```
File created:    5-Dec-2023 00:08:46  {WMEDLEY}<library>sketch>SKETCH-OPS.;1

      edit by:   rmk

   changes to:   (RECORDS AFFINETRANSFORMATION SKHISTEVENT SKEVENTTYPE SKETCHVIEW)

previous date:    3-May-2023 21:06:28  {WMEDLEY}<library>sketch>SKETCHOPS.;2

   Read Table:   INTERLISP

      Package:   INTERLISP

       Format:    XCCS

(RPAQQ SKETCH-OPSCOMS
       [                                                      ; functions that used to be on SKETCH
        (COMS ;; miscellaneous utility functions

              (FNS SK.FONTNAMELIST SCALE.REGION.OUT SK.SCALE.POSITION.INTO.VIEWER
                   SK.SCALE.POSITION.INTO.VIEWER.EXACT SK.MAKE.POSITION.INTEGER SCALE.POSITION.INTO.SKETCHW
                   UNSCALE UNSCALE.REGION)

              ;; misc IO functions

              (FNS STATUSPRINT CLEARPROMPTWINDOW CLOSEPROMPTWINDOW MYGETPROMPTWINDOW PROMPT.GETINPUT))
        (COMS ;; fns for dealing with display priorities

              (FNS SK.SEND.TO.BOTTOM SK.BRING.TO.TOP SK.SWITCH.PRIORITIES SK.SEL.AND.CHANGE.PRIORITY
                   SK.SEL.AND.SWITCH.PRIORITIES SK.SORT.ELTS.BY.PRIORITY SK.SORT.GELTS.BY.PRIORITY
                   SORT.CHANGESPECS.BY.NEW.PRIORITY SORT.CHANGESPECS.BY.OLD.PRIORITY SK.SEND.ELEMENTS.TO.BOTTOM
                   SK.BRING.ELEMENTS.TO.TOP SK.COPY.GLOBAL.ELEMENT.AND.PROPERTY.LIST)
              (FNS SK.ELEMENT.PRIORITY SK.SET.ELEMENT.PRIORITY SK.POP.NEXT.PRIORITY SK.PRIORITY.CELL
                   SK.HIGH.PRIORITY SK.LOW.PRIORITY))
        (COMS ;; functions for dealing with display elements.

              (FNS DRAW.LOCAL.SKETCH SET.PRIORITYIMPORTANT SK.FIGUREIMAGE)
              (COMS ;; functions for hardcopying

                    (FNS SKETCHW.HARDCOPYFN SK.LIST.IMAGE SK.HARDCOPYIMAGEW)
                    (FNS SK.DO.HARDCOPYIMAGEW.TOFILE SK.HARDCOPYIMAGEW.TOFILE SK.HARDCOPYIMAGEW.TOPRINTER
                         SK.LIST.IMAGE.ON.FILE)
                    (FNS \SK.LIST.PAGE.IMAGE SK.GetImageFile SK.PRINTER.FILE.CANDIDATE.NAME SK.SET.HARDCOPY.MODE
                         SK.UNSET.HARDCOPY.MODE SK.UPDATE.AFTER.HARDCOPY DEFAULTPRINTINGIMAGETYPE
                         SK.SWITCH.REGION.X.AND.Y)
                    (CONSTANTS MICASPERPT IMICASPERPT PTSPERMICA)))
        (COMS ;; fns to implement transformations on the elements

              (FNS SK.SEL.AND.TRANSFORM SK.TRANSFORM.ELEMENTS SK.TRANSFORM.ITEM SK.TRANSFORM.ELEMENT
                   SK.TRANSFORM.POINT SK.TRANSFORM.POINT.LIST SK.TRANSFORM.REGION SK.PUT.ELTS.ON.GRID
                   SK.TRANSFORM.GLOBAL.ELEMENTS GLOBALELEMENTP SKETCH.LIST.OF.ELEMENTSP
                   SK.TRANSFORM.SCALE.FACTOR SK.TRANSFORM.BRUSH SK.TRANSFORM.ARROWHEADS SCALE.BRUSH)
              (FNS TWO.PT.TRANSFORMATION.INPUTFN SK.TWO.PT.TRANSFORM.ELTS SK.SEL.AND.TWO.PT.TRANSFORM
                   SK.APPLY.AFFINE.TRANSFORM SK.COMPUTE.TWO.PT.TRANSFORMATION SK.COMPUTE.SLOPE
                   SK.THREE.PT.TRANSFORM.ELTS SK.COMPUTE.THREE.PT.TRANSFORMATION SK.SEL.AND.THREE.PT.TRANSFORM
                   THREE.PT.TRANSFORMATION.INPUTFN)
              (FNS SK.COPY.AND.TWO.PT.TRANSFORM.ELTS SK.SEL.COPY.AND.TWO.PT.TRANSFORM
                   SK.COPY.AND.THREE.PT.TRANSFORM.ELTS SK.SEL.COPY.AND.THREE.PT.TRANSFORM
                   SK.COPY.AND.TRANSFORM.ELEMENTS SK.COPY.AND.TRANSFORM.ITEM)
              (DECLARE%: DONTCOPY (RECORDS AFFINETRANSFORMATION))
              (UGLYVARS FIRSTPTMARK SECONDPTMARK THIRDPTMARK NEWFIRSTPTMARK NEWSECONDPTMARK)
              (GLOBALVARS FIRSTPTMARK SECONDPTMARK THIRDPTMARK NEWFIRSTPTMARK NEWSECONDPTMARK)
              (FILES MATMULT))
        (COMS                                                 ; functions for marking
              (FNS SK.SHOWMARKS MARKPOINT SK.MARKHOTSPOTS SK.MARK.SELECTION)
              (UGLYVARS POINTMARK SPOTMARKER)
              (GLOBALVARS POINTMARK SPOTMARKER)
              (CURSORS POINTREADINGCURSOR)
                                                              ; hit detection functions.
              (FNS SK.SELECT.ITEM IN.SKETCH.ELT? SK.MARK.HOTSPOT SK.MARK.POSITION SK.SELECT.ELT SK.DESELECT.ELT)
              (CONSTANTS (SK.POINT.WIDTH 4))
                                                              ; fns to support caching of hotspots.
              (FNS SK.HOTSPOT.CACHE SK.HOTSPOT.CACHE.FOR.OPERATION SK.BUILD.CACHE SK.ELEMENT.PROTECTED?
                   SK.HAS.SOME.HOTSPOTS SK.SET.HOTSPOT.CACHE SK.CREATE.HOTSPOT.CACHE SK.ELTS.FROM.HOTSPOT
                   SK.ADD.HOTSPOTS.TO.CACHE SK.ADD.HOTSPOTS.TO.CACHE1 SK.ADD.HOTSPOT.TO.CACHE
                   SK.REMOVE.HOTSPOTS.FROM.CACHE SK.REMOVE.HOTSPOTS.FROM.CACHE1 SK.REMOVE.HOTSPOT.FROM.CACHE
                   SK.REMOVE.VALUE.FROM.CACHE.BUCKET SK.FIND.CACHE.BUCKET SK.ADD.VALUE.TO.CACHE.BUCKET))
        (COMS                                                 ; grid stuff
              (FNS SK.SET.GRID SK.DISPLAY.GRID SK.DISPLAY.GRID.POINTS SK.REMOVE.GRID.POINTS SK.TAKE.DOWN.GRID
                   SK.SHOW.GRID SK.GRIDFACTOR SK.TURN.GRID.ON SK.TURN.GRID.OFF SK.MAKE.GRID.LARGER
                   SK.MAKE.GRID.SMALLER SK.CHANGE.GRID GRID.FACTOR1 LEASTPOWEROF2GT GREATESTPOWEROF2LT
                   SK.DEFAULT.GRIDFACTOR SK.PUT.ON.GRID MAP.WINDOW.ONTO.GRID MAP.SCREEN.ONTO.GRID
                   MAP.GLOBAL.PT.ONTO.GRID MAP.GLOBAL.REGION.ONTO.GRID MAP.WINDOW.POINT.ONTO.GLOBAL.GRID
                   MAP.WINDOW.ONTO.GLOBAL.GRID SK.UPDATE.GRIDFACTOR SK.MAP.FROM.WINDOW.TO.GLOBAL.GRID
                   SK.MAP.INPUT.PT.TO.GLOBAL SK.MAP.FROM.WINDOW.TO.NEAREST.GRID)
              (INITVARS (DEFAULTGRIDSIZE 8)
                     (DEFAULTMINGRIDSIZE 4)
                     (DEFAULTMAXGRIDSIZE 32)))
        (COMS                                                 ; history and undo stuff
              (FNS SK.ADD.HISTEVENT SK.SEL.AND.UNDO SK.UNDO.LAST SK.UNDO.NAME SKEVENTTYPEFNS
                  SK.TYPE.OF.FIRST.ARG)
              (FNS SK.DELETE.UNDO SK.ADD.UNDO)
              (FNS SK.CHANGE.UNDO SK.ELT.IN.SKETCH? SK.CHANGE.REDO SK.MOVE.UNDO SK.MOVE.REDO)
              (FNS SK.UNDO.UNDO SK.UNDO.MENULABEL SK.LABEL.FROM.TYPE)
              (DECLARE%: DONTCOPY (RECORDS SKHISTEVENT SKEVENTTYPE))
              (INITVARS (SKETCH.#.UNDO.ITEMS 30))
              (GLOBALVARS SKETCH.#.UNDO.ITEMS)
              (IFPROP EVENTFNS ADD DELETE CHANGE UNDO MOVE COPY ZOOM ANNOTATE LINK))
        (COMS                                                   ; functions for displaying the global coordinate space values.
              (FNS SHOW.GLOBAL.COORDS LOCATOR.CLOSEFN SKETCHW.FROM.LOCATOR SKETCHW.UPDATE.LOCATORS
                   LOCATOR.UPDATE UPDATE.GLOBAL.LOCATOR UPDATE.GLOBALCOORD.LOCATOR ADD.GLOBAL.DISPLAY
                   ADD.GLOBAL.GRIDDED.DISPLAY CREATE.GLOBAL.DISPLAYER UPDATE.GLOBAL.GRIDDED.COORD.LOCATOR)
              (VARS (SKETCHW.LASTCURSORPTX 0)
                    (SKETCHW.LASTCURSORY 0))
              (GLOBALVARS SKETCHW.LASTCURSORPTX SKETCHW.LASTCURSORPTY))
        (COMS                                                   ; fns for reading colors
              (FNS DISPLAYREADCOLORHLSLEVELS DISPLAYREADCOLORLEVEL DRAWREADCOLORBOX READ.CHANGE.COLOR READCOLOR1
                   READCOLORCOMMANDMENUSELECTEDFN READCOLOR2)
              (FNS CREATE.CNS.MENU)
              (VARS COLORMENUHEIGHT COLORMENUWIDTH)
              (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                                      LLCOLOR)))
        (COMS ;; functions that used to be taken from GRAPHZOOM.  Renamed and defined here so GRAPHZOOM isn't loaded.

              (FNS SK.ABSWXOFFSET SK.ABSWYOFFSET SK.UNSCALE.POSITION.FROM.VIEWER SK.SCALE.REGION))
        (COMS                                                   ; functions for zooming
              (FNS VIEWER.SCALE SKETCH.ZOOM SAME.ASPECT.RATIO SKETCH.DO.ZOOM SKETCH.NEW.VIEW ZOOM.UPDATE.ELT
                   SK.UPDATE.AFTER.SCALE.CHANGE SKETCH.AUTOZOOM SKETCH.GLOBAL.REGION.ZOOM)
              (INITVARS (AUTOZOOM.FACTOR 0.8)
                     (AUTOZOOM.REPAINT.TIME 3000))
              (CURSORS AUTOZOOMCURSOR ZOOMINCURSOR ZOOMOUTCURSOR)
              (GLOBALVARS AUTOZOOM.FACTOR AUTOZOOM.REPAINT.TIME ZOOMINCURSOR ZOOMOUTCURSOR))
        (COMS                                                   ; fns for changing the view
              (FNS SKETCH.HOME SK.FRAME.IT SK.FRAME.WINDOW.TO.SKETCH SK.MOVE.TO.VIEW SK.NAME.CURRENT.VIEW
                   SKETCH.ADD.VIEW SK.RESTORE.VIEW SK.FORGET.VIEW)
              (DECLARE%: DONTCOPY (RECORDS SKETCHVIEW)))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                                SKETCH SKETCHELEMENTS SKETCHOBJ SKETCHEDIT
                                                                INTERPRESS))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA STATUSPRINT])
```

;; functions that used to be on SKETCH

;; miscellaneous utility functions

```
(DEFINEQ
```

(**SK.FONTNAMELIST**

```
  [LAMBDA (FONTDESC)                                            (* rrb " 2-NOV-83 21:00")
    (LIST (FONTPROP FONTDESC 'FAMILY)
          (FONTPROP FONTDESC 'SIZE)
          (FONTPROP FONTDESC 'FACE])
```

(**SCALE.REGION.OUT**

```
  [LAMBDA (REGION SCALE)                                        (* rrb "30-Dec-85 17:24")

          (* scales a region into a windows coordinate space making sure that all of the region is covered e.g.
          rounds out.)

    (PROG [(ROUNDINGFACTOR (DIFFERENCE SCALE (QUOTIENT SCALE 20000.0]
          (RETURN (CREATEREGION (FIX (QUOTIENT (fetch (REGION LEFT) of REGION)
                                               SCALE))
                         (FIX (QUOTIENT (fetch (REGION BOTTOM) of REGION)
                                     SCALE))
                         (FIX (QUOTIENT (PLUS (fetch (REGION WIDTH) of REGION)
                                              ROUNDINGFACTOR)
                                     SCALE))
                         (FIX (QUOTIENT (PLUS (fetch (REGION HEIGHT) of REGION)
                                              ROUNDINGFACTOR)
                                     SCALE])
```

(**SK.SCALE.POSITION.INTO.VIEWER**

```
  [LAMBDA (POS SCALE)                                           (* rrb "11-Sep-86 14:35")
                                                                (* scales a position into window coordinates from global
                                                                coordinates.)
    (COND
       ((EQP SCALE 1)                                           (* avoid QUOTIENT)
        (SK.MAKE.POSITION.INTEGER POS))
       (T (create POSITION
                 XCOORD _ (FIXR (QUOTIENT (fetch (POSITION XCOORD) of POS)
                                       SCALE))
                 YCOORD _ (FIXR (QUOTIENT (fetch (POSITION YCOORD) of POS)
                                       SCALE])
```

```
(SK.SCALE.POSITION.INTO.VIEWER.EXACT
  [LAMBDA (POS SCALE)                                        (* rrb "30-Sep-86 15:28")

            (* * scales a position into global coordinates from window coordinates.
            Doesn't convert to the closest integer like SK.SCALE.POSITION.INTO.VIEWER)

    (create POSITION
           XCOORD _ (QUOTIENT (fetch (POSITION XCOORD) of POS)
                           SCALE)
           YCOORD _ (QUOTIENT (fetch (POSITION YCOORD) of POS)
                           SCALE])

(SK.MAKE.POSITION.INTEGER
  [LAMBDA (POS)                                              (* rrb "11-Sep-86 14:35")
                                                             (* makes sure a position has integer coordinates)
    (COND
       ((AND (FIXP (fetch (POSITION XCOORD) of POS))
             (FIXP (fetch (POSITION YCOORD) of POS)))        (* avoid creation if possible)
        POS)
       (T (create POSITION
                 XCOORD _ (FIXR (fetch (POSITION XCOORD) of POS))
                 YCOORD _ (FIXR (fetch (POSITION YCOORD) of POS])

(SCALE.POSITION.INTO.SKETCHW
  [LAMBDA (POS SKETCHW)                                      (* rrb "11-Jul-86 15:52")
                                                             (* scales a position into a sketch window using its scale factor.)
    (SK.SCALE.POSITION.INTO.VIEWER POS (VIEWER.SCALE SKETCHW])

(UNSCALE
  [LAMBDA (COORD SCALE)                                      (* unscales a coordinate)
    (TIMES COORD SCALE])

(UNSCALE.REGION
  [LAMBDA (REGION SCALE)                                     (* rrb "15-AUG-83 17:31")
                                                             (* scales a region from a window region to the larger coordinate
    space.)
    (CREATEREGION (TIMES SCALE (fetch (REGION LEFT) of REGION))
           (TIMES SCALE (fetch (REGION BOTTOM) of REGION))
           (TIMES SCALE (fetch (REGION WIDTH) of REGION))
           (TIMES SCALE (fetch (REGION HEIGHT) of REGION])
)

;; misc IO functions

(DEFINEQ

(STATUSPRINT
  [LAMBDA NEXPS                                              (* rrb "26-Jun-84 09:42")

            (* prints a list of expressions in the status window associated with another window.
            If the first arg is a window or a process, its prompt window is used.
            Otherwise, the global prompt window is used.)

    (OR (EQ NEXPS 0)
        (PROG (WIN (BEG 1))
              (COND
                 ((WINDOWP (ARG NEXPS 1))
                  (SETQ BEG 2)
                  (SETQ WIN (MYGETPROMPTWINDOW (ARG NEXPS 1)
                                   2)))
                 [(PROCESSP (ARG NEXPS 1))
                  (SETQ BEG 2)
                  (COND
                     ([AND (HASTTYWINDOWP (ARG NEXPS 1))
                           (SETQ WIN (OPENWP (PROCESS.TTY (ARG NEXPS 1]
                      (SETQ WIN (GETPROMPTWINDOW WIN)))
                     (T (SETQ WIN PROMPTWINDOW]
                 ((EQ (ARG NEXPS 1)
                      T)
                  (SETQ BEG 2)
                  (SETQ WIN (TTYDISPLAYSTREAM)))
                 [(HASTTYWINDOWP (THIS.PROCESS))
                  (SETQ WIN (GETPROMPTWINDOW (TTYDISPLAYSTREAM]
                 (T (SETQ WIN PROMPTWINDOW)))
              (for X from BEG to NEXPS do (PRIN1 (ARG NEXPS X)
                                                 WIN])

(CLEARPROMPTWINDOW
  [LAMBDA (W)                                                (* rrb "28-Nov-84 11:20")
```

(* clears the prompt window of a window. IF W is NIL, clears the global one.)

```
    (COND
       [(WINDOWP W)
        (PROG (PWIN)
              (AND (SETQ PWIN (GETPROMPTWINDOW W NIL NIL T))
                   (OPENWP PWIN)
                   (CLEARW PWIN]
       (T (CLRPROMPT])
```

(**CLOSEPROMPTWINDOW**

```
  [LAMBDA (WINDOW)                                              (* rrb "20-Nov-85 10:26")
                                                                (* clears and closes the prompt window for a window.)
    (PROG [(PROMPTW (OPENWP (GETPROMPTWINDOW WINDOW NIL NIL T]
          (COND
             (PROMPTW (CLEARW PROMPTW)
                    (DETACHWINDOW PROMPTW)
                    (CLOSEW PROMPTW])
```

(**MYGETPROMPTWINDOW**

```
  [LAMBDA (MAINW NLINES FONT DONTCREATE)                        (* rrb "28-Aug-85 11:10")
                                                                (* a version of GETPROMPTWINDOW that is locally closable.)
    (PROG ((PROMPTW (GETPROMPTWINDOW (ARG NEXPS 1)
                           2
                           (OR FONT (DEFAULTFONT 'DISPLAY))
                           DONTCREATE)))
          [COND
             (PROMPTW                                           (* make it locally closeable)
                    (WINDOWADDPROP PROMPTW 'CLOSEFN (FUNCTION DETACHWINDOW]
          (RETURN PROMPTW])
```

(**PROMPT.GETINPUT**

```
  [LAMBDA (WINDOW PROMPTSTRING DEFAULTSTRING DELIMITER.LIST)    (* rrb "23-May-84 14:39")
                                                                (* Ask for input (file names, &c) perhaps with a default.)
    (PROG (PROMPTWIN)
          (COND
             (WINDOW (SETQ PROMPTWIN (GETPROMPTWINDOW WINDOW))
                    (FRESHLINE PROMPTWIN))
             ((SETQ PROMPTWIN PROMPTWINDOW)
              (CLEARW PROMPTWIN)))
          (RETURN (PROMPTFORWORD PROMPTSTRING DEFAULTSTRING NIL PROMPTWIN NIL NIL
                         (OR DELIMITER.LIST (CHARCODE (EOL LF TAB ESCAPE)))
                         NIL])
```

)

;; fns for dealing with display priorities

```
(DEFINEQ
```

(**SK.SEND.TO.BOTTOM**

```
  [LAMBDA (W)                                                   (* rrb "24-Sep-86 16:39")
```

(* allows the user to select an element or group of elements and puts them on the bottom of the priority stack.)

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

(**SK.BRING.TO.TOP**

```
  [LAMBDA (W)                                                   (* rrb "24-Sep-86 16:39")
```

(* allows the user to select an element or group of elements and brings them to the top of the priority stack.)

```
    (SK.EVAL.AS.PROCESS (LIST 'SK.SEL.AND.CHANGE.PRIORITY (KWOTE W)
                              T)
           W])
```

(**SK.SWITCH.PRIORITIES**

```
  [LAMBDA (W)                                                   (* rrb "24-Sep-86 15:21")
```

(* allows the user to select two elements and switches their positions in the priority stack.)

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

(**SK.SEL.AND.CHANGE.PRIORITY**

```
  [LAMBDA (W TOTOPFLG)                                          (* rrb "24-Sep-86 16:39")
```

(* lets the user select one or more elements and moves them to the top or the bottom of the priority stack depending on WHERE)

```
    (PROG ((SELELTS (SK.SELECT.MULTIPLE.ITEMS W T)))
          (OR SELELTS (RETURN))
          (SETQ SELELTS (SK.SORT.ELTS.BY.PRIORITY SELELTS))
          (COND
             (TOTOPFLG (SK.BRING.ELEMENTS.TO.TOP SELELTS W))
             (T (SK.SEND.ELEMENTS.TO.BOTTOM SELELTS W])
```

(**SK.SEL.AND.SWITCH.PRIORITIES**

```
  [LAMBDA (W)                                                    (* rrb "26-Sep-86 16:14")

          (* lets the user select a group of elements and reorderes them from the top to bottom.)

    (PROG ((SELELTS (SK.SELECT.MULTIPLE.ITEMS W T))
           SKETCH GELT NEWGELT PRIORITY)
          (OR (CDR SELELTS)
              (RETURN))
          (OR (SETQ SKETCH (INSURE.SKETCH W))
              (RETURN))
          (SETQ SELELTS (SK.SORT.ELTS.BY.PRIORITY SELELTS))
          (SK.DO.AND.RECORD.CHANGES (for ELT in SELELTS as TOELT in (REVERSE SELELTS)
                                       collect (SETQ GELT (fetch (SCREENELT GLOBALPART) of ELT))
                                               (SETQ NEWGELT (SK.COPY.GLOBAL.ELEMENT.AND.PROPERTY.LIST GELT))
                                               [SK.SET.ELEMENT.PRIORITY NEWGELT (SETQ PRIORITY
                                                                                  (SK.ELEMENT.PRIORITY
                                                                                   (fetch (SCREENELT GLOBALPART)
                                                                                      of TOELT]
                                               (create SKHISTORYCHANGESPEC
                                                      NEWELT _ NEWGELT
                                                      OLDELT _ GELT
                                                      PROPERTY _ 'PRIORITY
                                                      NEWVALUE _ PRIORITY
                                                      OLDVALUE _ (SK.ELEMENT.PRIORITY GELT)))
                 W T T)
          (REDISPLAYW W])
```

(**SK.SORT.ELTS.BY.PRIORITY**

```
  [LAMBDA (LOCALELTS)                                            (* rrb "24-Sep-86 15:57")
                                                                 (* sorts a list of local elements by their priority top most element
    first)
    (SORT LOCALELTS (FUNCTION (LAMBDA (A B)
                                (GREATERP (SK.ELEMENT.PRIORITY (fetch (SCREENELT GLOBALPART) of A))
                                       (SK.ELEMENT.PRIORITY (fetch (SCREENELT GLOBALPART) of B])
```

(**SK.SORT.GELTS.BY.PRIORITY**

```
  [LAMBDA (GLOBALELTS)                                           (* rrb "25-Sep-86 15:19")
                                                                 (* sorts a list of local elements by their priority bottom most
                                                                 element first)
    (SORT GLOBALELTS (FUNCTION (LAMBDA (A B)
                                 (LESSP (SK.ELEMENT.PRIORITY A)
                                        (SK.ELEMENT.PRIORITY B])
```

(**SORT.CHANGESPECS.BY.NEW.PRIORITY**

```
  [LAMBDA (CHANGESPECLST)                                        (* rrb "25-Sep-86 13:51")

          (* sorts a list of changespecs so that the first change spec element in the list is the lowest priority, etc.)

    (SORT CHANGESPECLST (FUNCTION (LAMBDA (A B)
                                    (LESSP (SK.ELEMENT.PRIORITY (fetch (SKHISTORYCHANGESPEC NEWELT) of A))
                                           (SK.ELEMENT.PRIORITY (fetch (SKHISTORYCHANGESPEC NEWELT) of B])
```

