPTS)
              (COND
                 [(EQ ANGLEPT T)

          (* user moved the point that is both the radius pt and the angle pt.
          If it was the only point moved, don't move the angle pt, just the radius pt.)

                  (COND
                     ((NULL (CDR SELPTS))
                      (SETQ ANGLEPT (fetch (ARC ARCRADIUSPT) of GLOBALEL]
                 (T (SETQ ANGLEPT (PTPLUS ANGLEPT GLOBALDELTA]
          (RETURN (SK.CREATE.ARC.USING CENTERPT RADPT ANGLEPT (fetch (SCREENELT GLOBALPART) of ARCELT])
```

(ARC.INSIDEFN

```
  [LAMBDA (GARC WREG)                                                (* rrb "20-Jan-87 14:44")
                                                                     (* determines if the global arc GARC is inside of WREG.)
    (REGIONSINTERSECTP WREG (ARC.GLOBALREGIONFN GARC])
```

(ARC.REGIONFN

```
  [LAMBDA (ARCSCRELT)                                                (* rrb "30-May-85 12:23")
                                                                     (* returns the region occuppied by an arc.)

          (* uses the heuristic that the region containing the curve is not more than 10% larger than the knots.
          This was determined empirically on several curves.)

    (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (fetch (LOCALARC LOCALARCKNOTS)
                                                               of (fetch (SCREENELT LOCALPART) of ARCSCRELT)))
                           1.1)
           (IQUOTIENT [ADD1 (SK.BRUSH.SIZE (fetch (LOCALARC LOCALARCBRUSH) of (fetch (SCREENELT LOCALPART)
                                                                                     of ARCSCRELT]
                  2])
```

(ARC.GLOBALREGIONFN

```
  [LAMBDA (GARCELT)                                                  (* rrb "20-Jun-86 14:04")
```

```
                                                                (* returns the global region occupied by a global arc element.)
    (OR (fetch (ARC ARCREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARCELT))
        (PROG ((INDVARC (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARCELT))
               REGION)

          (* uses the heuristic that the region containing the curve is not more than 10% larger than the knots.
          This was determined empirically on several curves.)

              [SETQ REGION (INCREASEREGION (EXPANDREGION (REGION.CONTAINING.PTS (SK.COMPUTE.ARC.PTS
                                                                                 (fetch (ARC ARCCENTERPT)
                                                                                    of INDVARC)
                                                                                 (fetch (ARC ARCRADIUSPT)
                                                                                    of INDVARC)
                                                                                 (\SK.GET.ARC.ANGLEPT INDVARC
                                                                                        )
                                                                                 (fetch (ARC ARCDIRECTION)
                                                                                    of INDVARC)))
                                                         1.1)
                                   (SK.BRUSH.SIZE (fetch (ARC ARCBRUSH) of INDVARC]
              (replace (ARC ARCREGION) of INDVARC with REGION)
              (RETURN REGION])

(ARC.TRANSLATE
  [LAMBDA (GARCELT DELTAPOS)                                    (* rrb "15-Dec-86 15:20")
                                                                (* returns a global arc element which has the arc translated by
                                                                DELTAPOS.)
    (PROG ((GLOBALEL (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARCELT)))
          (RETURN (SK.CREATE.ARC.USING (PTPLUS (fetch (ARC ARCCENTERPT) of GLOBALEL)
                                               DELTAPOS)
                         (PTPLUS (fetch (ARC ARCRADIUSPT) of GLOBALEL)
                                DELTAPOS)
                         (COND
                            ((POSITIONP (fetch (ARC ARCANGLEPT) of GLOBALEL))
                             (PTPLUS (fetch (ARC ARCANGLEPT) of GLOBALEL)
                                    DELTAPOS))
                            (T                                  (* T marks greater than 360)
                               T))
                         GARCELT])

(ARC.TRANSFORMFN
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)          (* rrb "15-Dec-86 15:20")

          (* returns a copy of the global element that has had each of its control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG ((INDVPART (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           NEWGELT)
          (SETQ NEWGELT (SK.CREATE.ARC.USING (SK.TRANSFORM.POINT (fetch (ARC ARCCENTERPT) of INDVPART)
                                                    TRANSFORMFN TRANSFORMDATA)
                               (SK.TRANSFORM.POINT (fetch (ARC ARCRADIUSPT) of INDVPART)
                                      TRANSFORMFN TRANSFORMDATA)
                               (COND
                                  ((POSITIONP (fetch (ARC ARCANGLEPT) of INDVPART))
                                   (SK.TRANSFORM.POINT (fetch (ARC ARCANGLEPT) of INDVPART)
                                          TRANSFORMFN TRANSFORMDATA))
                                  (T                            (* T marks greater than 360)
                                     T))
                               GELT))                           (* update the brush too.)
          (replace (ARC ARCBRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT)
             with (SK.TRANSFORM.BRUSH (fetch (ARC ARCBRUSH) of INDVPART)
                         SCALEFACTOR))
          (replace (ARC ARCARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT)
             with (SK.TRANSFORM.ARROWHEADS (fetch (ARC ARCARROWHEADS) of INDVPART)
                         SCALEFACTOR))
          (SET.ARC.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT))
          [AND (EQ TRANSFORMFN 'SK.APPLY.AFFINE.TRANSFORM)
               (COND
                  ([COND
                      [(GREATERP 0.0 (fetch (AFFINETRANSFORMATION Ax) of TRANSFORMDATA))
                                                                (* x coord is reflected, switch direction unless Y is reflected also.)
                       (NOT (GREATERP 0.0 (fetch (AFFINETRANSFORMATION Ey) of TRANSFORMDATA]
                      (T (GREATERP 0.0 (fetch (AFFINETRANSFORMATION Ey)
                                              TRANSFORMDATA]    (* change the direction if the transformation reflects.)
                   (replace (ARC ARCDIRECTION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of NEWGELT)
                      with (NOT (fetch (ARC ARCDIRECTION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART]
          (RETURN NEWGELT])

(ARC.READCHANGEFN
  [LAMBDA (SKW SCRNELTS)                                        (* rrb "17-Dec-85 16:22")
                                                                (* changefn for arcs)
    (PROG (ASPECT HOW)
          (SETQ HOW (SELECTQ [SETQ ASPECT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                                         CENTERFLG _ T
                                                                         TITLE _ "Which aspect?"
```