(**SORT.CHANGESPECS.BY.OLD.PRIORITY**

```
  [LAMBDA (CHANGESPECLST)                                        (* rrb "25-Sep-86 13:54")

          (* sorts a list of changespecs so that the first change spec element in the list is the lowest priority, etc.)

    (SORT CHANGESPECLST (FUNCTION (LAMBDA (A B)
                                    (LESSP (SK.ELEMENT.PRIORITY (fetch (SKHISTORYCHANGESPEC OLDELT) of A))
                                           (SK.ELEMENT.PRIORITY (fetch (SKHISTORYCHANGESPEC OLDELT) of B])
```

(**SK.SEND.ELEMENTS.TO.BOTTOM**

```
  [LAMBDA (ELEMENTS VIEWER)                                      (* rrb "24-Sep-86 18:06")

          (* * sets the priority of elements so that they all appear on the bottom.
          ELEMENTS are sorted so the topmost element is first.)

    (PROG ((SKETCH (INSURE.SKETCH VIEWER))
           LOWEST GELT NEWGELT)
          (OR SKETCH (RETURN))                                   (* find the lowest priority element so that all these do below it.)
          (SETQ LOWEST (SK.LOW.PRIORITY SKETCH))
          (SK.DO.AND.RECORD.CHANGES (for ELT in ELEMENTS
                                             collect (SETQ LOWEST (SUB1 LOWEST))
                                                    (SETQ GELT (fetch (SCREENELT GLOBALPART) of ELT))
                                                    (SETQ NEWGELT (SK.COPY.GLOBAL.ELEMENT.AND.PROPERTY.LIST GELT))
                                                    (SK.SET.ELEMENT.PRIORITY NEWGELT LOWEST)
                                                    (create SKHISTORYCHANGESPEC
                                                           NEWELT _ NEWGELT
                                                           OLDELT _ GELT
                                                           PROPERTY _ 'PRIORITY
                                                           NEWVALUE _ LOWEST
                                                           OLDVALUE _ (SK.ELEMENT.PRIORITY GELT)))
                VIEWER T T)
          (SK.LOW.PRIORITY SKETCH LOWEST)
          (REDISPLAYW VIEWER])
```

(SK.BRING.ELEMENTS.TO.TOP

```
  [LAMBDA (ELEMENTS W)                                          (* rrb "26-Sep-86 16:15")
                                                                (* sets the priority of the elements ELEMENTS so that they are
                                                                on top.)
    (PROG ((SKETCH (INSURE.SKETCH W))
           HIGHEST GELT NEWGELT)
          (OR SKETCH (RETURN))
          (SETQ HIGHEST (SK.HIGH.PRIORITY SKETCH))

          (* the elements are ordered from highest to lowest, reverse them so that they stay in the same order.)

          (SK.DO.AND.RECORD.CHANGES (for ELT in (REVERSE ELEMENTS)
                                             collect (SETQ HIGHEST (ADD1 HIGHEST))
                                                    (SETQ GELT (fetch (SCREENELT GLOBALPART) of ELT))
                                                    (SETQ NEWGELT (SK.COPY.GLOBAL.ELEMENT.AND.PROPERTY.LIST GELT))
                                                    (SK.SET.ELEMENT.PRIORITY NEWGELT HIGHEST)
                                                    (create SKHISTORYCHANGESPEC
                                                           NEWELT _ NEWGELT
                                                           OLDELT _ GELT
                                                           PROPERTY _ 'PRIORITY
                                                           NEWVALUE _ HIGHEST
                                                           OLDVALUE _ (SK.ELEMENT.PRIORITY GELT)))
                W T T)
          (SK.HIGH.PRIORITY SKETCH HIGHEST)
          (REDISPLAYW W])
```

(SK.COPY.GLOBAL.ELEMENT.AND.PROPERTY.LIST

```
  [LAMBDA (GELT)                                                (* rrb "24-Sep-86 17:26")

          (* makes a copy of a global sketch element that has the property list copied as well.)

    (PROG ((COMGLOBPART (fetch (GLOBALPART COMMONGLOBALPART) of GELT)))
          (RETURN (create GLOBALPART
                         COMMONGLOBALPART _ (create COMMONGLOBALPART
                                                   MINSCALE _ (fetch (COMMONGLOBALPART MINSCALE) of COMGLOBPART)
                                                   MAXSCALE _ (fetch (COMMONGLOBALPART MAXSCALE) of COMGLOBPART)
                                                   SKELEMENTPROPLIST _ (APPEND (fetch (COMMONGLOBALPART
                                                                                          SKELEMENTPROPLIST)
                                                                                   of COMGLOBPART)))
                         INDIVIDUALGLOBALPART _ (fetch (GLOBALPART INDIVIDUALGLOBALPART) of GELT])
)

(DEFINEQ
```

(SK.ELEMENT.PRIORITY

```
  [LAMBDA (GELEMENT)                                            (* rrb "30-Aug-86 17:52")
                                                                (* fetchs the priority of an element.)
    (OR (LISTGET (fetch (GLOBALPART SKELEMENTPROPLIST) of GELEMENT)
               'PRI)
        0])
```

(SK.SET.ELEMENT.PRIORITY

```
  [LAMBDA (GELEMENT PRIORITY)                                   (* rrb "30-Aug-86 20:50")

          (* * sets the priority of an element.)
                                                                (* keeps the priority first because it is looked at every display.)
    (PROG ((PLIST (fetch (GLOBALPART SKELEMENTPROPLIST) of GELEMENT)))
          [COND
             [PLIST (COND
                       ((EQ (CAR PLIST)
                            'PRI)
                        (RPLACA (CDR PLIST)
                               PRIORITY))
                       (T (replace (GLOBALPART SKELEMENTPROPLIST) of GELEMENT with (CONS 'PRI (CONS PRIORITY PLIST
                                                                                                       ]
             (T (replace (GLOBALPART SKELEMENTPROPLIST) of GELEMENT with (LIST 'PRI PRIORITY]
          (RETURN PRIORITY])
```

```
(SK.POP.NEXT.PRIORITY
  [LAMBDA (SKETCH)                                                  (* rrb "24-Sep-86 17:19")
                                                                    (* gets the next highest priority)
    (PROG ((PRIORITYCELL (SK.PRIORITY.CELL SKETCH)))
          (RETURN (CAR (RPLACA PRIORITYCELL (ADD1 (CAR PRIORITYCELL])

(SK.PRIORITY.CELL
  [LAMBDA (SKETCH)                                                  (* rrb "24-Sep-86 17:16")
    (OR (GETSKETCHPROP SKETCH 'PRIRANGE)
        (PUTSKETCHPROP SKETCH 'PRIRANGE (CONS 0 0])

(SK.HIGH.PRIORITY
  [LAMBDA (SKETCH VALUE)                                            (* rrb "24-Sep-86 17:21")
                                                                    (* sets a new value of the highest priority element.)
    (PROG ((CELL (SK.PRIORITY.CELL SKETCH)))
          (RETURN (PROG1 (CAR CELL)
                        (COND
                           ((NUMBERP VALUE)
                            (RPLACA CELL VALUE))))])

(SK.LOW.PRIORITY
  [LAMBDA (SKETCH VALUE)                                            (* rrb "24-Sep-86 17:22")
                                                                    (* reads and sets a new value of the lowest priority element.)
    (PROG ((CELL (SK.PRIORITY.CELL SKETCH)))
          (RETURN (PROG1 (CDR CELL)
                        (COND
                           ((NUMBERP VALUE)
                            (RPLACD CELL VALUE))))])
)

;; functions for dealing with display elements.

(DEFINEQ

(DRAW.LOCAL.SKETCH
  [LAMBDA (LOCALSPECS STREAM STREAMREGION SCALE)                    ; Edited  3-May-2023 21:00 by lmm
                                                                    ; Edited  2-May-2023 13:28 by lmm
                                                                    ; Edited 24-Mar-92 14:00 by jds

    ;; draws the local specs on a stream

    ;; set priority of the stream in case mode is set to REPLACE or ERASE --- would be better to scan list looking for an element that actually has one
    ;; of these.

    (SET.PRIORITYIMPORTANT STREAM 1)
    [MAPSKETCHSPECS LOCALSPECS (FUNCTION SK.DRAWFIGURE)
           STREAM STREAMREGION (OR (NUMBERP SCALE)
                                   (AND (WINDOWP STREAM)
                                        (VIEWER.SCALE STREAM]       ; turn the priority off so that the rest of the file procedes at speed.
    (SET.PRIORITYIMPORTANT STREAM 0])

(SET.PRIORITYIMPORTANT
  [LAMBDA (STREAM TOVAL)                                            ; Edited  2-May-2023 09:10 by lmm
                                                                    (* rrb "26-Sep-86 15:11")
                                                                    (* sets the PriorityImportant variable in an interpress master.)
    (COND
       ((IMAGESTREAMTYPEP STREAM 'INTERPRESS)
        (APPENDINTEGER.IP STREAM TOVAL)
        (ISET.IP STREAM (\IPC PRIORITYIMPORTANT])

(SK.FIGUREIMAGE
  [LAMBDA (SCRITEMS LIMITREGION REGIONOFINTEREST)                   (* rrb "30-Sep-86 18:33")

             (* returns a bitmap which contains the image of the elements on SCRITEMS.
             And a lower left corner.)

    (RESETFORM (CURSOR WAITINGCURSOR)
           (PROG (REGION DSPSTREAM BITMAP LEFT BOTTOM LIMITDIM)
                 (COND
                    ((NULL SCRITEMS)
                     (RETURN)))
                 [COND
                    ((SCREENELEMENTP SCRITEMS)                      (* single item case.)
                     (SETQ REGION (SK.ITEM.REGION SCRITEMS)))
                    (T (SETQ REGION (SK.ITEM.REGION (CAR SCRITEMS)))
                       [for SCITEM in (CDR SCRITEMS) do (SETQ REGION (SK.UNIONREGIONS REGION (SK.ITEM.REGION
                                                                                                 SCITEM]
                                                                    (* order the elements by priority)
                       (SETQ SCRITEMS (REVERSE (SK.SORT.ELTS.BY.PRIORITY SCRITEMS]
                                                                    (* only some of the points are being moved, reduce the region to
              those.)
                 (AND REGIONOFINTEREST (SETQ REGION (OR (INTERSECTREGIONS REGION REGIONOFINTEREST)
                                                      REGION)))
              [COND
                 (LIMITREGION

          (* limit the size of the bitmap. This is used by copy insert functions that do not know how big the thing coming in is.)

                    (COND
                       ((GREATERP (fetch (REGION WIDTH) of REGION)
                               (SETQ LIMITDIM (fetch (REGION WIDTH) of LIMITREGION)))
                                                           (* reduce the width picking out the middle of the region)
                        (replace (REGION LEFT) of REGION with (PLUS (fetch (REGION LEFT) of REGION)
                                                                   (QUOTIENT (DIFFERENCE
                                                                                LIMITDIM
                                                                                (fetch (REGION WIDTH)
                                                                                   of REGION))
                                                                          2)))
                        (replace (REGION WIDTH) of REGION with LIMITDIM)))
                    (COND
                       ((GREATERP (fetch (REGION HEIGHT) of REGION)
                               (SETQ LIMITDIM (fetch (REGION HEIGHT) of LIMITREGION)))
                                                           (* reduce the height picking out the middle of the region)
                        (replace (REGION BOTTOM) of REGION with (PLUS (fetch (REGION BOTTOM) of REGION)
                                                                     (QUOTIENT (DIFFERENCE
                                                                                  LIMITDIM
                                                                                  (fetch (REGION HEIGHT)
                                                                                     of REGION))
                                                                            2)))
                        (replace (REGION HEIGHT) of REGION with LIMITDIM]
                                                           (* ADD1 is used to convert the possibly floating region
                                                           coordinates into fixed.)
              [SETQ DSPSTREAM (DSPCREATE (SETQ BITMAP (BITMAPCREATE (ADD1 (fetch (REGION WIDTH) of REGION))
                                                           (ADD1 (fetch (REGION HEIGHT) of REGION]
              (DSPXOFFSET [IMINUS (SETQ LEFT (FIXR (fetch (REGION LEFT) of REGION]
                     DSPSTREAM)
              (DSPYOFFSET [IMINUS (SETQ BOTTOM (FIXR (fetch (REGION BOTTOM) of REGION]
                     DSPSTREAM)

          (* this is because the default clipping region is smaller than the clipping region of the figure in extreme cases.)

              (DSPCLIPPINGREGION REGION DSPSTREAM)
              (DSPOPERATION 'PAINT DSPSTREAM)              (* to avoid carriage returns.)
              (DSPRIGHTMARGIN (PLUS 100 (fetch (REGION RIGHT) of REGION))
                     DSPSTREAM)
              (DRAW.LOCAL.SKETCH SCRITEMS DSPSTREAM REGION)
              (RETURN (create SKFIGUREIMAGE
                             SKFIGURE.LOWERLEFT _ (create POSITION
                                                         XCOORD _ LEFT
                                                         YCOORD _ BOTTOM)
                             SKFIGURE.BITMAP _ BITMAP])

)

;; functions for hardcopying

(DEFINEQ

(SKETCHW.HARDCOPYFN
  [LAMBDA (SKETCHW OPENIMAGESTREAM)                        ; Edited 20-Aug-92 13:33 by jds
                                                           ; dumps the sketch onto OPENIMAGESTREAM.
                                                           ; centers it within the DSPCLIPPINGREGION of
                                                           ; OPENIMAGESTREAM
    (PROG ((SKETCH (INSURE.SKETCH (SKETCH.FROM.VIEWER SKETCHW)))
           (PAGEREGION (DSPCLIPPINGREGION NIL OPENIMAGESTREAM))
           (SKETCHREGION (SKETCH.REGION.VIEWED SKETCHW))
           (SCALE (VIEWER.SCALE SKETCHW))
           SKETCHREGIONINPAGECOORDS PAGELEFTSPACE PAGEBOTTOMSPACE PAGETOSKETCHFACTOR SKETCHX)
          (OR SKETCH (RETURN))
          (SPAWN.MOUSE)

     ;; move the margins out of the way

     ;; smallp is to maintain compatibility with koto.  For Lute release, this could be increased. (DONE: JDS 8/2-0/92)

          (DSPLEFTMARGIN (MIN 0 (fetch (REGION LEFT) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPBOTTOMMARGIN (MIN 0 (fetch (REGION BOTTOM) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPTOPMARGIN (MAX MAX.FIXP (fetch (REGION TOP) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPRIGHTMARGIN (MAX MAX.FIXP (fetch (REGION RIGHT) of PAGEREGION))
                 OPENIMAGESTREAM)

     ;; PAGETOSKETCHFACTOR is the factor to multiply the page coordinates by to get into sketch coordinates.

          (STATUSPRINT SKETCHW "Hardcopying ...")
          [STREAMPROP OPENIMAGESTREAM 'PRINTOPTIONS (APPEND (LIST 'DOCUMENT.NAME (OR (SKETCH.TITLE SKETCHW)
                                                                                     "A Sketch"))
                                                            (STREAMPROP OPENIMAGESTREAM 'PRINTOPTIONS]
          (SETQ PAGETOSKETCHFACTOR (FQUOTIENT SCALE (DSPSCALE NIL OPENIMAGESTREAM)))
          (SETQ SKETCHREGIONINPAGECOORDS (SCALE.REGION.OUT SKETCHREGION PAGETOSKETCHFACTOR))
          (COND
             ((AND (IMAGESTREAMTYPEP OPENIMAGESTREAM 'INTERPRESS)
                   (GREATERP (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS)
                          (fetch (REGION WIDTH) of PAGEREGION))
                   (GREATERP (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS)
                          (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS)))
                                                             ; Print in landscape mode
                                                             ; only know the hack for interpress streams.
                                                             ; Hack to coerce interpress stream into landscapemode

              ;; It's Landscape mode. PRINTERMODE may be looked up by POLYSHADE.IP

              (NCONC (fetch (STREAM OTHERPROPS) of OPENIMAGESTREAM)
                     '(PRINTERMODE LANDSCAPE))
              (ROTATE.IP OPENIMAGESTREAM 90)
              (CONCATT.IP OPENIMAGESTREAM)
              (TRANSLATE.IP OPENIMAGESTREAM 0 -21590)
              (CONCATT.IP OPENIMAGESTREAM)
              (DSPCLIPPINGREGION (SETQ PAGEREGION (SK.SWITCH.REGION.X.AND.Y PAGEREGION))
                     OPENIMAGESTREAM)                        ; End HACK

              ))
          (SETQ PAGELEFTSPACE (QUOTIENT (DIFFERENCE (fetch (REGION WIDTH) of PAGEREGION)
                                               (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS))
                                     2))
          (SETQ PAGEBOTTOMSPACE (QUOTIENT (DIFFERENCE (fetch (REGION HEIGHT) of PAGEREGION)
                                                 (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS))
                                       2))

     ;; translate the sketch so that the lower left corner of the sketch region is at the lower left corner of the image on the page.

          [SETQ SKETCHX (TRANSLATE.SKETCH SKETCH (MINUS (TIMES (DIFFERENCE (SETQ PAGELEFTSPACE
                                                                                 (PLUS (fetch (REGION LEFT)
                                                                                          of PAGEREGION)
                                                                                       PAGELEFTSPACE))
                                                                           (fetch (REGION LEFT) of
                                                                                               SKETCHREGIONINPAGECOORDS
                                                                                 ))
                                                                  PAGETOSKETCHFACTOR))
                                  (MINUS (TIMES (DIFFERENCE (SETQ PAGEBOTTOMSPACE (PLUS (fetch (REGION BOTTOM)
                                                                                           of PAGEREGION)
                                                                                        PAGEBOTTOMSPACE))
                                                       (fetch (REGION BOTTOM) of SKETCHREGIONINPAGECOORDS))
                                                PAGETOSKETCHFACTOR] ; calculate the local parts for the interpress sketch.
          (SETQ SKETCHX (MAKE.LOCAL.SKETCH SKETCHX (CREATEREGION (TIMES PAGELEFTSPACE PAGETOSKETCHFACTOR)
                                                          (TIMES PAGEBOTTOMSPACE PAGETOSKETCHFACTOR)
                                                          (fetch (REGION WIDTH) of SKETCHREGION)
                                                          (fetch (REGION HEIGHT) of SKETCHREGION))
                                  PAGETOSKETCHFACTOR OPENIMAGESTREAM))
          (DRAW.LOCAL.SKETCH SKETCHX OPENIMAGESTREAM (CREATEREGION PAGELEFTSPACE PAGEBOTTOMSPACE
                                                              (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS)
                                                              (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS))
                 )
          (STATUSPRINT SKETCHW " done.")
          (RETURN OPENIMAGESTREAM])
```

(**SK.LIST.IMAGE**

```
  [LAMBDA (SKETCHW FILE IMAGETYPE DONTLISTFLG)               ; Edited 20-Aug-92 13:42 by jds

    ;; makes an image file from the sketch in a window even if it takes more than one page.

    (PROG ((SKETCH (INSURE.SKETCH (SKETCH.FROM.VIEWER SKETCHW)))
           (VIEWREGION (DSPCLIPPINGREGION NIL SKETCHW))
           (SCALE (VIEWER.SCALE SKETCHW))
           PAGEREGION OPENIMAGESTREAM PAGEOVERLAPMARGIN SKETCHREGION SKETCHLOCALELTS SKETCHREGIONINPAGECOORDS
           LEFTSTART BOTTOMSTART RIGHTEND BOTTOMEND PAGETOSKETCHFACTOR PAGEHEIGHTINSKETCHCOORDS
           PAGEWIDTHINSKETCHCOORDS)
          (OR SKETCH (RETURN))
          (SPAWN.MOUSE)
          (STATUSPRINT SKETCHW "Hardcopying ... ")
          (SETQ OPENIMAGESTREAM (OPENIMAGESTREAM FILE IMAGETYPE))
          (SETQ PAGEREGION (DSPCLIPPINGREGION NIL OPENIMAGESTREAM))
                                                             ; move the margins out of the way
          (DSPLEFTMARGIN (MIN 0 (fetch (REGION LEFT) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPBOTTOMMARGIN (MIN 0 (fetch (REGION BOTTOM) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPTOPMARGIN (MAX MAX.FIXP (fetch (REGION TOP) of PAGEREGION))
                 OPENIMAGESTREAM)
          (DSPRIGHTMARGIN (MAX MAX.FIXP (fetch (REGION RIGHT) of PAGEREGION))
                 OPENIMAGESTREAM)

     ;; calculate the local elements for all the sketch elements at this scale.  This is done because the region testing routines all work on local
     ;; elements.  The local elements will be made again for each page;  wasteful but should demonstrate the capability.

          (SETQ SKETCHLOCALELTS (for SKELT in (fetch (SKETCH SKETCHELTS) of SKETCH)
                                   collect (SK.LOCAL.FROM.GLOBAL SKELT SKETCHW SCALE)))
          (SETQ SKETCHREGION (SK.GLOBAL.REGION.OF.LOCAL.ELEMENTS SKETCHLOCALELTS SCALE))

     ;; PAGETOSKETCHFACTOR is the factor to multiply the page coordinates by to get into sketch coordinates.
```

```
            (SETQ PAGETOSKETCHFACTOR (FQUOTIENT SCALE (DSPSCALE NIL OPENIMAGESTREAM)))
            (SETQ SKETCHREGIONINPAGECOORDS (SCALE.REGION.OUT SKETCHREGION PAGETOSKETCHFACTOR))
                                                                   ; should check here for wider than high and rotate it or use
                                                                   ; landscape imagestream.
            [COND
               ((AND (ILESSP (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS)
                             (fetch (REGION WIDTH) of PAGEREGION))
                     (ILESSP (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS)
                             (fetch (REGION HEIGHT) of PAGEREGION)))  ; whole image fits on one page, center it
                (SETQ LEFTSTART (QUOTIENT (DIFFERENCE (fetch (REGION WIDTH) of PAGEREGION)
                                                     (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS))
                                         2))
                (SETQ BOTTOMSTART (QUOTIENT (DIFFERENCE (fetch (REGION HEIGHT) of PAGEREGION)
                                                       (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS))
                                           2))
                (\SK.LIST.PAGE.IMAGE OPENIMAGESTREAM SKETCHREGION SKETCHLOCALELTS PAGETOSKETCHFACTOR
                       (CREATEREGION LEFTSTART BOTTOMSTART (fetch (REGION WIDTH) of SKETCHREGIONINPAGECOORDS)
                              (fetch (REGION HEIGHT) of SKETCHREGIONINPAGECOORDS))
                       SCALE))
               (T                                                  ; put sketch on multiple pages.  Might also try scaling it to fit.
                                                                   ; leave a half inch so that the pages can be taped together.
                  (SETQ PAGEOVERLAPMARGIN (TIMES 36 (DSPSCALE NIL OPENIMAGESTREAM)))
                  (SETQ PAGEREGION (CREATEREGION (fetch (REGION LEFT) of PAGEREGION)
                                          (fetch (REGION BOTTOM) of PAGEREGION)
                                          (DIFFERENCE (fetch (REGION WIDTH) of PAGEREGION)
                                                 PAGEOVERLAPMARGIN)
                                          (DIFFERENCE (fetch (REGION HEIGHT) of PAGEREGION)
                                                 PAGEOVERLAPMARGIN)))
                  (SETQ PAGEWIDTHINSKETCHCOORDS (TIMES (fetch (REGION WIDTH) of PAGEREGION)
                                                       PAGETOSKETCHFACTOR))
                  (SETQ PAGEHEIGHTINSKETCHCOORDS (TIMES (fetch (REGION HEIGHT) of PAGEREGION)
                                                        PAGETOSKETCHFACTOR))

                  ;; adjust sketch region to center the image within the multiple pages.  This is mostly to cover the case of a wide but not high image
                  ;; that extents across multiple pages.

                  [COND
                     ([NOT (ZEROP (SETQ LEFTSTART (REMAINDER (fetch (REGION WIDTH) of SKETCHREGION)
                                                         PAGEWIDTHINSKETCHCOORDS]
                                                                   ; unless the sketch is right on a page boundary, leave half the
                                                                   ; room in front.
                      (SETQ LEFTSTART (QUOTIENT (DIFFERENCE PAGEWIDTHINSKETCHCOORDS LEFTSTART)
                                             2]
                  (SETQ LEFTSTART (DIFFERENCE (fetch (REGION LEFT) of SKETCHREGION)
                                         LEFTSTART))
                  [COND
                     ([NOT (ZEROP (SETQ BOTTOMSTART (REMAINDER (fetch (REGION HEIGHT) of SKETCHREGION)
                                                           PAGEHEIGHTINSKETCHCOORDS]
                                                                   ; unless the sketch is right on a page boundary, leave half the
                                                                   ; room in front.
                      (SETQ BOTTOMSTART (QUOTIENT (DIFFERENCE PAGEHEIGHTINSKETCHCOORDS BOTTOMSTART)
                                               2]
                  (SETQ BOTTOMSTART (DIFFERENCE (PLUS (fetch (REGION TOP) of SKETCHREGION)
                                                      BOTTOMSTART)
                                           PAGEHEIGHTINSKETCHCOORDS))
                  (SETQ BOTTOMEND (DIFFERENCE (fetch (REGION BOTTOM) of SKETCHREGION)
                                         PAGEHEIGHTINSKETCHCOORDS))
                  (SETQ RIGHTEND (fetch (REGION RIGHT) of SKETCHREGION))
                  (STATUSPRINT SKETCHW (TIMES (IQUOTIENT (DIFFERENCE (PLUS RIGHTEND (SUB1 PAGEWIDTHINSKETCHCOORDS
                                                                                          ))
                                                                     LEFTSTART)
                                                        PAGEWIDTHINSKETCHCOORDS)
                                              (IQUOTIENT (DIFFERENCE (PLUS BOTTOMSTART (SUB1
                                                                                        PAGEHEIGHTINSKETCHCOORDS
                                                                                           ))
                                                                     BOTTOMEND)
                                                        PAGEHEIGHTINSKETCHCOORDS))
                         " pgs...")
                  (bind (PGN _ 0) for PGBOTTOM from BOTTOMSTART to BOTTOMEND by (MINUS PAGEHEIGHTINSKETCHCOORDS)
                     as PGROW from 1
                     do                                            ; unless this is the first line of pages, put out new page.
                        (OR (EQ PGROW 1)
                            (DSPNEWPAGE OPENIMAGESTREAM))
                        (for PGLEFT from LEFTSTART to RIGHTEND by PAGEWIDTHINSKETCHCOORDS as PGCOL from 1
                           do                                      ; unless this is the first page on a line of pages, put out new
                                                                   ; page.
                              (OR (EQ PGCOL 1)
                                  (DSPNEWPAGE OPENIMAGESTREAM))
                              (\SK.LIST.PAGE.IMAGE OPENIMAGESTREAM (CREATEREGION PGLEFT PGBOTTOM
                                                                          PAGEWIDTHINSKETCHCOORDS
                                                                          PAGEHEIGHTINSKETCHCOORDS)
                                     SKETCHLOCALELTS PAGETOSKETCHFACTOR PAGEREGION SCALE)
                              (STATUSPRINT SKETCHW (SETQ PGN (ADD1 PGN))
                                     ",")

                              ;; code to put out matrix numbers that I couldn't get to work.  (COND ((IMAGESTREAMTYPEP
                              ;; OPENIMAGESTREAM (QUOTE PRESS)) (* Press does better at the left edge so put numbers on the right.)
                              ;; (COND ((LESSP (PLUS PGLEFT PAGEWIDTHINSKETCHCOORDS) (fetch (REGION RIGHT) of
                                        ;; SKETCHREGION)) (* unless this is the last page, print a page number in the area that is overlapped.) (* this
                                        ;; should change back to the default font of the stream but I don't know how to do that.) (MOVETO (fetch (REGION
                                        ;; WIDTH) of PAGEREGION) (PLUS (fetch (REGION HEIGHT) of PAGEREGION) (FONTPROP
                                        ;; OPENIMAGESTREAM (QUOTE DESCENT))) OPENIMAGESTREAM) (printout OPENIMAGESTREAM
                                        ;; PGROW ', ' PGCOL)))) ((NEQ PGCOL 1) (* Interpress and assumed all others look better at the right edge so
                                        ;; put the number on the left.) (* unless this is the first page, print a page number in the area that is overlapped.) (*
                                        ;; this should change back to the default font of the stream but I don't know how to do that.) (MOVETO 10
                                        ;; (FONTPROP OPENIMAGESTREAM (QUOTE DESCENT)) OPENIMAGESTREAM) (printout
                                        ;; OPENIMAGESTREAM PGROW ', ' PGCOL)))

                                        ]
          (SETQ LEFTSTART (CLOSEF OPENIMAGESTREAM))
          (STATUSPRINT SKETCHW "...done.")
          (RETURN LEFTSTART])

(SK.HARDCOPYIMAGEW
  [LAMBDA (SKW)                                                      ; Edited 20-Aug-92 13:46 by jds

    ;; spawns a process to hardcopy a viewer.  This is spawned so that the lock on the viewer is released.

    (ADD.PROCESS (LIST 'HARDCOPYIMAGEW (KWOTE SKW))
           'NAME
           'SketchHardcopy])
)

(DEFINEQ

(SK.DO.HARDCOPYIMAGEW.TOFILE
  [LAMBDA (W)                                                        (* rrb " 5-May-86 13:38")
                                                                     (* sketch version of HARDCOPYIMAGEW.TOFILE that accepts
    a candidate file name.)
    (RESETFORM (TTY.PROCESS (THIS.PROCESS))
           (LET [(FILE&TYPE (SK.GetImageFile (SK.PRINTER.FILE.CANDIDATE.NAME W]
                (COND
                   (FILE&TYPE (HARDCOPY.SOMEHOW W (CAR FILE&TYPE)
                                     (CDR FILE&TYPE])

(SK.HARDCOPYIMAGEW.TOFILE
  [LAMBDA (SKW)                                                      (* rrb " 5-May-86 13:34")

          (* spawns a process to hardcopy a viewer. This is spawned so that the lock on the viewer is released.)

    (ADD.PROCESS (LIST 'SK.DO.HARDCOPYIMAGEW.TOFILE (KWOTE SKW))
           'NAME
           'SketchHardcopy])

(SK.HARDCOPYIMAGEW.TOPRINTER
  [LAMBDA (SKW)                                                      (* rrb "10-Feb-86 14:31")

          (* spawns a process to hardcopy a viewer. This is spawned so that the lock on the viewer is released.)

    (ADD.PROCESS (LIST 'HARDCOPYIMAGEW.TOPRINTER (KWOTE SKW))
           'NAME
           'SketchHardcopy])

(SK.LIST.IMAGE.ON.FILE
  [LAMBDA (SKETCHW)                                                  (* rrb " 5-May-86 13:39")

          (* makes a file suitable for the default printing host of the current sketch.
          Pretty dumb about file names.)

    (RESETFORM (TTY.PROCESS (THIS.PROCESS))
           (LET [(FILE&TYPE (SK.GetImageFile (SK.PRINTER.FILE.CANDIDATE.NAME SKETCHW]
                (COND
                   (FILE&TYPE (SK.LIST.IMAGE SKETCHW (CAR FILE&TYPE)
                                     (CDR FILE&TYPE])
)

(DEFINEQ

(\SK.LIST.PAGE.IMAGE
  [LAMBDA (OPENIMAGESTREAM REGIONINSKETCH LOCALSKELTS PAGETOSKETCHFACTOR REGIONONPAGE SKETCHTOWINDOWFACTOR)
                                                                     (* rrb "30-Dec-85 17:29")
                                                                     (* draws the image of a set of sketch elements on an
                                                                     OPENIMAGESTREAM.)
    (PROG ((SCALEDSKETCHREGION (SCALE.REGION.OUT REGIONINSKETCH SKETCHTOWINDOWFACTOR))
           ELTSINREGION SKETCHX)
          (COND
             ((SETQ ELTSINREGION (for LOCALSKELT in LOCALSKELTS when (REGIONSINTERSECTP SCALEDSKETCHREGION
                                                                                (SK.ITEM.REGION LOCALSKELT))
                                        collect (fetch (SCREENELT GLOBALPART) of LOCALSKELT)))
```

```
            (* translate the sketch so that the right stuff appears in the region on the page.)

                [SETQ SKETCHX (TRANSLATE.SKETCH (create SKETCH
                                                       SKETCHELTS _ ELTSINREGION)
                                     (DIFFERENCE (fetch (REGION LEFT) of REGIONINSKETCH)
                                            (TIMES (fetch (REGION LEFT) of REGIONONPAGE)
                                                   PAGETOSKETCHFACTOR))
                                     (DIFFERENCE (fetch (REGION BOTTOM) of REGIONINSKETCH)
                                            (TIMES (fetch (REGION BOTTOM) of REGIONONPAGE)
                                                   PAGETOSKETCHFACTOR]
                (SETQ SKETCHX (MAKE.LOCAL.SKETCH SKETCHX (CREATEREGION 0 0 (fetch (REGION WIDTH) of REGIONINSKETCH)
                                                                (fetch (REGION HEIGHT) of REGIONINSKETCH))
                                     PAGETOSKETCHFACTOR OPENIMAGESTREAM T))
                (DRAW.LOCAL.SKETCH SKETCHX OPENIMAGESTREAM REGIONONPAGE])

(SK.GetImageFile
  [LAMBDA (CANDIDATE)                                              (* rrb " 5-May-86 10:41")
                                                                   (* version of GetImageFile that takes a candidate name.)
    (PROG ((FILE (PopUpWindowAndGetAtom "File name (CR to abort): " CANDIDATE))
           PRINTFILETYPE FILETYPEMENU EXTENSIONSUPPLIED EXTENSIONFORTYPE)
          (COND
             ((NULL FILE)
              (RETURN)))
          (SETQ FILETYPEMENU (MakeMenuOfImageTypes "File type?"))
          (COND
             ((SETQ PRINTFILETYPE (PRINTFILETYPE.FROM.EXTENSION FILE))
              (RETURN (CONS FILE PRINTFILETYPE)))
             (T (SETQ PRINTFILETYPE (MENU FILETYPEMENU))
                (COND
                   ((NULL PRINTFILETYPE)
                    (RETURN))
                   (T (RETURN (CONS FILE PRINTFILETYPE])

(SK.PRINTER.FILE.CANDIDATE.NAME
  [LAMBDA (VIEWER)                                                 (* rrb " 5-May-86 13:30")

            (* * returns the preferred printer file name for a viewer)

    (PROG ((FILENAME (SK.OUTPUT.FILE.NAME (SKETCH.TITLE VIEWER)))
           EXTENSION PRINTEXTENSION)
          (OR FILENAME (RETURN))
          [COND
             ((EQ (SELECTQ (SETQ PRINTEXTENSION (DEFAULTPRINTINGIMAGETYPE))
                           (INTERPRESS (SETQ PRINTEXTENSION 'IP))
                           NIL)
                  (FILENAMEFIELD FILENAME 'EXTENSION))

            (* file name has a printer extension for some reason, propose either a null extension or hdcpy extension.)

              (COND
                 (PRINTEXTENSION (SETQ PRINTEREXTENSION NIL))
                 (T (SETQ PRINTEREXTENSION 'HDCPY]
          (RETURN (PACKFILENAME 'EXTENSION PRINTEXTENSION 'BODY FILENAME])

(SK.SET.HARDCOPY.MODE
  [LAMBDA (SKETCHW IMAGETYPE)                                      (* rrb "28-Oct-85 16:43")

            (* * changes a sketch window to show things in hardcopy mode.)

    (PROG [NOWTYPE (IMAGETYPEX (OR IMAGETYPE (PRINTERTYPE]
          (RETURN (COND
                     ((OR (NOT (IMAGESTREAMTYPEP SKETCHW 'HARDCOPY))
                          (AND (SETQ NOWTYPE (HARDCOPYSTREAMTYPE SKETCHW))
                               (NEQ IMAGETYPEX NOWTYPE)))

            (* make the font of the stream be something that will not cause MAKEHARDCOPYSTREAM to barf on.)
                                                                   (* flip cursor because finding fonts can take a while.)
                      (SKED.CLEAR.SELECTION SKETCHW)
                      (RESETFORM (CURSOR WAITINGCURSOR)
                             (DSPFONT (DEFAULTFONT IMAGETYPE)
                                    SKETCHW)
                             (MAKEHARDCOPYSTREAM SKETCHW IMAGETYPE)
                             (SK.UPDATE.AFTER.HARDCOPY SKETCHW)))
                     (T                                            (* already in hardcopy mode.)
                        (STATUSPRINT SKETCHW "The display is already showing " IMAGETYPE " output spacing."])

(SK.UNSET.HARDCOPY.MODE
  [LAMBDA (SKETCHW)                                                (* rrb "28-Oct-85 16:43")

            (* * changes a sketch window to show things in normal display mode.)

    (COND
       ((IMAGESTREAMTYPEP (GETSTREAM SKETCHW 'OUTPUT)
               'HARDCOPY)
        (SKED.CLEAR.SELECTION SKETCHW)
        (UNMAKEHARDCOPYSTREAM SKETCHW)
        (SK.UPDATE.AFTER.HARDCOPY SKETCHW])

(SK.UPDATE.AFTER.HARDCOPY
  [LAMBDA (SKETCHW)                                              (* rrb "11-Jul-86 15:48")

          (* * goes through a sketch window updating those elements that have changed as a result of a change in mode between
          normal and hardcopy and redraws the screen.)

    (MAPSKETCHSPECS (LOCALSPECS.FROM.VIEWER SKETCHW)
           [FUNCTION (LAMBDA (SKELT SKW SCALE)
                       (COND
                          ((MEMB (fetch (SCREENELT GTYPE) of SKELT)
                                 '(TEXT TEXTBOX))
                           (ZOOM.UPDATE.ELT SKELT SKW]
           SKETCHW
           (VIEWER.SCALE SKETCHW))
    (REDISPLAYW SKETCHW])

(DEFAULTPRINTINGIMAGETYPE
  [LAMBDA NIL                                                    (* rrb "20-Mar-85 12:45")
                                                                 (* returns the image type of the default printer.)
                                                                 (* code copied from OPENIMAGESTREAM)
    (CAR (MKLIST (PRINTERPROP (PRINTERTYPE (OR (CAR (LISTP DEFAULTPRINTINGHOST))
                                               DEFAULTPRINTINGHOST))
                        'CANPRINT])

(SK.SWITCH.REGION.X.AND.Y
  [LAMBDA (REGION)                                               (* rrb " 3-Sep-85 14:50")
                                                                 (* switchs the X and Y dimensions of a region.)
    (CREATEREGION (fetch (REGION BOTTOM) of REGION)
           (fetch (REGION LEFT) of REGION)
           (fetch (REGION HEIGHT) of REGION)
           (fetch (REGION WIDTH) of REGION])
)

(DECLARE%: EVAL@COMPILE

(RPAQQ MICASPERPT 35.27778)

(RPAQQ IMICASPERPT 35)

(RPAQQ PTSPERMICA 0.02834646)

(CONSTANTS MICASPERPT IMICASPERPT PTSPERMICA)
)

;; fns to implement transformations on the elements

(DEFINEQ

(SK.SEL.AND.TRANSFORM
  [LAMBDA (W TRANSFORMFN TRANSFORMDATA)                          (* rrb "10-Dec-85 17:25")

          (* lets the user select some elements and moves all of their control points onto the grid.)

    (SK.TRANSFORM.ELEMENTS (SK.SELECT.MULTIPLE.ITEMS W T NIL 'MOVE)
           TRANSFORMFN TRANSFORMDATA W])

(SK.TRANSFORM.ELEMENTS
  [LAMBDA (SCRELTS TRANSFORMFN TRANSFORMDATA SKW)                (* rrb "26-Apr-85 09:08")

          (* changes SCRELTS to the elements that have had each of their control points transformed by transformfn.
          TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG (NEWGLOBALS)

          (* computes the scale factor inherent in the transformation so that it doesn't have to be done on every element that might
          need it. It major use is in scaling brush sizes.)

          (SETQ NEWGLOBALS (MAPCOLLECTSKETCHSPECS SCRELTS (FUNCTION SK.TRANSFORM.ITEM)
                                  TRANSFORMFN TRANSFORMDATA (SK.TRANSFORM.SCALE.FACTOR TRANSFORMFN
                                                                   TRANSFORMDATA)
                                  SKW))                          (* make a history entry.)
          (SK.ADD.HISTEVENT 'MOVE (for NEWG in NEWGLOBALS as OLDG in SCRELTS when NEWG
                                       collect (LIST (fetch (SCREENELT GLOBALPART) of OLDG)
                                                     NEWG))
                 SKW)
          (RETURN NEWGLOBALS])
```

```
(SK.TRANSFORM.ITEM
  [LAMBDA (SELELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR W)        (* rrb "26-Apr-85 09:09")
                                                                  (* SELELT is a sketch element that was selected for a
                                                                  transformation operation.)
    (PROG (NEWGLOBAL OLDGLOBAL)
          (COND
             ((SETQ NEWGLOBAL (SK.TRANSFORM.ELEMENT (SETQ OLDGLOBAL (fetch (SCREENELT GLOBALPART) of SELELT))
                                      TRANSFORMFN TRANSFORMDATA SCALEFACTOR))
              (SK.UPDATE.ELEMENT OLDGLOBAL NEWGLOBAL W T)
              (RETURN NEWGLOBAL])

(SK.TRANSFORM.ELEMENT
  [LAMBDA (GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR)            (* rrb "26-Apr-85 09:14")

          (* 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.)

    (APPLY* (SK.TRANSFORMFN (fetch (GLOBALPART GTYPE) of GELT))
           GELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR])

(SK.TRANSFORM.POINT
  [LAMBDA (PT TRANSFORMFN TRANSFORMDATA)                          (* applies a transformation function to a position and returns the
    transformed point.)
    (APPLY* TRANSFORMFN PT TRANSFORMDATA])

(SK.TRANSFORM.POINT.LIST
  [LAMBDA (PTLST TRANSFORMFN TRANSFORMDATA)                       (* transforms a list of points)
    (for PT in PTLST collect (SK.TRANSFORM.POINT PT TRANSFORMFN TRANSFORMDATA])

(SK.TRANSFORM.REGION
  [LAMBDA (REG TRANSFORMFN TRANSFORMDATA)                         (* rrb "31-May-85 10:42")
                                                                  (* applies a transformation function to a region and returns the
                                                                  transformed region)
    (PROG (LOWERLEFT UPPERRIGHT)

          (* 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.)

          (SETQ LOWERLEFT (SK.TRANSFORM.POINT (create POSITION
                                                      XCOORD _ (fetch (REGION LEFT) of REG)
                                                      YCOORD _ (fetch (REGION BOTTOM) of REG))
                                 TRANSFORMFN TRANSFORMDATA))
          (SETQ UPPERRIGHT (SK.TRANSFORM.POINT (create POSITION
                                                       XCOORD _ (fetch (REGION PRIGHT) of REG)
                                                       YCOORD _ (fetch (REGION PTOP) of REG))
                                  TRANSFORMFN TRANSFORMDATA))     (* transformation may have changed the relative positions of the
     upper right and lower left.)
          (RETURN (CREATEREGION (MIN (fetch (POSITION XCOORD) of LOWERLEFT)
                                     (fetch (POSITION XCOORD) of UPPERRIGHT))
                         (MIN (fetch (POSITION YCOORD) of LOWERLEFT)
                              (fetch (POSITION YCOORD) of UPPERRIGHT))
                         (ABS (DIFFERENCE (fetch (POSITION XCOORD) of UPPERRIGHT)
                                     (fetch (POSITION XCOORD) of LOWERLEFT)))
                         (ABS (DIFFERENCE (fetch (POSITION YCOORD) of UPPERRIGHT)
                                     (fetch (POSITION YCOORD) of LOWERLEFT])

(SK.PUT.ELTS.ON.GRID
  [LAMBDA (W)                                                     (* rrb "31-Jan-86 10:59")

          (* lets the user select some elements and moves all of their control points onto the grid.)

    (SK.EVAL.AS.PROCESS (LIST (FUNCTION SK.SEL.AND.TRANSFORM)
                              (KWOTE W)
                              (KWOTE (FUNCTION SK.PUT.ON.GRID))
                              (KWOTE (SK.GRIDFACTOR W)))
           W])

(SK.TRANSFORM.GLOBAL.ELEMENTS
  [LAMBDA (SCRELTS TRANSFORMFN TRANSFORMDATA)                     (* rrb "29-Apr-85 12:57")

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

    (MAPGLOBALSKETCHSPECS SCRELTS (FUNCTION SK.TRANSFORM.ELEMENT)
           TRANSFORMFN TRANSFORMDATA (SK.TRANSFORM.SCALE.FACTOR TRANSFORMFN TRANSFORMDATA])

(GLOBALELEMENTP
  [LAMBDA (ELT?)                                                  (* rrb "30-Dec-85 15:26")
```

```
            (* * returns ELT? if it is a global sketch element.)

    (AND (LISTP ELT?)
         (SKETCH.ELEMENT.NAMEP (fetch (GLOBALPART GTYPE) of ELT?))
        ELT?])

(SKETCH.LIST.OF.ELEMENTSP
  [LAMBDA (ELTS)                                            (* return T if ELTS is a list of sketch elements.)
    (AND (LISTP ELTS)
         (for ELT in ELTS always (GLOBALELEMENTP ELT])

(SK.TRANSFORM.SCALE.FACTOR
  [LAMBDA (TRANSFORMFN TRANSFORMDATA)                       (* rrb "29-Apr-85 12:09")

            (* calculates scaling factor based on the transform of points. Since the transform is arbitrary in x and y scaling, this can't
            really do the right thing so it computes the area a unit square would have after transformation and uses that.)

    (COND
       ((EQ TRANSFORMFN (FUNCTION SK.PUT.ON.GRID))          (* test for specially in case grid is larger than unit.
                                                            Don't change the scale.)
        1.0)
       (T (PROG ((ORG (SK.TRANSFORM.POINT (CONSTANT (create POSITION
                                                            XCOORD _ 0
                                                            YCOORD _ 0))
                             TRANSFORMFN TRANSFORMDATA))
                 (YUNIT (SK.TRANSFORM.POINT (CONSTANT (create POSITION
                                                              XCOORD _ 0
                                                              YCOORD _ 1))
                               TRANSFORMFN TRANSFORMDATA))
                 (XUNIT (SK.TRANSFORM.POINT (CONSTANT (create POSITION
                                                              XCOORD _ 1
                                                              YCOORD _ 0))
                               TRANSFORMFN TRANSFORMDATA)))
                (RETURN (SQRT (TIMES (DISTANCEBETWEEN YUNIT ORG)
                                     (DISTANCEBETWEEN XUNIT ORG])

(SK.TRANSFORM.BRUSH
  [LAMBDA (BRUSH SCALEFACTOR)                               (* rrb "26-Apr-85 09:34")
                                                            (* returns a brush scaled from size ORGSCALE to NEWSCALE.)
    (create BRUSH using BRUSH BRUSHSIZE _ (TIMES (fetch (BRUSH BRUSHSIZE) of BRUSH)
                                                 SCALEFACTOR])

(SK.TRANSFORM.ARROWHEADS
  [LAMBDA (ARROWHEADS SCALEFACTOR)                          (* rrb "26-Sep-85 12:17")
                                                            (* returns a arrowhead specification scaled by SCALEFACTOR)
    (AND ARROWHEADS (LIST (AND (CAR ARROWHEADS)
                               (create ARROWHEAD using (CAR ARROWHEADS)
                                                       ARROWLENGTH _ (TIMES (fetch (ARROWHEAD ARROWLENGTH)
                                                                                of (CAR ARROWHEADS))
                                                                            SCALEFACTOR)))
                          (AND (CADR ARROWHEADS)
                               (create ARROWHEAD using (CADR ARROWHEADS)
                                                       ARROWLENGTH _ (TIMES (fetch (ARROWHEAD ARROWLENGTH)
                                                                                of (CADR ARROWHEADS))
                                                                            SCALEFACTOR)))
                          (CADDR ARROWHEADS])

(SCALE.BRUSH
  [LAMBDA (BRUSH ORGSCALE NEWSCALE)                         (* rrb " 8-Sep-86 20:02")

            (* returns a brush scaled from size ORGSCALE to NEWSCALE.
            It will returns a size of 0 only if given a size of 0 This is so that brushes that scale down always show up.)

    (COND
       [(EQP ORGSCALE NEWSCALE)                             (* make unscaled case fast -
                                                            avoid floating point.)
        (PROG ((BRUSHSIZE (fetch (BRUSH BRUSHSIZE) of BRUSH)))
              (RETURN (create BRUSH using BRUSH BRUSHSIZE _ (COND
                                                              ((GREATERP 1.0 BRUSHSIZE)
                                                                 (* create a brush of at least 1)
                                                               (COND
                                                                  ((ZEROP BRUSHSIZE)
                                                                   0)
                                                                  (T 1)))
                                                              ((NOT (FIXP BRUSHSIZE))
                                                               (FIXR BRUSHSIZE))
                                                              (T (RETURN BRUSH]
       (T (PROG ((BRUSHSIZE (FQUOTIENT (FTIMES (fetch (BRUSH BRUSHSIZE) of BRUSH)
                                               ORGSCALE)
                                    NEWSCALE)))
                (RETURN (create BRUSH using BRUSH BRUSHSIZE _ (COND
                                                                ((ZEROP BRUSHSIZE)
                                                                         0)
                                                                        (T (IMAX 1 (FIXR BRUSHSIZE])
)

(DEFINEQ

(TWO.PT.TRANSFORMATION.INPUTFN
  [LAMBDA (WINDOW)                                                      (* rrb "11-Jul-86 15:54")

          (* reads four points from the user and returns the two point transformation that maps the first two into the second two.)

    (PROG ((SCALE (VIEWER.SCALE WINDOW))
           FIRSTPT SECONDPT THIRDPT FOURTHPT FIRSTLOCALPT SECONDLOCALPT THIRDLOCALPT FOURTHLOCALPT)
          (STATUSPRINT WINDOW "
                " "Indicate the first point to move.")
          (COND
             ((SETQ FIRSTPT (SK.GETGLOBALPOSITION WINDOW))
              (SK.MARK.POSITION (SETQ FIRSTLOCALPT (SK.SCALE.POSITION.INTO.VIEWER FIRSTPT SCALE))
                     WINDOW FIRSTPTMARK))
             (T (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                " "Indicate the second point to move.")
          (COND
             ((SETQ SECONDPT (SK.GETGLOBALPOSITION WINDOW))
              (SK.MARK.POSITION (SETQ SECONDLOCALPT (SK.SCALE.POSITION.INTO.VIEWER SECONDPT SCALE))
                     WINDOW SECONDPTMARK))
             (T                                                         (* erase first pt on way out)
                (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
                (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                " "Indicate the new position of the first point.")
          (COND
             ((SETQ THIRDPT (SK.GETGLOBALPOSITION WINDOW))
              (SK.MARK.POSITION (SETQ THIRDLOCALPT (SK.SCALE.POSITION.INTO.VIEWER THIRDPT SCALE))
                     WINDOW NEWFIRSTPTMARK))
             (T                                                         (* erase first and second pts on way out)
                (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
                (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
                (CLOSEPROMPTWINDOW WINDOW)
                (RETURN NIL)))
          (STATUSPRINT WINDOW "
                " "Indicate the new position of the second point.")
          (SETQ FOURTHPT (SK.GETGLOBALPOSITION WINDOW))
          (CLOSEPROMPTWINDOW WINDOW)                                    (* erase the point marks.)
          (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
          (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
          (SK.MARK.POSITION THIRDLOCALPT WINDOW NEWFIRSTPTMARK)
          (OR FOURTHPT (RETURN NIL))                                    (* keep the coefficients of the two necessary equations.)
          (RETURN (SK.COMPUTE.TWO.PT.TRANSFORMATION FIRSTPT SECONDPT THIRDPT FOURTHPT])

(SK.TWO.PT.TRANSFORM.ELTS
  [LAMBDA (W)                                                           (* rrb "31-Jan-86 10:59")

          (* lets the user select some elements and specify a two point transformation and applies the transformation to all of the
          points.)

    (SK.EVAL.AS.PROCESS (LIST (FUNCTION SK.SEL.AND.TWO.PT.TRANSFORM)
                              (KWOTE W))
           W])

(SK.SEL.AND.TWO.PT.TRANSFORM
  [LAMBDA (W)                                                           (* rrb "10-Dec-85 17:26")

          (* lets the user select some elements and specify a two point transformation and applies the transformation to all of the
          points.)

    (PROG NIL
          (SK.TRANSFORM.ELEMENTS (OR (SK.SELECT.MULTIPLE.ITEMS W T NIL 'MOVE)
                                     (RETURN))
                 (FUNCTION SK.APPLY.AFFINE.TRANSFORM)
                 (OR (TWO.PT.TRANSFORMATION.INPUTFN W)
                     (RETURN))
                 W])

(SK.APPLY.AFFINE.TRANSFORM
  [LAMBDA (GPOSITION AFFINETRANS)                                       (* rrb "28-Apr-85 16:05")

          (* * applies a tranformation to the point. AFFINETRANS is an instance of AFFINETRANSFORMATION)

    (create POSITION
           XCOORD _ (PLUS (TIMES (fetch (AFFINETRANSFORMATION Ax) of AFFINETRANS)
                               (fetch (POSITION XCOORD) of GPOSITION))
                        (TIMES (fetch (AFFINETRANSFORMATION By) of AFFINETRANS)
                               (fetch (POSITION YCOORD) of GPOSITION))
                        (fetch (AFFINETRANSFORMATION C) of AFFINETRANS))
          YCOORD _ (PLUS (TIMES (fetch (AFFINETRANSFORMATION Dx) of AFFINETRANS)
                               (fetch (POSITION XCOORD) of GPOSITION))
                        (TIMES (fetch (AFFINETRANSFORMATION Ey) of AFFINETRANS)
                               (fetch (POSITION YCOORD) of GPOSITION))
                        (fetch (AFFINETRANSFORMATION F) of AFFINETRANS])
```