```
                                                                                        ITEMS _
                                                                                        (APPEND
                                                                                         [COND
                                                                                            ((SKETCHINCOLORP)
                                                                                             '((Color 'BRUSHCOLOR "changes the
                                                                                                       color of the outline"]
                                                                                         '((Arrowheads 'ARROW "allows changing
                                                                                                  of arrow head
                                                                                                  charactistics.")
                                                                                           (Size 'SIZE "changes the size of the
                                                                                                 brush")
                                                                                           (Angle 'ANGLE "changes the amount of
                                                                                                  angle in the arc.")
                                                                                           (Dashing 'DASHING "changes the
                                                                                                  dashing of the line.")
                                                                                           (Direction 'DIRECTION "changes which
                                                                                                  way around the circle the arc
                                                                                                  is drawn."]
                                  (SIZE (READSIZECHANGE "Change size how?"))
                                  (ANGLE (READANGLE))
                                  (ARROW (READ.ARROW.CHANGE SCRNELTS))
                                  (DASHING (READ.DASHING.CHANGE))
                                  (DIRECTION (READARCDIRECTION))
                                  (BRUSHCOLOR [READ.COLOR.CHANGE "Change color how?" NIL
                                                         (fetch (BRUSH BRUSHCOLOR) of (GETSKETCHELEMENTPROP
                                                                                       (fetch (SCREENELT GLOBALPART)
                                                                                          of (CAR SCRNELTS))
                                                                                       'BRUSH])
                                  NIL))
                 (RETURN (AND HOW (LIST ASPECT HOW])

)

(DEFINEQ
```

(SK.COMPUTE.ARC.ANGLE.PT

```
  [LAMBDA (CENTERPT RADPT ANGLEPT)                                        (* rrb "26-Jun-86 17:04")

                 (* computes the intersection of the line CENTERPT ANGLEPT with the circle with center CENTERPT that goes through
                 RADPT.)

    (COND
       ((EQ ANGLEPT T)                                                    (* used to mark more than 360.0)
        T)
       (T (PROG ((RADIUS (DISTANCEBETWEEN CENTERPT RADPT))
                 (ARCANGLE (SK.COMPUTE.SLOPE.OF.LINE CENTERPT ANGLEPT)))
                (RETURN (create POSITION
                               XCOORD _ (PLUS (fetch (POSITION XCOORD) of CENTERPT)
                                              (TIMES RADIUS (COS ARCANGLE)))
                               YCOORD _ (PLUS (fetch (POSITION YCOORD) of CENTERPT)
                                              (TIMES RADIUS (SIN ARCANGLE])
```

(SK.COMPUTE.ARC.ANGLE.PT.FROM.ANGLE

```
  [LAMBDA (CENTERPT RADPT ANGLE)                                          (* rrb " 7-Jul-86 14:49")

                 (* computes the point on the circle with center CENTERPT that goes through RADPT that is angle ANGLE from RADPT.)

    (COND
       ((OR (GEQ ANGLE 360.0)
            (LEQ ANGLE -360.0))                                           (* T denotes all the way around.)
        T)
       (T (PROG ((RADIUS (DISTANCEBETWEEN CENTERPT RADPT))
                 (DELTA (PLUS (SK.COMPUTE.SLOPE.OF.LINE CENTERPT RADPT)
                              ANGLE)))
                (RETURN (create POSITION
                               XCOORD _ (PLUS (fetch (POSITION XCOORD) of CENTERPT)
                                              (TIMES RADIUS (COS DELTA)))
                               YCOORD _ (PLUS (fetch (POSITION YCOORD) of CENTERPT)
                                              (TIMES RADIUS (SIN DELTA])
```

(SK.COMPUTE.ARC.PTS

```
  [LAMBDA (CENTERPT RADIUSPT ARCPT DIRECTION)                             (* DECLARATIONS%: FLOATING)
                                                                          (* rrb " 5-May-86 14:11")
                                                                          (* computes a list of knots that a spline goes through to make an
    arc)
    (PROG ((RADIUS (DISTANCEBETWEEN CENTERPT RADIUSPT))
           (ALPHA (SK.COMPUTE.SLOPE.OF.LINE CENTERPT RADIUSPT))
           (OMEGA (SK.COMPUTE.SLOPE.OF.LINE CENTERPT ARCPT))
           (CENTERX (fetch (POSITION XCOORD) of CENTERPT))
           (CENTERY (fetch (POSITION YCOORD) of CENTERPT))
           PTLST ANGLEINCR DEGREESARC)
          [COND
             [DIRECTION                                                   (* if non-NIL go in a counterclockwise direction.)
                    (COND
                       ((GREATERP OMEGA ALPHA)
                    (SETQ OMEGA (DIFFERENCE OMEGA 360.0]
         (T (COND
               ((GREATERP ALPHA OMEGA)                          (* angle crosses angle change point, correct.)
                (SETQ OMEGA (PLUS OMEGA 360.0]

          (* calculate an increment close to 10.0 that is exact but always have at least 5 knots and don't have more than a knot every
          5 pts)

          [SETQ ANGLEINCR (FQUOTIENT (SETQ DEGREESARC (DIFFERENCE OMEGA ALPHA))
                                 (IMIN (IMAX (ABS (FIX (FQUOTIENT DEGREESARC 10.0)))
                                             5)
                                       (PROGN                   (* don't have more than a knot every 5 pts)
                                              (IMAX (ABS (FIX (QUOTIENT (TIMES RADIUS 6.3 (QUOTIENT DEGREESARC
                                                                                                    360.0))
                                                                        4)))
                                                    3]

          (* go from initial point to just past the last point. The just past (PLUS OMEGA
          (QUOTIENT ANGLEINCR 5.0)) picks up the case where the floating pt rounding error accumulates to be greater than the
          last point when it is very close to it.)

          [SETQ PTLST (for ANGLE from ALPHA to (PLUS OMEGA (QUOTIENT ANGLEINCR 5.0)) by ANGLEINCR
                         collect (create POSITION
                                        XCOORD _ (PLUS CENTERX (TIMES RADIUS (COS ANGLE)))
                                        YCOORD _ (PLUS CENTERY (TIMES RADIUS (SIN ANGLE]
                                                                (* add first and last points exactly.
                                                                (CONS RADIUSPT (NCONC1 PTLST
                                                                (create POSITION XCOORD _ (FIXR
                                                                (PLUS CENTERX (TIMES RADIUS
                                                                (COS OMEGA)))) YCOORD _ (FIXR
                                                                (PLUS CENTERY (TIMES RADIUS
                                                                (SIN OMEGA))))))))
          (RETURN PTLST])
```

(**SK.SET.ARC.DIRECTION**

```
  [LAMBDA (SKW NEWDIR)                                          (* rrb "31-May-85 17:29")

          (* * reads a value of arc direction and makes it the default)

    (PROG [(LOCALNEWDIR (OR NEWDIR (READARCDIRECTION "Which way should new arcs go?"]
          (RETURN (AND LOCALNEWDIR (replace (SKETCHCONTEXT SKETCHARCDIRECTION) of (WINDOWPROP SKW 'SKETCHCONTEXT)
                                      with (EQ LOCALNEWDIR 'CLOCKWISE])
```

(**SK.SET.ARC.DIRECTION.CW**

```
  [LAMBDA (SKW)                                                 (* sets the default to clockwise)
    (SK.SET.ARC.DIRECTION SKW 'CLOCKWISE])
```

(**SK.SET.ARC.DIRECTION.CCW**

```
  [LAMBDA (SKW)                                                 (* sets the default direction of arcs to counterclockwise)
    (SK.SET.ARC.DIRECTION SKW 'COUNTERCLOCKWISE])
```

(**SK.COMPUTE.SLOPE.OF.LINE**

```
  [LAMBDA (PT1 PT2)                                             (* rrb "31-May-85 12:26")
                                                                (* computes the angle of a line)
    (SK.COMPUTE.SLOPE (DIFFERENCE (fetch (POSITION XCOORD) of PT2)
                             (fetch (POSITION XCOORD) of PT1))
           (DIFFERENCE (fetch (POSITION YCOORD) of PT2)
                  (fetch (POSITION YCOORD) of PT1])
```

(**SK.CREATE.ARC.USING**

```
  [LAMBDA (CENTERPT RADPT ANGLEPT GARCELT)                      (* rrb "15-Dec-86 15:20")
                                                                (* creates an arc global element that is like another one but has
    different positions.)
    (SET.ARC.SCALES (create GLOBALPART
                           COMMONGLOBALPART _ (APPEND (fetch (GLOBALPART COMMONGLOBALPART) of GARCELT))
                           INDIVIDUALGLOBALPART _ (SET.ARC.ARROWHEAD.POINTS (create ARC
                                                                             using (fetch (GLOBALPART
                                                                                           INDIVIDUALGLOBALPART
                                                                                                )
                                                                                      of GARCELT)
                                                                                  ARCCENTERPT _ CENTERPT
                                                                                  ARCRADIUSPT _ RADPT
                                                                                  ARCANGLEPT _
                                                                                  (SK.COMPUTE.ARC.ANGLE.PT
                                                                                   CENTERPT RADPT ANGLEPT)
                                                                                  ARCREGION _ NIL])
```

(**SET.ARC.SCALES**

```
  [LAMBDA (GARCELT)                                             (* rrb "30-May-85 11:33")

          (* updates the scale fields of an arc. Called upon creation and when a point is moved.)
```

```
    (PROG [(RAD (DISTANCEBETWEEN (fetch (ARC ARCCENTERPT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                                   of GARCELT))
                      (fetch (ARC ARCRADIUSPT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GARCELT]
          (replace (GLOBALPART MAXSCALE) of GARCELT with RAD)
          (replace (GLOBALPART MINSCALE) of GARCELT with (QUOTIENT RAD 3000.0))
          (RETURN GARCELT])

)

(DEFINEQ

(SK.INSURE.DIRECTION
  [LAMBDA (DIR)                                                  (* rrb "16-Oct-85 16:11")

          (* decodes a DIRECTION spec which indicates whether an arc goes clockwise or counterclockwise.
          T is CLOCKWISE. NIL is COUNTERCLOCKWISE.)

    (SELECTQ DIR
        ((NIL COUNTERCLOCKWISE)
            NIL)
        ((T CLOCKWISE)
            T)
        (\ILLEGAL.ARC DIR])

)

(RPAQ? SK.NUMBER.OF.POINTS.IN.ARC 8)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SK.NUMBER.OF.POINTS.IN.ARC)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(TYPERECORD ARC (ARCCENTERPT ARCRADIUSPT ARCBRUSH ARCDASHING ARCINITSCALE ARCARROWHEADS ARCANGLEPT ARCDIRECTION
                      ARCREGION ARCARROWHEADPOINTS))

(RECORD LOCALARC ((LOCALARCCENTERPT LOCALARCRADIUSPT LOCALARCANGLEPT)
                  LOCALHOTREGION LOCALARCARROWHEADPTS LOCALARCBRUSH LOCALARCKNOTS LOCALARCDASHING))
)
)

(RPAQ ARC.RADIUS.CURSOR (CURSORCREATE '

                                'NIL 15 7))

(RPAQ ARC.ANGLE.CURSOR (CURSORCREATE '

                               'NIL 7 15))

(RPAQ CW.ARC.ANGLE.CURSOR (CURSORCREATE '

                                   'NIL 7 15))

(RPAQ CW.ARC.RADIUS.CURSOR (CURSORCREATE '

                                   'NIL 15 7))

(READVARS-FROM-STRINGS '(ARCICON)
       "({(READBITMAP)(20 13
       %"@@@@@@@@%"
       %"@AOH@@@@%"
       %"@COL@@@@%"
       %"@G@N@@@@%"
       %"@F@F@@@@%"
       %"@N@G@@@@%"
       %"@L@C@@@@%"
       %"@@@C@@@@%"
       %"@@@G@@@@%"
       %"@@@F@@@@%"
       %"@@@N@@@@%"
       %"@@@L@@@@%"
       %"@@@@@@@@%")})
       ")

;; property getting and setting stuff

(DEFINEQ

(GETSKETCHELEMENTPROP
  [LAMBDA (ELEMENT PROPERTY)                                     (* rrb "26-Jun-86 14:16")
                                                                 (* gets the property from a sketch element.)
            (* knows about and sets the system ones specially. All others go to the elements property list.)

    (SELECTQ PROPERTY
        (TYPE (fetch (GLOBALPART GTYPE) of ELEMENT))
        (SCALE (\SKELT.GET.SCALE ELEMENT))
        (REGION (SK.ELEMENT.GLOBAL.REGION ELEMENT))
        ((POSITION 1STCONTROLPT)
            (\SK.GET.1STCONTROLPT ELEMENT))
        (2NDCONTROLPT (\SK.GET.2NDCONTROLPT ELEMENT))
        (3RDCONTROLPT (\SK.GET.3RDCONTROLPT ELEMENT))
        (DATA (\SKELT.GET.DATA ELEMENT))
        (BRUSH (\SK.GET.BRUSH ELEMENT))
        (FILLING (\SK.GET.FILLING ELEMENT))
        (DASHING (\SK.GET.DASHING ELEMENT))
        (ARROWHEADS (\SK.GET.ARROWHEADS ELEMENT))
        (FONT (\SK.GET.FONT ELEMENT))
        (JUSTIFICATION
            (\SK.GET.JUSTIFICATION ELEMENT))
        (DIRECTION (\SK.GET.DIRECTION ELEMENT))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of ELEMENT)
              PROPERTY])

(\SK.GET.ARC.ANGLEPT
  [LAMBDA (INDVARCELT)                                          (* rrb "20-Jun-86 13:54")

            (* returns the arc point of an individual arc element. Special because T is used to denote arcs of greater than 360 degrees.)