(SK.COMPUTE.TWO.PT.TRANSFORMATION

```
  [LAMBDA (P1 P2 Q1 Q2)                                ; Edited 30-Jan-87 14:24 by rrb
                                                       (* computes the AFFINETRANSFORMATION necessary to take
    P1 into Q1 and P2 into Q2.)
    (PROG ((PX1 (fetch (POSITION XCOORD) of P1))
           (PY1 (fetch (POSITION YCOORD) of P1))
           (PX2 (fetch (POSITION XCOORD) of P2))
           (PY2 (fetch (POSITION YCOORD) of P2))
           (QX1 (fetch (POSITION XCOORD) of Q1))
           (QY1 (fetch (POSITION YCOORD) of Q1))
           (QX2 (fetch (POSITION XCOORD) of Q2))
           (QY2 (fetch (POSITION YCOORD) of Q2))
           (MATRIX2 (IDENTITY-3-BY-3))
           (SCRATCHMATRIX (IDENTITY-3-BY-3))
           MATRIX1 PDELTAX PDELTAY QDELTAX QDELTAY PLEN QLEN LENRATIO)

          (* compute the transformation that translates P1 to the origin, rotates it until P has the same angle as Q, scales it until P has
          the same length as Q then translates the new P1 to Q1.)

          (SETQ PDELTAX (DIFFERENCE PX2 PX1))
          (SETQ PDELTAY (DIFFERENCE PY2 PY1))
          (SETQ QDELTAX (DIFFERENCE QX2 QX1))
          (SETQ QDELTAY (DIFFERENCE QY2 QY1))          (* compute the length of segments P and Q.)
          [SETQ PLEN (SQRT (PLUS (TIMES PDELTAX PDELTAX)
                                 (TIMES PDELTAY PDELTAY]
          (COND
             ((ZEROP PLEN)
              (STATUSPRINT WINDOW "
                    " "The two source points can not be the same.")
              (RETURN)))
          [SETQ QLEN (SQRT (PLUS (TIMES QDELTAX QDELTAX)
                                 (TIMES QDELTAY QDELTAY]
          (COND
             ((ZEROP QLEN)
              (STATUSPRINT WINDOW "The two destination points can not be the same.")
              (RETURN)))

          (* ratio is done to map P onto Q because the scaling is done after the rotation.
          It could be done first if the mapping were done from Q onto P.)

          (SETQ LENRATIO (QUOTIENT QLEN PLEN))         (* translate P1 to origin.)

          (* use MATRIX1 and MATRIX2 to swap the running result back and forth since matrix multiplication routines don't allow the
          result to be stored in one of the arguments.)

          (SETQ MATRIX1 (TRANSLATE-3-BY-3 (MINUS PX1)
                               (MINUS PY1)))           (* Scale to make P the same length as Q.)
          (MATMULT-333 MATRIX1 (SCALE-3-BY-3 LENRATIO LENRATIO SCRATCHMATRIX)
                 MATRIX2)                              (* rotate it so that the slope of P is the same as Q.)
          (MATMULT-333 MATRIX2 (ROTATE-3-BY-3 (DEGREES-TO-RADIANS (DIFFERENCE (SK.COMPUTE.SLOPE PDELTAX PDELTAY
                                                                                 )
                                                                         (SK.COMPUTE.SLOPE QDELTAX QDELTAY)))
                                      SCRATCHMATRIX)
                 MATRIX1)

          (* translate the origin pt to Q1. This is complicated because Q1 needs to be translated, rotated and scaled into new
          coordinates.)

          (MATMULT-333 MATRIX1 (TRANSLATE-3-BY-3 QX1 QY1 SCRATCHMATRIX)
                 MATRIX2)                              (* return only the coefficients that make a difference.)
          (RETURN (create AFFINETRANSFORMATION
                         Ax _ (CL:AREF MATRIX2 0 0)
                         By _ (CL:AREF MATRIX2 1 0)
                         C _ (CL:AREF MATRIX2 2 0)
                         Dx _ (CL:AREF MATRIX2 0 1)
                         Ey _ (CL:AREF MATRIX2 1 1)
                         F _ (CL:AREF MATRIX2 2 1])
```

(SK.COMPUTE.SLOPE

```
  [LAMBDA (DELTAX DELTAY)                              (* rrb "31-May-85 10:09")
                                                       (* computes the angle of a line from the delta X and Y.)
    (COND
       ((ZEROP DELTAX)
        (COND
           ((GREATERP DELTAY 0)
            90.0)
           (T -90.0)))
       (T (PLUS (COND
                   ((GREATERP DELTAX 0)
                    0.0)
                   (T
```

(* if the line is sloping to the left, add 180 to it. This is done because we need to make sure that P1 gets mapped into Q1.)

```
                      180.0))
                (ARCTAN (FQUOTIENT DELTAY DELTAX])
```

(**SK.THREE.PT.TRANSFORM.ELTS**

```
  [LAMBDA (W)                                                     (* rrb "31-Jan-86 11:00")
```

(* lets the user select some elements and specify a three point transformation and applies the transformation to all of the points.)

```
    (SK.EVAL.AS.PROCESS (LIST (FUNCTION SK.SEL.AND.THREE.PT.TRANSFORM)
                              (KWOTE W))
           W])
```

(**SK.COMPUTE.THREE.PT.TRANSFORMATION**

```
  [LAMBDA (P1 P2 P3 Q1 Q2 Q3 ERRORFLG)                            (* rrb " 8-May-85 18:10")
```

(* computes the AFFINETRANSFORMATION necessary to take P1 into Q1, P2 into Q2 and P3 into Q3.)

```
    (PROG ((PX1 (fetch (POSITION XCOORD) of P1))
           (PY1 (fetch (POSITION YCOORD) of P1))
           (PX2 (fetch (POSITION XCOORD) of P2))
           (PY2 (fetch (POSITION YCOORD) of P2))
           (PX3 (fetch (POSITION XCOORD) of P3))
           (PY3 (fetch (POSITION YCOORD) of P3))
           (QX1 (fetch (POSITION XCOORD) of Q1))
           (QY1 (fetch (POSITION YCOORD) of Q1))
           (QX2 (fetch (POSITION XCOORD) of Q2))
           (QY2 (fetch (POSITION YCOORD) of Q2))
           (QX3 (fetch (POSITION XCOORD) of Q3))
           (QY3 (fetch (POSITION YCOORD) of Q3))
           DELTAPY12 DELTAPX12 DELTAPY23 A&DBOTTOM AX BY C DX EY F)
```

(* this is the computation dictated by solving the six equations of the form QX1 = aPX1 + bPY1 + c for a, b, c, d, e, and f.)

```
                                                                  (* save some subexpressions that are reused.)
          (SETQ DELTAPX12 (FDIFFERENCE PX1 PX2))
          (SETQ DELTAPY23 (FDIFFERENCE PY2 PY3))
          [COND
             ((ZEROP (SETQ DELTAPY12 (FDIFFERENCE PY1 PY2)))      (* need to divide by this number and it is zero)
              (COND
                 (ERRORFLG                                        (* this is the second attempt, all points must be horizontal)
                        (STATUSPRINT WINDOW "
                               " "All three source points cannot be in the same line.
                               If you meant this, you should use the TWO PT TRANSFORM.")
                        (RETURN))
                 (T                                               (* try switching two points)
                    (RETURN (SK.COMPUTE.THREE.PT.TRANSFORMATION P2 P3 P1 Q2 Q3 Q1 T]
          [COND
             ([ZEROP (SETQ A&DBOTTOM (FDIFFERENCE (FDIFFERENCE PX2 PX3)
                                            (FTIMES (FQUOTIENT DELTAPX12 DELTAPY12)
                                                   DELTAPY23]     (* need to divide by this number and it is zero)
              (COND
                 (ERRORFLG
```

(* this is the second attempt, maybe all points are collinear, in any case, can't continue.)

```
                        (STATUSPRINT WINDOW "
                               " "All three source points cannot be in the same line.
                               If you meant this, you should use the TWO PT TRANSFORM.")
                        (RETURN))
                 (T                                               (* try switching two points)
                    (RETURN (SK.COMPUTE.THREE.PT.TRANSFORMATION P2 P3 P1 Q2 Q3 Q1 T]
          (SETQ AX (FQUOTIENT (FDIFFERENCE (FDIFFERENCE QX2 QX3)
                                     (FQUOTIENT (FTIMES (FDIFFERENCE QX1 QX2)
                                                       DELTAPY23)
                                            DELTAPY12))
                          A&DBOTTOM))
          (SETQ DX (FQUOTIENT (FDIFFERENCE (FDIFFERENCE QY2 QY3)
                                     (FQUOTIENT (FTIMES (FDIFFERENCE QY1 QY2)
                                                       DELTAPY23)
                                            DELTAPY12))
                          A&DBOTTOM))
          (SETQ BY (FQUOTIENT (FDIFFERENCE (FDIFFERENCE QX1 QX2)
                                     (FTIMES AX DELTAPX12))
                          DELTAPY12))
          (SETQ EY (FQUOTIENT (FDIFFERENCE (FDIFFERENCE QY1 QY2)
                                     (FTIMES DX DELTAPX12))
                          DELTAPY12))
          [SETQ C (FDIFFERENCE QX1 (FPLUS (FTIMES AX PX1)
                                          (FTIMES BY PY1]
          [SETQ F (FDIFFERENCE QY1 (FPLUS (FTIMES DX PX1)
                                          (FTIMES EY PY1]
          (RETURN (create AFFINETRANSFORMATION
                         Ax _ AX
                         By _ BY
                         C _ C
                         Dx _ DX
                         Ey _ EY
                         F _ F])
```

```
(SK.SEL.AND.THREE.PT.TRANSFORM
  [LAMBDA (W)                                                (* rrb "10-Dec-85 17:26")

        (* lets the user select some elements and specify a three point transformation and applies the transformation to all of the
        points.)

    (PROG NIL
        (SK.TRANSFORM.ELEMENTS (OR (SK.SELECT.MULTIPLE.ITEMS W T NIL 'MOVE)
                                   (RETURN))
               (FUNCTION SK.APPLY.AFFINE.TRANSFORM)
               (OR (THREE.PT.TRANSFORMATION.INPUTFN W)
                   (RETURN))
               W])
```

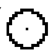
```
(THREE.PT.TRANSFORMATION.INPUTFN
  [LAMBDA (WINDOW)                                           (* rrb "11-Jul-86 15:54")

        (* reads six points from the user and returns the affine transformation that maps the first three into the second three)

    (PROG ((SCALE (VIEWER.SCALE WINDOW))
           FIRSTPT SECONDPT THIRDPT FOURTHPT FIFTHPT SIXTHPT FIRSTLOCALPT SECONDLOCALPT THIRDLOCALPT
           FOURTHLOCALPT FIFTHLOCALPT)
        (STATUSPRINT WINDOW "
               " "Indicate the first point to move.")
        (COND
           ((SETQ FIRSTPT (SK.GETGLOBALPOSITION WINDOW))
            (SK.MARK.POSITION (SETQ FIRSTLOCALPT (SK.SCALE.POSITION.INTO.VIEWER FIRSTPT SCALE))
                   WINDOW FIRSTPTMARK))
           (T (CLOSEPROMPTWINDOW WINDOW)
              (RETURN NIL)))
        (STATUSPRINT WINDOW "
               " "Indicate the second point to move.")
        (COND
           ((SETQ SECONDPT (SK.GETGLOBALPOSITION WINDOW))
            (SK.MARK.POSITION (SETQ SECONDLOCALPT (SK.SCALE.POSITION.INTO.VIEWER SECONDPT SCALE))
                   WINDOW SECONDPTMARK))
           (T                                                (* erase first pt on way out)
              (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
              (CLOSEPROMPTWINDOW WINDOW)
              (RETURN NIL)))
        (STATUSPRINT WINDOW "
               " "Indicate the third point to move.")
        (COND
           ((SETQ THIRDPT (SK.GETGLOBALPOSITION WINDOW))
            (SK.MARK.POSITION (SETQ THIRDLOCALPT (SK.SCALE.POSITION.INTO.VIEWER THIRDPT SCALE))
                   WINDOW THIRDPTMARK))
           (T                                                (* erase first and second pts on way out)
              (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
              (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
              (CLOSEPROMPTWINDOW WINDOW)
              (RETURN NIL)))
        (STATUSPRINT WINDOW "
               " "Indicate the new position of the first point.")
        (COND
           ((SETQ FOURTHPT (SK.GETGLOBALPOSITION WINDOW))
            (SK.MARK.POSITION (SETQ FOURTHLOCALPT (SK.SCALE.POSITION.INTO.VIEWER FOURTHPT SCALE))
                   WINDOW NEWFIRSTPTMARK))
           (T                                                (* erase first second and third pts on way out)
              (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
              (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
              (SK.MARK.POSITION THIRDLOCALPT WINDOW THIRDPTMARK)
              (CLOSEPROMPTWINDOW WINDOW)
              (RETURN NIL)))
        (STATUSPRINT WINDOW "
               " "Indicate the new position of the second point.")
        (COND
           ((SETQ FIFTHPT (SK.GETGLOBALPOSITION WINDOW))
            (SK.MARK.POSITION (SETQ FIFTHLOCALPT (SK.SCALE.POSITION.INTO.VIEWER FIFTHPT SCALE))
                   WINDOW NEWSECONDPTMARK))
           (T                                                (* erase first second and third pts on way out)
              (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
              (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
```

```
                  (SK.MARK.POSITION THIRDLOCALPT WINDOW THIRDPTMARK)
                  (SK.MARK.POSITION FOURTHLOCALPT WINDOW NEWFIRSTPTMARK)
                  (CLOSEPROMPTWINDOW WINDOW)
                  (RETURN NIL)))
            (STATUSPRINT WINDOW "
                  " "Indicate the new position of the third point.")
            (SETQ SIXTHPT (SK.GETGLOBALPOSITION WINDOW))
            (CLOSEPROMPTWINDOW WINDOW)                          (* erase the point marks.)
            (SK.MARK.POSITION FIRSTLOCALPT WINDOW FIRSTPTMARK)
            (SK.MARK.POSITION SECONDLOCALPT WINDOW SECONDPTMARK)
            (SK.MARK.POSITION THIRDLOCALPT WINDOW THIRDPTMARK)
            (SK.MARK.POSITION FOURTHLOCALPT WINDOW NEWFIRSTPTMARK)
            (SK.MARK.POSITION FIFTHLOCALPT WINDOW NEWSECONDPTMARK)
            (OR SIXTHPT (RETURN NIL))                           (* keep the coefficients of the two necessary equations.)
            (RETURN (SK.COMPUTE.THREE.PT.TRANSFORMATION FIRSTPT SECONDPT THIRDPT FOURTHPT FIFTHPT SIXTHPT])

)

(DEFINEQ

(SK.COPY.AND.TWO.PT.TRANSFORM.ELTS
  [LAMBDA (W)                                                   (* rrb "31-Jan-86 11:00")

            (* lets the user select some elements and specify a two point transformation and applies the transformation to all of the
            points.)

    (SK.EVAL.AS.PROCESS (LIST (FUNCTION SK.SEL.COPY.AND.TWO.PT.TRANSFORM)
                              (KWOTE W))
           W])

(SK.SEL.COPY.AND.TWO.PT.TRANSFORM
  [LAMBDA (W)                                                   (* rrb "10-Dec-85 17:26")

            (* lets the user select some elements and specify a two point transformation and applies the transformation to all copies of
            the points.)

    (PROG NIL
          (SK.COPY.AND.TRANSFORM.ELEMENTS (OR (SK.SELECT.MULTIPLE.ITEMS W T NIL 'COPY)
                                              (RETURN))
                 (FUNCTION SK.APPLY.AFFINE.TRANSFORM)
                 (OR (TWO.PT.TRANSFORMATION.INPUTFN W)
                     (RETURN))
                 W])

(SK.COPY.AND.THREE.PT.TRANSFORM.ELTS
  [LAMBDA (W)                                                   (* rrb "31-Jan-86 11:00")

            (* lets the user select some elements and specify a three point transformation and applies the transformation to copies of
            the elements)

    (SK.EVAL.AS.PROCESS (LIST (FUNCTION SK.SEL.COPY.AND.THREE.PT.TRANSFORM)
                              (KWOTE W))
           W])

(SK.SEL.COPY.AND.THREE.PT.TRANSFORM
  [LAMBDA (W)                                                   (* rrb "10-Dec-85 17:26")

            (* lets the user select some elements and specify a three point transformation and applies the transformation to copies of
            the elements)

    (PROG NIL
          (SK.COPY.AND.TRANSFORM.ELEMENTS (OR (SK.SELECT.MULTIPLE.ITEMS W T NIL 'COPY)
                                              (RETURN))
                 (FUNCTION SK.APPLY.AFFINE.TRANSFORM)
                 (OR (THREE.PT.TRANSFORMATION.INPUTFN W)
                     (RETURN))
                 W])

(SK.COPY.AND.TRANSFORM.ELEMENTS
  [LAMBDA (SCRELTS TRANSFORMFN TRANSFORMDATA SKW)               (* rrb " 8-May-85 17:08")

            (* changes copies of SCRELTS to the elements that have had each of their control points transformed by transformfn.
            TRANSFORMDATA is arbitrary data that is passed to tranformfn.)

    (PROG (NEWGLOBALS)

            (* computes the scale factor inherent in the transformation so that it doesn't have to be done on every element that might
            need it. It major use is in scaling brush sizes.)

          (SETQ NEWGLOBALS (MAPCOLLECTSKETCHSPECS SCRELTS (FUNCTION SK.COPY.AND.TRANSFORM.ITEM)
                                  TRANSFORMFN TRANSFORMDATA (SK.TRANSFORM.SCALE.FACTOR TRANSFORMFN
                                                                   TRANSFORMDATA)
                                  SKW))                         (* make a history entry.)
            (SK.ADD.HISTEVENT 'COPY NEWGLOBALS SKW)
            (RETURN NEWGLOBALS])

(SK.COPY.AND.TRANSFORM.ITEM
  [LAMBDA (SELELT TRANSFORMFN TRANSFORMDATA SCALEFACTOR W)          (* rrb "10-Mar-86 16:23")
                                                                    (* SELELT is a sketch element that was selected for a copy and
       transformation operation.)
     (PROG (NEWGLOBAL)
           (COND
              ((SETQ NEWGLOBAL (SK.TRANSFORM.ELEMENT (fetch (SCREENELT GLOBALPART) of SELELT)
                                      TRANSFORMFN TRANSFORMDATA SCALEFACTOR))
                                                                    (* clear the priority of the element.)
               (SK.SET.ELEMENT.PRIORITY NEWGLOBAL NIL)
               (SK.ADD.ELEMENT NEWGLOBAL W)
               (RETURN NEWGLOBAL])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD AFFINETRANSFORMATION (Ax By C Dx Ey F))
)
)

(READVARS-FROM-STRINGS '(FIRSTPTMARK SECONDPTMARK THIRDPTMARK NEWFIRSTPTMARK NEWSECONDPTMARK)
       "({(READBITMAP)(25 25
       %"AOCNB@@@%"
       %"AA@HF@@@%"
       %"AA@HB@@@%"
       %"AN@HB@@@%"
       %"A@@HB@@@%"
       %"A@@HB@@@%"
       %"A@@HOH@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@GO@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%")}  {(READBITMAP)(25 25
       %"AOCNG@@@%"
       %"AA@HHH@@%"
       %"AA@HAH@@%"
       %"AN@HG@@@%"
       %"A@@HL@@@%"
       %"A@@HH@@@%"
       %"A@@HOH@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@GO@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%")}  {(READBITMAP)(25 25
       %"AOCNG@@@%"
       %"AA@HHH@@%"
       %"AA@HAH@@%"
       %"AN@HF@@@%"
       %"A@@HAH@@%"
       %"A@@HHH@@%"
       %"A@@HG@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@GO@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%")}  {(READBITMAP)(25 25
       %"AAGJB@@@%"
       %"AIDBJ@@@%"
       %"AEDBJ@@@%"
       %"AEGBJ@@@%"
       %"ACDBJ@@@%"
       %"ACDBJ@@@%"
       %"AAGID@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@GO@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"AOCNB@@@%"
       %"AA@HF@@@%"
       %"AA@HB@@@%"
       %"AN@HB@@@%"
       %"A@@HB@@@%"
       %"A@@HB@@@%"
       %"A@@HOH@@%")}  {(READBITMAP)(25 25
       %"AAGJB@@@%"
       %"AIDBJ@@@%"
       %"AEDBJ@@@%"
       %"AEGBJ@@@%"
       %"ACDBJ@@@%"
       %"ACDBJ@@@%"
       %"AAGID@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@GO@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@H@@@@%"
       %"@@@@@@@@%"
       %"@@@@@@@@%"
       %"AOCNCH@@%"
       %"AA@HDD@@%"
       %"AA@H@D@@%"
       %"AN@HAH@@%"
       %"A@@HF@@@%"
       %"A@@HD@@@%"
       %"A@@HGL@@%")})
       ")

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS FIRSTPTMARK SECONDPTMARK THIRDPTMARK NEWFIRSTPTMARK NEWSECONDPTMARK)
)

(FILESLOAD MATMULT)
```