    (COND
       ((POSITIONP (fetch (ARC ARCANGLEPT) of INDVARCELT)))
       (T
            (* for arcs of greater than 360 degrees, the radiuspt is T and is marked as being the same as the radius pt.)

            (fetch (ARC ARCRADIUSPT) of INDVARCELT])

(\GETSKETCHELEMENTPROP1
  [LAMBDA (ELEMENT PROPERTY)

            (* * version of GETSKETCHELEMENTPROP that doesn't look for system properties.)

    (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of ELEMENT)
           PROPERTY])

(\SK.GET.BRUSH
  [LAMBDA (GELT)                                                (* rrb " 7-Dec-85 19:52")
                                                                (* gets the brush field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE BOX)
            (fetch (WIRE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        ((CIRCLE ARC)
            (fetch (CIRCLE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (ELLIPSE (fetch (ELLIPSE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (TEXTBOX (fetch (TEXTBOX TEXTBOXBRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'BRUSH])

(\SK.GET.FILLING
  [LAMBDA (GELT)                                                (* rrb " 7-Dec-85 18:58")
                                                                (* gets the filling field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((CLOSEDWIRE CLOSEDCURVE BOX)
            (fetch (CLOSEDWIRE CLOSEDWIREFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (CIRCLE (fetch (CIRCLE CIRCLEFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (ELLIPSE (fetch (ELLIPSE ELLIPSEFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (TEXTBOX (fetch (TEXTBOX TEXTBOXFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'FILLING])

(\SK.GET.ARROWHEADS
  [LAMBDA (GELT)                                                (* rrb " 7-Dec-85 19:17")
                                                                (* gets the arrowhead field from a global sketch element
                                                                instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (WIRE (fetch (WIRE WIREARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (OPENCURVE (fetch (OPENCURVE CURVEARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (ARC (fetch (ARC ARCARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'ARROWHEADS])

(\SK.GET.FONT
  [LAMBDA (GELT)                                                (* rrb " 7-Dec-85 19:22")
```

```
                                                                      (* gets the font field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((TEXT TEXTBOX)
            (fetch (TEXT FONT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'FONT])

(\SK.GET.JUSTIFICATION
  [LAMBDA (GELT)                                                      ; Edited  8-Jan-87 19:46 by rrb
                                                                      (* gets the justification field from a global sketch element
                                                                      instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((TEXT TEXTBOX)
            (fetch (TEXT TEXTSTYLE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'JUSTIFICATION])

(\SK.GET.DIRECTION
  [LAMBDA (GELT)                                                      (* rrb " 7-Dec-85 19:21")
                                                                      (* gets the direction field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (ARC (fetch (ARC ARCDIRECTION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'DIRECTION])

(\SK.GET.DASHING
  [LAMBDA (GELT)                                                      (* rrb " 7-Dec-85 20:05")
                                                                      (* gets the dashing field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((WIRE CIRCLE ARC)
            (fetch (WIRE OPENWIREDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        ((CLOSEDWIRE OPENCURVE CLOSEDCURVE BOX)
            (fetch (CLOSEDWIRE CLOSEDWIREDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (ELLIPSE (fetch (ELLIPSE DASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (TEXTBOX (fetch (TEXTBOX TEXTBOXDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
              'DASHING])

(PUTSKETCHELEMENTPROP
  [LAMBDA (ELEMENT PROPERTY VALUE SKETCHTOUPDATE)                     (* rrb "26-Jun-86 16:46")
                                                                      (* puts the property from a sketch element.)

            (* knows about and sets the system ones specially. All others go to the elements property list.)
                                                                      (* mostly not implemented yet.)
    (PROG1 (GETSKETCHELEMENTPROP ELEMENT PROPERTY)
        (AND (SELECTQ PROPERTY
                 (TYPE (ERROR "Can't change types"))
                 (SCALE (\SKELT.PUT.SCALE ELEMENT VALUE)
                        T)
                 (REGION (ERROR "Not implemented yet"))
                 ((POSITION 1STCONTROLPT)
                     (\SK.PUT.1STCONTROLPT ELEMENT VALUE))
                 (2NDCONTROLPT (\SK.PUT.2NDCONTROLPT ELEMENT VALUE))
                 (3RDCONTROLPT (\SK.PUT.3RDCONTROLPT ELEMENT VALUE))
                 (DATA (\SKELT.PUT.DATA ELEMENT VALUE SKETCHTOUPDATE))
                 (BRUSH (\SK.PUT.BRUSH ELEMENT VALUE SKETCHTOUPDATE))
                 (FILLING (\SK.PUT.FILLING ELEMENT VALUE))
                 (DASHING (\SK.PUT.DASHING ELEMENT VALUE))
                 (ARROWHEADS (\SK.PUT.ARROWHEADS ELEMENT VALUE))
                 (FONT (\SK.PUT.FONT ELEMENT VALUE))
                 (JUSTIFICATION
                     (\SK.PUT.JUSTIFICATION ELEMENT VALUE))
                 (DIRECTION (\SK.PUT.DIRECTION ELEMENT VALUE))
                 (PROG ((PLIST (fetch (GLOBALPART SKELEMENTPROPLIST) of ELEMENT)))
                       [COND
                          (PLIST (LISTPUT PLIST PROPERTY VALUE))
                          (T (replace (GLOBALPART SKELEMENTPROPLIST) of ELEMENT with (LIST PROPERTY VALUE]
                                                                      (* if it wasn't a system recognized property, return NIL so it won't
                 be redisplayed.)
                       (RETURN NIL)))
             SKETCHTOUPDATE
             (SKETCH.UPDATE SKETCHTOUPDATE ELEMENT)))])

(\SK.PUT.FILLING
  [LAMBDA (GELT NEWVALUE)                                             (* rrb "26-Jun-86 16:44")
                                                                      (* sets the filling field from a global sketch element instance.)
    (OR (SKFILLINGP NEWVALUE)
        (\ILLEGAL.ARG NEWVALUE))
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((CLOSEDWIRE CLOSEDCURVE BOX)
            (replace (CLOSEDWIRE CLOSEDWIREFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
               with NEWVALUE))
        (CIRCLE (replace (CIRCLE CIRCLEFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE)
)
        (ELLIPSE (replace (ELLIPSE ELLIPSEFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                    with NEWVALUE))
        (TEXTBOX (replace (TEXTBOX TEXTBOXFILLING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                    with NEWVALUE))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'FILLING NEWVALUE))
    T])
```

(**ADDSKETCHELEMENTPROP**

```
  [LAMBDA (ELEMENT PROPERTY VALUE SKETCHTOUPDATE)                    (* rrb "11-Dec-85 15:17")

            (* adds a value to the list of values for a property of a sketch element.)

    (PROG ((NOWVALUE (GETSKETCHELEMENTPROP ELEMENT PROPERTY)))
          (RETURN (PUTSKETCHELEMENTPROP ELEMENT PROPERTY [COND
                                                              ((NULL NOWVALUE)
                                                               (LIST VALUE))
                                                              ((NLISTP NOWVALUE)
                                                               (LIST NOWVALUE VALUE))
                                                              (T (APPEND NOWVALUE (CONS VALUE]
                         SKETCHTOUPDATE])
```

(**REMOVESKETCHELEMENTPROP**

```
  [LAMBDA (ELEMENT PROPERTY VALUE SKETCHTOUPDATE)                    (* rrb "11-Dec-85 15:17")

            (* removes a value to the list of values for a property of a sketch element.)

    (PROG ((NOWVALUE (GETSKETCHELEMENTPROP ELEMENT PROPERTY)))
          (RETURN (PUTSKETCHELEMENTPROP ELEMENT PROPERTY (COND
                                                              ((EQ NOWVALUE VALUE)
                                                               NIL)
                                                              ((NLISTP NOWVALUE)
                                                               NOWVALUE)
                                                              (T (REMOVE VALUE NOWVALUE)))
                         SKETCHTOUPDATE])
```

(**\SK.PUT.FONT**

```
  [LAMBDA (GELT NEWVALUE)                                            (* rrb "26-Jun-86 17:04")
                                                                     (* sets the font field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (TEXT (replace (TEXT FONT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with (SK.INSURE.TEXT
                                                                                                  NEWVALUE))
              (SK.UPDATE.TEXT.AFTER.CHANGE GELT))
        (TEXTBOX (replace (TEXTBOX FONT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with (SK.INSURE.TEXT
                                                                                                  NEWVALUE))
                 (SK.UPDATE.TEXTBOX.AFTER.CHANGE GELT))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'FONT NEWVALUE))
    T])
```

(**\SK.PUT.JUSTIFICATION**

```
  [LAMBDA (GELT NEWVALUE)                                            (* rrb "26-Jun-86 16:45")
                                                                     (* sets the justification field from a global sketch element
                                                                     instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (TEXT (replace (TEXT TEXTSTYLE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                 with (SK.INSURE.STYLE NEWVALUE SK.DEFAULT.TEXT.ALIGNMENT))
              (SK.UPDATE.TEXT.AFTER.CHANGE GELT))
        (TEXTBOX (replace (TEXTBOX TEXTSTYLE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                    with (SK.INSURE.STYLE NEWVALUE SK.DEFAULT.TEXT.ALIGNMENT))
                 (SK.UPDATE.TEXTBOX.AFTER.CHANGE GELT))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'JUSTIFICATION NEWVALUE))
    T])
```

(**\SK.PUT.DIRECTION**

```
  [LAMBDA (GELT NEWVALUE)                                            (* rrb "26-Jun-86 16:45")
                                                                     (* puts the direction field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (ARC (replace (ARC ARCDIRECTION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with (
                                                                                            SK.INSURE.DIRECTION
                                                                                                  NEWVALUE))
             (SK.UPDATE.ARC.AFTER.CHANGE GELT))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'DIRECTION NEWVALUE))
    T])
```

(**\SK.PUT.DASHING**

```
  [LAMBDA (GELT NEWVALUE)                                            (* rrb "26-Jun-86 16:44")
```

```
                                                                    (* sets the dashing field of a global sketch element.)
    (OR (NULL NEWVALUE)
        (DASHINGP NEWVALUE)
        (\ILLEGAL.ARG NEWVALUE))
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((WIRE CIRCLE ARC)
             (replace (WIRE OPENWIREDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE))
        ((CLOSEDWIRE OPENCURVE CLOSEDCURVE BOX)
             (replace (CLOSEDWIRE CLOSEDWIREDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                with NEWVALUE))
        (ELLIPSE (replace (ELLIPSE DASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE))
        (TEXTBOX (replace (TEXTBOX TEXTBOXDASHING) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                     with NEWVALUE))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'DASHING NEWVALUE))
    T])

(\SK.PUT.BRUSH
  [LAMBDA (GELT NEWVALUE SKETCHTOUPDATE)                            (* rrb "26-Jun-86 16:44")
                                                                    (* sets the brush field from a global sketch element instance.)
    (COND
       [(NUMBERP NEWVALUE)
        (SETQ NEWVALUE (create BRUSH
                              BRUSHSIZE _ NEWVALUE
                              BRUSHSHAPE _ 'ROUND]
       ((BRUSHP NEWVALUE))
       (T (\ILLEGAL.ARG NEWVALUE)))
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
             (replace (WIRE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE)
             (SK.UPDATE.WIRE.ELT.AFTER.CHANGE GELT))
        (BOX (replace (BOX BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE))
        (CIRCLE (replace (CIRCLE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE)
                (SK.UPDATE.CIRCLE.AFTER.CHANGE GELT))
        (ARC (replace (ARC ARCBRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE)
             (SK.UPDATE.ARC.AFTER.CHANGE GELT))
        (ELLIPSE (replace (ELLIPSE BRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE)
                (SK.UPDATE.ELLIPSE.AFTER.CHANGE GELT))
        (TEXTBOX (AND SKETCHTOUPDATE (SKETCH.CLEANUP SKETCHTOUPDATE))
                (replace (TEXTBOX TEXTBOXBRUSH) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE
                       )
                (SK.UPDATE.TEXTBOX.AFTER.CHANGE GELT))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'BRUSH NEWVALUE))
    T])

(\SK.PUT.ARROWHEADS
  [LAMBDA (GELT NEWVALUE)                                           ; Edited 21-Aug-2021 20:01 by larry
                                                                    (* sets the arrowhead field from a global sketch element
                                                                    instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        (WIRE (replace (WIRE WIREARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                 with (SK.INSURE.ARROWHEADS NEWVALUE))
              (SET.WIRE.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (OPENCURVE (replace (OPENCURVE CURVEARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                      with (SK.INSURE.ARROWHEADS NEWVALUE))
                   (SET.OPENCURVE.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (ARC (replace (ARC ARCARROWHEADS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with (
                                                                                              SK.INSURE.ARROWHEADS
                                                                                                     NEWVALUE))
             (SET.ARC.ARROWHEAD.POINTS (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               'ARROWHEADS NEWVALUE))
    T])

(SK.COPY.ELEMENT.PROPERTY.LIST
  [LAMBDA (ELEMENT OLDELEMENT)                                      (* rrb " 6-May-86 11:01")

            (* copies the property list of an element from OLDELEMENT if it is given, from itself otherwise.)