;; functions for marking

```
(DEFINEQ

(SK.SHOWMARKS
  [LAMBDA (W HOTSPOTCACHE)                                      (* rrb "29-Jan-85 18:04")
                                                                (* marks all of the hot spots of sketch elements in a figure
                                                                window.)
    (bind Y for BUCKET in HOTSPOTCACHE do (SETQ Y (CAR BUCKET))
                                          (for XBUCKET in (CDR BUCKET) do
                                                                (* there may be old buckets that don't contain any elements.)
```

```
                                                                              (AND (CDR XBUCKET)
                                                                                   (SK.MARK.HOTSPOT (CAR XBUCKET)
                                                                                          Y W SK.LOCATEMARK])

(MARKPOINT
  [LAMBDA (PT WINDOW MARK)                                                    (* rrb "12-May-85 18:50")

          (* marks a point in a window with a mark. The mark should be a bitmap.)

    (OR MARK (SETQ MARK SK.SELECTEDMARK))
    (PROG ((MARKWIDTH (BITMAPWIDTH MARK)))
          (RETURN (BITBLT MARK 0 0 WINDOW (IDIFFERENCE (fetch (POSITION XCOORD) of PT)
                                                 (LRSH MARKWIDTH 1))
                         (IDIFFERENCE (fetch (POSITION YCOORD) of PT)
                                (LRSH (fetch (BITMAP BITMAPHEIGHT) of MARK)
                                      1))
                         MARKWIDTH MARKWIDTH 'INPUT 'INVERT])

(SK.MARKHOTSPOTS
  [LAMBDA (SKETCHELT W MARK)                                                  (* rrb "12-May-85 18:59")
                                                                              (* marks the hotspots of a sketch element that are not already
                                                                              selected)
    (PROG [(HOTSPOTCACHE (SK.HOTSPOT.CACHE W))
           (SELECTEDELTS (WINDOWPROP W 'SKETCH.SELECTIONS]
          (for PTTAIL on (fetch (LOCALPART HOTSPOTS) of (fetch (SCREENELT LOCALPART) of SKETCHELT))
             unless (OR (MEMBER (CAR PTTAIL)
                                (CDR PTTAIL))
                        (for ELTSOFPT in (SK.ELTS.FROM.HOTSPOT (CAR PTTAIL)
                                                HOTSPOTCACHE)
                           thereis (MEMB ELTSOFPT SELECTEDELTS)))
             do

          (* mark points that aren't also hotspots of an already selected element or duplicate hot spots of this element.)

                (MARKPOINT (CAR PTTAIL)
                       W MARK])

(SK.MARK.SELECTION
  [LAMBDA (ELT SKW MARKBM)                                                    (* rrb " 9-May-85 10:42")
                                                                              (* marks or unmarks a selection.)
    (COND
       ((POSITIONP ELT)                                                       (* handle positions {points} specially.)
        (MARKPOINT ELT SKW MARKBM))
       (T (SK.MARKHOTSPOTS ELT SKW MARKBM])
)

(READVARS-FROM-STRINGS '(POINTMARK SPOTMARKER)
       "({(READBITMAP)(7 7
       %"HB@@%"
       %"DD@@%"
       %"BH@@%"
       %"A@@@%"
       %"BH@@%"
       %"DD@@%"
       %"HB@@%")}  {(READBITMAP)(12 12
       %"@B@@%"
       %"@G@@%"
       %"@G@@%"
       %"@G@@%"
       %"CHN@%"
       %"GHO@%"
       %"CHN@%"
       %"@G@@%"
       %"@G@@%"
       %"@G@@%"
       %"@B@@%"
       %"@@@@%")})
       ")

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS POINTMARK SPOTMARKER)
)

(RPAQ POINTREADINGCURSOR (CURSORCREATE '
```



```
                                         'NIL 7 7))
```

;; hit detection functions.

```
(DEFINEQ

(SK.SELECT.ITEM
  [LAMBDA (WINDOW ITEMFLG SELITEMS OPERATION)                        (* rrb "10-Dec-85 17:01")

         (* selects allows the user to select one of the sketch elements from the sketch WINDOW.
         If ITEMFLG is non-NIL, it returns the item selected, otherwise it returns the position.
         If SELITEMS is given it is used as the items to be marked and selected from.
         Keeps control and probably shouldn't)

    (PROG (HOTSPOTCACHE NOW PREVIOUS OLDPOS)
          (COND
             (SELITEMS                                               (* create a cache for the items to select from)
                  (SETQ HOTSPOTCACHE (SK.ADD.HOTSPOTS.TO.CACHE SELITEMS NIL)))
             [(SK.HAS.SOME.HOTSPOTS (SETQ HOTSPOTCACHE (SK.HOTSPOT.CACHE.FOR.OPERATION WINDOW OPERATION]
             (T                                                      (* no items, don't do anything.)
                (RETURN)))
          (TOTOPW WINDOW)
          (SK.SHOWMARKS WINDOW HOTSPOTCACHE)
          (until (MOUSESTATE (NOT UP)))
          (COND
             ((NOT (LASTMOUSESTATE (OR LEFT MIDDLE)))                (* for now not interested in anything besides left and middle.)
              (SK.SHOWMARKS WINDOW HOTSPOTCACHE)
              (RETURN)))                                             (* note current item selection.)
          (SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE (SETQ OLDPOS (CURSORPOSITION NIL WINDOW))
                           (NULL ITEMFLG)))
      FLIP
                                                                     (* turn off old selection.)
          (SK.DESELECT.ELT PREVIOUS WINDOW)
          (SK.SELECT.ELT (SETQ PREVIOUS NOW)
                 WINDOW)
      LP                                                             (* wait for a button up or move out of region)
          (COND
             ((NOT (MOUSESTATE (OR LEFT MIDDLE)))                    (* button up, selected item if one)
              (SK.DESELECT.ELT PREVIOUS WINDOW)
              (SK.SHOWMARKS WINDOW HOTSPOTCACHE)
              (RETURN PREVIOUS))
             ([EQUAL PREVIOUS (SETQ NOW (IN.SKETCH.ELT? HOTSPOTCACHE (CURSORPOSITION NIL WINDOW OLDPOS)
                                               (NULL ITEMFLG]
              (GO LP))
             (T (GO FLIP])
```

(IN.SKETCH.ELT?

```
  [LAMBDA (CACHE POS PTFLG)                                          (* rrb "21-Feb-85 13:47")
                                                                     (* returns the first element that POS is on.)
    (PROG ((Y (fetch (POSITION YCOORD) of POS))
           (X (fetch (POSITION XCOORD) of POS))
           (BESTMEASURE 1000)
           PTLEFT PTRIGHT PTTOP PTBOTTOM BESTELT BESTX BESTY YDIF THISDIF)
          (SETQ PTLEFT (DIFFERENCE X SK.POINT.WIDTH))
          (SETQ PTRIGHT (PLUS X SK.POINT.WIDTH))
          (SETQ PTBOTTOM (DIFFERENCE Y SK.POINT.WIDTH))
          (SETQ PTTOP (PLUS Y SK.POINT.WIDTH))
          [for YBUCKET in CACHE when (ILEQ (CAR YBUCKET)
                                           PTTOP)
             do (COND
                   ((ILESSP (CAR YBUCKET)
                           PTBOTTOM)                                 (* stop when Y gets too small.)
                    (RETURN)))
                (SETQ YDIF (ABS (DIFFERENCE (CAR YBUCKET)
                                       Y)))
                (for XBUCKET in (CDR YBUCKET) when (ILEQ (CAR XBUCKET)
                                                         PTRIGHT)
                   do (COND
                         ((ILESSP (CAR XBUCKET)
                                 PTLEFT)                             (* stop when X gets too small.)
                          (RETURN)))
                      (COND
                         ((CDR XBUCKET)                              (* this bucket has entries)
                          [SETQ THISDIF (PLUS YDIF (ABS (DIFFERENCE (CAR XBUCKET)
                                                                X]
                          (COND
                             ((ILESSP THISDIF BESTMEASURE)
                              (SETQ BESTMEASURE THISDIF)
                              (COND
                                 (PTFLG (SETQ BESTX (CAR XBUCKET))
                                        (SETQ BESTY (CAR YBUCKET)))
                                 (T (SETQ BESTELT (CADR XBUCKET]
          (RETURN (COND
                     (PTFLG (AND BESTX (create POSITION
                                              XCOORD _ BESTX
                                              YCOORD _ BESTY)))
                     (T BESTELT])
```

(SK.MARK.HOTSPOT

```
  [LAMBDA (X Y WINDOW MARK)                                          (* rrb "29-Jan-85 15:45")

         (* marks a point in a window with a mark. The mark should be a bitmap.)
```

```
    (PROG ((MARKWIDTH (BITMAPWIDTH MARK))
           HALFWIDTH)
          (RETURN (BITBLT MARK 0 0 WINDOW (IDIFFERENCE X (SETQ HALFWIDTH (LRSH MARKWIDTH 1)))
                         (IDIFFERENCE Y HALFWIDTH)
                         MARKWIDTH MARKWIDTH 'INPUT 'INVERT])

(SK.MARK.POSITION
  [LAMBDA (PT WINDOW MARKBITMAP)                              (* rrb "20-Apr-85 18:47")
                                                              (* marks a place on the sketch window WINDOW.)
    (SK.MARK.HOTSPOT (fetch (POSITION XCOORD) of PT)
           (fetch (POSITION YCOORD) of PT)
           WINDOW MARKBITMAP])

(SK.SELECT.ELT
  [LAMBDA (ELT FIGW MARKBM)                                   (* rrb " 3-Oct-84 11:18")
                                                              (* selects an item from a figure window.)
                                                              (* for now just mark it.)
    (AND ELT (SK.MARK.SELECTION ELT FIGW MARKBM])

(SK.DESELECT.ELT
  [LAMBDA (ELT SKW MARKBM)                                    (* rrb " 9-May-85 10:32")
                                                              (* turns off the selection marking of an item from a figure
                                                              window.)
    (AND ELT (SK.MARK.SELECTION ELT SKW MARKBM])
)

(DECLARE%: EVAL@COMPILE

(RPAQQ SK.POINT.WIDTH 4)

(CONSTANTS (SK.POINT.WIDTH 4))
)

;; fns to support caching of hotspots.

(DEFINEQ

(SK.HOTSPOT.CACHE
  [LAMBDA (SKW)                                               (* rrb "29-Jan-85 14:23")
                                                              (* retrieve the hotspot cache associated with a sketch window.)
    (WINDOWPROP SKW 'HOTSPOT.CACHE])

(SK.HOTSPOT.CACHE.FOR.OPERATION
  [LAMBDA (VIEWER OPERATION)                                  (* rrb "10-Dec-85 16:59")

            (* returns the hotspot cache for the elements in a viewer that are not protected against OPERATION.)

    (PROG (SCRELTS)
          (RETURN (COND
                     ((AND OPERATION (bind PROTECTION for SCRELT in (SETQ SCRELTS (LOCALSPECS.FROM.VIEWER VIEWER))
                                        thereis                 (* look for any element that disallows the current operation)
                                              (SK.ELEMENT.PROTECTED? (fetch (SCREENELT GLOBALPART) of SCRELT)
                                                     OPERATION)))
                                                              (* compute special cache)
                      (SK.BUILD.CACHE SCRELTS OPERATION))
                     (T                                       (* use the cache of all elements.)
                        (SK.HOTSPOT.CACHE VIEWER])

(SK.BUILD.CACHE
  [LAMBDA (SCRELTS SKETCHOP)                                  (* rrb "11-Dec-85 11:10")
                                                              (* Builds a cache of the elements in SCRELTS that aren't
                                                              protected against SKETCHOP.)
    (PROG (CACHE)
          (for ELT in SCRELTS when (NOT (SK.ELEMENT.PROTECTED? (fetch (SCREENELT GLOBALPART) of ELT)
                                               SKETCHOP))
             do (SETQ CACHE (SK.ADD.HOTSPOTS.TO.CACHE1 ELT CACHE)))
          (RETURN CACHE])

(SK.ELEMENT.PROTECTED?
  [LAMBDA (GELT HOW)                                          (* rrb " 5-Dec-85 11:16")
                                                              (* determines if GELT is protected against the operation HOW)
    (PROG [(PROTECTIONLST (GETSKETCHELEMENTPROP GELT 'PROTECTION]
          (RETURN (OR (EQMEMB HOW PROTECTIONLST)
                      (AND (NEQ HOW 'COPYSELECT)
                           (OR (EQMEMB T PROTECTIONLST)
                               (EQMEMB 'FROZEN PROTECTIONLST])

(SK.HAS.SOME.HOTSPOTS
  [LAMBDA (HOTSPOTCACHE)                                        (* rrb "17-Oct-85 11:18")
                                                                (* return T if there is a selectable point in HOTSPOTCACHE.)
    (for BUCKET in HOTSPOTCACHE when (SOME (CDR BUCKET)
                                           (FUNCTION CDR))
       do (RETURN T])

(SK.SET.HOTSPOT.CACHE
  [LAMBDA (SKW NEWCACHE)                                        (* rrb "29-Jan-85 14:23")
                                                                (* stores the hotspot cache associated with a sketch window.)
    (WINDOWPROP SKW 'HOTSPOT.CACHE NEWCACHE])

(SK.CREATE.HOTSPOT.CACHE
  [LAMBDA (SKW)                                                 (* rrb " 4-Feb-85 14:18")
                                                                (* creates the cache of hotspot locations for a sketch window.)
    (SK.SET.HOTSPOT.CACHE SKW (SK.ADD.HOTSPOTS.TO.CACHE (LOCALSPECS.FROM.VIEWER SKW)
                                     NIL])

(SK.ELTS.FROM.HOTSPOT
  [LAMBDA (POSITION CACHE)                                      (* rrb "29-Jan-85 13:47")
                                                                (* returns a list of local elements that have POSITION as one of
    their hotspots.)

          (* a cache is an alist of alist with the top descriminator being the Y value and the second one being the X value.)

    (PROG (TMP)
          (RETURN (AND (SETQ TMP (SK.FIND.CACHE.BUCKET (fetch (POSITION YCOORD) of POSITION)
                                        CACHE))
                       (SK.FIND.CACHE.BUCKET (fetch (POSITION XCOORD) of POSITION)
                              TMP])

(SK.ADD.HOTSPOTS.TO.CACHE
  [LAMBDA (ELTS CACHE)                                          (* rrb " 3-Feb-85 14:36")
                                                                (* adds a collection of hotspots to a cache.)
    (for ELT in ELTS do (SETQ CACHE (SK.ADD.HOTSPOTS.TO.CACHE1 ELT CACHE)))
    CACHE])

(SK.ADD.HOTSPOTS.TO.CACHE1
  [LAMBDA (LOCALELT CACHE)                                      (* rrb "29-Jan-85 14:55")
                                                                (* adds an elements hotspots to the cache.)
    (for HOTSPOT in (fetch (SCREENELT HOTSPOTS) of LOCALELT) do (SETQ CACHE (SK.ADD.HOTSPOT.TO.CACHE HOTSPOT
                                                                                   LOCALELT CACHE)))
    CACHE])

(SK.ADD.HOTSPOT.TO.CACHE
  [LAMBDA (POSITION ELT CACHE)                                  (* rrb "16-Sep-86 12:45")
                                                                (* adds a hotspot to a cache.)

          (* a cache is an alist of alist with the top descriminator being the Y value and the second one being the X value.)

    (PROG ((Y (fetch (POSITION YCOORD) of POSITION))
           (X (fetch (POSITION XCOORD) of POSITION)))
          (RETURN (COND
                     [(NULL CACHE)
                      (LIST (LIST Y (LIST X ELT]
                     ((GREATERP Y (CAAR CACHE))                 (* this element goes first Splice it onto the front.)
                      (RPLACD CACHE (CONS (CAR CACHE)
                                          (CDR CACHE)))
                      (RPLACA CACHE (LIST Y (LIST X ELT)))
                      CACHE)
                     ((EQ (CAAR CACHE)
                          Y)
                      (SK.ADD.VALUE.TO.CACHE.BUCKET X ELT (CDAR CACHE))
                      CACHE)
                     (T [for TAIL on CACHE do [AND (CDR TAIL)
                                                   (COND
                                                      ((EQ (CAADR TAIL)
                                                           Y)
                                                       (SK.ADD.VALUE.TO.CACHE.BUCKET X ELT (CDADR TAIL))
                                                       (RETURN))
                                                      ((GREATERP Y (CAADR TAIL))
                                                       (RPLACD TAIL (CONS (LIST Y (LIST X ELT))
                                                                          (CDR TAIL)))
                                                       (RETURN]
                           finally (NCONC1 CACHE (LIST Y (LIST X ELT]
                        CACHE])

(SK.REMOVE.HOTSPOTS.FROM.CACHE
  [LAMBDA (ELTS CACHE)                                          (* rrb "29-Jan-85 14:04")
                                                                (* removes a collection of hotspots from a cache.)
    (for ELT in ELTS do (SETQ CACHE (SK.REMOVE.HOTSPOTS.FROM.CACHE1 ELT CACHE])
(SK.REMOVE.HOTSPOTS.FROM.CACHE1
  [LAMBDA (LOCALELT CACHE)                                  (* rrb "29-Jan-85 13:45")
                                                            (* removes an elements hotspots to the cache.)
    (for HOTSPOT in (fetch (SCREENELT HOTSPOTS) of LOCALELT) do (SK.REMOVE.HOTSPOT.FROM.CACHE HOTSPOT LOCALELT
                                                                       CACHE])

(SK.REMOVE.HOTSPOT.FROM.CACHE
  [LAMBDA (POSITION ELT CACHE)                              (* rrb "16-Sep-86 12:45")
                                                            (* removes a hotspot to a cache.)

          (* a cache is an alist of alist with the top descriminator being the Y value and the second one being the X value.)

    (SK.REMOVE.VALUE.FROM.CACHE.BUCKET (fetch (POSITION XCOORD) of POSITION)
           ELT
           (FASSOC (fetch (POSITION YCOORD) of POSITION)
                  CACHE])

(SK.REMOVE.VALUE.FROM.CACHE.BUCKET
  [LAMBDA (VAL ELT BUCKET)                                  (* rrb "16-Sep-86 12:45")
                                                            (* removes ELT from the list of elements stored on BUCKET
                                                            under the key VAL.)

          (* leaves the x and y of the bucket because it seems easier than removing it and it may be used again in the case of
          changing an element by deleting it then adding it again.)

    (for TAIL on (FASSOC VAL (CDR BUCKET)) do (AND (CDR TAIL)
                                                   (COND
                                                      ((EQ (CADR TAIL)
                                                           ELT)
                                                       (RPLACD TAIL (CDDR TAIL])

(SK.FIND.CACHE.BUCKET
  [LAMBDA (VALUE CACHE)                                     (* rrb "16-Sep-86 12:46")

          (* internal function for searching the caching Alists. Returns the bucket if there is one;
          quits when a value is larger than the one asked for.)

    (for TAIL on CACHE do (COND
                             ((EQ (CAAR TAIL)
                                  VALUE)
                              (RETURN (CDAR TAIL)))
                             ((GREATERP VALUE (CAAR TAIL))
                              (RETURN NIL])

(SK.ADD.VALUE.TO.CACHE.BUCKET
  [LAMBDA (VAL ELT ALIST)                                   (* rrb "16-Sep-86 12:46")
                                                            (* adds ELT to the list of elements stored on ALIST under the
                                                            key VAL.)
    (COND
       ((NULL ALIST)                                        (* shouldn't ever happen.)
        NIL)
       ((GREATERP VAL (CAAR ALIST))                         (* this element goes first Splice it onto the front.)
        (RPLACD ALIST (CONS (CAR ALIST)
                            (CDR ALIST)))
        (RPLACA ALIST (LIST VAL ELT)))
       ((EQ (CAAR ALIST)
            VAL)                                            (* add it to the end of the first list.)
        (NCONC1 (CAR ALIST)
               ELT))
       (T (for TAIL on ALIST do [AND (CDR TAIL)
                                     (COND
                                        ((EQ (CAADR TAIL)
                                             VAL)
                                         (NCONC1 (CADR TAIL)
                                                ELT)
                                         (RETURN ALIST))
                                        ((GREATERP VAL (CAADR TAIL))
                                         (RPLACD TAIL (CONS (LIST VAL ELT)
                                                            (CDR TAIL)))
                                         (RETURN ALIST]
             finally (NCONC1 ALIST (LIST VAL ELT])
)

;; grid stuff

(DEFINEQ

(SK.SET.GRID
  [LAMBDA (SKETCHW)                                         (* rrb "25-Oct-84 12:40")
                                                            (* switches from grided to non-grided or vice versa.)
    (COND
       ((WINDOWPROP SKETCHW 'USEGRID)
        (SK.TURN.GRID.OFF SKETCHW))
       (T (SK.TURN.GRID.ON SKETCHW])
```

(SK.DISPLAY.GRID

```
  [LAMBDA (SKETCHW)                                          (* rrb "23-Sep-86 11:30")
                                                             (* displays the current grid.)

    (COND
       ((WINDOWPROP SKETCHW 'USEGRID))
       (T                                                    (* grid was not being used, turn it on.)
          (SK.TURN.GRID.ON SKETCHW T)))
    (WINDOWPROP SKETCHW 'GRIDUP T)
    (SK.DISPLAY.GRID.POINTS SKETCHW])
```

(SK.DISPLAY.GRID.POINTS

```
  [LAMBDA (SKETCHW NEWFLG)                                   (* rrb "16-Jan-85 10:09")
    (SK.SHOW.GRID (SK.GRIDFACTOR SKETCHW)
           SKETCHW NEWFLG])
```

(SK.REMOVE.GRID.POINTS

```
  [LAMBDA (SKETCHW)                                          (* rrb "23-Sep-86 11:28")
                                                             (* removes the grid by calling redisplay with the gridup property
    removed.)
    (COND
       ([NOT (GREATERP 3.0 (FQUOTIENT (SK.GRIDFACTOR SKETCHW)
                                  (VIEWER.SCALE SKETCHW]     (* if grid factor is less than 3.0 the grid isn't displayed)
        (WINDOWPROP SKETCHW 'GRIDUP (PROG1 (WINDOWPROP SKETCHW 'GRIDUP NIL)
                                           (REDISPLAYW SKETCHW])
```

(SK.TAKE.DOWN.GRID

```
  [LAMBDA (SKETCHW)                                          (* rrb "23-Sep-86 11:26")
                                                             (* takes down the grid if it is up.)
    (COND
       ((WINDOWPROP SKETCHW 'GRIDUP NIL)
        (SK.REMOVE.GRID.POINTS SKETCHW])
```

(SK.SHOW.GRID

```
  [LAMBDA (GRID SKW NEWFLG)                                  (* DECLARATIONS%: FLOATING)
                                                             (* rrb "23-Sep-86 11:03")
                                                             (* puts a grid of size GRID onto SKW.)
    (PROG ((SCALE (VIEWER.SCALE SKW))
           (REGION (SKETCH.REGION.VIEWED SKW)))
          (COND
             ((GREATERP 3.0 (FQUOTIENT GRID SCALE))          (* would be every point or so)
              (STATUSPRINT SKW (CONCAT (COND
                                          (NEWFLG "New")
                                          (T "Current"))
                                      " grid has a position every "
                                      (FQUOTIENT GRID SCALE)
                                      " screen points."))
              NIL)
             (T

          (* make a horizontal bitmap that has the X pattern then blt it at the proper Y places.)

                [PROG ((WREG (DSPCLIPPINGREGION NIL SKW))
                       SCALEDWREG SCALEDWLEFT HORIZPATTERN WWIDTH WLEFT GRIDLEFT SKREGLEFT SKREGLIMIT)
                      (SETQ WWIDTH (fetch (REGION WIDTH) of WREG))
                      (SETQ WLEFT (fetch (REGION LEFT) of WREG))
                      (SETQ HORIZPATTERN (BITMAPCREATE WWIDTH 1))
                      (SETQ SCALEDWREG (UNSCALE.REGION WREG SCALE))
                      (SETQ SCALEDWLEFT (fetch (REGION LEFT) of SCALEDWREG))
                      (SETQ GRIDLEFT (NEAREST.ON.GRID SCALEDWLEFT GRID))
                                                             (* put limit calculation outside of the loop.)
                      (SETQ SKREGLIMIT (PLUS SCALEDWLEFT (fetch (REGION WIDTH) of SCALEDWREG)))
                      (for X from GRIDLEFT to SKREGLIMIT by GRID
                         do (BITMAPBIT HORIZPATTERN (FIXR (FQUOTIENT (DIFFERENCE X SCALEDWLEFT)
                                                                     SCALE))
                                  0 1))
                      (SETQ SKREGLIMIT (PLUS (fetch (REGION BOTTOM) of SCALEDWREG)
                                             (fetch (REGION HEIGHT) of SCALEDWREG)))
                      (for Y from (NEAREST.ON.GRID (fetch (REGION BOTTOM) of SCALEDWREG)
                                         GRID)
                         to SKREGLIMIT by GRID do (BITBLT HORIZPATTERN 0 0 SKW WLEFT (FIXR (FQUOTIENT Y SCALE))
                                                         WWIDTH 1 'INPUT 'PAINT]
                (COND
                   ((GREATERP (FQUOTIENT GRID SCALE)
                           (QUOTIENT (MIN (WINDOWPROP SKW 'HEIGHT)
                                          (WINDOWPROP SKW 'WIDTH))
                                  3))                        (* there aren't enough visible points so tell the user how far
                                                             apart they are.)
```

```
                    (STATUSPRINT SKW (CONCAT (COND
                                                  (NEWFLG "New")
                                                  (T "Current"))
                                              " grid has a position every "
                                              (FIXR (FQUOTIENT GRID SCALE))
                                              " screen points."])
```

(SK.GRIDFACTOR

```
  [LAMBDA (SKETCHW GRIDSIZE)                                          (* rrb "25-Oct-84 12:34")

                (* sets the grid factor of a window to GRIDSIZE. Returns the previous setting.
                The actual use of the grid is determined by (QUOTE USEGRID) property.)

    (COND
       ((NUMBERP GRIDSIZE)
        (WINDOWPROP SKETCHW 'GRIDFACTOR GRIDSIZE))
       (GRIDSIZE (\ILLEGAL.ARG GRIDSIZE)
              (WINDOWPROP SKETCHW 'GRIDFACTOR))
       (T (WINDOWPROP SKETCHW 'GRIDFACTOR])
```

(SK.TURN.GRID.ON

```
  [LAMBDA (SKETCHW QUIETFLG)                                          (* rrb "25-Oct-84 12:04")
                                                                      (* turns the grid on.)
    (COND
       ((WINDOWPROP SKETCHW 'USEGRID T)
        (OR QUIETFLG (STATUSPRINT SKETCHW "The grid was already in use."])
```

(SK.TURN.GRID.OFF

```
  [LAMBDA (SKETCHW)                                                   (* rrb "25-Oct-84 12:03")
                                                                      (* turns the grid off.)
    (COND
       ((WINDOWPROP SKETCHW 'USEGRID NIL)
        (SK.TAKE.DOWN.GRID SKETCHW))
       (T (STATUSPRINT SKETCHW "The grid was not is use."])
```

(SK.MAKE.GRID.LARGER

```
  [LAMBDA (SKETCHW)                                                   (* rrb "23-Sep-86 10:51")
                                                                      (* makes the grid larger. If the grid is off, it turns it on.)
    (SK.CHANGE.GRID [PROG ((NOWGRID (SK.GRIDFACTOR SKETCHW)))
                          (RETURN (COND
                                     ((EQP NOWGRID 0.5)               (* if going from half to one, switch to integer scale factors)
                                      1)
                                     (T (TIMES NOWGRID 2]
           SKETCHW])
```

(SK.MAKE.GRID.SMALLER

```
  [LAMBDA (SKETCHW)                                                   (* rrb "23-Sep-86 10:48")
                                                                      (* makes the grid smaller. If the grid is off, it turns it on.)
    (SK.CHANGE.GRID [PROG ((NOWGRID (SK.GRIDFACTOR SKETCHW)))
                          (RETURN (COND
                                     ((EQ NOWGRID 1)                  (* if going from one to half, switch from integer scale factors to
                                                                      floating)
                                      0.5)
                                     (T (QUOTIENT NOWGRID 2]
           SKETCHW])
```

(SK.CHANGE.GRID

```
  [LAMBDA (NEWGRID SKETCHW)                                           (* rrb " 1-Feb-85 15:52")

                (* changes the grid of a window. Turns the grid on if it isn't already on.)

    (SK.TURN.GRID.ON SKETCHW T)
    (AND (WINDOWPROP SKETCHW 'GRIDUP)
         (SK.REMOVE.GRID.POINTS SKETCHW))
    (SK.GRIDFACTOR SKETCHW NEWGRID)
    (AND (WINDOWPROP SKETCHW 'GRIDUP)
         (SK.DISPLAY.GRID.POINTS SKETCHW T])
```

(GRID.FACTOR1

```
  [LAMBDA (REALHEIGHT HEIGHTONSCREEN NPTS)                            (* rrb "19-Jun-84 17:26")

                (* returns the greatest power of two such that REALHEIGHT maps onto SCREENHEIGHT leaving at least NPTS per grid.)

    (LEASTPOWEROF2GT (FQUOTIENT (FTIMES NPTS REALHEIGHT)
                                HEIGHTONSCREEN])
```

(LEASTPOWEROF2GT

```
  [LAMBDA (FLOATP)                                                    (* rrb "23-Sep-86 10:57")
```

```
            (* returns the number which is the least power of two that is greater than FLOATP.)

    (PROG [(LOG2 (FQUOTIENT (LOG FLOATP)
                     (CONSTANT (LOG 2]
          (RETURN (COND
                     [(FGREATERP LOG2 0.0)                          (* keep the grid integer)
                      (FIX (COND
                              ((EQUAL LOG2 (FLOAT (FIX LOG2)))      (* special case of exact hit.)
                               (EXPT 2.0 (FIX LOG2)))
                              (T (EXPT 2.0 (ADD1 (FIX LOG2]
                     (T (EXPT 2.0 (FIX LOG2])
```

(**GREATESTPOWEROF2LT**

```
  [LAMBDA (FLOATP)                                                  (* rrb " 9-Jul-85 17:43")

            (* returns the number which is the greatest power of two that is less than FLOATP.)

    (PROG [(LOG2 (FQUOTIENT (LOG FLOATP)
                     (CONSTANT (LOG 2]
          (RETURN (COND
                     ((FGREATERP LOG2 0.0)
                      (EXPT 2.0 (FIX LOG2)))
                     ((EQUAL LOG2 (FLOAT (FIX LOG2)))               (* special case of exact hit.)
                      (EXPT 2.0 (FIX LOG2)))
                     (T (EXPT 2.0 (SUB1 (FIX LOG2])
```

(**SK.DEFAULT.GRIDFACTOR**

```
  [LAMBDA (SKETCHW)                                                 (* rrb "25-Nov-85 17:46")
                                                                    (* returns the default grid factor for a window.
                                                                    Starts at about a quarter inch.)
    (GRID.FACTOR1 (fetch (REGION HEIGHT) of (SKETCH.REGION.VIEWED SKETCHW))
           (WINDOWPROP SKETCHW 'HEIGHT)
           DEFAULTGRIDSIZE])
```

(**SK.PUT.ON.GRID**

```
  [LAMBDA (GPOSITION GRID)                                          (* rrb " 7-Feb-85 11:32")
                                                                    (* returns the grid point that is closest to GPOSITION.)
    (create POSITION
           XCOORD _ (NEAREST.ON.GRID (fetch (POSITION XCOORD) of GPOSITION)
                           GRID)
           YCOORD _ (NEAREST.ON.GRID (fetch (POSITION YCOORD) of GPOSITION)
                           GRID])
```

(**MAP.WINDOW.ONTO.GRID**

```
  [LAMBDA (X SCALE GRID)                                            (* rrb "20-Jun-84 16:53")
                                                                    (* maps from a window point onto the window point that is
                                                                    closest to GRID.)
    (FIXR (QUOTIENT (NEAREST.ON.GRID (TIMES X SCALE)
                           GRID)
                SCALE])
```

(**MAP.SCREEN.ONTO.GRID**

```
  [LAMBDA (X SCALE GRID WOFFSET)                                    (* rrb "20-Jun-84 16:22")

            (* maps a screen coordinate into the screen coordinate that is closest to the grid of a window with offset WOFFSET.)

    (COND
       ((OR (NOT GRID)
            (EQ GRID 0)
            (EQP GRID 0.0))
        X)
       (T (IPLUS (MAP.WINDOW.ONTO.GRID (IDIFFERENCE X WOFFSET)
                        SCALE GRID)
                WOFFSET])
```

(**MAP.GLOBAL.PT.ONTO.GRID**

```
  [LAMBDA (PT SKW)                                                  (* rrb " 7-Feb-85 11:33")

            (* If the grid is in use, maps from a point in global coordinates into the closest grid point in global coordinates.)

    (COND
       ((WINDOWPROP SKW 'USEGRID)
        (SK.PUT.ON.GRID PT (SK.GRIDFACTOR SKW)))
       (T PT])
```

(**MAP.GLOBAL.REGION.ONTO.GRID**

```
  [LAMBDA (GREGION SKW)                                             (* rrb "25-Jan-85 10:50")

            (* If the grid is in use, maps from a region in global coordinates into the closest larger region in global coordinates.)
```

```
    (COND
       [(WINDOWPROP SKW 'USEGRID)
        (PROG ((GRID (SK.GRIDFACTOR SKW))
               HALFGRID NEWLEFT NEWBOTTOM)
              (SETQ HALFGRID (QUOTIENT GRID 2.0))
              (RETURN (CREATEREGION (SETQ NEWLEFT (NEARREST.ON.GRID (DIFFERENCE (fetch (REGION LEFT) of GREGION)
                                                                                 HALFGRID)
                                                                   GRID))
                             (SETQ NEWBOTTOM (NEAREST.ON.GRID (DIFFERENCE (fetch (REGION BOTTOM) of GREGION)
                                                                          HALFGRID)
                                                              GRID))
                             (DIFFERENCE (NEAREST.ON.GRID (PLUS (fetch (REGION RIGHT) of GREGION)
                                                                HALFGRID)
                                                          GRID)
                                    NEWLEFT)
                             (DIFFERENCE (NEAREST.ON.GRID (PLUS (fetch (REGION TOP) of GREGION)
                                                                HALFGRID)
                                                          GRID)
                                    NEWBOTTOM]
       (T GREGION])
```

(MAP.WINDOW.POINT.ONTO.GLOBAL.GRID

```
  [LAMBDA (PT SCALE GRID)                                        (* rrb " 1-Feb-85 14:08")

          (* maps from a point in window coordinates into the closest grid point in global coordinates.)

    (create POSITION
           XCOORD _ (MAP.WINDOW.ONTO.GLOBAL.GRID (fetch (POSITION XCOORD) of PT)
                           SCALE GRID)
           YCOORD _ (MAP.WINDOW.ONTO.GLOBAL.GRID (fetch (POSITION YCOORD) of PT)
                           SCALE GRID])
```

(MAP.WINDOW.ONTO.GLOBAL.GRID

```
  [LAMBDA (X SCALE GRID)                                         (* rrb " 1-Feb-85 14:08")
                                                                 (* maps from a window point onto the window point that is
                                                                 closest to GRID.)
    (NEAREST.ON.GRID (TIMES X SCALE)
           GRID])
```

(SK.UPDATE.GRIDFACTOR

```
  [LAMBDA (SKW OLDSCALE)                                         (* rrb "25-Nov-85 17:46")
                                                                 (* determines the size of the grid for the newly scaled window.)
    (PROG ((OLDGRID (SK.GRIDFACTOR SKW))
           X)
          (SK.GRIDFACTOR SKW (GRID.FACTOR1 (fetch (REGION HEIGHT) of (SKETCH.REGION.VIEWED SKW))
                                    (WINDOWPROP SKW 'HEIGHT)
                                    (IMIN DEFAULTMAXGRIDSIZE (FQUOTIENT OLDGRID OLDSCALE])
```

(SK.MAP.FROM.WINDOW.TO.GLOBAL.GRID

```
  [LAMBDA (POSITION SKETCHW)                                     (* rrb "11-Jul-86 15:56")

          (* maps from a position in a window to the corresponding global position taking into account the grid if it is in use.)

    (COND
       ((WINDOWPROP SKETCHW 'USEGRID)
        (MAP.WINDOW.POINT.ONTO.GLOBAL.GRID POSITION (VIEWER.SCALE SKETCHW)
               (SK.GRIDFACTOR SKETCHW)))
       (T (SK.UNSCALE.POSITION.FROM.VIEWER POSITION (VIEWER.SCALE SKETCHW])
```

(SK.MAP.INPUT.PT.TO.GLOBAL

```
  [LAMBDA (POSSPEC SKETCHW)                                      (* rrb "11-Jul-86 15:52")

          (* maps from a position ala GETSKWPOSITION in a window to the corresponding global position
          (POSITION is a list of (GRIDON? position)))

    (AND POSSPEC (COND
                    ((EQ (fetch (INPUTPT INPUT.ONGRID?) of POSSPEC)
                         'GLOBAL)
                     (fetch (INPUTPT INPUT.GLOBALPOSITION) of POSSPEC))
                    ((fetch (INPUTPT INPUT.ONGRID?) of POSSPEC)
                     (MAP.WINDOW.POINT.ONTO.GLOBAL.GRID (fetch (INPUTPT INPUT.POSITION) of POSSPEC)
                            (VIEWER.SCALE SKETCHW)
                            (SK.GRIDFACTOR SKETCHW)))
                    (T

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

                       (SK.MAP.FROM.WINDOW.TO.NEAREST.GRID (fetch (INPUTPT INPUT.POSITION) of POSSPEC)
                              (VIEWER.SCALE SKETCHW)
                              T])
```

```
(SK.MAP.FROM.WINDOW.TO.NEAREST.GRID
  [LAMBDA (POSITION SCALE NOMOVEFLG)                                 (* rrb " 3-Oct-85 14:16")

            (* maps from a point in a window to the closest grid position in the global space that has a distance between the points of
            less than 1.0)

    (PROG [(GRID (COND
                   (NOMOVEFLG

            (* if NOMOVEFLG is on, use a grid small enough that the mapping into and out of coordinate space will leave POSITION
            unchanged. For most uses, this is too fine.)

                          (GREATESTPOWEROF2LT SCALE))
                   (T (LEASTPOWEROF2GT (TIMES SCALE 2]
          (RETURN (create POSITION
                         XCOORD _ (NEAREST.ON.GRID (TIMES (fetch (POSITION XCOORD) of POSITION)
                                                          SCALE)
                                         GRID)
                         YCOORD _ (NEAREST.ON.GRID (TIMES (fetch (POSITION YCOORD) of POSITION)
                                                          SCALE)
                                         GRID])
)

(RPAQ? DEFAULTGRIDSIZE 8)

(RPAQ? DEFAULTMINGRIDSIZE 4)

(RPAQ? DEFAULTMAXGRIDSIZE 32)
```

;; history and undo stuff

```
(DEFINEQ

(SK.ADD.HISTEVENT
  [LAMBDA (EVENTTYPE EVENTARGS SKETCHW)                              (* rrb "11-Jan-85 18:04")
                                                                     (* puts a history event on a sketch window.)
                                                                     (* trim to a given length)
    (PROG [(HISTLST (WINDOWPROP SKETCHW 'SKETCHHISTORY]
          (WINDOWPROP SKETCHW 'SKETCHHISTORY (CONS (create SKHISTEVENT
                                                          EVENTTYPE _ EVENTTYPE
                                                          EVENTARGS _ EVENTARGS)
                                                   (COND
                                                      ((GREATERP SKETCH.#.UNDO.ITEMS (LENGTH HISTLST))
                                                                     (* there is room for one more)
                                                       HISTLST)
                                                      (T (REMOVE.LAST HISTLST])

(SK.SEL.AND.UNDO
  [LAMBDA (SKW)                                                      (* rrb " 5-Dec-85 17:18")
                                                                     (* gives the user a choice of past events to undo.)
    (SKED.CLEAR.SELECTION SKW)
    (PROG [EVENT UNDOFN (HISTLST (WINDOWPROP SKW 'SKETCHHISTORY]
          (COND
             ((NULL HISTLST)
              (STATUSPRINT SKW "Nothing to undo.")
              (RETURN)))
          (COND
             ([SETQ EVENT (\CURSOR.IN.MIDDLE.MENU (create MENU
                                                         ITEMS _ (for EVENT in HISTLST
                                                                    collect (LIST (SK.UNDO.NAME EVENT)
                                                                                  EVENT))
                                                         WHENSELECTEDFN _ (FUNCTION CADR)
                                                         TITLE _ "Select event to undo"
                                                         WHENHELDFN _ (FUNCTION (LAMBDA (ITEM MENU BUTTON)
                                                                                  (PROMPTPRINT "Will undo this
                                                                                         event."]
              (COND
                 ((fetch (SKHISTEVENT UNDONE?) of EVENT)

            (* can't undo already undone event. They are included in the menu to provide session continuity.)

                  (STATUSPRINT SKW "That event has already been undone.")
                  (RETURN NIL))
                 ([NULL (SETQ UNDOFN (fetch (SKEVENTTYPE SKUNDOFN) of (SKEVENTTYPEFNS (fetch (SKHISTEVENT
                                                                                                EVENTTYPE)
                                                                                          of EVENT]
                  (STATUSPRINT SKW "Can't undo that event.")
                  (RETURN NIL)))
              (COND
                 ((APPLY* UNDOFN (fetch (SKHISTEVENT EVENTARGS) of EVENT)
                         SKW EVENT)                                  (* only add to history list if something happened.)
                  (replace (SKHISTEVENT UNDONE?) of EVENT with T)
                  (SK.ADD.HISTEVENT 'UNDO EVENT SKW))
                 ((NOT (EQ UNDOFN 'SK.UNDO.UNDO))
                  (STATUSPRINT SKW "Element subsequently modified, can't undo"])
```

```
(SK.UNDO.LAST
  [LAMBDA (SKW)                                                   (* rrb " 5-Dec-85 17:19")
                                                                  (* undoes the first not yet undone history event.)
    (SKED.CLEAR.SELECTION SKW)
    (PROG [EVENT UNDOFN (HISTLST (WINDOWPROP SKW 'SKETCHHISTORY]
          (COND
             ((NULL HISTLST)
              (STATUSPRINT SKW "Nothing to undo.")
              (RETURN)))
          (COND
             [(SETQ EVENT (for HISTEVENT in HISTLST when [AND (NOT (EQ (fetch (SKHISTEVENT EVENTTYPE) of HISTEVENT
                                                                             )
                                                                           'UNDO))
                                                                (NOT (fetch (SKHISTEVENT UNDONE?) of HISTEVENT))
                                                                (SETQ UNDOFN (fetch (SKEVENTTYPE SKUNDOFN)
                                                                                of (SKEVENTTYPEFNS
                                                                                    (fetch (SKHISTEVENT EVENTTYPE)
                                                                                       of HISTEVENT]
                             do (RETURN HISTEVENT)))
              (COND
                 ((APPLY* UNDOFN (fetch (SKHISTEVENT EVENTARGS) of EVENT)
                         SKW EVENT)                               (* only add to history list if something happened.)
                  (STATUSPRINT SKW (SK.UNDO.NAME EVENT)
                         " event undone.")
                  (replace (SKHISTEVENT UNDONE?) of EVENT with T)
                  (SK.ADD.HISTEVENT 'UNDO EVENT SKW))
                 ((NOT (EQ UNDOFN 'SK.UNDO.UNDO))
                  (STATUSPRINT SKW "Element subsequently modified, can't undo"]
             (T (STATUSPRINT SKW "
                       " "All events have been undone.  Use the '?UNDO' subcommand to undo an UNDO command."])

(SK.UNDO.NAME
  [LAMBDA (HISTEVENT)                                             (* rrb "17-Apr-84 11:27")
                                                                  (* returns the menu label for HISTEVENT.)
    (APPLY* (fetch (SKEVENTTYPE SKUNDONAMEFN) of (SKEVENTTYPEFNS (fetch (SKHISTEVENT EVENTTYPE) of HISTEVENT)))
           HISTEVENT])

(SKEVENTTYPEFNS
  [LAMBDA (EVENTTYPE)                                             (* rrb "17-Apr-84 11:02")
                                                                  (* returns the list of type related functions associated with
                                                                  EVENTTYPE.)
    (GETPROP EVENTTYPE 'EVENTFNS])

(SK.TYPE.OF.FIRST.ARG
  [LAMBDA (HISTEVENT NOMARKUNDOFLG)                               (* rrb "11-Dec-85 15:20")

          (* returns a name suitable for a menu label for an history event by combining the event name with the type of its arg.)

    (PROG ((ARGS (fetch (SKHISTEVENT EVENTARGS) of HISTEVENT))
           (TYPE (fetch (SKHISTEVENT EVENTTYPE) of HISTEVENT)))
          (RETURN (CONCAT (COND
                             ((AND (NULL NOMARKUNDOFLG)
                                   (fetch (SKHISTEVENT UNDONE?) of HISTEVENT))
                              "*")
                             (T " "))
                         TYPE " " (COND
                                     ((CDR ARGS)
                                      '"a group")
                                     (T (SELECTQ TYPE
                                            ((GROUP UNGROUP FREEZE UNFREEZE)
                                                 "")
                                            ((MOVE CHANGE)
                                                 (SK.LABEL.FROM.TYPE (fetch (GLOBALPART GTYPE)
                                                                        of (CAAR ARGS))))
                                            (SK.LABEL.FROM.TYPE (fetch (GLOBALPART GTYPE) of (CAR ARGS])
)

(DEFINEQ

(SK.DELETE.UNDO
  [LAMBDA (EVENTARGS SKW)                                         (* rrb "11-Sep-84 14:57")
                                                                  (* undoes a delete event)
    (PROG (CHANGED?)
          [for GELT in EVENTARGS do (COND
                                       ((SK.ADD.ELEMENT GELT SKW)
                                        (SETQ CHANGED? T]
          (RETURN CHANGED?])

(SK.ADD.UNDO
  [LAMBDA (EVENTARGS SKW)                                         (* rrb "30-Dec-85 16:18")
```

```
                                                                          (* undoes an add event)
    (SK.DELETE.ELEMENT2 EVENTARGS SKW 'DON'T])

)

(DEFINEQ

(SK.CHANGE.UNDO
  [LAMBDA (EVENTARGS SKW)                                                 (* rrb "24-Sep-86 17:09")
                                                                          (* undoes a change event)

          (* the args for a change event are the old {previous} global part of the element and the new global part of the element, the
          property that was changed, the new value and the old value.)

    (PROG [CHANGED? NOWELT PREVELT (WHENCHANGEDFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                                         'WHENCHANGEDFN]
          [for EVENT in EVENTARGS do (SETQ NOWELT (fetch (SKHISTORYCHANGESPEC NEWELT) of EVENT))
                                     (SETQ PREVELT (fetch (SKHISTORYCHANGESPEC OLDELT) of EVENT))
                                                                          (* apply the whenchangedfn if the element is still in the sketch.)
                                     (COND
                                        [(AND WHENCHANGEDFN (SK.ELT.IN.SKETCH? NOWELT SKW)
                                              (EQ (APPLY* WHENCHANGEDFN SKW NOWELT (fetch (SKHISTORYCHANGESPEC
                                                                                           PROPERTY)
                                                                                      of EVENT)
                                                          (fetch (SKHISTORYCHANGESPEC OLDVALUE) of EVENT)
                                                          (fetch (SKHISTORYCHANGESPEC NEWVALUE) of EVENT))
                                                  'DON'T]
                                        ((SK.UPDATE.ELEMENT NOWELT PREVELT SKW NIL T)
                                         (SETQ CHANGED? T]
          (RETURN CHANGED?])

(SK.ELT.IN.SKETCH?
  [LAMBDA (ELEMENT SKETCH)                                                (* determines if an element is in a sketch.)
    (MEMBER ELEMENT (SKETCH.ELEMENTS.OF.SKETCH SKETCH])

(SK.CHANGE.REDO
  [LAMBDA (EVENTARGS SKW)                                                 (* rrb "24-Sep-86 17:10")
                                                                          (* redoes a change event)
    (PROG [CHANGED? NEWELT OLDELT (WHENCHANGEDFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                                        'WHENCHANGEDFN]
          [for EVENT in EVENTARGS do (SETQ NEWELT (fetch (SKHISTORYCHANGESPEC NEWELT) of EVENT))
                                     (SETQ OLDELT (fetch (SKHISTORYCHANGESPEC OLDELT) of EVENT))
                                                                          (* apply the whenchangedfn if the element is still in the sketch.)
                                     (COND
                                        [(AND WHENCHANGEDFN (SK.ELT.IN.SKETCH? OLDELT SKW)
                                              (EQ (APPLY* WHENCHANGEDFN SKW OLDELT (fetch (SKHISTORYCHANGESPEC
                                                                                           PROPERTY)
                                                                                      of EVENT)
                                                          (fetch (SKHISTORYCHANGESPEC NEWVALUE) of EVENT)
                                                          (fetch (SKHISTORYCHANGESPEC OLDVALUE) of EVENT))
                                                  'DON'T]
                                        ((SK.UPDATE.ELEMENT OLDELT NEWELT SKW NIL T)
                                         (SETQ CHANGED? T]
          (OR CHANGED? (STATUSPRINT SKW "That sketch element has been changed by something else, can't redo."])

(SK.MOVE.UNDO
  [LAMBDA (EVENTARGS SKW)                                                 (* rrb "24-Sep-86 17:10")
                                                                          (* undoes a move event)

          (* the args for a move event are the old {previous} global part of the element and the new global part of the element, and the
          amount of the move.)

    (PROG [CHANGED? NOWELT PREVELT (WHENMOVEDFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                                       'WHENMOVEDFN]
          [for EVENT in EVENTARGS do (SETQ NOWELT (CADR EVENT))
                                     (SETQ PREVELT (CAR EVENT))     (* apply the WHENMOVEDFN if the element is still in the
                                                                          sketch.)
                                     (COND
                                        [(AND WHENMOVEDFN (SK.ELT.IN.SKETCH? NOWELT SKW)
                                              (EQ (APPLY* WHENMOVEDFN SKW (CONS T NOWELT)
                                                          (CADDR EVENT))
                                                  'DON'T]
                                        ((SK.UPDATE.ELEMENT NOWELT PREVELT SKW NIL T)
                                         (SETQ CHANGED? T]
          (RETURN CHANGED?])

(SK.MOVE.REDO
  [LAMBDA (EVENTARGS SKW)                                                 (* rrb "24-Sep-86 17:10")
                                                                          (* redoes a move event)
    (PROG [CHANGED? NEWELT OLDELT (WHENMOVEDFN (GETSKETCHPROP (INSURE.SKETCH SKW)
                                                      'WHENMOVEDFN]
          [for EVENT in EVENTARGS do (SETQ NEWELT (CADR EVENT))
                                     (SETQ OLDELT (CAR EVENT))      (* apply the WHENMOVEDFN if the element is still in the
                                                                          sketch.)
                                        (COND
                                           [(AND WHENMOVEDFN (SK.ELT.IN.SKETCH? OLDELT SKW)
                                                 (EQ (APPLY* WHENMOVEDFN SKW OLDELT (CADDR EVENT))
                                                     'DON'T]
                                           ((SK.UPDATE.ELEMENT OLDELT NEWELT SKW NIL T)
                                            (SETQ CHANGED? T]
          (OR CHANGED? (STATUSPRINT SKW "That sketch element has been changed by something else, can't redo."])
)

(DEFINEQ

(SK.UNDO.UNDO
  [LAMBDA (UNDONEEVENT SKW THISEVENT)                               (* rrb "18-Apr-84 15:32")
                                                                    (* undoes an UNDO event by calling the REDO fn of that event
                                                                    type.)
    (PROG (REDOFN)
          (COND
             ([SETQ REDOFN (fetch (SKEVENTTYPE SKREDOFN) of (SKEVENTTYPEFNS (fetch (SKHISTEVENT EVENTTYPE)
                                                                                   of UNDONEEVENT]
              (APPLY* REDOFN (fetch (SKHISTEVENT EVENTARGS) of UNDONEEVENT)
                     SKW)
              (replace (SKHISTEVENT UNDONE?) of UNDONEEVENT with NIL)
                                                                    (* remove the undo event from the history list.)
              (WINDOWDELPROP SKW 'SKETCHHISTORY THISEVENT))
             (T (STATUSPRINT SKW "Can't undo that event.")))

          (* always return NIL so the undoing of an undo event won't be added as an event.)

          (RETURN NIL])

(SK.UNDO.MENULABEL
  [LAMBDA (UNDOEVENT)                                               (* rrb "18-Sep-84 11:53")

          (* returns a name suitable for a menu label for an UNDO history event by combining the event name with the type of its arg.)

    (CONCAT "undo" (SK.TYPE.OF.FIRST.ARG (fetch (SKHISTEVENT EVENTARGS) of UNDOEVENT)
                          T])

(SK.LABEL.FROM.TYPE
  [LAMBDA (SKELEMENTTYPE)                                           (* rrb " 4-Jun-85 13:40")

          (* takes a type name and returns the label for it. These two are different because the names changed since the first sketchs
          were made.)

    (SELECTQ SKELEMENTTYPE
         (WIRE 'LINE)
         (OPENCURVE 'CURVE)
         (CLOSEDWIRE 'POLYGON)
         SKELEMENTTYPE])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SKHISTEVENT (EVENTTYPE EVENTARGS UNDONE?))

(RECORD SKEVENTTYPE (SKUNDOFN SKUNDONAMEFN SKREDOFN))
)
)

(RPAQ? SKETCH.#.UNDO.ITEMS 30)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCH.#.UNDO.ITEMS)
)

(PUTPROPS ADD EVENTFNS (SK.ADD.UNDO SK.TYPE.OF.FIRST.ARG SK.DELETE.UNDO))

(PUTPROPS DELETE EVENTFNS (SK.DELETE.UNDO SK.TYPE.OF.FIRST.ARG SK.ADD.UNDO))

(PUTPROPS CHANGE EVENTFNS (SK.CHANGE.UNDO SK.TYPE.OF.FIRST.ARG SK.CHANGE.REDO))

(PUTPROPS UNDO EVENTFNS (SK.UNDO.UNDO SK.UNDO.MENULABEL SHOULDNT))

(PUTPROPS MOVE EVENTFNS (SK.MOVE.UNDO SK.TYPE.OF.FIRST.ARG SK.MOVE.REDO))

(PUTPROPS COPY EVENTFNS (SK.ADD.UNDO SK.TYPE.OF.FIRST.ARG SK.DELETE.UNDO))

;; functions for displaying the global coordinate space values.

(DEFINEQ
(SHOW.GLOBAL.COORDS
  [LAMBDA (XCOORD YCOORD W)                                  (* rrb " 5-Jun-85 18:30")
                                                             (* converts to global coordinates and displays it in W)
    (DSPRESET W)
    (COND
       ((AND (EQP XCOORD (FIX XCOORD))
             (EQP YCOORD (FIX YCOORD)))
        (printout W .F6.0 XCOORD " x" "  " T .F6.0 YCOORD " y" "  "))
       (T (printout W .F8.2 XCOORD " x" "  " T .F8.2 YCOORD " y" "  "])

(LOCATOR.CLOSEFN
  [LAMBDA (GCOORDW)                                          (* rrb " 7-May-85 09:41")

            (* close function for a window that is keeping track of the global coordinate system.
            It breaks the link to itself.)

    (DETACHWINDOW GCOORDW])

(SKETCHW.FROM.LOCATOR
  [LAMBDA (GCOORDW)                                          (* rrb " 7-May-85 09:40")
                                                             (* returns the active window if any that points to GCOORDW)
    (for W in (ACTIVEWINDOWS) when (MEMB GCOORDW (ATTACHEDWINDOWS W)) do (RETURN W])

(SKETCHW.UPDATE.LOCATORS
  [LAMBDA (W)                                                (* rrb " 7-May-85 10:06")

            (* a cursor moved function for a sketch that shows the coordinates cursor in global coordinates.)

    (AND (INSIDEP (DSPCLIPPINGREGION NIL W)
                (LASTMOUSEX W)
                (LASTMOUSEY W))
         (for LOCATOR in (ATTACHEDWINDOWS W) when (MEMB (FUNCTION LOCATOR.CLOSEFN)
                                                        (WINDOWPROP LOCATOR 'CLOSEFN))
            do (LOCATOR.UPDATE LOCATOR W])

(LOCATOR.UPDATE
  [LAMBDA (LOCATORW SKW)                                     (* rrb "22-May-85 11:09")
                                                             (* updates the position of the locator coordinates.)

            (* there are three kinds of locators%: real coordinate, gridded real coordinates and latitude longitude, although lat lon has
            been deimplemented.)

    (SELECTQ (WINDOWPROP LOCATORW 'LOCATORTYPE)
         (GLOBALCOORD (UPDATE.GLOBALCOORD.LOCATOR LOCATORW SKW))
         (GLOBALGRIDDEDCOORD
              (UPDATE.GLOBAL.GRIDDED.COORD.LOCATOR LOCATORW SKW))
         (LATLON (UPDATE.LATLON.LOCATOR LOCATORW SKW))
         (SHOULDNT])

(UPDATE.GLOBAL.LOCATOR
  [LAMBDA (SKETCHW)                                          (* rrb "19-APR-83 14:19")
                                                             (* checks to see if the latitude longitude display needs to be
                                                             updated.)
    (COND
       ([OR (AND (NEQ SKETCHW.LASTCURSORPTX (SETQ SKETCHW.LASTCURSORPTX (LASTMOUSEX SKETCHW)))
                 (SETQ SKETCHW.LASTCURSORPTY (LASTMOUSEY SKETCHW)))
            (NEQ SKETCHW.LASTCURSORPTY (SETQ SKETCHW.LASTCURSORPTY (LASTMOUSEY SKETCHW]
                                                             (* call it if either point has changed.)
        (SKETCHW.UPDATE.LOCATORS SKETCHW])

(UPDATE.GLOBALCOORD.LOCATOR
  [LAMBDA (GCOORDW W)                                        (* rrb "11-Jul-86 15:52")

            (* a cursor moved function for a map that shows the coordinates cursor in global coordinates.)

    (PROG (SCALE)
          (OR GCOORDW (RETURN))
          (OR (SETQ SCALE (VIEWER.SCALE W))
              (RETURN))
          (SHOW.GLOBAL.COORDS (UNSCALE (LASTMOUSEX W)
                                       SCALE)
                 (UNSCALE (LASTMOUSEY W)
                        SCALE)
                 GCOORDW])

(ADD.GLOBAL.DISPLAY
  [LAMBDA (SKW TYPE)                                         (* rrb "28-Aug-85 11:10")

            (* creates a locator which gives the coordinates of the cursor in SKW in global coordinates.)
    (PROG [(LOCATOR (CREATE.GLOBAL.DISPLAYER (FONTCREATE BOLDFONT)
                          (COND
                             ((EQ TYPE 'GRID)
                              "cursor grid location")
                             (T "cursor location in sketch"]
          (ATTACHWINDOW LOCATOR SKW 'BOTTOM 'RIGHT 'LOCALCLOSE)
          [WINDOWPROP LOCATOR 'LOCATORTYPE (COND
                                               ((EQ TYPE 'GRID)
                                                'GLOBALGRIDDEDCOORD)
                                               (T 'GLOBALCOORD]
          (WINDOWPROP SKW 'CURSORMOVEDFN (FUNCTION SKETCHW.UPDATE.LOCATORS))
          (RETURN LOCATOR])
```