    (replace (GLOBALPART SKELEMENTPROPLIST) of ELEMENT with (APPEND (fetch (GLOBALPART SKELEMENTPROPLIST)
                                                                           of (OR OLDELEMENT ELEMENT])

(SKETCH.UPDATE
  [LAMBDA (SKETCH ELEMENTS)                                         (* rrb " 6-Dec-85 14:40")
                                                                    (* updates all or part of a sketch.)
    (PROG ((SKSTRUC (INSURE.SKETCH SKETCH))
           ALLVIEWERS)
          (SETQ ALLVIEWERS (ALL.SKETCH.VIEWERS SKSTRUC))
          (COND
             ((NULL ELEMENTS)
              (for SKW in ALLVIEWERS do (SK.UPDATE.AFTER.SCALE.CHANGE SKW)))
             ((GLOBALELEMENTP ELEMENTS)
              (SKETCH.UPDATE1 ELEMENTS ALLVIEWERS))
             ((LISTP ELEMENTS)
              (for ELT in ELEMENTS do (SKETCH.UPDATE1 ELT ALLVIEWERS)))
             (T (\ILLEGAL.ARG ELEMENTS])

(SKETCH.UPDATE1
  [LAMBDA (GELT VIEWERS)                                     (* rrb "26-Sep-86 14:49")
                                                             (* updates the element GELT in the sketch viewers VIEWERS.)
    (bind SELECTION for SKW in VIEWERS do (COND
                                            ((AND [SCREENELEMENTP (SETQ SELECTION (fetch (TEXTELTSELECTION
                                                                                           SKTEXTELT)
                                                                                 of (WINDOWPROP SKW
                                                                                           'SELECTION]
                                                  (EQ GELT (fetch (SCREENELT GLOBALPART) of SELECTION)))
                                                             (* if the element being updated is the current text selection, clear
                                             the selection.)
                                             (SKED.CLEAR.SELECTION SKW)))
                                          (SK.UPDATE.ELEMENT1 GELT GELT SKW T])

(\SKELT.GET.SCALE
  [LAMBDA (GELT)                                             (* rrb "29-Oct-85 13:44")
                                                             (* gets the scale field from a global sketch element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
         ((TEXT TEXTBOX SKIMAGEOBJ BITMAPELT)
              (fetch (TEXT INITIALSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         ((WIRE OPENCURVE CIRCLE ARC)
              (fetch (WIRE OPENWIREINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         ((CLOSEDWIRE CLOSEDCURVE BOX)
              (fetch (CLOSEDWIRE CLOSEDWIREINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         (ELLIPSE (fetch (ELLIPSE ELLIPSEINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         NIL])

(\SKELT.PUT.SCALE
  [LAMBDA (GELT NEWVALUE)                                    (* rrb "16-Oct-85 21:24")
                                                             (* sets the scale field of a global sketch element instance.)
    (COND
       ((NUMBERP NEWVALUE)
        (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
             ((TEXT TEXTBOX SKIMAGEOBJ BITMAPELT)
                  (replace (TEXT INITIALSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT) with NEWVALUE))
             ((WIRE OPENCURVE CIRCLE ARC)
                  (replace (WIRE OPENWIREINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                     with NEWVALUE))
             ((CLOSEDWIRE CLOSEDCURVE BOX)
                  (replace (CLOSEDWIRE CLOSEDWIREINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                     with NEWVALUE))
             (ELLIPSE (replace (ELLIPSE ELLIPSEINITSCALE) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)
                         with NEWVALUE))
             NIL))
       (T (\ILLEGAL.ARG NEWVALUE])

(\SKELT.PUT.DATA
  [LAMBDA (GELT NEWVALUE SKETCHTOUPDATE)                     (* rrb "26-Jun-86 16:40")
                                                             (* changes the data of a sketch element.)

             (* this is harder than it seems because all of the dependent fields must be updated also -
             lots of grubby details duplicated.)

    (PROG ((INDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
              (GROUP (COND
                        ([OR (NLISTP NEWVALUE)
                             (NOT (EVERY NEWVALUE (FUNCTION GLOBALELEMENTP]
                         (\ILLEGAL.ARG NEWVALUE)))
                     (replace (GROUP LISTOFGLOBALELTS) of INDVELT with NEWVALUE)
                     (SK.UPDATE.GROUP.AFTER.CHANGE GELT))
              ((TEXT TEXTBOX)                                (* before changing the text element, make sure any interactive
                                                             editing is closed off.)
                   (AND SKETCHTOUPDATE (SKETCH.CLEANUP SKETCHTOUPDATE))
                   (SK.REPLACE.TEXT.IN.ELEMENT GELT (SK.INSURE.TEXT NEWVALUE)))
              (BITMAPELT (replace (BITMAPELT SKBITMAP) of INDVELT with NEWVALUE))
              (SKIMAGEOBJ (replace (SKIMAGEOBJ SKIMAGEOBJ) of INDVELT with NEWVALUE)
                          (SK.UPDATE.IMAGEOBJECT.AFTER.CHANGE GELT))
              ((WIRE OPENCURVE CLOSEDWIRE CLOSEDCURVE)
                   (replace (WIRE LATLONKNOTS) of INDVELT with NEWVALUE)
                   (SK.UPDATE.WIRE.ELT.AFTER.CHANGE GELT))
              (RETURN NIL))
          (RETURN T])

(SK.REPLACE.TEXT.IN.ELEMENT
  [LAMBDA (GTEXTELT NEWSTRS)                                 (* rrb "15-Dec-85 18:00")
                                                             (* changes the characters in a text or textbox element.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GTEXTELT)
        (TEXTBOX (replace (TEXTBOX LISTOFCHARACTERS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT)
                    with (OR NEWSTRS (CONS "")))
                  (SK.UPDATE.TEXTBOX.AFTER.CHANGE GTEXTELT))
        (TEXT (replace (TEXT LISTOFCHARACTERS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GTEXTELT)
                  with NEWSTRS)
              (SK.UPDATE.TEXT.AFTER.CHANGE GTEXTELT))
        (\ILLEGAL.ARG GTEXTELT))
    GTEXTELT])
```

(\SKELT.GET.DATA

```
  [LAMBDA (GELT)                                              (* rrb " 6-Dec-85 14:52")
                                                              (* changes the data of a sketch element.)
    (PROG ((INDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
          (RETURN (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
                      (GROUP (fetch (GROUP LISTOFGLOBALELTS) of INDVELT))
                      ((TEXT TEXTBOX)
                          (fetch (TEXT LISTOFCHARACTERS) of INDVELT))
                      (BITMAPELT (fetch (BITMAPELT SKBITMAP) of INDVELT))
                      (SKIMAGEOBJ (fetch (SKIMAGEOBJ SKIMAGEOBJ) of INDVELT))
                      ((WIRE OPENCURVE CLOSEDWIRE CLOSEDCURVE)
                          (fetch (WIRE LATLONKNOTS) of INDVELT))
                      (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
                             'DATA])
```

(\SK.GET.1STCONTROLPT