(**ADD.GLOBAL.GRIDDED.DISPLAY**

```
  [LAMBDA (SKW)                                          (* adds a locator that shows the nearest grid location.)
    (ADD.GLOBAL.DISPLAY SKW 'GRID])
```

(**CREATE.GLOBAL.DISPLAYER**

```
  [LAMBDA (FONT TITLE)                                   (* rrb " 7-May-85 09:59")
                                                         (* creates a window for displaying latitude longitude.)
    (PROG ((GCOORDW (CREATEW (CREATEREGION 0 0 (WIDTHIFWINDOW (STRINGWIDTH "11111111.1111    " FONT))
                                    (HEIGHTIFWINDOW (ITIMES 2 (FONTPROP FONT 'HEIGHT))
                                           T))
                          (OR TITLE "Real Coordinates")
                          NIL T)))

          (* extra space on stringwidth is to allow for the fact that printout translates into PRIN1 rather than PRIN3.)

          (DSPFONT FONT GCOORDW)
          (DSPRESET GCOORDW)                             (* reset its coordinates to the upper left)
          (WINDOWPROP GCOORDW 'CLOSEFN (FUNCTION LOCATOR.CLOSEFN))
          (RETURN GCOORDW])
```

(**UPDATE.GLOBAL.GRIDDED.COORD.LOCATOR**

```
  [LAMBDA (GCOORDW W)                                    (* rrb "11-Jul-86 15:52")

          (* a cursor moved function for a map that shows the coordinates cursor in global coordinates.)

    (PROG (SCALE)
          (OR GCOORDW (RETURN))
          (OR (SETQ SCALE (VIEWER.SCALE W))
              (RETURN))
          (COND
             [(WINDOWPROP W 'USEGRID)
              (PROG ((GRID (SK.GRIDFACTOR W))
                     XGRID YGRID)
                    (SETQ YGRID (MAP.WINDOW.ONTO.GLOBAL.GRID (LASTMOUSEY W)
                                       SCALE GRID))
                    (COND
                       ([OR [NOT (EQP (SETQ XGRID (MAP.WINDOW.ONTO.GLOBAL.GRID (LASTMOUSEX W)
                                                        SCALE GRID))
                                      (WINDOWPROP GCOORDW 'XCOORD]
                            (NOT (EQP YGRID (WINDOWPROP GCOORDW 'YCOORD]

          (* only update if one of the values has changed. This is done here but not in the ungridded case because it is handled by the
          cursor moved fn.)

                        (WINDOWPROP GCOORDW 'XCOORD XGRID)
                        (WINDOWPROP GCOORDW 'YCOORD YGRID)
                        (SHOW.GLOBAL.COORDS XGRID YGRID GCOORDW]
             (T (SHOW.GLOBAL.COORDS (UNSCALE (LASTMOUSEX W)
                                            SCALE)
                       (UNSCALE (LASTMOUSEY W)
                              SCALE)
                       GCOORDW])
)

(RPAQQ SKETCHW.LASTCURSORPTX 0)

(RPAQQ SKETCHW.LASTCURSORY 0)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS SKETCHW.LASTCURSORPTX SKETCHW.LASTCURSORPTY)
)
```

;; fns for reading colors

```
(DEFINEQ
```

(**DISPLAYREADCOLORHLSLEVELS**

```
  [LAMBDA (HLS WIN)                                      (* rrb "17-Jul-85 15:10")
```

```
                                                                      (* displays a hue lightness saturation triple in the color reading
                                                                      window.)
    (PROG (LEVEL)
          (DISPLAYREADCOLORLEVEL (SETQ LEVEL (HLSLEVEL HLS 'HUE))
                 (LEVELFROMHLSVALUE 'HUE LEVEL)
                 HUEREGION WIN)
          (DISPLAYREADCOLORLEVEL (SETQ LEVEL (HLSLEVEL HLS 'LIGHTNESS))
                 (LEVELFROMHLSVALUE 'LIGHTNESS LEVEL)
                 LIGHTNESSREGION WIN)
          (DISPLAYREADCOLORLEVEL (SETQ LEVEL (HLSLEVEL HLS 'SATURATION))
                 (LEVELFROMHLSVALUE 'SATURATION LEVEL)
                 SATURATIONREGION WIN])
```

(DISPLAYREADCOLORLEVEL

```
  [LAMBDA (PRINTLEVEL BARLEVEL REGION WINDOW)                         ; Edited 12-Jun-90 15:14 by mitani
                                                                      (* displays the value of a primary color in a color bar region.)
    (COND
       ((FIXP PRINTLEVEL)
        (MOVETO (DIFFERENCE (fetch (REGION LEFT) of REGION)
                       4)
               VALBTM WINDOW)
        (PRIN1 PRINTLEVEL WINDOW)                                     (* overstrike extra digits in case the old value was larger.)
        (PRIN1 "  " WINDOW))
       (T                                                             (* floating point values)
          (MOVETO (DIFFERENCE (fetch (REGION LEFT) of REGION)
                         10)
                 VALBTM WINDOW)
          (printout WINDOW .F5.3 PRINTLEVEL)))
    (FILLINREGION REGION BARLEVEL GRAYSHADE WINDOW])
```

(DRAWREADCOLORBOX

```
  [LAMBDA (TITLELEFT TITLE WINDOW)                                    (* rrb "17-Jul-85 14:20")

          (* draws the box and title for a display bar for an rgb or hls quantity.
          Returns a dotted pair of the region the box occuppied and the left most position printed in.)

    (PROG (XPOS REGION)
          (MOVETO TITLELEFT 4 WINDOW)
          (SETQ XPOS (DSPXPOSITION NIL WINDOW))
          (PRIN1 TITLE WINDOW)
          (OUTLINEREGION (SETQ REGION (create REGION
                                              LEFT _ (CENTEREDLEFT 10 XPOS (SETQ XPOS (DSPXPOSITION NIL WINDOW)))
                                              BOTTOM _ (PLUS 4 (FONTPROP WIN 'HEIGHT))
                                              WIDTH _ 10
                                              HEIGHT _ 256))
                 2 NIL WINDOW)
          (RETURN (CONS REGION XPOS])
```

(READ.CHANGE.COLOR

```
  [LAMBDA (MSG)                                                       (* reads a color from the user and returns it)
    BLACKCOLOR])
```

(READCOLOR1

```
  [LAMBDA (MSG ALLOWNONEFLG NOWCOLOR)                                 (* rrb "19-Dec-85 12:02")
                                                                      (* lets the user select a color.)
    (PROG [(WIN (CREATEW (MAKEWITHINREGION (CREATEREGION LASTMOUSEX LASTMOUSEY COLORMENUWIDTH COLORMENUHEIGHT)
                                WHOLEDISPLAY)
                       (OR MSG "Enter a color:  Left in rectangle sets level.")))
           VAL REDREGION GREENREGION BLUEREGION HUEREGION LIGHTNESSREGION SATURATIONREGION
           (INITCOLOR (AND NOWCOLOR (INSURE.RGB.COLOR NOWCOLOR T]
          [SETQ REDREGION (CAR (SETQ VAL (DRAWREADCOLORBOX 10 " RED " WIN]
          [SETQ GREENREGION (CAR (SETQ VAL (DRAWREADCOLORBOX (IPLUS (CDR VAL)
                                                                    5)
                                                  "GREEN" WIN]
          [SETQ BLUEREGION (CAR (SETQ VAL (DRAWREADCOLORBOX (IPLUS (CDR VAL)
                                                                   5)
                                                 " BLUE" WIN]
          [SETQ HUEREGION (CAR (SETQ VAL (DRAWREADCOLORBOX (IPLUS (CDR VAL)
                                                                  20)
                                                " hue " WIN]
          [SETQ LIGHTNESSREGION (CAR (SETQ VAL (DRAWREADCOLORBOX (CDR VAL)
                                                      " light " WIN]
          [SETQ SATURATIONREGION (CAR (SETQ VAL (DRAWREADCOLORBOX (CDR VAL)
                                                       " sat " WIN]
          (ADDMENU (create MENU
                          ITEMS _ [APPEND [COND
                                             (ALLOWNONEFLG '(("No color" 'NONE "specifies that no color should
                                                                   be used."]
                                         '((OK 'OK "Returns the displayed color.")
                                           (Abort 'ABORT "Aborts this operation."]
                          CENTERFLG _ T
                          MENUBORDERSIZE _ 1
                          WHENSELECTEDFN _ (FUNCTION READCOLORCOMMANDMENUSELECTEDFN))
```

```
                WIN
                (create POSITION
                       XCOORD _ (PLUS (CDR VAL)
                                      10)
                       YCOORD _ 100))
         [SETQ VAL (COND
                      (INITCOLOR (READCOLOR2 WIN (fetch (RGB RED) of INITCOLOR)
                                              (fetch (RGB GREEN) of INITCOLOR)
                                              (fetch (RGB BLUE) of INITCOLOR)))
                      (T (READCOLOR2 WIN 0 0 0]
         (CLOSEW WIN)
         (RETURN VAL])
```

(**READCOLORCOMMANDMENUSELECTEDFN**

```
  [LAMBDA (ITEM MENU BUTTON)                                         (* rrb "18-Jul-85 11:01")

         (* when selected function for the menu that sits in the read color window.
         Puts the value OK or ABORT on the window if selected.)

    (WINDOWPROP (WFROMMENU MENU)
           'MENUCOMMAND
           (CADADR ITEM])
```

(**READCOLOR2**

```
  [LAMBDA (WIN REDLEVEL GREENLEVEL BLUELEVEL)                        (* rrb "29-Oct-85 12:29")
                                                                     (* internal function to READCOLOR which polls mouse and
                                                                     updates fields.)
    (PROG ((VALBTM (IPLUS (fetch (REGION BOTTOM) of REDREGION)
                          264))
           LEVEL LASTX LASTY HLS)
         (PROGN (DISPLAYREADCOLORLEVEL REDLEVEL REDLEVEL REDREGION WIN)
                (DISPLAYREADCOLORLEVEL GREENLEVEL GREENLEVEL GREENREGION WIN)
                (DISPLAYREADCOLORLEVEL BLUELEVEL BLUELEVEL BLUEREGION WIN))
         (DISPLAYREADCOLORHLSLEVELS (SETQ HLS (RGBTOHLS REDLEVEL GREENLEVEL BLUELEVEL))
                WIN)
      WAITLP
                                                                     (* check if menu command was pressed.)
         (SELECTQ (WINDOWPROP WIN 'MENUCOMMAND)
              (OK (RETURN (create RGB
                                  RED _ REDLEVEL
                                  GREEN _ GREENLEVEL
                                  BLUE _ BLUELEVEL)))
              (NONE (RETURN 'NONE))
              (ABORT (RETURN NIL))
              NIL)
         [COND
            ((MOUSESTATE LEFT)
             (COND
                [[SETQ COLOR (COND
                                ((INSIDEP REDREGION (SETQ LASTX (LASTMOUSEX WIN))
                                         (SETQ LASTY (LASTMOUSEY WIN)))
                                 'RED)
                                ((INSIDEP GREENREGION LASTX LASTY)
                                 'GREEN)
                                ((INSIDEP BLUEREGION LASTX LASTY)
                                 'BLUE]
                 (until (MOUSESTATE (NOT LEFT))
                    do                                               (* as long as left is down, adjust the color.)
                       (COND
                          ((NEQ [SETQ LEVEL (IMIN 255 (IMAX 0 (IDIFFERENCE (LASTMOUSEY WIN)
                                                                      (fetch (REGION BOTTOM) of REDREGION]
                                (SELECTQ COLOR
                                     (RED REDLEVEL)
                                     (GREEN GREENLEVEL)
                                     BLUELEVEL))                     (* see if color level has changed.)
                           (SELECTQ COLOR
                               (RED (DISPLAYREADCOLORLEVEL (SETQ REDLEVEL LEVEL)
                                           REDLEVEL REDREGION WIN))
                               (GREEN (DISPLAYREADCOLORLEVEL (SETQ GREENLEVEL LEVEL)
                                             GREENLEVEL GREENREGION WIN))
                               (DISPLAYREADCOLORLEVEL (SETQ BLUELEVEL LEVEL)
                                      BLUELEVEL BLUEREGION WIN))
                           (DISPLAYREADCOLORHLSLEVELS (SETQ HLS (RGBTOHLS REDLEVEL GREENLEVEL BLUELEVEL))
                                  WIN]
                ([SETQ COLOR (COND
                                ((INSIDEP HUEREGION (SETQ LASTX (LASTMOUSEX WIN))
                                         (SETQ LASTY (LASTMOUSEY WIN)))
                                 'HUE)
                                ((INSIDEP LIGHTNESSREGION LASTX LASTY)
                                 'LIGHTNESS)
                                ((INSIDEP SATURATIONREGION LASTX LASTY)
                                 'SATURATION]
                 (until (MOUSESTATE (NOT LEFT))
                    do                                               (* as long as red is down, adjust the color.)
                       (COND
                              ((NOT (EQUAL [SETQ LEVEL (HLSVALUEFROMLEVEL COLOR
                                                          (IMIN 255 (IMAX 0 (IDIFFERENCE (LASTMOUSEY WIN)
                                                                                  (fetch (REGION BOTTOM)
                                                                                     of REDREGION]
                                          (HLSLEVEL HLS COLOR)))   (* see if color level has changed.)
                               (HLSLEVEL HLS COLOR LEVEL)
                               (SELECTQ COLOR
                                   (HUE (DISPLAYREADCOLORLEVEL LEVEL (LEVELFROMHLSVALUE 'HUE LEVEL)
                                             HUEREGION WIN))
                                   (LIGHTNESS (DISPLAYREADCOLORLEVEL LEVEL (LEVELFROMHLSVALUE 'LIGHTNESS LEVEL)
                                                   LIGHTNESSREGION WIN))
                                   (DISPLAYREADCOLORLEVEL LEVEL (LEVELFROMHLSVALUE 'SATURATION LEVEL)
                                          SATURATIONREGION WIN))   (* set the color levels of the current color and update that display
                              also.)
                               (SETQ LEVEL (HLSTORGB HLS))
                               (PROGN (DISPLAYREADCOLORLEVEL (SETQ REDLEVEL (CAR LEVEL))
                                            REDLEVEL REDREGION WIN)
                                      (DISPLAYREADCOLORLEVEL (SETQ GREENLEVEL (CADR LEVEL))
                                            GREENLEVEL GREENREGION WIN)
                                      (DISPLAYREADCOLORLEVEL (SETQ BLUELEVEL (CADDR LEVEL))
                                            BLUELEVEL BLUEREGION WIN]
          (BLOCK)
          (GO WAITLP])
)

(DEFINEQ

(CREATE.CNS.MENU
  [LAMBDA NIL                                             (* rrb "17-Jul-85 21:14")
                                                          (* creates the CNS menu.)
                                                          (* Not fully implemented. Use
                                                          STYLESHEET.WHENSELECTEDFN to set items from level
                                                          bars.)
    (SETQ CNS.STYLE
     (CREATE.STYLE 'ITEM.TITLES '(Saturation Lightness Tint Hue)
            'ITEM.TITLE.FONT
            '(TIMESROMAN 14 BOLD)
            'ITEMS
            [LIST (create MENU
                         ITEMS _ '(Grayish Moderate Strong Vivid))
                  (create MENU
                         ITEMS _ '(Black ("Very Dark" 'VeryDark)
                                         Dark Medium Light ("Very Light" 'VeryLight)
                                         White))
                  (create MENU
                         ITEMS _ '(Orange Orangish Red Reddish Yellow Yellowish Green Greenish Blue Bluish
                                         Purple Purplish Brown Brownish))
                  (create MENU
                         ITEMS _ '(Red Orange Yellow Green Blue Purple Brown]
            'SELECTION
            '("" "" "" "")
            'NEED.NOT.FILL.IN T))
    (STYLESHEET CNS.STYLE])
)

(RPAQQ COLORMENUHEIGHT 320)

(RPAQQ COLORMENUWIDTH 360)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       LLCOLOR)
)

;; functions that used to be taken from GRAPHZOOM.  Renamed and defined here so GRAPHZOOM isn't loaded.

(DEFINEQ

(SK.ABSWXOFFSET
  [LAMBDA (NEWX W)                                        (* rrb "29-MAR-83 11:27")
                                                          (* sets the offset of a window.)
    (WXOFFSET (IDIFFERENCE (WXOFFSET NIL W)
                     NEWX)
           W])

(SK.ABSWYOFFSET
  [LAMBDA (NEWY W)                                        (* rrb "29-MAR-83 11:28")
                                                          (* sets the offset of a window.)
    (WYOFFSET (IDIFFERENCE (WYOFFSET NIL W)
                     NEWY)
           W])
```

```
(SK.UNSCALE.POSITION.FROM.VIEWER
  [LAMBDA (POSITION SCALE)                                    (* rrb " 1-APR-83 16:05")
                                                              (* unscales a point in a window out into the larger coordinate
                                                              space.)

    (create POSITION
           XCOORD _ (TIMES (fetch (POSITION XCOORD) of POSITION)
                           SCALE)
           YCOORD _ (TIMES (fetch (POSITION YCOORD) of POSITION)
                           SCALE])

(SK.SCALE.REGION
  [LAMBDA (REGION SCALE)                                      (* rrb "16-Sep-86 12:38")
                                                              (* scales a region into a windows coordinate space.)

    (COND
       [(EQP SCALE 1)                                         (* make unscaled case fast but make sure it is integer.)
        (COND
           ((AND (FIXP (fetch (REGION LEFT) of REGION))
                 (FIXP (fetch (REGION BOTTOM) of REGION))
                 (FIXP (fetch (REGION WIDTH) of REGION))
                 (FIXP (fetch (REGION HEIGHT) of REGION)))
            REGION)
           (T (CREATEREGION (FIXR (fetch (REGION LEFT) of REGION))
                     (FIXR (fetch (REGION BOTTOM) of REGION))
                     (FIXR (fetch (REGION WIDTH) of REGION))
                     (FIXR (fetch (REGION HEIGHT) of REGION]
       (T (CREATEREGION (FIXR (QUOTIENT (fetch (REGION LEFT) of REGION)
                                     SCALE))
                 (FIXR (QUOTIENT (fetch (REGION BOTTOM) of REGION)
                              SCALE))
                 (FIXR (QUOTIENT (fetch (REGION WIDTH) of REGION)
                              SCALE))
                 (FIXR (QUOTIENT (fetch (REGION HEIGHT) of REGION)
                              SCALE])
)
```