```
  [LAMBDA (GELT PROPERTY)                                     (* rrb " 9-Dec-85 11:33")
                                                              (* gets the first control point field from a global sketch element
                                                              instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((TEXT CIRCLE ARC ELLIPSE)
            (fetch (TEXT LOCATIONLATLON) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        ((TEXTBOX BOX)
            (LOWERLEFTCORNER (fetch (BOX GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))))
        ((BITMAPELT SKIMAGEOBJ)
            (LOWERLEFTCORNER (fetch (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of (fetch (GLOBALPART
                                                                                     INDIVIDUALGLOBALPART)
                                                                              of GELT))))
        ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
            (CAR (fetch (WIRE LATLONKNOTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))))
        (GROUP (fetch (GROUP GROUPCONTROLPOINT) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               PROPERTY])
```

(\SK.PUT.1STCONTROLPT

```
  [LAMBDA (GELT NEWPOSITION)                                  (* rrb "26-Jun-86 16:22")
                                                              (* changes the first control point field from a global sketch
                                                              element instance.)
    (OR (POSITIONP NEWPOSITION)
        (\ILLEGAL.ARG NEWPOSITION))
    (PROG ((INDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           X)
          (SELECTQ (CAR INDVELT)
              (TEXT (replace (TEXT LOCATIONLATLON) of INDVELT with NEWPOSITION)
                    (SK.UPDATE.TEXT.AFTER.CHANGE GELT))
              (CIRCLE (replace (CIRCLE CENTERLATLON) of INDVELT with NEWPOSITION)
                      (SK.UPDATE.CIRCLE.AFTER.CHANGE GELT))
              (ARC (replace (ARC ARCCENTERPT) of INDVELT with NEWPOSITION)
                   (SK.UPDATE.ARC.AFTER.CHANGE GELT))
              (ELLIPSE (replace (ELLIPSE ELLIPSECENTERLATLON) of INDVELT with NEWPOSITION)
                       (SK.UPDATE.ELLIPSE.AFTER.CHANGE GELT))
              (TEXTBOX (replace (TEXTBOX TEXTBOXREGION) of INDVELT with (create REGION using (fetch (BOX
                                                                                                        GLOBALREGION
                                                                                                        )
                                                                                                  of INDVELT)
                                                                                               LEFT _
                                                                                               (fetch (POSITION XCOORD
                                                                                                             )
                                                                                                  of NEWPOSITION)
                                                                                               BOTTOM _
                                                                                               (fetch (POSITION YCOORD
                                                                                                             )
                                                                                                  of NEWPOSITION)))
                       (SK.UPDATE.TEXTBOX.AFTER.CHANGE GELT))
              (BOX (replace (BOX GLOBALREGION) of INDVELT with (create REGION using (fetch (BOX GLOBALREGION)
                                                                                           of INDVELT)
                                                                                        LEFT _ (fetch (POSITION XCOORD)
                                                                                                  of NEWPOSITION)
                                                                                        BOTTOM _ (fetch (POSITION YCOORD
                                                                                                             )
                                                                                                  of NEWPOSITION)))
                   (SK.UPDATE.BOX.AFTER.CHANGE GELT))
              (SKIMAGEOBJ (replace (SKIMAGEOBJ SKIMOBJ.GLOBALREGION) of INDVELT with (create REGION
                                                                                             using (fetch (SKIMAGEOBJ
                                                                                                                                    SKIMOBJ.GLOBALREGION
                                                                                                                                            )
                                                                                                                                       of INDVELT)
                                                                                                                                   LEFT _
                                                                                                                                   (fetch (POSITION
                                                                                                                                            XCOORD)
                                                                                                                                       of NEWPOSITION)
                                                                                                                                   BOTTOM _
                                                                                                                                   (fetch (POSITION
                                                                                                                                            YCOORD)
                                                                                                                                       of NEWPOSITION)))
                                         (SK.UPDATE.IMAGEOBJECT.AFTER.CHANGE GELT))
                    (BITMAPELT (replace (BITMAPELT SKBITMAPREGION) of INDVELT with (create REGION
                                                                                       using (fetch (BITMAPELT
                                                                                                      SKBITMAPREGION)
                                                                                                of INDVELT)
                                                                                          LEFT _ (fetch (POSITION
                                                                                                           XCOORD)
                                                                                                    of NEWPOSITION)
                                                                                          BOTTOM _ (fetch (POSITION
                                                                                                            YCOORD)
                                                                                                     of NEWPOSITION)))
)
                    ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
                        [COND
                           ((SETQ X (fetch (WIRE LATLONKNOTS) of INDVELT))
                                                                    (* there is at least one knot)
                            (RPLACA X NEWPOSITION))
                           (T (replace (WIRE LATLONKNOTS) of INDVELT with (CONS NEWPOSITION]
                        (SK.UPDATE.WIRE.ELT.AFTER.CHANGE GELT))
                    (GROUP                                          (* change the position of the control point without changing the
                                                                    group.)
                           (replace (GROUP GROUPCONTROLPOINT) of INDVELT with NEWPOSITION))
                    (RETURN NIL))
                (RETURN T])
```

(\SK.GET.2NDCONTROLPT

```
  [LAMBDA (GELT)                                                    (* rrb " 9-Dec-85 11:32")
                                                                    (* gets the second control point field from a global sketch
                                                                    element instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
        ((CIRCLE ARC ELLIPSE)
            (fetch (CIRCLE RADIUSLATLON) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
        ((TEXTBOX BOX)
            (UPPERRIGHTCORNER (fetch (BOX GLOBALREGION) of (fetch (GLOBALPART INDIVIDUALGLOBALPART)
                                                              of GELT))))
        ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
            (CADR (fetch (WIRE LATLONKNOTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))))
        (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
               '2NDCONTROLPT])
```

(\SK.PUT.2NDCONTROLPT