;; functions for zooming

```
(DEFINEQ

(VIEWER.SCALE
  [LAMBDA (WIN)                                               (* rrb "11-Jul-86 15:49")
                                                              (* returns the scale of a sketch viewer)

    (WINDOWPROP WIN 'SCALE])

(SKETCH.ZOOM
  [LAMBDA (SKW)                                               (* rrb " 8-May-85 18:11")
                                                              (* changes the scale of the figure being looked at in a window.)
    (PROG (NEWREG)
          (PROMPTPRINT "Specify the part of this figure that will be seen after the zoom.
                It can be either larger or smaller than the present window size.")
          (SETQ NEWREG (GETWREGION SKW (FUNCTION SAME.ASPECT.RATIO)
                              SKW 4 4))
          (CLRPROMPT)
          (COND
             ((NULL (REGIONSINTERSECTP NEWREG (DSPCLIPPINGREGION NIL SKW)))
                                                              (* if it doesn't overlap this window, don't move.)
              (STATUSPRINT SKW "Specified region was entirely outside the window.  Not changed."))
             (T (SKETCH.DO.ZOOM SKW NEWREG])

(SAME.ASPECT.RATIO
  [LAMBDA (FIXPT MOVEPT WIN)                                  (* rrb "29-MAR-83 11:13")
                                                              (* new region function that keeps the same aspect ratio as a
                                                              window.)

    (COND
       ((NULL MOVEPT)
        FIXPT)
       (T (PROG ((REG (DSPCLIPPINGREGION NIL WIN))
                 (YMOVE (fetch (POSITION YCOORD) of MOVEPT))
                 (XFIX (fetch (POSITION XCOORD) of FIXPT))
                 (XMOVE (fetch (POSITION XCOORD) of MOVEPT))
                 (YFIX (fetch (POSITION YCOORD) of FIXPT))
                 WID)                                         (* use height as the deciding point.)
                [SETQ WID (ABS (QUOTIENT (ITIMES (fetch (REGION WIDTH) of REG)
                                                (IDIFFERENCE YMOVE YFIX))
                                     (fetch (REGION HEIGHT) of REG]
                (RETURN (create POSITION
                               XCOORD _ (COND
                                           ((IGREATERP XFIX XMOVE)
                                            (IDIFFERENCE XFIX WID))
                                           (T (IPLUS XFIX WID)))
                               YCOORD _ YMOVE])
```

```
(SKETCH.DO.ZOOM
  [LAMBDA (SKETCHW NEWREGION)                                          (* rrb "11-Jul-86 15:57")

            (* moves the viewing region of a window to be over NEWREGION which is in window coordinates.)

    (PROG (NEWSCALE (OLDSCALE (VIEWER.SCALE SKETCHW))
                (OLDREG (DSPCLIPPINGREGION NIL SKETCHW)))              (* scale on the basis of heights.)
          [SETQ NEWSCALE (FTIMES OLDSCALE (FQUOTIENT (fetch (REGION HEIGHT) of NEWREGION)
                                                    (fetch (REGION HEIGHT) of OLDREG]
          (WINDOWPROP SKETCHW 'SCALE NEWSCALE)
          (SK.ABSWXOFFSET (FIXR (FQUOTIENT (FTIMES (fetch (REGION LEFT) of NEWREGION)
                                                  OLDSCALE)
                                        NEWSCALE))
                 SKETCHW)
          (SK.ABSWYOFFSET (FIXR (FQUOTIENT (FTIMES (fetch (REGION BOTTOM) of NEWREGION)
                                                  OLDSCALE)
                                        NEWSCALE))
                 SKETCHW)
          (SK.UPDATE.GRIDFACTOR SKETCHW OLDSCALE)
          (SK.UPDATE.AFTER.SCALE.CHANGE SKETCHW])

(SKETCH.NEW.VIEW
  [LAMBDA (SKW)                                                        (* rrb "11-Jul-86 15:51")
                                                                       (* opens a new view onto the sketch viewed by SKW.)
    (WINDOWPROP (SKETCHW.CREATE (SKETCH.FROM.VIEWER SKW)
                       NIL NIL NIL (VIEWER.SCALE SKW)
                       T
                       (SK.GRIDFACTOR SKW))
          'DONTQUERYCHANGES T])

(ZOOM.UPDATE.ELT
  [LAMBDA (ELT SKW)                                                    (* rrb "29-Jan-85 14:40")

            (* destructively updates the local part of an element in response to a zoom or hardcopy command.)

    (PROG ((CACHE (SK.HOTSPOT.CACHE SKW)))
          (SK.REMOVE.HOTSPOTS.FROM.CACHE1 ELT CACHE)
          (replace (SCREENELT LOCALPART) of ELT with (fetch (SCREENELT LOCALPART) of (SK.LOCAL.FROM.GLOBAL
                                                                                        (fetch (SCREENELT GLOBALPART)
                                                                                           of ELT)
                                                                                        SKW)))
          (SK.ADD.HOTSPOTS.TO.CACHE1 ELT CACHE)
          (RETURN ELT])

(SK.UPDATE.AFTER.SCALE.CHANGE
  [LAMBDA (SKETCHW STOPIFMOUSEDOWN)                                    (* rrb "19-Mar-86 15:05")

            (* called to update the display and local elements after a window has had a scale change.)

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

    (PROG ([SKETCH (fetch (SKETCH SKETCHELTS) of (INSURE.SKETCH (SKETCH.FROM.VIEWER SKETCHW]
           NEWREGION INNEW? LOCALELT)                                  (* take down the caret.)
          (SKED.CLEAR.SELECTION SKETCHW T)
          (SK.UPDATE.REGION.VIEWED SKETCHW)
          (WINDOWPROP SKETCHW 'PICKFONTCACHE NIL)
          (SETQ NEWREGION (SKETCH.REGION.VIEWED SKETCHW))
          [for GELT in SKETCH do (SETQ INNEW? (SK.INSIDE.REGION GELT NEWREGION))
                                 (COND
                                    [(SETQ LOCALELT (SK.LOCAL.ELT.FROM.GLOBALPART GELT SKETCHW))
                                     (COND
                                        (INNEW?                        (* is still in but must have its local adjusted to the new scale.)
                                               (ZOOM.UPDATE.ELT LOCALELT SKETCHW))
                                        (T                             (* if it is not supposed to be in the new region, remove it.)
                                           (SK.DELETE.ITEM LOCALELT SKETCHW]
                                    (INNEW?                            (* just came in)
                                           (SK.ADD.ITEM GELT SKETCHW]
          (DSPFILL NIL NIL 'REPLACE SKETCHW)
          (SKETCHW.REPAINTFN SKETCHW NIL STOPIFMOUSEDOWN T])

(SKETCH.AUTOZOOM
  [LAMBDA (SKW)                                                        (* rrb "10-Sep-86 16:51")

            (* allows the user to pick a point and zooms to or from that point according to the cursor.)

    (RESETFORM (CURSOR AUTOZOOMCURSOR)
           (PROG [SKETCHREG NEWSKETCHREG PTX PTY SCALE LFT BTM WID HGHT DISPLAYSTOPPED
                        (WINDOWREG (WINDOWPROP SKW 'REGION]
                 (STATUSPRINT SKW "left button enlarges; middle reduces.")

            (* zoom by a constant factor that keeps the point that the cursor is on at the same location.)
```

```
                 [until (AND (MOUSESTATE (NOT UP))
                             (NOT (INSIDE? WINDOWREG LASTMOUSEX LASTMOUSEY))
                             (OR (NOT (EQ DISPLAYSTOPPED 'STOPPED))
                                 (PROGN                            (* last display didn't finish)
                                        (SKETCH.GLOBAL.REGION.ZOOM SKW NEWSKETCHREG T)
                                        T)))
                    do (COND
                          ((LASTMOUSESTATE (OR LEFT MIDDLE))
                           [SETQ PTX (TIMES (LASTMOUSEX SKW)
                                            (SETQ SCALE (VIEWER.SCALE SKW]
                           (SETQ PTY (TIMES (LASTMOUSEY SKW)
                                            SCALE))
                           (SETQ SKETCHREG (SKETCH.REGION.VIEWED SKW))
                           (SETQ LFT (fetch (REGION LEFT) of SKETCHREG))
                           (SETQ BTM (fetch (REGION BOTTOM) of SKETCHREG))
                           (SETQ WID (fetch (REGION WIDTH) of SKETCHREG))
                           (SETQ HGHT (fetch (REGION HEIGHT) of SKETCHREG))
                           (COND
                              ([SETQ NEWSKETCHREG (COND
                                                     ((LASTMOUSESTATE LEFT)
                                                                   (* zoom in)
                                                      (CREATEREGION (FDIFFERENCE PTX (TIMES (DIFFERENCE PTX LFT)
                                                                                            AUTOZOOM.FACTOR))
                                                             (FDIFFERENCE PTY (TIMES AUTOZOOM.FACTOR
                                                                                     (DIFFERENCE PTY BTM)))
                                                             (TIMES WID AUTOZOOM.FACTOR)
                                                             (TIMES HGHT AUTOZOOM.FACTOR)))
                                                     ((LASTMOUSESTATE MIDDLE)
                                                                   (* zoom out)
                                                      (CREATEREGION (FDIFFERENCE PTX (QUOTIENT (DIFFERENCE PTX
                                                                                                      LFT)
                                                                                            AUTOZOOM.FACTOR))
                                                             (FDIFFERENCE PTY (QUOTIENT (DIFFERENCE PTY BTM)
                                                                                     AUTOZOOM.FACTOR))
                                                             (QUOTIENT WID AUTOZOOM.FACTOR)
                                                             (QUOTIENT HGHT AUTOZOOM.FACTOR]
                               (CURSOR (COND
                                          ((LASTMOUSESTATE LEFT)
                                           ZOOMINCURSOR)
                                          (T ZOOMOUTCURSOR)))
                               (SETQ DISPLAYSTOPPED (SKETCH.GLOBAL.REGION.ZOOM SKW NEWSKETCHREG T))
                               (CURSOR AUTOZOOMCURSOR]
                 (CLOSEPROMPTWINDOW SKW])
```

(SKETCH.GLOBAL.REGION.ZOOM

```
  [LAMBDA (SKETCHW NEWREGION STOPIFMOUSEDOWN)                      ; Edited  9-Jan-87 08:45 by rrb

          (* moves the viewing region of a window to be over NEWREGION which is in sketch coordinates.)

    (PROG (WIDTHSCALE HEIGHTSCALE NEWSCALE NEWLEFT NEWSCALE NEWBOTTOM (OLDSCALE (VIEWER.SCALE SKETCHW))
                 (WINDOWREG (DSPCLIPPINGREGION NIL SKETCHW)))      (* scale on the basis of which ever dimension make the region
                                                                   fit.)
          (SKED.CLEAR.SELECTION SKETCHW)
          (COND
             ([GREATERP (SETQ HEIGHTSCALE (FQUOTIENT (fetch (REGION HEIGHT) of NEWREGION)
                                                 (fetch (REGION HEIGHT) of WINDOWREG)))
                     (SETQ WIDTHSCALE (FQUOTIENT (fetch (REGION WIDTH) of NEWREGION)
                                             (fetch (REGION WIDTH) of WINDOWREG]
                                                                   (* height is largest scale)
              (SETQ NEWSCALE HEIGHTSCALE))
             (T (SETQ NEWSCALE WIDTHSCALE)))                       (* center the extra width)
          (SETQ NEWLEFT (FIXR (FQUOTIENT (DIFFERENCE (fetch (REGION LEFT) of NEWREGION)
                                                (QUOTIENT (DIFFERENCE (TIMES (fetch (REGION WIDTH) of WINDOWREG)
                                                                             NEWSCALE)
                                                                  (fetch (REGION WIDTH) of NEWREGION))
                                                       2))
                                     NEWSCALE)))                   (* center the extra height)
          (SETQ NEWBOTTOM (FIXR (FQUOTIENT (DIFFERENCE (fetch (REGION BOTTOM) of NEWREGION)
                                                  (QUOTIENT (DIFFERENCE (TIMES (fetch (REGION HEIGHT)
                                                                                      of WINDOWREG)
                                                                               NEWSCALE)
                                                                    (fetch (REGION HEIGHT) of NEWREGION))
                                                         2))
                                       NEWSCALE)))
          (COND
             [(EQUAL OLDSCALE NEWSCALE)                            (* scale hasn't changed, just scroll)
              (RETURN (SKETCHW.SCROLLFN SKETCHW (DIFFERENCE NEWLEFT (fetch (REGION LEFT) of WINDOWREG))
                             (DIFFERENCE NEWBOTTOM (fetch (REGION BOTTOM) of WINDOWREG]
             (T (WINDOWPROP SKETCHW 'SCALE NEWSCALE)
                (SK.ABSWXOFFSET NEWLEFT SKETCHW)
                (SK.ABSWYOFFSET NEWBOTTOM SKETCHW)
                (SK.UPDATE.GRIDFACTOR SKETCHW OLDSCALE)
                (RETURN (SK.UPDATE.AFTER.SCALE.CHANGE SKETCHW STOPIFMOUSEDOWN])
)
```

```
(RPAQ? AUTOZOOM.FACTOR 0.8)

(RPAQ? AUTOZOOM.REPAINT.TIME 3000)

(RPAQ AUTOZOOMCURSOR (CURSORCREATE '

                                'NIL 7 8))

(RPAQ ZOOMINCURSOR (CURSORCREATE '

                             'NIL 7 8))

(RPAQ ZOOMOUTCURSOR (CURSORCREATE '

                               'NIL 7 8))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS AUTOZOOM.FACTOR AUTOZOOM.REPAINT.TIME ZOOMINCURSOR ZOOMOUTCURSOR)
)
```

;; fns for changing the view

```
(DEFINEQ

(SKETCH.HOME
  [LAMBDA (SKW)                                               (* rrb " 7-May-85 12:43")
                                                              (* changes the scale of the figure being looked at in a window.)
    (PROG NIL
          (WINDOWPROP SKW 'SCALE 1.0)
          (WXOFFSET (WXOFFSET NIL SKW)
                 SKW)
          (WYOFFSET (WYOFFSET NIL SKW)
                 SKW)
          (SK.UPDATE.AFTER.SCALE.CHANGE SKW])

(SK.FRAME.IT
  [LAMBDA (SKW)                                               (* rrb "23-Oct-85 10:44")
                                                              (* changes the region being viewed so that the entire sketch just
    fits.)
    (PROG ((SKETCH (INSURE.SKETCH SKW)))
          (COND
             ((NULL (fetch (SKETCH SKETCHELTS) of SKETCH))
              (STATUSPRINT SKW "There is nothing in this sketch."))
             (T (SKETCH.GLOBAL.REGION.ZOOM SKW (SKETCH.REGION.OF.SKETCH SKETCH])

(SK.FRAME.WINDOW.TO.SKETCH
  [LAMBDA (SKW)                                               (* rrb "24-Sep-86 10:17")

          (* reshapes the window so that the sketch at the current scale just fits inside the window.)

    (PROG ((SKETCH (INSURE.SKETCH SKW)))
          (COND
             ((NULL (fetch (SKETCH SKETCHELTS) of SKETCH))
              (STATUSPRINT SKW "There is nothing in this sketch."))
             (T                                               (* make sure the region isn't larger than the screen.)
                (PROG ((LOCALREGION (INCREASEREGION (SK.SCALE.REGION (SKETCH.REGION.OF.SKETCH SKETCH)
                                                           (VIEWER.SCALE SKW))
                                           1))
                       ATWINS HOWATTED WININTERIOR WREGION BORDER)

          (* 1 point increase is because the region function for boxes is one too small in the width and height, i.e.
          doesn't include the bit for the edge.)

                      (COND
                         ((OR (GREATERP (fetch (REGION WIDTH) of LOCALREGION)
                                     (BITMAPWIDTH (SCREENBITMAP SKW)))
                              (GREATERP (fetch (REGION HEIGHT) of LOCALREGION)
                                     (DIFFERENCE (BITMAPHEIGHT (SCREENBITMAP SKW))
                                            12)))

          (* leave room at the top for part of the title so the user can use popup menu)

                          (STATUSPRINT SKW "The window would have to be larger than the screen."))
                         (T (CLOSEPROMPTWINDOW SKW)
                            (SETQ HOWATTED (for ATW in (SETQ ATWINS (ATTACHEDWINDOWS SKW))
                                              collect (DETACHWINDOW ATW)))
                            (SETQ WININTERIOR (DSPCLIPPINGREGION NIL SKW))
                            (SETQ WREGION (WINDOWPROP SKW 'REGION))
                                                              (* move the coordinate system to lower left corner and display
                                                              the image there.)
                            (SCROLLW SKW (DIFFERENCE (fetch (REGION LEFT) of WININTERIOR)
                                                (fetch (REGION LEFT) of LOCALREGION))
                                   (DIFFERENCE (fetch (REGION BOTTOM) of WININTERIOR)
                                                     (fetch (REGION BOTTOM) of LOCALREGION)))
                                    [SHAPEW SKW (CREATEREGION (fetch (REGION LEFT) of WREGION)
                                                     (fetch (REGION BOTTOM) of WREGION)
                                                     (PLUS (fetch (REGION WIDTH) of LOCALREGION)
                                                           (DIFFERENCE (fetch (REGION WIDTH) of WREGION)
                                                                  (fetch (REGION WIDTH) of WININTERIOR)))
                                                     (PLUS (fetch (REGION HEIGHT) of LOCALREGION)
                                                           (DIFFERENCE (fetch (REGION HEIGHT) of WREGION)
                                                                  (fetch (REGION HEIGHT) of WININTERIOR]
                                    (for ATW in ATWINS as HOWAT in HOWATTED do (ATTACHWINDOW ATW SKW (CAR HOWAT)
                                                                                  (CDR HOWAT])
```

(SK.MOVE.TO.VIEW

```
  [LAMBDA (SKW VIEW)                                          (* rrb "28-Jun-85 18:16")

          (* restores a view by changing the position and scale of the figure being looked at in a window.)

    (PROG ((NEWSCALE (fetch (SKETCHVIEW VIEWSCALE) of VIEW))
           (OLDSCALE (WINDOWPROP SKW 'SCALE))
           SKREGWIDTH SKREGHEIGHT)
          (WINDOWPROP SKW 'SCALE NEWSCALE)
          (WXOFFSET (WXOFFSET NIL SKW)
                 SKW)
          (WXOFFSET (IMINUS (QUOTIENT (DIFFERENCE (fetch (SKETCHVIEW VIEWXPOSITION) of VIEW)
                                             (TIMES (QUOTIENT (WINDOWPROP SKW 'WIDTH)
                                                           2)
                                                    NEWSCALE))
                                   NEWSCALE))
                 SKW)
          (WYOFFSET (WYOFFSET NIL SKW)
                 SKW)
          (WYOFFSET (IMINUS (QUOTIENT (DIFFERENCE (fetch (SKETCHVIEW VIEWYPOSITION) of VIEW)
                                             (TIMES (QUOTIENT (WINDOWPROP SKW 'HEIGHT)
                                                           2)
                                                    NEWSCALE))
                                   NEWSCALE))
                 SKW)
          (SK.UPDATE.GRIDFACTOR SKW OLDSCALE)
          (SK.UPDATE.AFTER.SCALE.CHANGE SKW])
```

(SK.NAME.CURRENT.VIEW

```
  [LAMBDA (SKW)                                               (* rrb "11-Jul-86 15:52")

          (* reads a name from the user and adds the current view to the list of views)

    (PROG [(SKETCH (INSURE.SKETCH SKW))
           (NAME (MKATOM (PROMPT.GETINPUT SKW "Name for this view: "]
          (COND
             (NAME [PUTSKETCHPROP SKETCH 'VIEWS (APPEND (GETSKETCHPROP SKETCH 'VIEWS)
                                                       (CONS (create SKETCHVIEW
                                                                    VIEWNAME _ NAME
                                                                    VIEWSCALE _ (VIEWER.SCALE SKW)
                                                                    VIEWPOSITION _ (REGION.CENTER (
                                                                                   SKETCH.REGION.VIEWED
                                                                                          SKW]
                   (STATUSPRINT SKW " ... done."])
```

(SKETCH.ADD.VIEW

```
  [LAMBDA (SKETCH NAME SCALE CENTERPOSITION)                  (* rrb "25-Nov-85 18:27")
                                                              (* Adds a view to SKETCH.)
    (PROG ((SKETCH (INSURE.SKETCH SKETCH)))
          (COND
             (NAME (PUTSKETCHPROP SKETCH 'VIEWS (APPEND (GETSKETCHPROP SKETCH 'VIEWS)
                                                       (CONS (create SKETCHVIEW
                                                                    VIEWNAME _ NAME
                                                                    VIEWSCALE _ (OR (NUMBERP SCALE)
                                                                                    (\ILLEGAL.ARG SCALE))
                                                                    VIEWPOSITION _ (OR (POSITIONP CENTERPOSITION
                                                                                              )
                                                                                       (\ILLEGAL.ARG
                                                                                              CENTERPOSITION])
```

(SK.RESTORE.VIEW

```
  [LAMBDA (SKW)                                               (* rrb " 6-Nov-85 09:56")

          (* puts up a menu of the previously saved places in the sketch and moves to the one selected.)

    (PROG [(VIEW (\CURSOR.IN.MIDDLE.MENU (create MENU
                                              ITEMS _ (CONS '(Home 'HOME "returns to the origin at the
                                                                   original scale")
                                                            (for SAVEDVIEW in (GETSKETCHPROP (INSURE.SKETCH
                                                                                                  SKW)
                                                                                     'VIEWS)
```

```
                                                                              collect (LIST (fetch (SKETCHVIEW VIEWNAME)
                                                                                                of SAVEDVIEW)
                                                                                            (KWOTE SAVEDVIEW)
                                                                                            "returns the view to this
                                                                                            location.")))
                                                           TITLE _ "Which view?"
                                                           CENTERFLG _ T]    (* treat home specially so the user will always have one way
                                                                             back.)
          (COND
             ((EQ VIEW 'HOME)
              (SKETCH.HOME SKW))
             (VIEW (SK.MOVE.TO.VIEW SKW VIEW])

(SK.FORGET.VIEW
  [LAMBDA (SKW)                                                      (* rrb " 6-Nov-85 09:57")

          (* puts up a menu of the previously saved places in the sketch and lets the user select one to forget.)

    (PROG ((SKETCH (INSURE.SKETCH SKW))
           VIEWS ONETOFORGET)
          (SETQ VIEWS (GETSKETCHPROP SKETCH 'VIEWS))
          (COND
             ((NULL VIEWS)
              (STATUSPRINT SKW "There are no saved views.  They are created with the 'Save view' command.")
              (RETURN)))
          (SETQ ONETOFORGET (MENU (create MENU
                                          ITEMS _ (for SAVEDVIEW in VIEWS collect (LIST (fetch (SKETCHVIEW VIEWNAME)
                                                                                            of SAVEDVIEW)
                                                                                      (KWOTE SAVEDVIEW)
                                                                                      "removes this view."))
                                          TITLE _ "Which view?"
                                          CENTERFLG _ T)))
          (COND
             (ONETOFORGET (PUTSKETCHPROP SKETCH 'VIEWS (REMOVE ONETOFORGET VIEWS))
                    (STATUSPRINT SKW "View " (fetch (SKETCHVIEW VIEWNAME) of ONETOFORGET)
                           " forgotten."])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD SKETCHVIEW (VIEWNAME VIEWSCALE VIEWPOSITION)
       (RECORD VIEWPOSITION (VIEWXPOSITION . VIEWYPOSITION)))
)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       SKETCH SKETCHELEMENTS SKETCHOBJ SKETCHEDIT INTERPRESS)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA STATUSPRINT)
)
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

RECORD INDEX

CONSTANT INDEX