```
  [LAMBDA (GELT NEWPOSITION)                                        (* rrb "26-Jun-86 16:38")
                                                                    (* changes the second control point field from a global sketch
                                                                    element instance.)
    (OR (POSITIONP NEWPOSITION)
        (\ILLEGAL.ARG NEWPOSITION))
    (PROG ((INDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           X)
          (SELECTQ (CAR INDVELT)
              (CIRCLE (replace (CIRCLE RADIUSLATLON) of INDVELT with NEWPOSITION)
                      (SK.UPDATE.CIRCLE.AFTER.CHANGE GELT))
              (ARC (replace (ARC ARCRADIUSPT) of INDVELT with NEWPOSITION)
                   (SK.UPDATE.ARC.AFTER.CHANGE GELT))
              (ELLIPSE (replace (ELLIPSE SEMIMINORLATLON) of INDVELT with NEWPOSITION)
                       (SK.UPDATE.ELLIPSE.AFTER.CHANGE GELT))
              (BOX (SETQ X (fetch (BOX GLOBALREGION) of INDVELT))
                   [replace (BOX GLOBALREGION) of INDVELT with (create REGION using X WIDTH _
                                                                    (DIFFERENCE (fetch (POSITION
                                                                                         XCOORD)
                                                                                   of NEWPOSITION)
                                                                           (fetch (REGION LEFT)
                                                                              of X))
                                                                    HEIGHT _
                                                                    (DIFFERENCE (fetch (POSITION
                                                                                         YCOORD)
                                                                                   of NEWPOSITION)
                                                                           (fetch (REGION BOTTOM)
                                                                              of X]
                   (SK.UPDATE.BOX.AFTER.CHANGE GELT))
              (TEXTBOX (SETQ X (fetch (TEXTBOX TEXTBOXREGION) of INDVELT))
                       [replace (TEXTBOX TEXTBOXREGION) of INDVELT with (create REGION
                                                                           using X WIDTH _
                                                                              (DIFFERENCE (fetch (POSITION
                                                                                                                             XCOORD)
                                                                                                                          of NEWPOSITION)
                                                                                                               (fetch (REGION LEFT)
                                                                                                                  of X))
                                                                                                     HEIGHT _
                                                                                                     (DIFFERENCE (fetch (POSITION
                                                                                                                             YCOORD)
                                                                                                                          of NEWPOSITION)
                                                                                                               (fetch (REGION BOTTOM)
                                                                                                                  of X]
                        (SK.UPDATE.TEXTBOX.AFTER.CHANGE GELT))
              ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
                   (COND
                      ((NULL (SETQ X (fetch (WIRE LATLONKNOTS) of INDVELT)))
                                                                   (* doesn't have a first knot, give it one at 0 . 0)
                       (replace (WIRE LATLONKNOTS) of INDVELT with (LIST '(0 . 0)
                                                                         NEWPOSITION)))
                      ((NULL (CDR X))
                       (replace (WIRE LATLONKNOTS) of INDVELT with (LIST (CAR X)
                                                                         NEWPOSITION)))
                      (T                                           (* there is at least one knot)
                         (RPLACA (CDR X)
                                NEWPOSITION)))
                   (SK.UPDATE.WIRE.ELT.AFTER.CHANGE GELT))
              (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
                     '2NDCONTROLPT NEWPOSITION))
          (RETURN T])
```

(\SK.GET.3RDCONTROLPT

```
  [LAMBDA (GELT)                                                   (* rrb "20-Jun-86 13:55")
                                                                   (* gets the third control point field from a global sketch element
                                                                   instance.)
    (SELECTQ (fetch (GLOBALPART GTYPE) of GELT)
         (ELLIPSE (fetch (ELLIPSE SEMIMAJORLATLON) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         (ARC (\SK.GET.ARC.ANGLEPT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT)))
         ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
              (CADDR (fetch (WIRE LATLONKNOTS) of (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))))
         (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
                '3RDCONTROLPT])
```

(\SK.PUT.3RDCONTROLPT

```
  [LAMBDA (GELT NEWPOSITION)                                       (* rrb "10-Jul-86 11:15")
                                                                   (* changes the third control point field from a global sketch
                                                                   element instance.)
    (PROG ((INDVELT (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT))
           X)
          (RETURN (COND
                     ((EQ (CAR INDVELT)
                          'ARC)                                    (* handle ARC specially because it will convert the number of
                                                                   degrees to a point.)
                      (COND
                         ((POSITIONP NEWPOSITION)
                          (replace (ARC ARCANGLEPT) of INDVELT with (SK.COMPUTE.ARC.ANGLE.PT (fetch (ARC
                                                                                                    ARCCENTERPT
                                                                                                        )
                                                                                                 of INDVELT)
                                                                           (fetch (ARC ARCRADIUSPT) of INDVELT)
                                                                           NEWPOSITION)))
                         ((NUMBERP NEWPOSITION)
                          (replace (ARC ARCANGLEPT) of INDVELT with (SK.COMPUTE.ARC.ANGLE.PT.FROM.ANGLE
                                                                     (fetch (ARC ARCCENTERPT) of INDVELT)
                                                                     (fetch (ARC ARCRADIUSPT) of INDVELT)
                                                                     NEWPOSITION)))
                         (T (\ILLEGAL.ARG NEWPOSITION)))
                      (SK.UPDATE.ARC.AFTER.CHANGE GELT)
                      T)
                     (T (OR (POSITIONP NEWPOSITION)
                            (\ILLEGAL.ARG NEWPOSITION))
                        (SELECTQ (CAR INDVELT)
                             (ELLIPSE (replace (ELLIPSE SEMIMAJORLATLON) of INDVELT with NEWPOSITION)
                                      (SK.UPDATE.ELLIPSE.AFTER.CHANGE GELT))
                             ((WIRE CLOSEDWIRE OPENCURVE CLOSEDCURVE)
                                  (COND
                                     ((NULL (SETQ X (fetch (WIRE LATLONKNOTS) of INDVELT)))
                                                                   (* doesn't have a first knot, give it one at 0 . 0)
                                      (replace (WIRE LATLONKNOTS) of INDVELT with (LIST '(0 . 0)
                                                                                        '(0 . 0)
                                                                                        NEWPOSITION)))
                                     ((NULL (CDR X))
                                      (replace (WIRE LATLONKNOTS) of INDVELT with (LIST (CAR X)
                                                                                        '(0 . 0)
                                                                                        NEWPOSITION)))
                                     ((NULL (CDDR X))
                                      (replace (WIRE LATLONKNOTS) of INDVELT with (LIST (CAR X)
                                                                                        (CADR X)
```

```
                                                                                                           NEWPOSITION)))
                                              (T                                 (* there is at least one knot)
                                                 (RPLACA (CDDR X)
                                                        NEWPOSITION)))
                                          (SK.UPDATE.WIRE.ELT.AFTER.CHANGE GELT))
                                     (LISTPUT (fetch (GLOBALPART SKELEMENTPROPLIST) of GELT)
                                            '3RDCONTROLPT NEWPOSITION))
                                T])

)

(DEFINEQ

(LOWERLEFTCORNER
  [LAMBDA (REGION)                                                       (* returns a position which is the lower left corner of a region.)
    (CREATEPOSITION (FETCH (REGION LEFT) OF REGION)
           (FETCH (REGION BOTTOM) OF REGION])

(UPPERRIGHTCORNER
  [LAMBDA (REGION)                                                       (* rrb "16-Oct-85 21:10")
                                                                         (* returns a position which is the lower left corner of a region.)
    (CREATEPOSITION (fetch (REGION RIGHT) of REGION)
           (fetch (REGION TOP) of REGION])

)
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX