File created: 15-Mar-94 10:48:02 **{DSK}<users>nilsson>mnw>HLDISPLAY.;5**

changes to: (FNS \GETREGION.CHECKBASEPT DSPYSCREENTOWINDOW DSPXSCREENTOWINDOW \GETREGION.CHECKOPPT GETGRIDBOXREGION NEAREST/PT/ON/GRID EDITBMBUTTONFN)

previous date: 25-Feb-94 14:50:58 {DSK}<users>nilsson>mnw>HLDISPLAY.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ HLDISPLAYCOMS
       (                                                    ; GRID functions
        (FNS GRID GRIDXCOORD GRIDYCOORD LEFTOFGRIDCOORD BOTTOMOFGRIDCOORD SHADEGRIDBOX)
                                                            ; Low level compatibility and extensions
        (FNS INSIDE?)
        [COMS                                               ; Mouse selection code
              (FNS MOUSESTATE-EXPR MOUSESTATE-NAME)
              (PROP ARGNAMES MOUSESTATE LASTMOUSESTATE UNTILMOUSESTATE KEYSETSTATE LASTKEYSETSTATE)
              (EXPORT (DECLARE%: DOCOPY (MACROS MOUSESTATE LASTMOUSESTATE UNTILMOUSESTATE KEYSETSTATE
                                                LASTKEYSETSTATE))
                      (DECLARE%: DONTCOPY (MACROS WITHIN))
                      (ADDVARS (GLOBALVARS LASTMOUSEX LASTMOUSEY LASTMOUSEBUTTONS]
                                                            ; High Level Display utilities
        (FNS DECODEBUTTONS)
        (FNS PTDIFFERENCE PTPLUS)
        (COMS                                               ; User interaction for regions, etc
              (FNS GETPOSITION GETBOXPOSITION DSPYSCREENTOWINDOW DSPXSCREENTOWINDOW GETREGION \GETREGION.PACKPTS
                   \GETREGION.CHECKBASEPT \GETREGION.CHECKOPPT \GETREGIONTRACKWITHBOX \UPDATEXYANDBOX
                   GETBOXREGION \TRACKWITHBOX MOVEBOX DRAWGRAYBOX BLTHLINE BLTVLINE SETCORNER GETSCREENPOSITION
                   GETBOXSCREENPOSITION GETSCREENREGION GETBOXSCREENREGION)
              ;; Old-medley-window-system versions of generic box/position functions
              (FNS \MEDW.GETSCREENPOSITION \MEDW.GETBOXSCREENPOSITION \MEDW.GETSCREENREGION)
              (FNS GETGRIDBOXREGION \RANGELIMIT)
              (FNS MOUSECONFIRM)
              (CURSORS MOUSECONFIRMCURSOR))
        (FNS NEAREST/PT/ON/GRID PTON10GRID NEAREST/MULTIPLE)
        (EXPORT (MACROS IABS))
        (UGLYVARS DASHEDSHADE)
        (GLOBALVARS CROSSHAIRS EXPANDINGBOX FORCEPS BOXCURSOR LOCKEDSPOT OLDEXPANDINGBOX LowerLeftCursor
               UpperRightCursor UpperLeftCursor LowerRightCursor)
        (CURSORS CROSSHAIRS EXPANDINGBOX FORCEPS BOXCURSOR LOCKEDSPOT OLDEXPANDINGBOX LowerLeftCursor
               UpperRightCursor UpperLeftCursor LowerRightCursor)
        (FNS \SW2BM COMPOSEREGS TRANSLATEREG)
        (COMS                                               ; Bitmap and shade editors
              (FNS EDITBM EDITBMSCROLLFN EDITBMCLOSEFN TILEAREA EDITBMBUTTONFN \EDITBM/PUTUP/DISPLAY
                   \EDITBMHOWMUCH EDITBMRESHAPEFN EDITBMREPAINTFN UPDATE/SHADE/DISPLAY
                   UPDATE/BM/DISPLAY/SELECTED/REGION SHOWBUTTON RESETGRID.NEW RESETGRID \READBMDIMENSIONS
                   EDITSHADE \BITMAPFROMTEXTURE EDITSHADEREPAINTFN GRAYBOXAREA \SHADEBITS READHOTSPOT WBOX
                   \CLEARBM EDITBMTEXTURE)
              (DECLARE%: DONTCOPY (RECORDS BUTTON)
                     (MACROS BITMASK UPDATE/BM/DISPLAY))
              (DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (DARKBITSHADE 23130)
                                                    (NORMALGRIDSQUARE 16)
                                                    (NOTINUSEGRAY 42405)
                                                    (EDITBMMENU)
                                                    (EDITBMWINDOWMENU)
                                                    (GRIDSIZEMENU)
                                                    (CLICKWAITTIME 250)))
              (DECLARE%: DOEVAL@COMPILE DONTCOPY (GLOBALVARS DARKBITSHADE NORMALGRIDSQUARE NOTINUSEGRAY
                                                             EDITBMMENU CLICKWAITTIME))
              (CONSTANTS (GRIDTHICKNESS 2)
                     (MINGRIDSQUARE 8)
                     (MAXGRIDWIDTH 199)
                     (MAXGRIDHEIGHT 175)
                     (BMWINDOWSHADE 33410)))
        (FNS SCALEBM BLTPATTERN BLTPATTERN.REPLACEDISPLAY BLTPATTERN.GENERIC)
        (FNS EXPANDBITMAP EXPANDBM SHRINKBITMAP \FAST4BIT)
        (FUNCTIONS ROTATE-BITMAP ROTATE-BITMAP-LEFT)
        (PROP FILETYPE HLDISPLAY)
        (UGLYVARS \4BITEXPANSIONTABLE)))
```

;; GRID functions

```
(DEFINEQ
```

(**GRID**

```
  [LAMBDA (GRIDSPEC WIDTH HEIGHT BORDER DS GRIDSHADE)           ; Edited 8-Dec-88 16:12 by SHIH
```

;; Draws a grid

```
    (PROG ((X0 (fetch (REGION LEFT) of GRIDSPEC))
           (Y0 (fetch (REGION BOTTOM) of GRIDSPEC))
           (SQWIDTH (fetch (REGION WIDTH) of GRIDSPEC))
           (SQHEIGHT (fetch (REGION HEIGHT) of GRIDSPEC))
           (GRIDSHADE (COND
                          ((TEXTUREP GRIDSHADE))
                          (T BLACKSHADE)))
           LINELENGTH TWICEBORDER MAXIMUMCOLOR TOTALHEIGHT GRIDBM TEMPBM)
          (SETQ TOTALHEIGHT (ITIMES HEIGHT SQHEIGHT))
          (COND
             ((OR (ZEROP BORDER)
                  (NULL BORDER))                                    ; don't draw anything.
              (RETURN))
             [(NUMBERP BORDER)
              (SETQ TWICEBORDER (ITIMES BORDER 2))
              (PROGN ;; draw vertical lines use BITBLT so that we don't have to correct for the width of the line since line drawing will put the
                     ;; coordinate in the middle.
                     (BLTSHADE GRIDSHADE DS X0 Y0 BORDER TOTALHEIGHT 'REPLACE)
                     (for X from (IDIFFERENCE (IPLUS X0 SQWIDTH)
                                              BORDER)
                        to (IDIFFERENCE (IPLUS X0 (ITIMES (SUB1 WIDTH)
                                                          SQWIDTH))
                                        BORDER)
                        by SQWIDTH do (BLTSHADE GRIDSHADE DS X Y0 TWICEBORDER TOTALHEIGHT 'REPLACE))
                     (BLTSHADE GRIDSHADE DS (IDIFFERENCE (IPLUS X0 (ITIMES WIDTH SQWIDTH))
                                                         BORDER)
                           Y0 BORDER TOTALHEIGHT 'REPLACE))
              (PROGN                                                ; draw horizontal lines
                     (BLTSHADE GRIDSHADE DS X0 Y0 (SETQ LINELENGTH (ITIMES WIDTH SQWIDTH))
                           BORDER
                           'REPLACE)
                     (for Y from (IDIFFERENCE (IPLUS Y0 SQHEIGHT)
                                              BORDER)
                        to (IDIFFERENCE (IPLUS Y0 (ITIMES (SUB1 HEIGHT)
                                                          SQHEIGHT))
                                        BORDER)
                        by SQHEIGHT do (BLTSHADE GRIDSHADE DS X0 Y LINELENGTH TWICEBORDER 'REPLACE))
                     (BLTSHADE GRIDSHADE DS X0 (IDIFFERENCE (IPLUS Y0 TOTALHEIGHT)
                                                            BORDER)
                           LINELENGTH BORDER 'REPLACE]
             [(EQ BORDER 'POINT)                                    ; put a point in the lower left corner of each box
              (if (WINDOWP DS)
                  then (SETQ TEMPBM (WINDOWPROP DS 'TEMPBM))
                       (SETQ GRIDBM (WINDOWPROP DS 'GRIDBM))
                       (if (NOT GRIDBM)
                           then (SETQ GRIDBM (BITMAPCREATE SQWIDTH SQHEIGHT))
                                (WINDOWPROP DS 'GRIDBM GRIDBM))
                       (BLTSHADE WHITESHADE GRIDBM 0 0)              ; Clear temporary bitmap.
                       (BLTSHADE BLACKSHADE GRIDBM 0 0 1 1 'REPLACE)
                                                                     ; Put spot down.
                                                                     ; Fill up temporary bitmap.
                       (BLTPATTERN GRIDBM 0 0 SQWIDTH SQHEIGHT DS X0 Y0 (ITIMES WIDTH SQWIDTH)
                              (ITIMES HEIGHT SQHEIGHT)
                              'PAINT TEMPBM)
                 else [SETQ MAXIMUMCOLOR (MAXIMUMCOLOR (BITSPERPIXEL (DSPDESTINATION NIL DS]
                      ;; Crufty slow original code.
                      (for X from X0 to (IPLUS X0 (ITIMES WIDTH SQWIDTH)) by SQWIDTH
                         do (for Y from Y0 to (IPLUS Y0 TOTALHEIGHT) by SQHEIGHT
                               do (BITMAPBIT DS X Y MAXIMUMCOLOR]
             (T (\ILLEGAL.ARG BORDER])
```

(GRIDXCOORD

```
  [LAMBDA (XPOS GRIDSPEC)                                           (* rrb "21-MAR-83 13:04")
    (PROG [(GX (IDIFFERENCE XPOS (fetch (REGION LEFT) of GRIDSPEC]
                                                                    (* because (IQUOTIENT -1 2) is 0 instead of -1 like we would
                                                                    like)
          (RETURN (COND
                     ((IGEQ GX 0)
                      (IQUOTIENT GX (fetch (REGION WIDTH) of GRIDSPEC)))
                     (T (SUB1 (IQUOTIENT GX (fetch (REGION WIDTH) of GRIDSPEC])
```

(GRIDYCOORD

```
  [LAMBDA (YPOS GRIDSPEC)                                           (* rrb "21-MAR-83 13:07")
    (PROG [(GY (IDIFFERENCE YPOS (fetch (REGION BOTTOM) of GRIDSPEC]
                                                                    (* because (IQUOTIENT -1 2) is 0 instead of -1 like we would
                                                                    like)
          (RETURN (COND
                     ((IGEQ GY 0)
                      (IQUOTIENT GY (fetch (REGION HEIGHT) of GRIDSPEC)))
                     (T (SUB1 (IQUOTIENT GY (fetch (REGION HEIGHT) of GRIDSPEC])
```

(LEFTOFGRIDCOORD

```
  [LAMBDA (GRIDX GRIDSPEC)                                          (* rrb "19-MAR-82 09:20")
                                                                    (* returns the Left position of a grid location.)
    (IPLUS (fetch (REGION LEFT) of GRIDSPEC)
           (ITIMES (fetch (REGION WIDTH) of GRIDSPEC)
                  GRIDX])
```

(BOTTOMOFGRIDCOORD

```
  [LAMBDA (GRIDY GRIDSPEC)                                          (* rrb "19-MAR-82 09:38")
    (IPLUS (fetch (REGION BOTTOM) of GRIDSPEC)
           (ITIMES (fetch (REGION HEIGHT) of GRIDSPEC)
                  GRIDY])
```

(SHADEGRIDBOX

```
  [LAMBDA (X Y SHADE OPERATION GRIDSPEC GRIDBORDER DS)              ; Edited  1-Sep-87 17:41 by FS
                                                                    (* shades the interior of a grid box.)
    (PROG ((BORDER (OR (FIXP GRIDBORDER)
                       0)))
          (BLTSHADE SHADE DS (IPLUS (LEFTOFGRIDCOORD X GRIDSPEC)
                                    BORDER)
                 (IPLUS (BOTTOMOFGRIDCOORD Y GRIDSPEC)
                        BORDER)
                 (IDIFFERENCE (fetch (REGION WIDTH) of GRIDSPEC)
                              (ITIMES BORDER 2))
                 (IDIFFERENCE (fetch (REGION HEIGHT) of GRIDSPEC)
                              (ITIMES BORDER 2))
                 OPERATION)                                         (* if this is POINT grid, set lower left corner.)
          (COND
             ((EQ GRIDBORDER 'POINT)
              (BITMAPBIT DS (LEFTOFGRIDCOORD X GRIDSPEC)
                     (BOTTOMOFGRIDCOORD Y GRIDSPEC)
                     (MAXIMUMCOLOR (BITSPERPIXEL (DSPDESTINATION NIL DS])

)

;; Low level compatibility and extensions

(DEFINEQ

## (INSIDE?

  [LAMBDA (BOX X Y)                                                 (* rrb "19-MAR-82 09:32")
    (AND (WITHIN (OR X LASTMOUSEX)
                 (fetch (REGION LEFT) of BOX)
                 (fetch (REGION WIDTH) of BOX))
         (WITHIN (OR Y LASTMOUSEY)
                 (fetch (REGION BOTTOM) of BOX)
                 (fetch (REGION HEIGHT) of BOX])

)

;; Mouse selection code

(DEFINEQ

## (MOUSESTATE-EXPR

  [LAMBDA (EXPR MOUSEONLYFLG)                                       (* rrb " 5-Apr-84 17:05")

          (* if MOUSEONLYFLG is non-NIL, the testing should be done only on the mouse buttons.
          MOUSEONLYFLG will be passed in as T by MOUSESTATE but will get reset if any of the names are not mouse button
          names.)

    (PROG (NAMEMASK (MOUSEBUTTONMASK 7))
          (RETURN (COND
                     [(NLISTP EXPR)
                      (COND
                         [(EQ EXPR 'UP)
                          (LIST 'ZEROP (COND
                                          (MOUSEONLYFLG (LIST 'LOGAND MOUSEBUTTONMASK 'LASTMOUSEBUTTONS))
                                          (T 'LASTMOUSEBUTTONS]
                         (T

          (* MOUSEONLYFLG can be ignored on this branch because it is generating code for the case where the user is listing the
          button names and if he includes keyset names you want to include them anyway.)

                               (LIST 'NEQ (LIST 'LOGAND 'LASTMOUSEBUTTONS (MOUSESTATE-NAME EXPR))
                                     0]
                     ((EQ (CAR EXPR)
                          'ONLY)
                      (COND
                         ((SETQ NAMEMASK (MOUSESTATE-NAME (CADR EXPR)
                                                          MOUSEONLYFLG)))
                         ((SETQ NAMEMASK (MOUSESTATE-NAME (CADR EXPR)
                                                          NIL))         (* non-mouse buttons were named, use all keys.)
                          (SETQ MOUSEONLYFLG NIL)))
                      (LIST 'EQ (COND
```

```
                                    (MOUSEONLYFLG (LIST 'LOGAND MOUSEBUTTONMASK 'LASTMOUSEBUTTONS))
                                    (T 'LASTMOUSEBUTTONS))
                              NAMEMASK))
                        ([EVERY EXPR (FUNCTION (LAMBDA (X)
                                                  (AND (ATOM X)
                                                       (NEQ X 'UP]
```

(* Cant use LOGx trick for UP as it is a disjunct not a key selector)

```
                         (SELECTQ (CAR EXPR)
                              (OR [LIST 'NEQ 0 (LIST 'LOGAND 'LASTMOUSEBUTTONS (CONS 'LOGOR
                                                                                     (MAPCAR (CDR EXPR)
                                                                                            (FUNCTION
                                                                                             MOUSESTATE-NAME])
                              (AND [LIST 'EQ (CONS 'LOGOR (MAPCAR (CDR EXPR)
                                                                  (FUNCTION MOUSESTATE-NAME)))
                                         (LIST 'LOGAND 'LASTMOUSEBUTTONS (CONS 'LOGOR (MAPCAR
                                                                                        (CDR EXPR)
                                                                                        (FUNCTION MOUSESTATE-NAME])
                              (NOT (COND
                                      ((CDDR EXPR)
                                       (SHOULDNT)))
                                   [LIST 'ZEROP (LIST 'LOGAND 'LASTMOUSEBUTTONS (MOUSESTATE-NAME (CADR EXPR])
                              (HELP (CAR EXPR)
                                    " unrecognized mouse key operator")))
                        (T (CONS (CAR EXPR)
                                 (MAPCAR (CDR EXPR)
                                         (FUNCTION (LAMBDA (OPT)
                                                           (MOUSESTATE-EXPR OPT MOUSEONLYFLG])
```

(MOUSESTATE-NAME

```
  [LAMBDA (KEYNAME MOUSEONLYFLG)                          (* rrb "13-JUN-82 11:17")
                                                          (* return the numeric code for a mouse or keyset key.)
    (SELECTQ KEYNAME
         ((LEFT RED)
             4)
         ((RIGHT BLUE)
             2)
         ((YELLOW MIDDLE)
             1)
         (COND
            ((NOT MOUSEONLYFLG)                           (* if wants mouse only, return NIL)
             (SELECTQ KEYNAME
                  (LEFTKEY 128)
                  (LEFTMIDDLEKEY
                      64)
                  (MIDDLEKEY 32)
                  (RIGHTMIDDLEKEY
                      16)
                  (RIGHTKEY 8)
                  (HELP KEYNAME " is not a recognized key name."])
)

(PUTPROPS MOUSESTATE ARGNAMES (BUTTONFORM))

(PUTPROPS LASTMOUSESTATE ARGNAMES (BUTTONFORM))

(PUTPROPS UNTILMOUSESTATE ARGNAMES (BUTTONFORM INTERVAL))

(PUTPROPS KEYSETSTATE ARGNAMES (BUTTONFORM))

(PUTPROPS LASTKEYSETSTATE ARGNAMES (BUTTONFORM))

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS MOUSESTATE MACRO (ARGS (LIST 'PROGN '(GETMOUSESTATE)
                                       (MOUSESTATE-EXPR (CAR ARGS)
                                              T))))

(PUTPROPS LASTMOUSESTATE MACRO (ARGS (MOUSESTATE-EXPR (CAR ARGS)
                                             T)))

(PUTPROPS UNTILMOUSESTATE MACRO [ARGS (COND
                                         [(AND (CDR ARGS)
                                               (CADR ARGS)
                                               (NEQ (CADR ARGS)
                                                    T))
```

(* time argument is given and is not T or NIL; compile in time keeping loop.)

```
                                          (LIST 'PROG [LIST (LIST 'TIMEOUT (LIST 'IPLUS '(CLOCK 0)
```

```
                                                                    (LIST 'OR
                                                                          (LIST 'NUMBERP
                                                                                (CADR ARGS))
                                                                          100)))
                                                        '(NOWTIME (CLOCK 0]
                                                  'LP
                                                  [LIST 'COND (LIST (CONS 'MOUSESTATE (LIST (CAR ARGS)
                                                                                            T))
                                                                    '(RETURN T]
                                                  '(COND
                                                      ((IGREATERP (CLOCK0 NOWTIME)
                                                                  TIMEOUT)
                                                       (RETURN NIL))
                                                      (T (\BACKGROUND)))
                                                  '(GO LP]
                                           (T (LIST 'PROG NIL 'LP [LIST 'COND (LIST (CONS 'MOUSESTATE
                                                                                          (LIST (CAR ARGS)
                                                                                                T))
                                                                                    '(RETURN T]
                                                    '(\BACKGROUND)
                                                    '(GO LP])

(PUTPROPS KEYSETSTATE MACRO [ARGS (LIST 'PROGN '(GETMOUSESTATE)
                                        (MOUSESTATE-EXPR (CAR ARGS])

(PUTPROPS LASTKEYSETSTATE MACRO (ARGS (MOUSESTATE-EXPR (CAR ARGS))))
)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS WITHIN MACRO [(A B C)
                        (AND (IGEQ A B)
                             (ILESSP A (IPLUS B C])
)
)

(ADDTOVAR GLOBALVARS LASTMOUSEX LASTMOUSEY LASTMOUSEBUTTONS)
```

;; END EXPORTED DEFINITIONS

;; High Level Display utilities

```
(DEFINEQ
```

(DECODEBUTTONS

```
  [LAMBDA (BUTTONSTATE)
    (DECLARE (GLOBALVARS LASTMOUSEBUTTONS))                    (* rrb " 9-JAN-82 14:20")
```

(* return a list of the buttons and keys that are down from a button state.)

```
    (OR (SMALLP BUTTONSTATE)
        (SETQ BUTTONSTATE LASTMOUSEBUTTONS))
    (NCONC (AND (NEQ 0 (LOGAND BUTTONSTATE 4))
                (CONS 'LEFT))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 2))
                (CONS 'RIGHT))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 1))
                (CONS 'MIDDLE))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 128))
                (CONS 'LEFTKEY))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 64))
                (CONS 'LEFTMIDDLEKEY))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 32))
                (CONS 'MIDDLEKEY))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 16))
                (CONS 'RIGHTMIDDLEKEY))
           (AND (NEQ 0 (LOGAND BUTTONSTATE 8))
                (CONS 'RIGHTKEY])

)

(DEFINEQ
```

(PTDIFFERENCE

```
  [LAMBDA (PT1 PT2)                                            (* rrb "24-JAN-83 14:54")
                                                               (* adds two positions)
    (create POSITION
           XCOORD _ (DIFFERENCE (fetch (POSITION XCOORD) of PT1)
                                (fetch (POSITION XCOORD) of PT2))
           YCOORD _ (DIFFERENCE (fetch (POSITION YCOORD) of PT1)
                                (fetch (POSITION YCOORD) of PT2])
```

(PTPLUS

```
  [LAMBDA (PT1 PT2)                                                    (* rrb "24-JAN-83 14:54")
                                                                       (* adds two positions)
    (create POSITION
           XCOORD _ (PLUS (fetch (POSITION XCOORD) of PT1)
                          (fetch (POSITION XCOORD) of PT2))
           YCOORD _ (PLUS (fetch (POSITION YCOORD) of PT1)
                          (fetch (POSITION YCOORD) of PT2])

)

;; User interaction for regions, etc

(DEFINEQ

(GETPOSITION
  [LAMBDA (WINDOW CURSOR)                                              ; Edited 27-Aug-87 16:56 by FS
                                                                       ; Get position with cursor
    (fetch (SCREENPOSITION POSITION) of (GETSCREENPOSITION WINDOW CURSOR])

(GETBOXPOSITION
  [LAMBDA (BOXWIDTH BOXHEIGHT ORGX ORGY WINDOW PROMPTMSG)              ; Edited 17-Jan-94 14:01 by sybalsky:mv:envos
    ;; gets a box position, returning the lower left corner.  During the moving the outline of the box is displayed.  If ORGX is given, the box is originally
    ;; drawn at that location and the nearest corner to the cursor is snapped to the cursor position.
    (fetch (SCREENPOSITION POSITION) of (GETBOXSCREENPOSITION BOXWIDTH BOXHEIGHT ORGX ORGY WINDOW PROMPTMSG])

(DSPYSCREENTOWINDOW
  [LAMBDA (Y DS)                                                       ; Edited 15-Mar-94 10:41 by sybalsky
                                                                       (* transforms an y coordinate from screen coordinates into
                                                                       window coordinates)
    (IDIFFERENCE Y (fetch (\DISPLAYDATA DDYOFFSET) of (\GETDISPLAYDATA DS])

(DSPXSCREENTOWINDOW
  [LAMBDA (X DS)                                                       ; Edited 15-Mar-94 10:41 by sybalsky
                                                                       (* transforms an x coordinate from screen coordinates into
                                                                       window coordinates)
    (IDIFFERENCE X (fetch (\DISPLAYDATA DDXOFFSET) of (\GETDISPLAYDATA DS])

(GETREGION
  [LAMBDA (MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG INITCORNERS)
                                                                       ; Edited 17-Jan-94 14:02 by sybalsky:mv:envos
                                                                       ; accepts region from the user.
    (fetch (SCREENREGION REGION) of (GETSCREENREGION MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG
                                     INITCORNERS])

(\GETREGION.PACKPTS
  [LAMBDA NIL                                                          (* rrb "12-Dec-83 18:01")
                                                                       (* copy from variable into position for the constraint checks.)
    (replace (POSITION XCOORD) of BASEPT with BASEX)
    (replace (POSITION YCOORD) of BASEPT with BASEY)
    (replace (POSITION XCOORD) of OPPT with OPPX)
    (replace (POSITION YCOORD) of OPPT with OPPY])

(\GETREGION.CHECKBASEPT
  [LAMBDA (NEWREGFNS BASEPT)                                           ; Edited 15-Mar-94 10:40 by sybalsky
    ;; called by GETREGION to check the constraints imposed on the base point by the user functions.
    ;; if the new region fns is a list, apply them in order.
    (bind USERPT for FN in NEWREGFNS do ;; call user fn on base pt
                                        ;; copying the user return point is time cnsuming but necessary to isolate the system from user code.
                                        (SETQ USERPT (APPLY* FN BASEPT NIL NEWREGIONFNARG))
                                        (COND
                                           ((NOT (POSITIONP USERPT))
                                            (ERROR "non-POSITION returned by NEWREGIONFN" USERPT))
                                           (T (replace (POSITION XCOORD) of BASEPT with (fetch (POSITION XCOORD)
                                                                                               of USERPT))
                                              (replace (POSITION YCOORD) of BASEPT with (fetch (POSITION YCOORD)
                                                                                               of USERPT])

(\GETREGION.CHECKOPPT
  [LAMBDA (MINWID MINHGHT NEWREGFNS BASEPT OPPT)                       ; Edited 15-Mar-94 10:40 by sybalsky

          (* called by GETREGION to check the constraints imposed by the minimum sizes and user functions.
          It assumes BASEPT and OPPT are POSITIONs set to the fixed corner BASEPT and moving corner OPPT.)

    (PROG ((BASEX (fetch (POSITION XCOORD) of BASEPT))
           (BASEY (fetch (POSITION YCOORD) of BASEPT))
           (OPPX (fetch (POSITION XCOORD) of OPPT))
```

```
          (OPPY (fetch (POSITION YCOORD) of OPPT))
          USERPT)                                          (* check for minimum height and width constraints.)
         (AND [COND
                 [(IGREATERP BASEX OPPX)
                  (COND
                     ((ILESSP (IDIFFERENCE BASEX OPPX)
                              MINWID)
                      (SETQ OPPX (IDIFFERENCE BASEX MINWID]
                 ((ILESSP (IDIFFERENCE OPPX BASEX)
                          MINWID)
                  (SETQ OPPX (IPLUS BASEX MINWID]
              (replace (POSITION XCOORD) of OPPT with OPPX))
         (AND [COND
                 [(IGREATERP BASEY OPPY)
                  (COND
                     ((ILESSP (IDIFFERENCE BASEY OPPY)
                              MINHGHT)
                      (SETQ OPPY (IDIFFERENCE BASEY MINHGHT]
                 ((ILESSP (IDIFFERENCE OPPY BASEY)
                          MINHGHT)
                  (SETQ OPPY (IPLUS BASEY MINHGHT]
              (replace (POSITION YCOORD) of OPPT with OPPY))      (* if the new region fns is a list, apply them in order.)
         (for FN in NEWREGFNS do (SETQ USERPT (APPLY* FN BASEPT OPPT NEWREGIONFNARG))
                                 (COND
                                    ((NOT (POSITIONP USERPT))
                                     (ERROR "non-POSITION returned by NEWREGIONFN" USERPT))
                                    (T (replace (POSITION XCOORD) of OPPT with (fetch (POSITION XCOORD) of USERPT))
                                       (replace (POSITION YCOORD) of OPPT with (fetch (POSITION YCOORD) of USERPT])
```

(**GETREGIONTRACKWITHBOX**

```
[LAMBDA NIL                                                   (* hdj "19-Sep-86 14:40")
```

;; tracks a box sized between BASEX BASEY and OPPX OPPY until the left or middle mouse button go down.

```
(DECLARE (GLOBALVARS \CURSORDESTINATION DASHEDSHADE)
         (USEDFREE BASEX BASEY OPPX OPPY)
         (LOCALVARS . T))
(PROG (OLDCURSOR NOERROR XTEMP YTEMP OLDMOUSEX OLDMOUSEY POSTEMP THRUONCE WIDTH HEIGHT DESTINATION MAXX MAXY
           )
      (SETQ WIDTH (IDIFFERENCE BASEX OPPX))
      (SETQ HEIGHT (IDIFFERENCE BASEY OPPY))
      (SETQ DESTINATION \CURSORDESTINATION)
      (SETQ MAXX (SUB1 (BITMAPWIDTH DESTINATION)))
      (SETQ MAXY (SUB1 (BITMAPHEIGHT DESTINATION)))
      (DRAWGRAYBOX OPPX OPPY BASEX BASEY DESTINATION DASHEDSHADE)
```

;; go thru the loop at least once so that checking of user function against the first point is always done.

```
      [SETQ NOERROR (ERSETQ (until (AND THRUONCE (MOUSESTATE (OR LEFT MIDDLE)))
                               do (SETQ THRUONCE T)
                                  (COND
                                     ((LASTMOUSESTATE RIGHT)
                                      (SETQ OLDCURSOR (CURSOR FORCEPS))
                                      (until (MOUSESTATE (NOT RIGHT)))
                                      (CURSOR OLDCURSOR)      ; switch to drag nearest corner
                                      [COND
                                         ((COND
                                             ((IGREATERP BASEX OPPX)
                                              (IGREATERP LASTMOUSEX (IQUOTIENT (IPLUS OPPX BASEX)
                                                                               2)))
                                             (T (IGREATERP (IQUOTIENT (IPLUS OPPX BASEX)
                                                                      2)
                                                           LASTMOUSEX)))
                                                                      ; switch X
                                          (swap OPPX BASEX)
                                          (SETQ WIDTH (IDIFFERENCE BASEX OPPX]
                                      [COND
                                         ((COND
                                             ((IGREATERP BASEY OPPY)
                                              (IGREATERP LASTMOUSEY (IQUOTIENT (IPLUS OPPY BASEY)
                                                                               2)))
                                             (T (IGREATERP (IQUOTIENT (IPLUS OPPY BASEY)
                                                                      2)
                                                           LASTMOUSEY)))
                                                                      ; switch Y
                                          (swap OPPY BASEY)
                                          (SETQ HEIGHT (IDIFFERENCE BASEY OPPY]
                                      (\CURSORPOSITION OPPX OPPY))
                                     ((OR (NOT (EQ LASTMOUSEX OLDMOUSEX))
                                          (NOT (EQ LASTMOUSEY OLDMOUSEY)))
                                                                      ; the cursor has moved, check user constraints.
                                      (SETQ OLDMOUSEX LASTMOUSEX)
                                      (SETQ OLDMOUSEY LASTMOUSEY)
                                                                      ; make sure the base corner {which is opposite the one tracked
                                                                      ; with the mouse} is on the screen.
                                      [replace (POSITION XCOORD) of BASEPT
                                         with (IMAX 0 (IMIN MAXX (IPLUS OLDMOUSEX WIDTH]
                                      [replace (POSITION YCOORD) of BASEPT
```

```
                                        with (IMAX 0 (IMIN MAXY (IPLUS OLDMOUSEY HEIGHT]
                                    (\GETREGION.CHECKBASEPT NEWREGFNS BASEPT)
                                    (SETQ XTEMP (fetch (POSITION XCOORD) of BASEPT))
                                    (SETQ YTEMP (fetch (POSITION YCOORD) of BASEPT))
                                    (COND
                                       ((NOT (AND (IEQP BASEX XTEMP)
                                                  (IEQP BASEY YTEMP)
                                                  (EQ \CURSORDESTINATION DESTINATION)))
                                                        ; move the box
                                        (SETQ XTEMP (IDIFFERENCE XTEMP BASEX))
                                        (SETQ YTEMP (IDIFFERENCE YTEMP BASEY))
                                        (DRAWGRAYBOX OPPX OPPY BASEX BASEY DESTINATION DASHEDSHADE)
                                        (SETQ DESTINATION \CURSORDESTINATION)
                                        (SETQ MAXX (SUB1 (BITMAPWIDTH DESTINATION)))
                                        (SETQ MAXY (SUB1 (BITMAPHEIGHT DESTINATION)))
                                        (SETQ OPPX (IPLUS OPPX XTEMP))
                                        (SETQ OPPY (IPLUS OPPY YTEMP))
                                        (SETQ BASEX (IPLUS BASEX XTEMP))
                                        (SETQ BASEY (IPLUS BASEY YTEMP))
                                        (COND
                                           (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN)))
                                        (DRAWGRAYBOX OPPX OPPY BASEX BASEY DESTINATION DASHEDSHADE]
          (DRAWGRAYBOX OPPX OPPY BASEX BASEY DESTINATION DASHEDSHADE)
          (COND
             ((NULL NOERROR)                                    ; pass back ^E
              (ERROR!])
```

(\UPDATEXYANDBOX

```
  [LAMBDA (BASEPTCHANGE? DESTINATION SHADE)                     (* kbr%: " 3-Feb-86 12:44")
```

(* moves the values in BASEPT and OPPT into the variables BASEX BASEY OPPX OPPY and updates the image on the screen if it has changed.)

```
    (PROG (TEMPX TEMPY)
          (COND
             [(EQ DESTINATION \CURSORDESTINATION)                (* Cursor destination hasn't changed.
                                                                 Add to old image. *)
              [COND
                 (BASEPTCHANGE?                                 (* the base point might have changed, check it too.)
                        (SETQ TEMPX (fetch (POSITION XCOORD) of BASEPT))
                        (SETQ TEMPY (fetch (POSITION YCOORD) of BASEPT))
                        (COND
                           ((NOT (AND (IEQP BASEX TEMPX)
                                      (IEQP BASEY TEMPY)))      (* move the box)
                            (MOVEBOX OPPX OPPY BASEX BASEY (SETQ BASEX TEMPX)
                                   (SETQ BASEY TEMPY)
                                   DESTINATION SHADE]
              (SETQ TEMPX (fetch (POSITION XCOORD) of OPPT))
              (SETQ TEMPY (fetch (POSITION YCOORD) of OPPT))
              (COND
                 ((NOT (AND (IEQP OPPX TEMPX)
                            (IEQP OPPY TEMPY)))                 (* move the box)
                  (MOVEBOX BASEX BASEY OPPX OPPY (SETQ OPPX TEMPX)
                         (SETQ OPPY TEMPY)
                         DESTINATION SHADE)
                  (SETCORNER BASEX BASEY OPPX OPPY]
             (T
```

(* Cursor moved to new screen. Can't get new image by adding to old image.
*)

```
                (DRAWGRAYBOX BASEX BASEY OPPX OPPY DESTINATION SHADE)
                (SETQ BASEX (fetch (POSITION XCOORD) of BASEPT))
                (SETQ BASEY (fetch (POSITION YCOORD) of BASEPT))
                (SETQ OPPX (fetch (POSITION XCOORD) of OPPT))
                (SETQ OPPY (fetch (POSITION YCOORD) of OPPT))
                (DRAWGRAYBOX BASEX BASEY OPPX OPPY \CURSORDESTINATION SHADE)
                (SETCORNER BASEX BASEY OPPX OPPY])
```

(GETBOXREGION

```
  [LAMBDA (WIDTH HEIGHT ORGX ORGY WINDOW PROMPTMSG)             ; Edited 17-Jan-94 14:02 by sybalsky:mv:envos
```

;; returns a region width by height positioned where user says.

```
    (fetch (SCREENREGION REGION) of (GETBOXSCREENREGION WIDTH HEIGHT ORGX ORGY WINDOW PROMPTMSG])
```

(\TRACKWITHBOX

```
  [LAMBDA (SHADE)                                               ; Edited 31-Aug-87 12:45 by FS
```

;; tracks the cursor with a box from corner ORGX ORGY with dimensions BOXWIDTH and BOXHEIGHT until the left or middle button changes.
;; Implements the convention that the RIGHT button can be used to change corners. Returns non-NIL unless an error occurred. Returns the result
;; by setting freely the variables ORGX ORGY BOXWIDTH BOXHEIGHT

```
    (DECLARE (SPECVARS ORGX ORGY BOXWIDTH BOXHEIGHT))
    (PROG (OLDCURSOR ORGLEFTMIDDLE NOERROR MLMASK DESTINATION)
          [SETQ MLMASK (CONSTANT (LOGOR (MOUSESTATE-NAME 'LEFT)
```

```
                                    (MOUSESTATE-NAME 'MIDDLE]
          (SETQ DESTINATION \CURSORDESTINATION)
          (SETQ ORGLEFTMIDDLE (LOGAND MLMASK LASTMOUSEBUTTONS))
          (DRAWGRAYBOX ORGX ORGY (IPLUS ORGX BOXWIDTH)
                 (IPLUS ORGY BOXHEIGHT)
                 DESTINATION SHADE)
          [SETQ NOERROR (ERSETQ (until (PROGN (GETMOUSESTATE)
                                              (NOT (EQ (LOGAND MLMASK LASTMOUSEBUTTONS)
                                                       ORGLEFTMIDDLE)))
                                do (COND
                                      ((LASTMOUSESTATE RIGHT)
                                       (SETQ OLDCURSOR (CURSOR FORCEPS))
                                       (until (MOUSESTATE (NOT RIGHT)))
                                       (CURSOR OLDCURSOR)     ; switch to drag nearest corner
                                       [COND
                                          ((COND
                                              [(IGREATERP BOXWIDTH 0)
                                               (IGREATERP LASTMOUSEX (IPLUS ORGX (IQUOTIENT BOXWIDTH 2]
                                              (T (IGREATERP (IPLUS ORGX (IQUOTIENT BOXWIDTH 2))
                                                        LASTMOUSEX)))
                                                        ; switch X
                                           (SETQ ORGX (IPLUS ORGX BOXWIDTH))
                                           (SETQ BOXWIDTH (IMINUS BOXWIDTH]
                                       [COND
                                          ((COND
                                              [(IGREATERP BOXHEIGHT 0)
                                               (IGREATERP LASTMOUSEY (IPLUS ORGY (IQUOTIENT BOXHEIGHT 2]
                                              (T (IGREATERP (IPLUS ORGY (IQUOTIENT BOXHEIGHT 2))
                                                        LASTMOUSEY)))
                                                        ; switch Y
                                           (SETQ ORGY (IPLUS ORGY BOXHEIGHT))
                                           (SETQ BOXHEIGHT (IMINUS BOXHEIGHT]
                                       (\CURSORPOSITION ORGX ORGY))
                                      (T (COND
                                            ((NOT (AND (IEQP ORGX LASTMOUSEX)
                                                       (IEQP ORGY LASTMOUSEY)))
                                                        ; the cursor has moved, move the box by erasing old box and
                                                        ; drawing new box.  *
                                             (DRAWGRAYBOX ORGX ORGY (IPLUS ORGX BOXWIDTH)
                                                    (IPLUS ORGY BOXHEIGHT)
                                                    DESTINATION SHADE)
                                             (SETQ ORGX LASTMOUSEX)
                                             (SETQ ORGY LASTMOUSEY)
                                             (SETQ DESTINATION \CURSORDESTINATION)
                                             (COND
                                                (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN)))
                                             (DRAWGRAYBOX ORGX ORGY (IPLUS ORGX BOXWIDTH)
                                                    (IPLUS ORGY BOXHEIGHT)
                                                    DESTINATION SHADE]
          (DRAWGRAYBOX ORGX ORGY (IPLUS ORGX BOXWIDTH)
                 (IPLUS ORGY BOXHEIGHT)
                 DESTINATION SHADE)
          (COND
             ((NULL NOERROR)                                    ; pass back ^E
              (ERROR!])
```

(MOVEBOX

```
  [LAMBDA (X1 Y1 X2 Y2 X3 Y3 DESTINATION SHADE)                 ; Edited 25-Aug-87 15:52 by FS
                                                                ; moves the opposite corner of a box from {X2,Y2} to {X3,Y3}.
    (.WHILE.CURSOR.DOWN. (BLTHLINE Y1 X2 X3 DESTINATION SHADE)
           (BLTVLINE X1 Y2 Y3 DESTINATION SHADE)
           (BLTHLINE Y2 X1 X2 DESTINATION SHADE)
           (BLTHLINE Y3 X1 X3 DESTINATION SHADE)
           (BLTVLINE X2 Y1 Y2 DESTINATION SHADE)
           (BLTVLINE X3 Y1 Y3 DESTINATION SHADE])
```

(DRAWGRAYBOX

```
  [LAMBDA (X1 Y1 X2 Y2 DESTINATION SHADE)                       (* kbr%: " 3-Feb-86 12:47")
                                                                (* Put a gray box in window or bitmap DESTINATION)
    (.WHILE.CURSOR.DOWN. (BLTHLINE Y1 X1 X2 DESTINATION SHADE)
           (BLTVLINE X1 Y1 Y2 DESTINATION SHADE)
           (BLTHLINE Y2 X1 X2 DESTINATION SHADE)
           (BLTVLINE X2 Y1 Y2 DESTINATION SHADE])
```

(BLTHLINE

```
  [LAMBDA (Y XA XB DESTINATION SHADE)                           ; Edited  1-Sep-87 17:43 by FS
    (BLTSHADE SHADE DESTINATION (IMIN XA XB)
           Y
           (IABS (IDIFFERENCE XB XA))
           2
           'INVERT])
```

(BLTVLINE

```
  [LAMBDA (X YA YB DESTINATION SHADE)                    ; Edited  1-Sep-87 17:43 by FS
    (BLTSHADE SHADE DESTINATION X (IMIN YA YB)
           2
           (IABS (IDIFFERENCE YB YA))
           'INVERT])
```

(SETCORNER

```
  [LAMBDA (X1 Y1 X2 Y2)                                  (* edited%: "26-Jan-86 13:15")
                                                         (* sets the cursor shape for the box from x1,y1 to x2, y2)
    (DECLARE (GLOBALVARS LowerLeftCursor LowerRightCursor UpperLeftCursor UpperRightCursor))
    (PROG (NEWCURSOR OLDCURSOR)
          [SETQ NEWCURSOR (COND
                             ((IGREATERP X2 X1)          (* moving to left)
                              (COND
                                 ((IGREATERP Y2 Y1)      (* moving up)
                                  UpperRightCursor)
                                 (T LowerRightCursor)))
                             (T                          (* moving to right)
                                (COND
                                   ((IGREATERP Y2 Y1)
                                    UpperLeftCursor)
                                   (T LowerLeftCursor]   (* only call cursor if it changes (less flicker on software cursors))
          (SETQ OLDCURSOR (CURSOR))
          (COND
             ((NOT (EQ NEWCURSOR OLDCURSOR))
              (CURSOR NEWCURSOR)
              (\CURSORPOSITION X2 Y2])
```

(GETSCREENPOSITION

```
  [LAMBDA (WINDOW CURSOR)                                ; Edited 17-Jan-94 14:32 by sybalsky:mv:envos
    ;; Get screenposition with cursor.  If WINDOW, then screenposition should be on same screen as WINDOW and in WINDOW's coordinate system.
    ;; *
    (OR (NULL WINDOW)
        (SETQ WINDOW (WFROMDS WINDOW)))
    (WINDOWOP 'SCGETSCREENPOSITION (COND
                                      (WINDOW (FETCH (WINDOW SCREEN) OF WINDOW))
                                      (T \CURSORSCREEN))
           WINDOW CURSOR])
```

(GETBOXSCREENPOSITION

```
  [LAMBDA (BOXWIDTH BOXHEIGHT ORGX ORGY WINDOW PROMPTMSG)        ; Edited 17-Jan-94 14:32 by sybalsky:mv:envos
    ;; gets a box position, returning the lower left corner.  During the moving the outline of the box is displayed.  If ORGX is given, the box is originally
    ;; drawn at that location and the nearest corner to the cursor is snapped to the cursor position.
    (WINDOWOP 'SCGETBOXSCREENPOSITION \CURSORSCREEN BOXWIDTH BOXHEIGHT ORGX ORGY WINDOW PROMPTMSG])
```

(GETSCREENREGION

```
  [LAMBDA (MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG INITCORNERS)
                                                         ; Edited 17-Jan-94 14:32 by sybalsky:mv:envos
                                                         ; accepts region from the user.
    ;; accepts region from the user.  INITCORNERS lets caller specify size of initial ghost box.  It is a list of the form (BASEX BASEY OPPX OPPY)
    (WINDOWOP 'SCGETSCREENREGION \CURSORSCREEN MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG
           INITCORNERS])
```

(GETBOXSCREENREGION

```
  [LAMBDA (WIDTH HEIGHT ORGX ORGY WINDOW PROMPTMSG)      ; Edited  7-Dec-88 16:36 by SHIH
    ;; returns a screenregion width by height positioned where user says.
    (PROG (SCREENPOS)
          (SETQ SCREENPOS (GETBOXSCREENPOSITION WIDTH HEIGHT ORGX ORGY WINDOW PROMPTMSG))
          (RETURN (create SCREENREGION
                          SCREEN _ (fetch (SCREENPOSITION SCREEN) of SCREENPOS)
                          LEFT _ (fetch (SCREENPOSITION XCOORD) of SCREENPOS)
                          BOTTOM _ (fetch (SCREENPOSITION YCOORD) of SCREENPOS)
                          WIDTH _ WIDTH
                          HEIGHT _ HEIGHT])

)

;; Old-medley-window-system versions of generic box/position functions

(DEFINEQ

(\MEDW.GETSCREENPOSITION
  [LAMBDA (SCREEN WINDOW CURSOR)                         ; Edited 17-Jan-94 14:15 by sybalsky:mv:envos
    ;; Get screenposition with cursor.  If WINDOW, then screenposition should be on same screen as WINDOW and in WINDOW's coordinate system.
    ;; *
    (OR (NULL WINDOW)
        (SETQ WINDOW (WFROMDS WINDOW)))
```

```
    (RESETFORM (CURSOR (OR CURSOR CROSSHAIRS))
           [until (MOUSESTATE LEFT) do (COND
                                          (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN]
                                                                  ; wait until the cursor is down
           [COND
              (WINDOW (until (AND (MOUSESTATE (NOT LEFT))
                                  (EQ \CURSORSCREEN (fetch (WINDOW SCREEN) of WINDOW)))
                      do (COND
                             (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN]
           ;; if a window was specified, then wait until the left button comes up, or until the cursor leaves the screen of the window
           (COND
              ((NULL WINDOW)
               (until (MOUSESTATE (NOT LEFT)))
               (create SCREENPOSITION
                       SCREEN _ LASTSCREEN
                       XCOORD _ LASTMOUSEX
                       YCOORD _ LASTMOUSEY))
              (T (create SCREENPOSITION
                         SCREEN _ LASTSCREEN
                         XCOORD _ (LASTMOUSEX WINDOW)
                         YCOORD _ (LASTMOUSEY WINDOW])
```

(**MEDW.GETBOXSCREENPOSITION**

```
  [LAMBDA (SCREEN BOXWIDTH BOXHEIGHT ORGX ORGY WINDOW PROMPTMSG) ; Edited 17-Jan-94 14:18 by sybalsky:mv:envos
```

```
    ;; gets a box position, returning the lower left corner.  During the moving the outline of the box is displayed.  If ORGX is given, the box is originally
    ;; drawn at that location and the nearest corner to the cursor is snapped to the cursor position.
    (RESETFORM (CURSOR BOXCURSOR)
           (PROG ((MOUSEDOWNFLG (MOUSESTATE (OR LEFT MIDDLE)))
                  SHADE)
                 (COND
                    ((AND (FIXP ORGX)
                          (FIXP ORGY))                    ; origin given, move cursor to nearest corner.
                     [COND
                        ((IGREATERP LASTMOUSEX (IPLUS ORGX (IQUOTIENT BOXWIDTH 2)))
                         (SETQ ORGX (IPLUS ORGX BOXWIDTH))
                         (SETQ BOXWIDTH (IMINUS BOXWIDTH]
                     [COND
                        ((IGREATERP LASTMOUSEY (IPLUS ORGY (IQUOTIENT BOXHEIGHT 2)))
                         (SETQ ORGY (IPLUS ORGY BOXHEIGHT))
                         (SETQ BOXHEIGHT (IMINUS BOXHEIGHT]
                     (\CURSORPOSITION ORGX ORGY))
                    (T (SETQ ORGX LASTMOUSEX)
                       (SETQ ORGY LASTMOUSEY)))
                 (AND PROMPTMSG (PROMPTPRINT PROMPTMSG))
                 (SETQ SHADE GRAYSHADE)
             TRACKLP
                                                          ; track the cursor with a box ghost until the left or middle button
                                                          ; changes.
                 (\TRACKWITHBOX SHADE)
                 [COND
                    ((AND (NULL MOUSEDOWNFLG)
                          (LASTMOUSESTATE (NOT UP)))
                     (SETQ MOUSEDOWNFLG T)
                     (CURSOR CROSSHAIRS))
                    ((AND MOUSEDOWNFLG (LASTMOUSESTATE UP))
                     (AND PROMPTMSG (CLRPROMPT))
                     (RETURN (COND
                                 (WINDOW (create SCREENPOSITION
                                                 SCREEN _ LASTSCREEN
                                                 XCOORD _ (DSPXSCREENTOWINDOW (IMIN ORGX (IPLUS ORGX BOXWIDTH))
                                                                              WINDOW)
                                                 YCOORD _ (DSPYSCREENTOWINDOW (IMIN ORGY (IPLUS ORGY BOXHEIGHT)
                                                                                                )
                                                                              WINDOW)))
                                 (T (create SCREENPOSITION
                                            SCREEN _ LASTSCREEN
                                            XCOORD _ (IMIN ORGX (IPLUS ORGX BOXWIDTH))
                                            YCOORD _ (IMIN ORGY (IPLUS ORGY BOXHEIGHT]
                 (GO TRACKLP])
```

(**MEDW.GETSCREENREGION**

```
  [LAMBDA (SCREEN MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG INITCORNERS)
                                                          ; Edited 17-Jan-94 14:17 by sybalsky:mv:envos
                                                          ; accepts region from the user.
```

```
    ;; accepts region from the user.  INITCORNERS lets caller specify size of initial ghost box.  It is a list of the form (BASEX BASEY OPPX OPPY)

;;; Why is INITCORNERS not two positions?  gbn

    (RESETFORM (CURSOR EXPANDINGBOX)
           (PROG (DESTINATION SHADE BASEX BASEY OPPX OPPY OLDMOUSEX OLDMOUSEY INITLEFT INITRIGHT INITBOTTOM
                              INITTOP BASEPT OPPT NEWMOUSEX NEWMOUSEY DOWNFLG BEGCLOCK NOTTIMEDOUT NEWREGFNS)
                 (SETQ BASEPT (create POSITION))
                 (SETQ OPPT (create POSITION))
```

```
(SETQ MINWIDTH (COND
                   ((FIXP MINWIDTH))
                   (T 0)))
(SETQ MINHEIGHT (COND
                    ((FIXP MINHEIGHT))
                    (T 0)))
(SETQ NEWREGFNS (MKLIST NEWREGIONFN))
(SETQ SHADE GRAYSHADE)
(SETQ NOTTIMEDOUT T)
(SETQ DESTINATION \CURSORDESTINATION)
[COND
   [INITCORNERS                                    ; setup box by initcorners.
          (COND
             ((AND (EQ 4 (LENGTH INITCORNERS))
                   (for X in INITCORNERS always (FIXP X)))
              (SETQ BASEX (CAR INITCORNERS))
              (SETQ BASEY (CADR INITCORNERS))
              (SETQ OPPX (CADDR INITCORNERS))
              (SETQ OPPY (CADDDR INITCORNERS)))
             (T (\ILLEGAL.ARG INITCORNERS]
   (T                                              ; start with the cursor in the lower right corner of the ghost box.
      (GETMOUSESTATE)
      (SETQ OPPX LASTMOUSEX)
      (SETQ OPPY LASTMOUSEY)
      [COND
         ((ILESSP (SETQ BASEX (IDIFFERENCE OPPX MINWIDTH))
                  0)
          ;; arrange things so that the whole box if possible is on the screen.  If this is not possible, the lower right corner is on
          ;; the screen.
          (SETQ OPPX (SUB1 (IMIN MINWIDTH \CURSORDESTWIDTH)))
          (SETQ BASEX (IDIFFERENCE OPPX MINWIDTH]
      (COND
         ((IGEQ (SETQ BASEY (IPLUS OPPY MINHEIGHT))
                \CURSORDESTHEIGHT)
          ;; if the top corner would be off the screen, move the bottom corner as low as necessary but limited to the bottom of
          ;; the screen.
          (SETQ OPPY (IMAX 0 (IDIFFERENCE \CURSORDESTHEIGHT MINHEIGHT)))
          (SETQ BASEY (IPLUS OPPY MINHEIGHT]
(\CURSORPOSITION OPPX OPPY)                        ; wait for the user to put down the first corner.
(\GETREGIONTRACKWITHBOX)
[COND
   ((AND INITREGION (LASTMOUSESTATE MIDDLE))       ; switch the nearest corner of INITREGION to the cursor and
                                                   ; track it.
                                                   ; Pull from closest corner, ie.  set BASEX,Y to be opposite corner
    (SETQ BASEX (COND
                   ((ILESSP (SETQ OPPX LASTMOUSEX)
                            (IQUOTIENT [IPLUS (SETQ INITLEFT (fetch (REGION LEFT) of INITREGION))
                                              (SETQ INITRIGHT (IPLUS INITLEFT
                                                                     (fetch (REGION WIDTH)
                                                                        of INITREGION]
                                       2))         ; pointing at left half of box, so make origin be in right
                    INITRIGHT)
                   (T                              ; pointing at right half of box,
                      INITLEFT)))
    (SETQ BASEY (COND
                   ((ILESSP (SETQ OPPY LASTMOUSEY)
                            (IQUOTIENT [IPLUS (SETQ INITBOTTOM (fetch (REGION BOTTOM)
                                                                  of INITREGION))
                                              (SETQ INITTOP (IPLUS INITBOTTOM
                                                                   (fetch (REGION HEIGHT)
                                                                      of INITREGION]
                                       2))
                    INITTOP)
                   (T INITBOTTOM]                  ; copy from variable into position for the constraint checks.
(\GETREGION.PACKPTS)
(\GETREGION.CHECKOPPT MINWIDTH MINHEIGHT NEWREGFNS BASEPT OPPT)
(SETQ OPPX (fetch (POSITION XCOORD) of OPPT))
(SETQ OPPY (fetch (POSITION YCOORD) of OPPT))      ; Now draw the initial box
(SETQ DESTINATION \CURSORDESTINATION)
(DRAWGRAYBOX BASEX BASEY OPPX OPPY DESTINATION SHADE)
(SETCORNER BASEX BASEY OPPX OPPY)
(SETQ BEGCLOCK (CLOCK 0))
(COND
   [[ERSETQ (until (PROGN (GETMOUSESTATE)
                          (COND
                             [NOTTIMEDOUT          ; wait to see if user was clicking to mark a corner
                                    (COND
                                       ((\CLOCKGREATERP BEGCLOCK CLICKWAITTIME)
                                        (SETQ NOTTIMEDOUT NIL]
                             (DOWNFLG (LASTMOUSESTATE UP))
                             ((LASTMOUSESTATE (NOT UP))
                                                   ; mouse button when down, continue tracking until it goes up.
                              (SETQ DOWNFLG T)
                              NIL)))
                 do (COND
```

```
                                    [(LASTMOUSESTATE (AND RIGHT (OR LEFT MIDDLE)))
                                     (CURSOR FORCEPS)
                                     (until (MOUSESTATE (NOT RIGHT)))
                                                                  ; Switch to nearest corner
                                     (COND
                                        ((IGEQ (IABS (IDIFFERENCE LASTMOUSEX OPPX))
                                               (IABS (IDIFFERENCE LASTMOUSEX BASEX)))
                                         (swap BASEX OPPX)))
                                     (COND
                                        ((IGEQ (IABS (IDIFFERENCE LASTMOUSEY OPPY))
                                               (IABS (IDIFFERENCE LASTMOUSEY BASEY)))
                                         (swap BASEY OPPY)))
                                     (\GETREGION.PACKPTS)
                                     (\GETREGION.CHECKBASEPT NEWREGFNS BASEPT)
                                     (\GETREGION.CHECKOPPT MINWIDTH MINHEIGHT NEWREGFNS BASEPT OPPT)
                                     (SETCORNER BASEX BASEY OPPX OPPY)
                                     (\UPDATEXYANDBOX T DESTINATION SHADE)
                                     (SETQ DESTINATION \CURSORDESTINATION)
                                     (COND
                                        (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN]
                                    ((OR (NOT (EQ LASTMOUSEX OLDMOUSEX))
                                         (NOT (EQ LASTMOUSEY OLDMOUSEY)))
                                                                  ; the cursor has moved, check user constraints.
                                     (replace (POSITION XCOORD) of OPPT with (SETQ OLDMOUSEX LASTMOUSEX))
                                     (replace (POSITION YCOORD) of OPPT with (SETQ OLDMOUSEY LASTMOUSEY))
                                     (\GETREGION.CHECKOPPT MINWIDTH MINHEIGHT NEWREGFNS BASEPT OPPT)
                                     (\UPDATEXYANDBOX NIL DESTINATION SHADE)
                                     (SETQ DESTINATION \CURSORDESTINATION)
                                     (COND
                                        (BACKGROUNDCURSOREXITFN (APPLY* BACKGROUNDCURSOREXITFN]
                                                                  ; erase box image.
                          (DRAWGRAYBOX BASEX BASEY OPPX OPPY DESTINATION SHADE)
                          (RETURN (create SCREENREGION
                                          SCREEN _ \CURSORSCREEN
                                          LEFT _ (IMIN BASEX OPPX)
                                          BOTTOM _ (IMIN BASEY OPPY)
                                          WIDTH _ (IABS (IDIFFERENCE OPPX BASEX))
                                          HEIGHT _ (IABS (IDIFFERENCE BASEY OPPY]
                         (T                                       ; ^E take down box.
                            (DRAWGRAYBOX BASEX BASEY OPPX OPPY DESTINATION SHADE)
                            (ERROR!])

)

(DEFINEQ

(GETGRIDBOXREGION
  [LAMBDA (MINWIDTH MINHEIGHT GRIDSPEC GRIDINTERIOR WINDOW)       ; Edited 15-Mar-94 10:43 by sybalsky

    ;; Like GETREGION, it lets you sweep out a region, but only within the grid specified by GRIDSPEC and limited to the interior of GRIDREGION
    ;; within WINDOW.

    (LET* [NEWREGION [GRIDREGION (create REGION using GRIDINTERIOR LEFT _ [\DSPTRANSFORMX
                                                                           (fetch (REGION LEFT) of GRIDINTERIOR)
                                                                           (fetch (STREAM IMAGEDATA)
                                                                              of (WINDOWPROP WINDOW 'DSP]
                                                                 BOTTOM _ (\DSPTRANSFORMY (fetch (REGION BOTTOM) of
                                                                                                                  GRIDINTERIOR
                                                                                                                  )
                                                                                          (fetch (STREAM IMAGEDATA)
                                                                                             of (WINDOWPROP WINDOW 'DSP]
                          (RAWREGION (GETREGION 0 0 NIL (FUNCTION (LAMBDA (BASEPT OPPT FNARG)
                                                                    (COND
                                                                       ((AND OPPT (INSIDE? GRIDREGION
                                                                                           (fetch (POSITION XCOORD)
                                                                                              of OPPT)
                                                                                           (fetch (POSITION YCOORD)
                                                                                              of OPPT)))
                                                                        OPPT)
                                                                       [OPPT (CREATEPOSITION (\RANGELIMIT
                                                                                              (fetch (REGION LEFT)
                                                                                                 of GRIDREGION)
                                                                                              (fetch (POSITION XCOORD)
                                                                                                 of OPPT)
                                                                                              (fetch (REGION RIGHT)
                                                                                                 of GRIDREGION))
                                                                                             (\RANGELIMIT (fetch (REGION BOTTOM)
                                                                                                             of GRIDREGION)
                                                                                                          (fetch (POSITION YCOORD)
                                                                                                             of OPPT)
                                                                                                          (fetch (REGION TOP)
                                                                                                             of GRIDREGION]
                                                                       ((INSIDE? GRIDREGION (fetch (POSITION XCOORD)
                                                                                               of BASEPT)
                                                                                            (fetch (POSITION YCOORD) of BASEPT))
                                                                        BASEPT)
                                                                       (T (CREATEPOSITION (\RANGELIMIT
                                                                                           (fetch (REGION LEFT)
```

```
                                                                                    of GRIDREGION)
                                                                                 (fetch (POSITION XCOORD)
                                                                                    of BASEPT)
                                                                                 (fetch (REGION RIGHT)
                                                                                    of GRIDREGION))
                                                                           (\RANGELIMIT (fetch (REGION BOTTOM)
                                                                                          of GRIDREGION)
                                                                                  (fetch (POSITION YCOORD)
                                                                                     of BASEPT)
                                                                                  (fetch (REGION TOP) of GRIDREGION
                                                                                                    ]
        (SETQ NEWREGION (CREATEREGION (GRIDXCOORD (DSPXSCREENTOWINDOW (fetch (REGION LEFT) of RAWREGION)
                                                                      WINDOW)
                                                 GRIDSPEC)
                                      (GRIDYCOORD (DSPYSCREENTOWINDOW (fetch (REGION BOTTOM) of RAWREGION)
                                                                      WINDOW)
                                                  GRIDSPEC)
                                      0 0))
        (replace (REGION WIDTH) of NEWREGION with (- (ADD1 (GRIDXCOORD (DSPXSCREENTOWINDOW
                                                                        (fetch (REGION RIGHT) of RAWREGION)
                                                                       WINDOW)
                                                                  GRIDSPEC))
                                                     (fetch (REGION LEFT) of NEWREGION)))
        (replace (REGION HEIGHT) of NEWREGION with (- (ADD1 (GRIDYCOORD (DSPYSCREENTOWINDOW
                                                                         (fetch (REGION TOP) of RAWREGION)
                                                                         WINDOW)
                                                                   GRIDSPEC))
                                                      (fetch (REGION BOTTOM) of NEWREGION)))
        NEWREGION])
```

(\RANGELIMIT

```
  [LAMBDA (MIN VAL MAX)
    (IMAX MIN (IMIN MAX VAL])

)

(DEFINEQ
```

(MOUSECONFIRM

```
  [LAMBDA (PROMPTSTRING HELPSTRING WINDOW DON'TCLEAR/MAINW)      (* bvm%: " 2-May-86 15:19")

          (* * Changes the cursor to a "little mouse" ; prints a prompt; and waits for the user to press and then release a mouse
          button. If the LEFT was the final one release then return T otherwise return NIL --
          uses PROMPTWINDOW unless provided a window * *)

    (DECLARE (GLOBALVARS MOUSECONFIRMCURSOR))
    (LET ((HELPSTR (COND
                       ((EQ HELPSTRING T)
                        NIL)
                       ((NULL HELPSTRING)
                        "Click LEFT to confirm, RIGHT to abort.")
                       (T HELPSTRING)))
          PWINDOW)
         (COND
            ((EQ PROMPTSTRING T)
             (SETQ PROMPTSTRING NIL)))
         (COND
            [(OR PROMPTSTRING HELPSTR)
             [FRESHLINE (OR WINDOW (SETQ WINDOW (COND
                                                   [(WINDOWP DON'TCLEAR/MAINW)
                                                                 (* Open a prompt window from this window)
                                                    (SETQ PWINDOW (GETPROMPTWINDOW DON'TCLEAR/MAINW
                                                                                   (COND
                                                                                      ((NULL PROMPTSTRING)
                                                                                       HELPSTR)
                                                                                      ((NULL HELPSTR)
                                                                                       PROMPTSTRING)
                                                                                      (T (CONCAT PROMPTSTRING "  " HELPSTR
                                                                                                 ]
                                                   (T PROMPTWINDOW]
              [COND
                 (PROMPTSTRING (printout WINDOW PROMPTSTRING)
                               (COND
                                  (HELPSTR (SPACES 2 WINDOW]
              (COND
                 (HELPSTR (printout WINDOW HELPSTR]
            (T                                                  (* Didn't print anything, so don't clear anything)
               (SETQ DON'TCLEAR/MAINW T)))
         (PROG1 (RESETFORM (CURSOR MOUSECONFIRMCURSOR)
                           (until (MOUSESTATE (OR LEFT MIDDLE RIGHT)))
                           (bind (LEFTDOWN _ (LASTMOUSESTATE LEFT)) until (MOUSESTATE UP) do
                                                                 (* If buttons are still down, but not LEFT, user must have
                                                                 changed mind)
                                                                                    (SETQ LEFTDOWN
                                                                                     (LASTMOUSESTATE LEFT))
                             finally (RETURN LEFTDOWN)))
```

```
            (COND
               (PWINDOW                                    (* Close prompt window)
                      (CLOSEW PWINDOW))
               ((NULL DON'TCLEAR/MAINW)
                (CLEARW WINDOW)))])

)

(RPAQ MOUSECONFIRMCURSOR (CURSORCREATE '
                                      'NIL 8 8))

(DEFINEQ

(NEAREST/PT/ON/GRID
  [LAMBDA (PT GRIDN)                                       ; Edited 15-Mar-94 10:40 by sybalsky
                                                           (* finds the point on a grid of multiple GRIDN closest to PT.)
    (create POSITION
           XCOORD _ (NEAREST/MULTIPLE (fetch (POSITION XCOORD) of PT)
                          GRIDN)
           YCOORD _ (NEAREST/MULTIPLE (fetch (POSITION YCOORD) of PT)
                          GRIDN)
       smashing PT])

(PTON10GRID
  [LAMBDA (FIXEDPT MOVINGPT)                               (* rrb " 6-AUG-81 08:45")
                                                           (* insists that a pt be on a 10 grid)
    (COND
       (MOVINGPT (NEAREST/PT/ON/GRID MOVINGPT 10))
       (T (NEAREST/PT/ON/GRID FIXEDPT 10])

(NEAREST/MULTIPLE
  [LAMBDA (X N)                                            (* rrb " 6-AUG-81 08:42")
                                                           (* finds the multiple of N that is nearest to X)
    (COND
       ((IGREATERP X 0)
        (ITIMES (IQUOTIENT (IPLUS X (IQUOTIENT N 2))
                           N)
                N))
       (T (ITIMES (IQUOTIENT (IDIFFERENCE X (IQUOTIENT N 2))
                             N)
                  N])

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS IABS MACRO [OPENLAMBDA (A)
                       (COND
                          ((IGEQ A 0)
                           A)
                          (T (IMINUS A])
)

;; END EXPORTED DEFINITIONS

(READVARS-FROM-STRINGS '(DASHEDSHADE)
       "({(READBITMAP)(16 16
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"@@OO%"
       %"OO@@%"
       %"OO@@%"
       %"OO@@%"
       %"OO@@%"
       %"OO@@%"
       %"OO@@%"
       %"OO@@%")})
       ")

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS CROSSHAIRS EXPANDINGBOX FORCEPS BOXCURSOR LOCKEDSPOT OLDEXPANDINGBOX LowerLeftCursor
       UpperRightCursor UpperLeftCursor LowerRightCursor)
)
```

```
(RPAQ CROSSHAIRS (CURSORCREATE '
                        'NIL 7 7))

(RPAQ EXPANDINGBOX (CURSORCREATE '
                       'NIL 0 13))

(RPAQ FORCEPS (CURSORCREATE '
                     'NIL 7 15))

(RPAQ BOXCURSOR (CURSORCREATE '
                       'NIL 7 7))

(RPAQ LOCKEDSPOT (CURSORCREATE '
                       'NIL 7 7))

(RPAQ OLDEXPANDINGBOX (CURSORCREATE '
                            'NIL 7 7))

(RPAQ LowerLeftCursor (CURSORCREATE '
                       'NIL 0 0))

(RPAQ UpperRightCursor (CURSORCREATE '
                        'NIL 15 15))

(RPAQ UpperLeftCursor (CURSORCREATE '
                       'NIL 0 15))

(RPAQ LowerRightCursor (CURSORCREATE '
                        'NIL 15 0))

(DEFINEQ
```

(\SW2BM

```
  [LAMBDA (P PR Q QR)                                   (* edited%: "26-Jan-86 13:23")

          (* Switches the areas of P and Q defined by the regions PR and QR respectively)

    (PROG (PL PH PW PB QL QH QW QB)
          [COND
             (PR (SETQ PL (fetch (REGION LEFT) of PR))
                 (SETQ PB (fetch (REGION BOTTOM) of PR))
                 (SETQ PH (fetch (REGION HEIGHT) of PR))
                 (SETQ PW (fetch (REGION WIDTH) of PR)))
             (T (SETQ PL (SETQ PB 0))
                (SETQ PW (fetch (BITMAP BITMAPWIDTH) of P))
                (SETQ PH (fetch (BITMAP BITMAPHEIGHT) of P]
          [COND
             (QR (SETQ QL (fetch (REGION LEFT) of QR))
                 (SETQ QB (fetch (REGION BOTTOM) of QR))
                 (SETQ QW (fetch (REGION WIDTH) of QR))
                 (SETQ QH (fetch (REGION HEIGHT) of QR)))
             (T (SETQ QL (SETQ QB 0))
                (SETQ QW (fetch (BITMAP BITMAPWIDTH) of Q))
                (SETQ QH (fetch (BITMAP BITMAPHEIGHT) of Q]
          (PROG ((CL (IMAX (IMINUS PL)
                           (IMINUS QL)
                           0))
                 (CB (IMAX (IMINUS PB)
                           (IMINUS QB)
                           0)))
                (PROG ((XP (IPLUS CL PL))
                       (YP (IPLUS CB PB))
                       (XQ (IPLUS CL QL))
                       (YQ (IPLUS CB QB))
                       CW CH)
                      (SETQ CW (IMIN (IDIFFERENCE (IMIN (fetch (BITMAP BITMAPWIDTH) of P)
                                                        (IPLUS PL PW))
                                                  XP)
                                     (IDIFFERENCE (IMIN (fetch (BITMAP BITMAPWIDTH) of Q)
                                                        (IPLUS QL QW))
                                                  XQ)))
                      (SETQ CH (IMIN (IDIFFERENCE (IMIN (fetch (BITMAP BITMAPHEIGHT) of P)
                                                        (IPLUS PB PH))
                                                  YP)
```

```
                                    (IDIFFERENCE (IMIN (fetch (BITMAP BITMAPHEIGHT) of Q)
                                                       (IPLUS QB QH))
                                                 YQ)))
                            (UNINTERRUPTABLY
                                (BITBLT P XP YP Q XQ YQ CW CH 'INPUT 'INVERT)
                                (BITBLT Q XQ YQ P XP YP CW CH 'INPUT 'INVERT)
                                (BITBLT P XP YP Q XQ YQ CW CH 'INPUT 'INVERT))])
```

(COMPOSEREGS

```
  [LAMBDA (INNER OUTER)                                          (* rrb "19-MAR-82 09:35")
```

(* Converts INNER from OUTER relative coords to same units as OUTER - inverse of TRANSLATEREGS)

```
    (create REGION
           LEFT _ (IPLUS (fetch (REGION LEFT) of OUTER)
                         (fetch (REGION LEFT) of INNER))
           BOTTOM _ (IPLUS (fetch (REGION BOTTOM) of OUTER)
                           (fetch (REGION BOTTOM) of INNER))
       using INNER])
```

(TRANSLATEREG

```
  [LAMBDA (INNER OUTER)                                          (* rrb "19-MAR-82 09:35")
                                                                 (* Translates a nested INNER region to OUTER region relative
                                                                 coordinates)
    (create REGION
           LEFT _ (IDIFFERENCE (fetch (REGION LEFT) of INNER)
                               (fetch (REGION LEFT) of OUTER))
           BOTTOM _ (IDIFFERENCE (fetch (REGION BOTTOM) of INNER)
                                 (fetch (REGION BOTTOM) of OUTER))
           WIDTH _ (fetch (REGION WIDTH) of INNER)
           HEIGHT _ (fetch (REGION HEIGHT) of INNER])

)

;; Bitmap and shade editors

(DEFINEQ
```

(EDITBM

```
  [LAMBDA (BMSPEC)                                               ; Edited 31-Aug-87 12:28 by FS
```

;;; A simple bitmap editor.

;; The edit part of the display is from 0 to MAXGRIDWIDTH in width and from 0 to MAXGRIDHEIGHT in height. The commands and display area ;; for the bitmap being edited are above the edit region.

```
    (DECLARE (GLOBALVARS \CURSORDESTWIDTH \CURSORDESTHEIGHT))
    (PROG (BMW BMWINTERIOR BMWWIDTH BMWHEIGHT WIDTH HEIGHT BM CR ORIGBM GRIDSQUARE BPP ORIGBPP ORIGWIDTH)
                                                                 ; set ORIGBM to the input bitmap if any and BM to a copy of it for
                                                                 ; editting.
          [COND
             ((OR (EQ BMSPEC CursorBitMap)
                  (AND (EQ BMSPEC 'CursorBitMap)
                       (SETQ BMSPEC CursorBitMap)))              ; editing cursor, save old value and make changes to the original.
              (SETQ ORIGBM (BITMAPCOPY CursorBitMap))
              (SETQ BM CursorBitMap))
             [(BITMAPP BMSPEC)
              (SETQ BM (BITMAPCOPY (SETQ ORIGBM BMSPEC]
             [(LITATOM BMSPEC)
              (COND
                 ([BITMAPP (SETQ ORIGBM (EVALV BMSPEC 'EDITBM]   ; use value.
                  (SETQ BM (BITMAPCOPY ORIGBM)))
                 (T (SETQ ORIGBM NIL)
                    (SETQ BM (\READBMDIMENSIONS]
             ((REGIONP BMSPEC)                                   ; if BMSPEC is a region, treat it as a region of the screen.
              (SETQ BM (BITMAPCREATE (fetch (REGION WIDTH) of BMSPEC)
                                     (fetch (REGION HEIGHT) of BMSPEC)
                                     (BITSPERPIXEL \CURSORDESTINATION)))
                                                                 ; note that bm has initial bits in it.
              (SETQ ORIGBM BMSPEC)
              (BITBLT \CURSORDESTINATION (fetch (REGION LEFT) of BMSPEC)
                     (fetch (REGION BOTTOM) of BMSPEC)
                     BM 0 0 NIL NIL 'INPUT 'REPLACE))
             ((WINDOWP BMSPEC)
              (SETQ ORIGBM BMSPEC)
              ;; FS: Seems too big below, why not ClipRegion's Width & Height? That's all that's used...
              (SETQ BM (BITMAPCREATE (WINDOWPROP BMSPEC 'WIDTH)
                                     (WINDOWPROP BMSPEC 'HEIGHT)
                                     (BITSPERPIXEL BMSPEC)))     ; open the window and bring it to the top.
              (TOTOPW BMSPEC)
              (SETQ CR (DSPCLIPPINGREGION NIL BMSPEC))
              (BITBLT BMSPEC (fetch (REGION LEFT) of CR)
                     (fetch (REGION BOTTOM) of CR)
                     BM 0 0 (fetch (REGION WIDTH) of CR)
```

```
                (fetch (REGION HEIGHT) of CR)))
      (T                                                    ; otherwise create a bitmap
         (SETQ BM (\READBMDIMENSIONS]
   (if (OR (EQ (BITMAPHEIGHT BM)
               0)
           (EQ (BITMAPWIDTH BM)
               0))
       then (ERROR "Can't edit a bitmap with no bits in it." BMSPEC))
   (SETQ BPP (BITSPERPIXEL \CURSORDESTINATION))
   (SETQ ORIGBPP (fetch (BITMAP BITMAPBITSPERPIXEL) of BM))
   [COND
      ((NOT (EQ BPP ORIGBPP))
       ;; save the actual number of bits per pixel and set it to BPP in the bitmap being edited so that it can be BITBLT ed on the screen.
       (SETQ ORIGWIDTH (fetch (BITMAP BITMAPWIDTH) of BM))
       (replace (BITMAP BITMAPBITSPERPIXEL) of BM with BPP)
       (SETQ WIDTH (IQUOTIENT (ITIMES ORIGBPP ORIGWIDTH)
                              BPP))
       (replace (BITMAP BITMAPWIDTH) of BM with WIDTH))
      (T (SETQ WIDTH (fetch (BITMAP BITMAPWIDTH) of BM]
   (SETQ HEIGHT (fetch (BITMAP BITMAPHEIGHT) of BM))
;; Calculate a default window size.  Start by calculating the grid size from the bitmap size.
   (SETQ GRIDSQUARE (IMAX (IMIN (IQUOTIENT (IDIFFERENCE (IQUOTIENT (ITIMES \CURSORDESTWIDTH 2)
                                                                    3)
                                                         GRIDTHICKNESS)
                                           WIDTH)
                                (IQUOTIENT (IDIFFERENCE (IQUOTIENT (ITIMES \CURSORDESTHEIGHT 2)
                                                                   3)
                                                        (ITIMES GRIDTHICKNESS 2))
                                           (ADD1 HEIGHT))
                                NORMALGRIDSQUARE)
                          MINGRIDSQUARE))
   (SETQ BMWWIDTH (IMIN (IPLUS (ITIMES GRIDSQUARE WIDTH)
                               GRIDTHICKNESS)
                        (IQUOTIENT (ITIMES \CURSORDESTWIDTH 2)
                               3)))
   (SETQ BMWHEIGHT (IMIN (IPLUS (ITIMES HEIGHT (ADD1 GRIDSQUARE))
                                (ITIMES GRIDTHICKNESS 2)
                                1)
                         (IQUOTIENT (ITIMES \CURSORDESTHEIGHT 2)
                                3)))
   (SETQ BMW (CREATEW (GETBOXREGION (WIDTHIFWINDOW BMWWIDTH)
                                    (HEIGHTIFWINDOW BMWHEIGHT T)
                                    NIL NIL NIL "Indicate the position for the Bitmap Edit window.")
                      "Bitmap Editor"))
   (WINDOWPROP BMW 'BM BM)
   (WINDOWPROP BMW 'SCROLLFN (FUNCTION EDITBMSCROLLFN))
   (WINDOWPROP BMW 'RESHAPEFN (FUNCTION EDITBMRESHAPEFN))
   (WINDOWPROP BMW 'REPAINTFN (FUNCTION EDITBMREPAINTFN))
   (WINDOWPROP BMW 'BUTTONEVENTFN (FUNCTION EDITBMBUTTONFN))
   (WINDOWPROP BMW 'CLOSEFN (FUNCTION EDITBMCLOSEFN))
   (WINDOWPROP BMW 'XOFFSET 0)
   (WINDOWPROP BMW 'YOFFSET 0)
   (WINDOWPROP BMW 'DXOFFSET 0)
   (WINDOWPROP BMW 'DYOFFSET 0)
   (WINDOWPROP BMW 'ORIGINALBITMAP ORIGBM)
   (WINDOWPROP BMW 'FINISHEDFLG NIL)
   (WINDOWPROP BMW 'COLOR (MAXIMUMCOLOR BPP))
   (WINDOWPROP BMW 'GRIDON T)                               ; call reshapefn to initialize the display and values
   (EDITBMRESHAPEFN BMW NIL NIL NIL (NOT ORIGBM))           ; start a mouse process in case this process is the mouse
                                                            ; process.
   (SPAWN.MOUSE)
   (while (NOT (WINDOWPROP BMW 'FINISHEDFLG)) do (DISMISS 500))
                                                            ; remove the closefn before closing the window.
   (WINDOWPROP BMW 'CLOSEFN NIL)
   (CLOSEW BMW)
   (COND
      ((NOT (EQ ORIGBPP BPP))
       (replace (BITMAP BITMAPBITSPERPIXEL) of BM with ORIGBPP)
       (replace (BITMAP BITMAPWIDTH) of BM with ORIGWIDTH)))
   (RETURN (COND
              ((EQ T (WINDOWPROP BMW 'FINISHEDFLG))         ; editor exited via ok, stuff contents into original bitmap.
               (COND
                  ((EQ BMSPEC CursorBitMap)                 ; editting happened in original, leave it alone.
                   CursorBitMap)
                  ((REGIONP ORIGBM)                         ; put it back into the screen.
                   (BITBLT BM 0 0 \CURSORDESTINATION (fetch (REGION LEFT) of ORIGBM)
                           (fetch (REGION BOTTOM) of ORIGBM)
                           (fetch (REGION WIDTH) of ORIGBM)
                           (fetch (REGION HEIGHT) of ORIGBM)
                           'INPUT
                           'REPLACE)
                   BM)
                  ((WINDOWP ORIGBM)                         ; put it back into the window
                   (BITBLT BM 0 0 ORIGBM (fetch (REGION LEFT) of CR)
                           (fetch (REGION BOTTOM) of CR)
```

```
                              (fetch (REGION WIDTH) of CR)
                              (fetch (REGION HEIGHT) of CR)
                              'INPUT
                              'REPLACE)
                        BM)
                       (ORIGBM (BITBLT BM 0 0 ORIGBM 0 0 WIDTH HEIGHT)
                               [COND
                                  ((AND BMSPEC (LITATOM BMSPEC))
                                                          ; if spec was an atom without a bm value, set it.  in the
                                                          ; environment above EDITBM.
                                   (MARKASCHANGED BMSPEC 'VARS)
                                   (STKEVAL 'EDITBM (LIST 'SETQQ BMSPEC BM]
                               ORIGBM)
                        (T BM)))
                  (T                                      ; error exit, if cursor return it to original value.
                     (COND
                        ((EQ BMSPEC CursorBitMap)
                         (BITBLT ORIGBM NIL NIL CursorBitMap)))
                     (ERROR!])
```

(EDITBMSCROLLFN

```
  [LAMBDA (W DX DY)                                       ; Edited 31-Aug-87 13:29 by FS
                                                          ; Do scrolling for the bitmap editor.
    (PROG (GRIDSPEC REG WHEIGHT WWIDTH (DXGRID 0)
                 (DYGRID 0)
                 EXTENT EXTENTWIDTH EXTENTHEIGHT GILEFT GIBOTTOM GIHEIGHT GWIDTH GHEIGHT GRIDINTERIOR EBMXLIMIT
                 EBMYLIMIT EBMXOFFSET EBMYOFFSET BM BITMAPWIDTH BITMAPHEIGHT BITSWIDE BITSHIGH DXOFFSET DYOFFSET
                 )
          (SETQ GRIDSPEC (WINDOWPROP W 'GRIDSPEC))
          (SETQ REG (WINDOWPROP W 'REGION))
          (SETQ WHEIGHT (WINDOWPROP W 'HEIGHT))
          (SETQ WWIDTH (WINDOWPROP W 'WIDTH))
          (SETQ GRIDINTERIOR (WINDOWPROP W 'GRIDINTERIOR))
          (SETQ EBMXOFFSET (WINDOWPROP W 'XOFFSET))
          (SETQ EBMYOFFSET (WINDOWPROP W 'YOFFSET))
          (SETQ BM (WINDOWPROP W 'BM))
          (SETQ BITMAPWIDTH (fetch BITMAPWIDTH of BM))
          (SETQ BITMAPHEIGHT (fetch BITMAPHEIGHT of BM))
          (SETQ BITSWIDE (WINDOWPROP W 'BITSWIDE))
          (SETQ BITSHIGH (WINDOWPROP W 'BITSHIGH))
          (SETQ DXOFFSET (WINDOWPROP W 'DXOFFSET))
          (SETQ DYOFFSET (WINDOWPROP W 'DYOFFSET))
          (SETQ EBMXLIMIT (IPLUS EBMXOFFSET BITSWIDE))
          (SETQ EBMYLIMIT (IPLUS EBMYOFFSET BITSHIGH))
          (COND
             (GRIDSPEC (SETQ GILEFT (fetch (REGION LEFT) of GRIDINTERIOR))
                    (SETQ GIBOTTOM (fetch (REGION BOTTOM) of GRIDINTERIOR))
                    (SETQ GIHEIGHT (fetch (REGION HEIGHT) of GRIDINTERIOR))
                    (SETQ GWIDTH (fetch (REGION WIDTH) of GRIDSPEC))
                    (SETQ GHEIGHT (fetch (REGION HEIGHT) of GRIDSPEC))
                    (SETQ EXTENT (WINDOWPROP W 'EXTENT))
                    (SETQ EXTENTWIDTH (fetch (REGION WIDTH) of EXTENT))
                    (SETQ EXTENTHEIGHT (fetch (REGION HEIGHT) of EXTENT))
                                                          ; Make a horizontal adjustment
                    (COND
                       ((FLOATP DX)                       ; Horizontal thumbing
                        [WINDOWPROP W 'XOFFSET (SETQ EBMXOFFSET (FIX (TIMES (IDIFFERENCE BITMAPWIDTH BITSWIDE)
                                                                            DX]
                        (replace (REGION LEFT) of EXTENT with (IMINUS (QUOTIENT (TIMES EBMXOFFSET EXTENTWIDTH)
                                                                                BITMAPWIDTH)))
                                                          (* BLTSHADE WHITESHADE W GILEFT GIBOTTOM
                                                          SCREENWIDTH SCREENHEIGHT
                                                          (QUOTE REPLACE) GRIDINTERIOR)
                        (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W T))
                       ((ILESSP DX 0)                     ; moving to the left.
                                                          ; determine how many grid points to move.
                        (SETQ DXGRID (IMIN (GRIDXCOORD (IMINUS DX)
                                                       GRIDSPEC)
                                           (IDIFFERENCE BITMAPWIDTH EBMXLIMIT)))
                        (COND
                           ((NOT (IGREATERP DXGRID 0))    ; right edge is at the right margin
                            (RETURN)))
                        (WINDOWPROP W 'XOFFSET (SETQ EBMXOFFSET (IPLUS EBMXOFFSET DXGRID)))
                                                          ; update EXTENT bar
                        (replace (REGION LEFT) of EXTENT with (IMAX (IMINUS (QUOTIENT (TIMES EBMXOFFSET
                                                                                             EXTENTWIDTH)
                                                                                      BITMAPWIDTH))
                                                                    (IMINUS EXTENTWIDTH)))
                                                          ; move image to the left.
                        (BITBLT W (IPLUS GILEFT (TIMES DXGRID GWIDTH))
                               GIBOTTOM W GILEFT GIBOTTOM SCREENWIDTH SCREENHEIGHT 'INPUT 'REPLACE NIL
                               GRIDINTERIOR)              ; clear the newly exposed area.
                        (BLTSHADE WHITESHADE W (IPLUS GILEFT (TIMES (IDIFFERENCE BITSWIDE DXGRID)
                                                                    GWIDTH))
                               GIBOTTOM SCREENWIDTH SCREENHEIGHT 'REPLACE GRIDINTERIOR)
                        (RESETGRID.NEW BM GRIDSPEC DXGRID BITSHIGH (IDIFFERENCE BITSWIDE DXGRID)
```

```
                0 W))
            ((ILESSP 0 DX)                          ; determine how many grid point to the left to move.
             (SETQ DXGRID (IMIN EBMXOFFSET (GRIDXCOORD DX GRIDSPEC)))
             (COND
                ((NOT (IGREATERP DXGRID 0))         ; left edge is at the left margin
                 (RETURN)))
             (WINDOWPROP W 'XOFFSET (SETQ EBMXOFFSET (IDIFFERENCE EBMXOFFSET DXGRID)))
                                                    ; update REGION bar
             (replace (REGION LEFT) of EXTENT with (IMIN (IMINUS (IQUOTIENT (TIMES EBMXOFFSET
                                                                                   EXTENTWIDTH)
                                                                            BITMAPWIDTH))
                                                         0))
                                                    ; move image to the right.
             (BITBLT W GILEFT GIBOTTOM W (IPLUS GILEFT (TIMES DXGRID GWIDTH))
                    GIBOTTOM SCREENWIDTH SCREENHEIGHT 'INPUT 'REPLACE NIL GRIDINTERIOR)
                                                    ; clear the newly exposed area.
             (BLTSHADE WHITESHADE W GILEFT GIBOTTOM (TIMES DXGRID GWIDTH)
                    GIHEIGHT
                    'REPLACE)
             (RESETGRID.NEW BM GRIDSPEC DXGRID BITSHIGH 0 0 W)))
                                                    ; Make a vertical adjustment
         (COND
            ((FLOATP DY)                            ; Vertical Thumbing
             [WINDOWPROP W 'YOFFSET (SETQ EBMYOFFSET (FIX (TIMES (IDIFFERENCE BITMAPHEIGHT BITSHIGH)
                                                                 (FDIFFERENCE 1.0 DY]
                                                    ; set EXTENT bar
             (replace (REGION BOTTOM) of EXTENT with (IMINUS (QUOTIENT (TIMES EBMYOFFSET EXTENTHEIGHT)
                                                                       BITMAPHEIGHT)))
                                                    ; Clear Window
                                                    (* BLTSHADE WHITESHADE W GILEFT GIBOTTOM
                                                    SCREENWIDTH SCREENHEIGHT
                                                    (QUOTE REPLACE) GRIDINTERIOR)
                                                    ; Repaint the image using grid function
             (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W T))
            ((ILESSP DY 0)                          ; determine how many squares to move down.
             (SETQ DYGRID (IMIN (IDIFFERENCE (fetch (BITMAP BITMAPHEIGHT) of BM)
                                        EBMYLIMIT)
                                (GRIDYCOORD (IMIN GIHEIGHT (IMINUS DY))
                                        GRIDSPEC)))
             (COND
                ((NOT (IGREATERP DYGRID 0))         ; top edge is at the top margin
                 (RETURN)))
             (WINDOWPROP W 'YOFFSET (SETQ EBMYOFFSET (IPLUS EBMYOFFSET DYGRID)))
             (replace (REGION BOTTOM) of EXTENT with (IMAX (IMINUS (QUOTIENT (TIMES EBMYOFFSET
                                                                                   EXTENTHEIGHT)
                                                                             BITMAPHEIGHT))
                                                           (IMINUS EXTENTHEIGHT)))
             (BITBLT W GILEFT (IPLUS GIBOTTOM (ITIMES DYGRID GHEIGHT))
                    W GILEFT GIBOTTOM SCREENWIDTH SCREENHEIGHT 'INPUT 'REPLACE NIL GRIDINTERIOR)
                                                    (* BLTSHADE WHITESHADE W GILEFT
                                                    (IPLUS GIBOTTOM (ITIMES (IDIFFERENCE BITSHIGH
                                                    DYGRID) GHEIGHT)) SCREENWIDTH SCREENHEIGHT
                                                    (QUOTE REPLACE) GRIDINTERIOR)
             (RESETGRID.NEW BM GRIDSPEC BITSWIDE DYGRID 0 (IDIFFERENCE BITSHIGH DYGRID)
                    W T))
            ((ILESSP 0 DY)                          ; moving up;  determine how may grid squares to move.
             (SETQ DYGRID (IMIN EBMYOFFSET (GRIDYCOORD (IMIN GIHEIGHT DY)
                                                  GRIDSPEC)))
             (COND
                ((NOT (IGREATERP DYGRID 0))         ; bottom edge is at the bottom margin
                 (RETURN)))
             (WINDOWPROP W 'YOFFSET (SETQ EBMYOFFSET (IDIFFERENCE EBMYOFFSET DYGRID)))
             (replace (REGION BOTTOM) of EXTENT with (IMIN (IMINUS (QUOTIENT (TIMES EBMYOFFSET
                                                                                   EXTENTHEIGHT)
                                                                             BITMAPHEIGHT))
                                                           0))
             (BITBLT W GILEFT GIBOTTOM W GILEFT (IPLUS GIBOTTOM (ITIMES DYGRID GHEIGHT))
                    SCREENWIDTH SCREENHEIGHT 'INPUT 'REPLACE NIL GRIDINTERIOR)
                                                    (* BLTSHADE WHITESHADE W GILEFT GIBOTTOM
                                                    (fetch (REGION WIDTH) of GRIDINTERIOR)
                                                    (ITIMES DYGRID GHEIGHT) (QUOTE REPLACE))
             (RESETGRID.NEW BM GRIDSPEC BITSWIDE DYGRID 0 0 W T)))
         ;; This call to GRID is unnecessary as the grid dots get filled in earlier.
         ;; (COND ((WINDOWPROP W 'GRIDON) (GRID GRIDSPEC BITSWIDE BITSHIGH 'POINT W)))
         [COND
            ([OR (ILESSP EBMXOFFSET DXOFFSET)
                 (ILESSP EBMYOFFSET DYOFFSET)
                 [IGREATERP (IPLUS EBMXOFFSET BITSWIDE)
                        (IPLUS DXOFFSET (WINDOWPROP W 'BMDISPLAYWIDTH]
                 (IGREATERP (IPLUS EBMYOFFSET BITSHIGH)
                        (IPLUS DYOFFSET (WINDOWPROP W 'BMDISPLAYHEIGHT]
                                                    ; Adjust the display region left lower corner so the selected
                                                    ; region is near the center.
             [WINDOWPROP W 'DXOFFSET (SETQ DXOFFSET (IMAX 0 (IMIN (IDIFFERENCE (fetch (BITMAP
                                                                                    BITMAPWIDTH
                                                                                     )
```

```
                                                                                    of BM)
                                                                              (WINDOWPROP W
                                                                                    'BMDISPLAYWIDTH))
                                                                        (IDIFFERENCE
                                                                         (IPLUS EBMXOFFSET (LRSH BITSWIDE 1
                                                                                                  ))
                                                                         (LRSH (WINDOWPROP W
                                                                                    'BMDISPLAYWIDTH)
                                                                               1]
                        (WINDOWPROP W 'DYOFFSET (SETQ DYOFFSET (IMAX 0 (IMIN (IDIFFERENCE (fetch (BITMAP
                                                                                                   BITMAPHEIGHT
                                                                                                     )
                                                                                    of BM)
                                                                              (WINDOWPROP W
                                                                                    'BMDISPLAYHEIGHT))
                                                                        (IDIFFERENCE
                                                                         (IPLUS EBMYOFFSET (LRSH BITSHIGH 1
                                                                                                  ))
                                                                         (LRSH (WINDOWPROP W
                                                                                    'BMDISPLAYHEIGHT)
                                                                               1]
                     (UPDATE/BM/DISPLAY BM W])
```

(EDITBMCLOSEFN

```
[LAMBDA (BMW)                                          ; Edited 23-Feb-94 16:07 by turpiN:mv:envos
```

;; the close function for a bitmap edit window. For now do what a STOP would have done.

;; FS: Assuming this window won't be reused, flush the temporary bm.

```
(WINDOWPROP BMW 'TEMPBM NIL)
(WINDOWPROP BMW 'GRIDBM NIL)
(WINDOWPROP BMW 'FINISHEDFLG 'KILL)
(COND
   ((WINDOWPROP BMW 'COORDWIN)
    (DETACHWINDOW (WINDOWPROP BMW 'COORDWIN)
           BMW)
    (CLOSEW (WINDOWPROP BMW 'COORDWIN))
    (WINDOWPROP BMW 'COORDWIN NIL])
```

(TILEAREA

```
[LAMBDA (LFT BTM WDTH HGHT SRCBM WIN)                  ; Edited 27-Aug-87 21:20 by FS
```

;; lays tiles out in an area of a window. This function only provided for backwards compatibility.

```
(BLTPATTERN.REPLACEDISPLAY SRCBM 0 0 (BITMAPWIDTH SRCBM)
       (BITMAPHEIGHT SRCBM)
       WIN LFT BTM WDTH HGHT])
```

(EDITBMBUTTONFN

```
[LAMBDA (W)                                            ; Edited 15-Mar-94 10:33 by sybalsky
                                                       ; Edited  5-Mar-92 15:54 by jds
```

;; inner function of bitmap editor.

```
(DECLARE (GLOBALVARS \CURRENTCURSOR))
(PROG (GRIDX0 GRIDY0 BITMAPWIDTH BITMAPHEIGHT NEWGRIDSIZE PAINTW ORIGBM GRIDSPEC GRIDINTERIOR BM BITSWIDE
             BITSHIGH WREGION XOFFSET YOFFSET DXOFFSET DYOFFSET DISPLAYREGION EXTENT BITSPERPIXEL CURSORBM)
      (SETQ GRIDSPEC (WINDOWPROP W 'GRIDSPEC))
      (SETQ GRIDINTERIOR (WINDOWPROP W 'GRIDINTERIOR))
      (SETQ BM (WINDOWPROP W 'BM))
      (SETQ BITSWIDE (WINDOWPROP W 'BITSWIDE))
      (SETQ BITSHIGH (WINDOWPROP W 'BITSHIGH))
      (SETQ WREGION (WINDOWPROP W 'REGION))
      (SETQ XOFFSET (WINDOWPROP W 'XOFFSET))
      (SETQ YOFFSET (WINDOWPROP W 'YOFFSET))
      (SETQ DXOFFSET (WINDOWPROP W 'DXOFFSET))
      (SETQ DYOFFSET (WINDOWPROP W 'DYOFFSET))
      (SETQ DISPLAYREGION (WINDOWPROP W 'DISPLAYREGION))
      (SETQ EXTENT (WINDOWPROP W 'EXTENT))
      (SETQ GRIDX0 (fetch (REGION LEFT) of GRIDSPEC))
      (SETQ GRIDY0 (fetch (REGION BOTTOM) of GRIDSPEC))
      (SETQ BITMAPWIDTH (fetch (BITMAP BITMAPWIDTH) of BM))
      (SETQ BITMAPHEIGHT (fetch (BITMAP BITMAPHEIGHT) of BM))
      (SETQ BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of BM))
      (SETQ COLOR (WINDOWPROP W 'COLOR))
```

;; mark the region of the bitmap that is being editted.

```
      (COND
         ((INSIDE? GRIDINTERIOR (LASTMOUSEX W)
                   (LASTMOUSEY W))
```

;; if cursor is inside, shade it.

```
          (\SHADEBITS BM GRIDSPEC GRIDINTERIOR W BITSWIDE BITSHIGH COLOR))
         ((INSIDE? DISPLAYREGION (LASTMOUSEX W)
                   (LASTMOUSEY W))
```

;; Run the menu foe re-windowing into the whole bitmap

```
(SELECTQ [MENU (COND
                  ((type? MENU EDITBMWINDOWMENU)
                   EDITBMWINDOWMENU)
                  ((SETQ EDITBMWINDOWMENU (create MENU
                                                  ITEMS _ '((Move 'Move "Selects a different part
                                                                        of the bitmap to edit."))
                                                  CENTERFLG _ T]
      (Move                                         ; move the editing window's location on the bitmap.
            (PROG (POS)
                  [SETQ POS (GETBOXPOSITION BITSWIDE BITSHIGH [IPLUS 4 (fetch (REGION LEFT)
                                                                           of WREGION)
                                                                      (- XOFFSET
                                                                         (WINDOWPROP W 'DXOFFSET]
                                            (IPLUS (WINDOWPROP W 'BMDISPLAYBOTTOM)
                                                   (- YOFFSET (WINDOWPROP W 'DYOFFSET))
                                                   4
                                                   (fetch (REGION BOTTOM) of WREGION]
                  [WINDOWPROP W 'XOFFSET (SETQ XOFFSET
                                               (IMIN (IDIFFERENCE BITMAPWIDTH BITSWIDE)
                                                     (IMAX [IPLUS (WINDOWPROP W 'DXOFFSET)
                                                                  (- (fetch (POSITION XCOORD) of POS)
                                                                     (IPLUS 4 (fetch (REGION LEFT)
                                                                                 of WREGION]
                                                           0]
                  [WINDOWPROP W 'YOFFSET
                         (SETQ YOFFSET
                               (IMAX 0 (IMIN (- BITMAPHEIGHT BITSHIGH)
                                             (- (IPLUS (WINDOWPROP W 'DYOFFSET)
                                                       (- (fetch (POSITION YCOORD) of POS)
                                                          (IPLUS (fetch (REGION BOTTOM) of WREGION)
                                                                 4)))
                                                (WINDOWPROP W 'BMDISPLAYBOTTOM]
                  (replace (REGION LEFT) of EXTENT with (IMINUS (QUOTIENT (TIMES XOFFSET
                                                                                 (fetch (REGION WIDTH)
                                                                                    of EXTENT))
                                                                           BITMAPWIDTH)))
                  (replace (REGION BOTTOM) of EXTENT with (IMINUS (QUOTIENT (TIMES YOFFSET
                                                                                   (fetch (REGION
                                                                                           HEIGHT
                                                                                           )
                                                                                      of EXTENT))
                                                                            BITMAPHEIGHT)))
                  [COND
                     ([OR (ILESSP XOFFSET DXOFFSET)
                          (ILESSP YOFFSET DYOFFSET)
                          [IGREATERP (IPLUS XOFFSET BITSWIDE)
                                 (IPLUS DXOFFSET (WINDOWPROP W 'BMDISPLAYWIDTH]
                          (IGREATERP (IPLUS YOFFSET BITSHIGH)
                                 (IPLUS DYOFFSET (WINDOWPROP W 'BMDISPLAYHEIGHT]
                      ;; Adjust the display region left lower corner so the selected region is near the center.
                      [WINDOWPROP W 'DXOFFSET (SETQ DXOFFSET
                                                    (IMAX 0 (IMIN (- (fetch (BITMAP BITMAPWIDTH)
                                                                        of BM)
                                                                     (WINDOWPROP W 'BMDISPLAYWIDTH))
                                                                  (- (IPLUS XOFFSET (LRSH BITSWIDE 1))
                                                                     (LRSH (WINDOWPROP W
                                                                                 'BMDISPLAYWIDTH)
                                                                           1]
                      (WINDOWPROP W 'DYOFFSET (SETQ DYOFFSET
                                                    (IMAX 0 (IMIN (- (fetch (BITMAP BITMAPHEIGHT)
                                                                        of BM)
                                                                     (WINDOWPROP W 'BMDISPLAYHEIGHT))
                                                                  (- (IPLUS YOFFSET (LRSH BITSHIGH 1))
                                                                     (LRSH (WINDOWPROP W
                                                                                 'BMDISPLAYHEIGHT)
                                                                           1]
                                                    (* DSPFILL GRIDINTERIOR WHITESHADE
                                                    (QUOTE REPLACE) W)
                  (UPDATE/BM/DISPLAY BM W)
            ;; FS: More useless code: (COND ((WINDOWPROP W 'GRIDON) (GRID GRIDSPEC BITSWIDE BITSHIGH 'POINT
            ;; W)))
                  (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W T)))
      NIL))
((LASTMOUSESTATE LEFT)
 (UPDATE/BM/DISPLAY/SELECTED/REGION W)
 (SETQ CURSORBM (BITMAPCREATE 16 16 (BITSPERPIXEL BM)))
 (BITBLT BM NIL NIL CURSORBM)
 [RESETFORM (CURSOR (CURSORCREATE CURSORBM NIL (fetch (CURSOR CUHOTSPOTX) of \CURRENTCURSOR)
                                  (fetch (CURSOR CUHOTSPOTY) of \CURRENTCURSOR)))
        (until (MOUSESTATE (NOT LEFT]
 (UPDATE/BM/DISPLAY/SELECTED/REGION W))
(T ;; the region being editted is inverted while the menu is active.  Each command must make sure that it is recomplemented.
   (UPDATE/BM/DISPLAY/SELECTED/REGION W)
   (SELECTQ [MENU (COND
```

```
                        ((type? MENU EDITBMMENU)
                         EDITBMMENU)
                        (T (SETQ EDITBMMENU (create MENU
                                                    ITEMS _ [APPEND (COND
                                                                       [(COLORDISPLAYP)
                                                                        '((Color 'Color "Choose color
                                                                                 to set bits with"]
                                                                       (T NIL))
                                                                    '((Paint 'Paint "Calls the window
                                                                             PAINT command on the
                                                                             bitmap.")
                                                                      (ShowAsTile 'ShowAsTile "tiles the
                                                                             upper part of the edit
                                                                             window with the bitmap.")
                                                                      (Grid% On/Off 'GridOnOff "Grid
                                                                             On/Off Switch")
                                                                      (GridSize_ 'GridSize_ "Allows
                                                                             setting of the size of a
                                                                             bit in the edit area.")
                                                                      (Reset 'Reset "Sets the bitmap
                                                                             back to the state at the
                                                                             start of this edit
                                                                             session.")
                                                                      (Clear 'Clear "Sets the entire
                                                                             bitmap to 0")
                                                                      (Blacken 'Blacken "Blacken a
                                                                             region of bits")
                                                                      (ClearBits 'ClearBits "Clear a
                                                                             region of bits")
                                                                      (Show% Coordinates 'ShowCoord
                                                                             "Toggle coordinate display
                                                                             window, displays on
                                                                             bit-changes")
                                                                      (Cursor_ 'Cursor_ "Puts the bitmap
                                                                             into the cursor and exits
                                                                             the editor.")
                                                                      (OK 'OK "Leaves the edit
                                                                          session.")
                                                                      (Abort 'Abort "Restores the bitmap
                                                                             to its original values and
                                                                             leaves the editor."]
                                                    CENTERFLG _ T]
                              (OK (WINDOWPROP W 'FINISHEDFLG T)
                                  (COND
                                     ((WINDOWPROP W 'COORDWIN)
                                      (DETACHWINDOW (WINDOWPROP W 'COORDWIN)
                                             W)
                                      (CLOSEW (WINDOWPROP W 'COORDWIN))
                                      (WINDOWPROP W 'COORDWIN NIL))))
                              (Abort (WINDOWPROP W 'FINISHEDFLG 'KILL)
                                     (COND
                                        ((WINDOWPROP W 'COORDWIN)
                                         (DETACHWINDOW (WINDOWPROP W 'COORDWIN)
                                                W)
                                         (CLOSEW (WINDOWPROP W 'COORDWIN))
                                         (WINDOWPROP W 'COORDWIN NIL))))
                              (Reset ;; allow the user to choose between everything or just visible part.  This also give the user a chance to change their
                                     ;; mind.
                                     (COND
                                        ((SELECTQ (\EDITBMHOWMUCH BM BITSWIDE BITSHIGH "RESET how much?")
                                              (VISIBLE [COND
                                                          [(SETQ ORIGBM (WINDOWPROP W 'ORIGINALBITMAP))
                                                           (COND
                                                              ((REGIONP ORIGBM)
                                                               (BITBLT \CURSORDESTINATION (IPLUS XOFFSET
                                                                                                 (fetch (REGION LEFT)
                                                                                                    of ORIGBM))
                                                                       (IPLUS YOFFSET (fetch (REGION BOTTOM) of ORIGBM))
                                                                       BM XOFFSET YOFFSET BITSWIDE BITSHIGH 'INPUT
                                                                       'REPLACE))
                                                              (T (BITBLT ORIGBM XOFFSET YOFFSET BM XOFFSET YOFFSET
                                                                         BITSWIDE BITSHIGH]
                                                          (T (BLTSHADE WHITESHADE BM XOFFSET YOFFSET BITSWIDE BITSHIGH
                                                                    'REPLACE]
                                                    T)
                                              (WHOLE [COND
                                                        [(SETQ ORIGBM (WINDOWPROP W 'ORIGINALBITMAP))
                                                         (COND
                                                            ((REGIONP ORIGBM)
                                                             (BITBLT \CURSORDESTINATION (fetch (REGION LEFT) of ORIGBM)
                                                                     (fetch (REGION BOTTOM) of ORIGBM)
                                                                     BM))
                                                            (T (BITBLT ORIGBM NIL NIL BM]
                                                        (T (BLTSHADE WHITESHADE BM NIL NIL NIL NIL 'REPLACE]
                                                    T)
                                              (PROGN (UPDATE/BM/DISPLAY/SELECTED/REGION W)
```

```
                        NIL))
                (\EDITBM/PUTUP/DISPLAY W BM GRIDSPEC GRIDINTERIOR BITSWIDE BITSHIGH))))
(Clear ;; allow the user to choose between everything or just visible part.  This also give the user a chance to change their
       ;; mind.
       (COND
          ((SELECTQ (\EDITBMHOWMUCH BM BITSWIDE BITSHIGH "CLEAR how much?")
                (VISIBLE (BLTSHADE WHITESHADE BM XOFFSET YOFFSET BITSWIDE BITSHIGH
                                   'REPLACE)
                         T)
                (WHOLE (\CLEARBM BM)
                       T)
                (PROGN (UPDATE/BM/DISPLAY/SELECTED/REGION W)
                       NIL))
           (DSPFILL GRIDINTERIOR WHITESHADE 'REPLACE W)
           (COND
              ((WINDOWPROP W 'GRIDON)
               (GRID GRIDSPEC BITSWIDE BITSHIGH 'POINT W)))
           (UPDATE/BM/DISPLAY BM W))))
(Blacken (LET ((REG (GETGRIDBOXREGION 0 0 GRIDSPEC GRIDINTERIOR W)))
           (BLTSHADE BLACKSHADE BM (+ (fetch (REGION LEFT) of REG)
                                      XOFFSET)
                  (+ (fetch (REGION BOTTOM) of REG)
                     YOFFSET)
                  (fetch (REGION WIDTH) of REG)
                  (fetch (REGION HEIGHT) of REG)
                  'REPLACE)
           (UPDATE/BM/DISPLAY BM W)
           (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W)))
(ClearBits (LET ((REG (GETGRIDBOXREGION 0 0 GRIDSPEC GRIDINTERIOR W)))
             (BLTSHADE WHITESHADE BM (+ (fetch (REGION LEFT) of REG)
                                        XOFFSET)
                    (+ (fetch (REGION BOTTOM) of REG)
                       YOFFSET)
                    (fetch (REGION WIDTH) of REG)
                    (fetch (REGION HEIGHT) of REG)
                    'REPLACE)
             (UPDATE/BM/DISPLAY BM W)
             (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W)))
(ShowCoord [LET [(COORDWIN (WINDOWPROP W 'COORDWIN]
             (COND
                (COORDWIN (DETACHWINDOW COORDWIN W)
                       (CLOSEW COORDWIN)
                       (WINDOWPROP W 'COORDWIN NIL))
                (T (ATTACHWINDOW (SETQ COORDWIN (CREATEW '(0 0 70 32)
                                                        "Coordinates" NIL T))
                                 W
                                 'TOP
                                 'LEFT)
                   (WINDOWPROP W 'COORDWIN COORDWIN])
(GridOnOff (COND
             ((NOT (WINDOWPROP W 'GRIDON)) ; Turn Grid On
              (WINDOWPROP W 'GRIDON T)
              (GRID GRIDSPEC BITSWIDE BITSHIGH 'POINT W)
              (UPDATE/BM/DISPLAY BM W)
              NIL)
             (T                                 ; Turn off grid
                (WINDOWPROP W 'GRIDON NIL) (* DSPFILL (create REGION LEFT _ 0 BOTTOM _ 0 WIDTH _
                                           (ADD1 (fetch (REGION WIDTH) of GRIDINTERIOR)) HEIGHT _
                                           (ADD1 (fetch (REGION HEIGHT) of GRIDINTERIOR)))
                                           WHITESHADE (QUOTE REPLACE) W)
                (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 W T)
                (UPDATE/BM/DISPLAY BM W)
                NIL)))
(GridSize_                                      ; sets the grid square size and calls the reshapefn.
          (COND
             ([SETQ NEWGRIDSIZE
               (NUMBERP (MENU (COND
                                ((TYPENAMEP GRIDSIZEMENU 'MENU)
                                 GRIDSIZEMENU)
                                (T (SETQ GRIDSIZEMENU
                                     (create MENU
                                             ITEMS _
                                             '(3 4 5 6 7 8 12 16 20 24 28 32)
                                             MENUROWS _ 4]
              (WINDOWPROP W 'GRIDSQUARE NEWGRIDSIZE)
              (EDITBMRESHAPEFN W))))
(ShowAsTile                                     ; tiles the upper part of the window with the bitmap so the user
                                                ; can see what it would be as a shade.
          (UPDATE/SHADE/DISPLAY BM W))
(Paint                                          ; call the window paint command on the contents of the bitmap.
       [SETQ PAINTW (CREATEW (create REGION
                                     LEFT _ (IQUOTIENT (- SCREENWIDTH BITMAPWIDTH)
                                                       2)
                                     BOTTOM _ (IQUOTIENT (- SCREENHEIGHT BITMAPHEIGHT)
                                                         2)
                                     WIDTH _ (WIDTHIFWINDOW BITMAPWIDTH)
```

```
                                                    HEIGHT _ (HEIGHTIFWINDOW BITMAPHEIGHT NIL]
                        (OPENW PAINTW)
                        (BITBLT BM 0 0 PAINTW)
                        (PAINTW PAINTW)
                        (COND
                           ((MENU (create MENU
                                          ITEMS _ '((YES T "Will put the newly painted bits back in the bitmap
                                                           being editted.")
                                                    (NO NIL "Will discard the painted bits, not changing the
                                                           bitmap being editted."))
                                          TITLE _ "Put change into bitmap?"
                                          CENTERFLG _ T))
                            (BITBLT PAINTW 0 0 BM)
                            (\EDITBM/PUTUP/DISPLAY W BM GRIDSPEC GRIDINTERIOR BITSWIDE BITSHIGH)))
                        (CLOSEW PAINTW)                    ; set PAINTW so that space can be reclaimed
                        (SETQ PAINTW))
                  (Cursor_                                 ; Stuffs lower left part of image into the cursor and sets the
                                                           ; hotspot.
                     (READHOTSPOT BM GRIDSPEC GRIDINTERIOR W)
                                                           (* WINDOWPROP W (QUOTE FINISHEDFLG) T)
                     )
                  (Color (WINDOWPROP W 'COLOR (OR (MENU (COLORMENU BITSPERPIXEL))
                                                  COLOR)))
                  (UPDATE/BM/DISPLAY/SELECTED/REGION W])
```

(\EDITBM/PUTUP/DISPLAY

```
  [LAMBDA (WINDOW BM GRIDSPEC GRIDINTERIOR BITSWIDE BITSHIGH)    ; Edited 31-Aug-87 13:05 by FS
                                                                 (* initializes the display for the bitmap editor.)
                                                                 (* DSPFILL GRIDINTERIOR WHITESHADE
                                                                 (QUOTE REPLACE) WINDOW)
                                                                 (* COND ((WINDOWPROP WINDOW
                                                                 (QUOTE GRIDON)) (GRID GRIDSPEC BITSWIDE BITSHIGH
                                                                 (QUOTE POINT) WINDOW)))
    (RESETGRID.NEW BM GRIDSPEC BITSWIDE BITSHIGH 0 0 WINDOW T)
    (UPDATE/BM/DISPLAY BM WINDOW])
```

(\EDITBMHOWMUCH

```
  [LAMBDA (BM EDITWIDTH EDITHEIGHT TITLEQ)                       (* kbr%: " 2-Sep-85 19:44")
                                                                 (* asks the user how much to clear)
    (MENU (COND
             ((OR (IGREATERP (fetch (BITMAP BITMAPWIDTH) of BM)
                             EDITWIDTH)
                  (IGREATERP (fetch (BITMAP BITMAPHEIGHT) of BM)
                             EDITHEIGHT))
              (create MENU
                      TITLE _ TITLEQ
                      ITEMS _ '((VisiblePart 'VISIBLE "Operates on just the part visible in the edit region")
                                (WholeBitmap 'WHOLE "Operates on the entire bitmap"))
                      CENTERFLG _ T))
             (T (create MENU
                        TITLE _ TITLEQ
                        ITEMS _ '((WholeBitmap 'WHOLE "Operates on the entire bitmap"))
                        CENTERFLG _ T])
```

(EDITBMRESHAPEFN

```
  [LAMBDA (BMEDITWINDOW OLDIMAGE OLDREGION OLDSCREENREGION ZEROBMFLG)
                                                                 ; Edited  7-Dec-88 17:00 by SHIH
    ;; allows the bitmap edit window to be reshaped to enlarge the editting area.  This is also called to set up the image during initialization.
    (PROG (BMWINTERIORWIDTH BMWINTERIORHEIGHT EDITAREABITWIDTH EDITAREABITHEIGHT GRIDSQUARE GRIDINTERIOR
                BITMAPWIDTH BMDISPLAYWIDTH BMDISPLAYBOTTOM BMDISPLAYHEIGHT BITMAPHEIGHT
                (BM (WINDOWPROP BMEDITWINDOW 'BM))
                MINCOMMANDAREAWIDTH EXTENTWIDTH EXTENTHEIGHT TEMPBM)
          (SETQ MINCOMMANDAREAWIDTH 30)
          (SETQ BITMAPWIDTH (fetch (BITMAP BITMAPWIDTH) of BM))
          (SETQ BITMAPHEIGHT (fetch (BITMAP BITMAPHEIGHT) of BM))
          (SETQ BMWINTERIORWIDTH (WINDOWPROP BMEDITWINDOW 'WIDTH))
    ;; leave room at the top for the full size display area.  But not more than half of the window.
          (SETQ BMWINTERIORHEIGHT (IMAX (IDIFFERENCE (WINDOWPROP BMEDITWINDOW 'HEIGHT)
                                                     (IPLUS BITMAPHEIGHT GRIDTHICKNESS))
                                        (IQUOTIENT (WINDOWPROP BMEDITWINDOW 'HEIGHT)
                                                   2)))
    ;; if the user hasn't set it, determine the grid size as the largest size which fits the interior but not larger than NORMALGRIDSQUARE nor smaller
    ;; than MINGRIDSQUARE.  If GRIDSQUARE was specified, reset it to NIL so that if reshaped it will be recalculated.
          (SETQ GRIDSQUARE (OR (WINDOWPROP BMEDITWINDOW 'GRIDSQUARE NIL)
                               (IMAX (IMIN (IQUOTIENT BMWINTERIORWIDTH BITMAPWIDTH)
                                           (IQUOTIENT BMWINTERIORHEIGHT BITMAPHEIGHT)
                                           NORMALGRIDSQUARE)
                                     MINGRIDSQUARE)))            ; calculate how many bits will be displayed at once.
          (SETQ EDITAREABITWIDTH (IMIN (IQUOTIENT BMWINTERIORWIDTH GRIDSQUARE)
                                       BITMAPWIDTH))
          (WINDOWPROP BMEDITWINDOW 'BITSWIDE EDITAREABITWIDTH)
```

```
(SETQ EDITAREABITHEIGHT (IMIN (IQUOTIENT BMWINTERIORHEIGHT GRIDSQUARE)
                              BITMAPHEIGHT))          ; calculate offset of display and command regions at the top of
                                                      ; the window.
(WINDOWPROP BMEDITWINDOW 'BITSHIGH EDITAREABITHEIGHT)
(SETQ BMDISPLAYBOTTOM (IPLUS (ITIMES GRIDSQUARE EDITAREABITHEIGHT)
                             GRIDTHICKNESS))
(SETQ BMDISPLAYWIDTH (IMIN BITMAPWIDTH (IDIFFERENCE BMWINTERIORWIDTH MINCOMMANDAREAWIDTH)))
```

;; put the offset --- the lower left coordinate --- in the same place unless the new shape allows more to be shown past the upper right corner.

```
(WINDOWPROP BMEDITWINDOW 'XOFFSET (IMIN (WINDOWPROP BMEDITWINDOW 'XOFFSET)
                                        (IDIFFERENCE BITMAPWIDTH EDITAREABITWIDTH)))
(WINDOWPROP BMEDITWINDOW 'YOFFSET (IMIN (WINDOWPROP BMEDITWINDOW 'YOFFSET)
                                        (IDIFFERENCE BITMAPHEIGHT EDITAREABITHEIGHT)))
                                                      ; Center edit square
(SETQ GRIDINTERIOR (create REGION
                          LEFT _ (IQUOTIENT (IDIFFERENCE BMWINTERIORWIDTH (ITIMES EDITAREABITWIDTH
                                                                                  GRIDSQUARE))
                                            2)
                          BOTTOM _ (IQUOTIENT (IDIFFERENCE BMDISPLAYBOTTOM (ITIMES EDITAREABITHEIGHT
                                                                                   GRIDSQUARE))
                                              2)
                          WIDTH _ (ITIMES EDITAREABITWIDTH GRIDSQUARE)
                          HEIGHT _ (ITIMES EDITAREABITHEIGHT GRIDSQUARE)))
(WINDOWPROP BMEDITWINDOW 'GRIDINTERIOR GRIDINTERIOR)
(WINDOWPROP BMEDITWINDOW 'BMDISPLAYBOTTOM BMDISPLAYBOTTOM)
(WINDOWPROP BMEDITWINDOW 'BMDISPLAYWIDTH BMDISPLAYWIDTH)
(WINDOWPROP BMEDITWINDOW 'BMDISPLAYHEIGHT (SETQ BMDISPLAYHEIGHT (IDIFFERENCE (WINDOWPROP BMEDITWINDOW
                                                                                        'HEIGHT)
                                                                              BMDISPLAYBOTTOM)))
(WINDOWPROP BMEDITWINDOW 'DISPLAYREGION
       (create REGION
              LEFT _ 0
              BOTTOM _ BMDISPLAYBOTTOM
              WIDTH _ BMDISPLAYWIDTH
              HEIGHT _ BMDISPLAYHEIGHT))
(WINDOWPROP BMEDITWINDOW 'GRIDSPEC (create REGION
                                           LEFT _ (fetch (REGION LEFT) of GRIDINTERIOR)
                                           BOTTOM _ (fetch (REGION BOTTOM) of GRIDINTERIOR)
                                           WIDTH _ GRIDSQUARE
                                           HEIGHT _ GRIDSQUARE))
(SETQ EXTENTHEIGHT (QUOTIENT (TIMES BITMAPHEIGHT (WINDOWPROP BMEDITWINDOW 'HEIGHT))
                             EDITAREABITHEIGHT))
[SETQ EXTENTWIDTH (IDIFFERENCE (QUOTIENT (TIMES BITMAPWIDTH BMWINTERIORWIDTH)
                                         EDITAREABITWIDTH)
                               (WINDOWPROP BMEDITWINDOW 'BORDER]
(WINDOWPROP BMEDITWINDOW 'EXTENT (CREATEREGION (MINUS (QUOTIENT (TIMES (WINDOWPROP BMEDITWINDOW
                                                                        'XOFFSET)
                                                                  EXTENTWIDTH)
                                                           BITMAPWIDTH))
                                               (MINUS (QUOTIENT (TIMES (WINDOWPROP BMEDITWINDOW 'YOFFSET)
                                                                       EXTENTHEIGHT)
                                                                BITMAPHEIGHT))
                                               EXTENTWIDTH EXTENTHEIGHT))
```

;; Build & cache a temporary bitmap.

;; Could make only (min (bitmapheight bm) (iquotient (bitmapheight window) scale)), except if user changes scale, bitmap might be too small. So,
;; make sufficiently large just to be safe.

```
(SETQ TEMPBM (WINDOWPROP BMEDITWINDOW 'TEMPBM))
(LET ((TEMPBM.W BMWINTERIORWIDTH)
      (TEMPBM.H (IMIN BITMAPHEIGHT EDITAREABITHEIGHT)))
     (if (OR (NOT TEMPBM)
             (OR (< (BITMAPWIDTH TEMPBM)
                    TEMPBM.W)
                 (< (BITMAPHEIGHT TEMPBM)
                    TEMPBM.H)))
         then (SETQ TEMPBM (BITMAPCREATE TEMPBM.W TEMPBM.H (FETCH (BITMAP BITMAPBITSPERPIXEL)
                                                                  OF BM)))
              (WINDOWPROP BMEDITWINDOW 'TEMPBM TEMPBM)))
(EDITBMREPAINTFN BMEDITWINDOW NIL ZEROBMFLG])
```

(EDITBMREPAINTFN

```
[LAMBDA (WIN REGION ZEROBM)                           ; Edited  8-Dec-88 14:38 by SHIH
```

;; redisplays a bitmap editting window If ZEROBM is non-NIL, it doesn't bother to display the bits.

```
(PROG [(GRIDSPEC (WINDOWPROP WIN 'GRIDSPEC))
       (EDITAREABITWIDTH (WINDOWPROP WIN 'BITSWIDE))
       (EDITAREABITHEIGHT (WINDOWPROP WIN 'BITSHIGH))
       (BM (WINDOWPROP WIN 'BM]
      (CLEARW WIN)                                    ; gray the area above the edit grid that is not bitmap display area.
      (BLTSHADE NOTINUSEGRAY WIN (+ (WINDOWPROP WIN 'BMDISPLAYWIDTH)
                                    GRIDTHICKNESS)
             (WINDOWPROP WIN 'BMDISPLAYBOTTOM))
```

;; put in the display of the full sized bitmap.

```
      (UPDATE/BM/DISPLAY BM WIN)
```

```
;; FS: Now that RESETGRID displays the grid, don't need the call to GRID.
      (if ZEROBM
          then (if (WINDOWPROP WIN 'GRIDON)
                   then (GRID GRIDSPEC EDITAREABITWIDTH EDITAREABITHEIGHT 'POINT WIN))
        else (RESETGRID.NEW BM GRIDSPEC EDITAREABITWIDTH EDITAREABITHEIGHT 0 0 WIN])
```

(UPDATE/SHADE/DISPLAY

```
  [LAMBDA (BM WIN)                                          (* rrb "20-JUN-82 16:53")
                                                            (* displays BM as if it were a shade.)
    (PROG [(BOTTOM (WINDOWPROP WIN 'BMDISPLAYBOTTOM]
          (TILEAREA 0 BOTTOM (WINDOWPROP WIN 'WIDTH)
                 (IDIFFERENCE (WINDOWPROP WIN 'HEIGHT)
                        BOTTOM)
                 BM WIN])
```

(UPDATE/BM/DISPLAY/SELECTED/REGION

```
  [LAMBDA (W)                                               ; Edited  1-Sep-87 17:48 by FS
                                                            (* Shade the selected region of the bitmap display area.)
    (COND
       ([OR (IGREATERP (fetch (BITMAP BITMAPWIDTH) of (WINDOWPROP W 'BM))
                 (WINDOWPROP W 'BITSWIDE))
            (IGREATERP (fetch (BITMAP BITMAPHEIGHT) of (WINDOWPROP W 'BM))
                 (WINDOWPROP W 'BITSHIGH]                   (* only invert the region being editted if it is less than the entire
                                                            bitmap.)
        (BLTSHADE BLACKSHADE W (IDIFFERENCE (WINDOWPROP W 'XOFFSET)
                                    (WINDOWPROP W 'DXOFFSET))
              (IDIFFERENCE (IPLUS (WINDOWPROP W 'BMDISPLAYBOTTOM)
                                  (WINDOWPROP W 'YOFFSET))
                    (WINDOWPROP W 'DYOFFSET))
              (WINDOWPROP W 'BITSWIDE)
              (WINDOWPROP W 'BITSHIGH)
              'INVERT])
```

(SHOWBUTTON

```
  [LAMBDA (BUTTON DS)                                       (* rrb "27-JUL-81 10:59")
                                                            (* displays a menu box and its title.)
    (PROG ((BLOCK (fetch (BUTTON REGION) of BUTTON)))
          (WBOX BLOCK NIL NIL DS)                           (* Display the title in the middle of the box)
          (CENTERPRINTINREGION (fetch (BUTTON LABEL) of BUTTON)
                 BLOCK DS])
```

(RESETGRID.NEW

```
  [LAMBDA (BM GRIDSPEC WIDTH HEIGHT ORIGX ORIGY WINDOW DOCLEARFLG)
                                                            ; Edited  8-Dec-88 14:36 by SHIH
    ;; Copies the contents of a bitmap into the edit display grid of window.  ORIGX & Y are used to offest into both bitmap and destination window.
    (LET (XOFFSET YOFFSET MAXX MAXY SHADE XSCALE YSCALE TEMPBM)
         (SETQ XSCALE (fetch (REGION WIDTH) of GRIDSPEC))
         (SETQ YSCALE (fetch (REGION HEIGHT) of GRIDSPEC))
         (if (NULL ORIGX)
             then (SETQ ORIGX 0))
         (if (NULL ORIGY)
             then (SETQ ORIGY 0))
         (SETQ XOFFSET (WINDOWPROP WINDOW 'XOFFSET))
         (SETQ YOFFSET (WINDOWPROP WINDOW 'YOFFSET))
         (SETQ MAXX (IPLUS ORIGX WIDTH -1))
         (SETQ MAXY (IPLUS ORIGY HEIGHT -1))
         (SETQ TEMPBM (WINDOWPROP WINDOW 'TEMPBM))
         ;; Use SCALEBM.  Bitmap destination must be empty (white).
         (if DOCLEARFLG
             then (BLTSHADE WHITESHADE WINDOW (LEFTOFGRIDCOORD ORIGX GRIDSPEC)
                          (BOTTOMOFGRIDCOORD ORIGY GRIDSPEC)
                          (CL:* WIDTH XSCALE)
                          (CL:* HEIGHT YSCALE)
                          'REPLACE))
         (SCALEBM BM (+ ORIGX XOFFSET)
                (+ ORIGY YOFFSET)
                WINDOW
                (LEFTOFGRIDCOORD ORIGX GRIDSPEC)
                (BOTTOMOFGRIDCOORD ORIGY GRIDSPEC)
                WIDTH HEIGHT XSCALE YSCALE TEMPBM)
         ;; Texture the pixels correctly (note that Bltshade has a different meaning on color BMs, so only shade if its B/W).  DARKBITSHADE MUST
         ;; be a number, but try and be robust anyway.
         (IF (= 1 (BITSPERPIXEL BM))
             THEN (BLTSHADE (if (NUMBERP DARKBITSHADE)
                                then (- -1 DARKBITSHADE)
                              else DARKBITSHADE)
                          WINDOW
                          (LEFTOFGRIDCOORD ORIGX GRIDSPEC)
                          (BOTTOMOFGRIDCOORD ORIGY GRIDSPEC)
                          (CL:* WIDTH XSCALE)
```

```
                        (CL:* HEIGHT YSCALE)
                        'ERASE))

    ;; Add grid

    (if (WINDOWPROP WINDOW 'GRIDON)
        then (if (OR (NEQ ORIGX (CAR GRIDSPEC))
                     (NEQ ORIGY (CADR GRIDSPEC)))
                 then (SETQ GRIDSPEC (COPYALL GRIDSPEC))
                      (replace (REGION LEFT) of GRIDSPEC with (LEFTOFGRIDCOORD ORIGX GRIDSPEC))
                      (replace (REGION BOTTOM) of GRIDSPEC with (BOTTOMOFGRIDCOORD ORIGY GRIDSPEC)))
             (GRID GRIDSPEC WIDTH HEIGHT 'POINT WINDOW])
```

(RESETGRID

```
[LAMBDA (BM GRIDSPEC WIDTH HEIGHT ORGX ORGY W)                ; Edited  7-Dec-88 16:58 by SHIH

    ;; copies the contents of a bitmap into the edit display grid.

    ;; This is no longer called from HLDISPLAY, and is probably obsolete.  Thus code commented out, below.

    ;; (PROG (XOFFSET YOFFSET MAXX MAXY SHADE) (COND ((NULL ORGX) (SETQ ORGX 0))) (COND ((NULL ORGY) (SETQ ORGY 0)))
    ;; (SETQ XOFFSET (WINDOWPROP W 'XOFFSET)) (SETQ YOFFSET (WINDOWPROP W 'YOFFSET)) (SETQ MAXX (IPLUS ORGX WIDTH
    ;; -1)) (SETQ MAXY (IPLUS ORGY HEIGHT -1)) (for Y from ORGY to MAXY do (for X from ORGX to MAXX do (SETQ SHADE
    ;; (EDITBMTEXTURE BM (IPLUS X XOFFSET) (IPLUS Y YOFFSET))) (SHADEGRIDBOX X Y SHADE 'REPLACE GRIDSPEC (COND ((NULL
    ;; (WINDOWPROP W 'GRIDON)) 0) (T 'POINT)) W))))

    NIL])
```

(\READBMDIMENSIONS

```
[LAMBDA NIL                                                   (* gbn%: "26-Jan-86 15:57")
                                                              (* asks the user for dimensions of a bitmap and creates it.)
  (PROG (WIDTH HEIGHT)
    WIDTHLP
        (PRIN1 "How wide would you like the bitmap to be? " T)
        (COND
           ([NOT (NUMBERP (SETQ WIDTH (READ T]
            (PRIN1 "?" T)
            (TERPRI T)
            (GO WIDTHLP))
           ((ILESSP WIDTH 1)
            (PRIN1 "WIDTH must be positive." T)
            (TERPRI T)
            (GO WIDTHLP)))
    HEIGHTLP
        (PRIN1 "How high would you like the bitmap to be? " T)
        (COND
           ([NOT (NUMBERP (SETQ HEIGHT (READ T]
            (PRIN1 "?" T)
            (TERPRI T)
            (GO HEIGHTLP))
           ((ILESSP HEIGHT 1)
            (PRIN1 "HEIGHT must be positive." T)
            (TERPRI T)
            (GO HEIGHTLP)))
        (RETURN (BITMAPCREATE WIDTH HEIGHT (BITSPERPIXEL \CURSORDESTINATION])
```

(EDITSHADE

```
[LAMBDA (SHADE)                                               ; Edited 10-Oct-89 12:08 by jds

    ;; a simple shade editor.

    (PROG (SHADEBM QUITREGION SHADEREGION BMWIDTH BMHEIGHT GRIDINTERIOR GRIDSPEC X Y SEDW BOXSIZE SHOWREGION)
          [SETQ SHADEBM (COND
                           ((BITMAPP SHADE)
                            (CREATETEXTUREFROMBITMAP SHADE))
                           ((FIXP SHADE)
                            (\BITMAPFROMTEXTURE SHADE))
                           ((EQ SHADE T)
                            (BITMAPCREATE 16 16))
                           ((NULL SHADE)
                            (BITMAPCREATE 4 4))
                           (T (\ILLEGAL.ARG SHADE]
          (SETQ QUITREGION (CREATEREGION 72 150 50 20))
          (SETQ SHOWREGION (CREATEREGION 125 150 100 20))
          (SETQ SHADEREGION (CREATEREGION 10 185 272 100))
          (SETQ SEDW (CREATEW (GETBOXREGION 300 300 NIL NIL NIL "Indicate position of Shade edit window.")))
          (SETQ BMWIDTH (BITMAPWIDTH SHADEBM))
          (SETQ BMHEIGHT (BITMAPHEIGHT SHADEBM))
          (SETQ BOXSIZE (IMIN (IQUOTIENT 144 BMHEIGHT)
                              (IQUOTIENT 256 BMWIDTH)))
          (WINDOWPROP SEDW 'PROCESS (THIS.PROCESS))
          (WINDOWPROP SEDW 'REPAINTFN 'EDITSHADEREPAINTFN)
          (WINDOWPROP SEDW 'QUITREGION QUITREGION)
          (WINDOWPROP SEDW 'SHOWREGION SHOWREGION)
          (WINDOWPROP SEDW 'GRIDSPEC (SETQ GRIDSPEC (CREATEREGION (SETQ X (IQUOTIENT (- 292 (ITIMES BOXSIZE
                                                                                                    BMWIDTH))
                                                                                       2))
                                                                  (SETQ Y (IQUOTIENT (- 150 (ITIMES BOXSIZE BMHEIGHT))
```

```
                                                                    2))
                                                          BOXSIZE BOXSIZE)))
        [WINDOWPROP SEDW 'GRIDINTERIOR (SETQ GRIDINTERIOR (CREATEREGION X Y (ITIMES BOXSIZE BMWIDTH)
                                                                  (ITIMES BOXSIZE BMHEIGHT]
        (WINDOWPROP SEDW 'SHADEBM SHADEBM)
        (WINDOWPROP SEDW 'SHADEREGION SHADEREGION)
        (WINDOWPROP SEDW 'XOFFSET 0)
        (WINDOWPROP SEDW 'YOFFSET 0)
        (EDITSHADEREPAINTFN SEDW)
        (RESETLST
            (RESETSAVE NIL (LIST 'CLOSEW SEDW))
            [do (DSPFILL SHADEREGION (COND
                                        ((EQ BMWIDTH 4)        ; bitblt doesn't like bitmaps that are not 16 by 16.0
                                         (CREATETEXTUREFROMBITMAP SHADEBM))
                                        (T SHADEBM))
                        'TEXTURE SEDW)
                (until (MOUSESTATE (OR LEFT MIDDLE RIGHT)) do (TOTOPW SEDW)
                                                              (BLOCK))
                (COND
                   [(LASTMOUSESTATE RIGHT)
                    (ERSETQ (DOWINDOWCOM (WHICHW LASTMOUSEX LASTMOUSEY]
                   ((EQ 'STOP (until (MOUSESTATE UP) bind (XPIXEL YPIXEL)
                                 do (TOTOPW SEDW)
                                    [COND
                                       [(INSIDE? GRIDINTERIOR (SETQ X (LASTMOUSEX SEDW))
                                              (SETQ Y (LASTMOUSEY SEDW)))
                                        (COND
                                           ((AND (STRICTLY/BETWEEN (SETQ XPIXEL (GRIDXCOORD X GRIDSPEC))
                                                        -1 BMWIDTH)
                                                 (STRICTLY/BETWEEN (SETQ YPIXEL (GRIDYCOORD Y GRIDSPEC))
                                                        -1 BMHEIGHT))
                                            (SHADEGRIDBOX XPIXEL YPIXEL (COND
                                                                          ((LASTMOUSESTATE LEFT)
                                                                           DARKBITSHADE)
                                                                          (T WHITESHADE))
                                                  'REPLACE GRIDSPEC 'POINT SEDW)
                                            (BITMAPBIT SHADEBM XPIXEL YPIXEL (COND
                                                                                ((LASTMOUSESTATE LEFT)
                                                                                 1)
                                                                                (T 0]
                                       [(INSIDE? QUITREGION X Y)
                                        (DSPFILL QUITREGION BLACKSHADE 'INVERT SEDW)
                                        (RETURN (until (MOUSESTATE UP)
                                                   do (COND
                                                         ((NOT (INSIDE? QUITREGION (LASTMOUSEX SEDW)
                                                                       (LASTMOUSEY SEDW)))
                                                          (DSPFILL QUITREGION BLACKSHADE 'INVERT SEDW)
                                                          (RETURN)))
                                                   finally (DSPFILL QUITREGION BLACKSHADE 'INVERT SEDW)
                                                           ; close window.
                                                           (RETURN 'STOP]
                                       ((INSIDE? SHOWREGION X Y)
                                        (DSPFILL SHOWREGION BLACKSHADE 'INVERT SEDW)
                                        (RETURN (until (MOUSESTATE UP)
                                                   do (COND
                                                         ((NOT (INSIDE? SHOWREGION (LASTMOUSEX SEDW)
                                                                       (LASTMOUSEY SEDW)))
                                                          (DSPFILL SHOWREGION BLACKSHADE 'INVERT SEDW)
                                                          (RETURN)))
                                                   finally (DSPFILL SHOWREGION BLACKSHADE 'INVERT SEDW)
                                                           ; close window.
                                                           (PRINTOUT (GETPROMPTWINDOW SEDW 1)
                                                                  T "Texture:  " (CREATETEXTUREFROMBITMAP
                                                                                    SHADEBM]
                                    (BLOCK)))
                             (RETURN])
        (RETURN (COND
                   ((AND (OR (NUMBERP SHADE)
                             (NULL SHADE))
                         (EQ BMWIDTH 4)
                         (EQ BMHEIGHT 4))                  ; user passed in a number or NIL, give them a number back.
                    (CREATETEXTUREFROMBITMAP SHADEBM))
                   (T SHADEBM])
```

(\BITMAPFROMTEXTURE

```
  [LAMBDA (FIXP)                                           (* rrb "16-May-84 14:56")
                                                           (* returns a 4 by 4 bitmap that contains the texture represented
                                                           by FIXP.)
    (PROG ((SHADE (BITMAPCREATE 4 4)))
          [for X from 0 to 3
             do (for Y from 0 to 3 do (COND
                                         ([NOT (EQ 0 (LOGAND FIXP (\BITMASK (IPLUS (ITIMES (IDIFFERENCE 3 Y)
                                                                                           4)
                                                                                   X]
                                          (BITMAPBIT SHADE X Y 1]
          (RETURN SHADE])
```

(EDITSHADEREPAINTFN

```
  [LAMBDA (WIN)                                                    ; Edited 10-Oct-89 12:04 by jds
                                                                   ; redisplays an edit shade window.

    (PROG (GRIDSPEC SHADE BMWIDTH BMHEIGHT)
          (SETQ GRIDSPEC (WINDOWPROP WIN 'GRIDSPEC))
          (SETQ SHADE (WINDOWPROP WIN 'SHADEBM))
          (SETQ BMWIDTH (BITMAPWIDTH SHADE))
          (SETQ BMHEIGHT (BITMAPHEIGHT SHADE))
          (SHOWBUTTON (create BUTTON
                              REGION _ (WINDOWPROP WIN 'QUITREGION)
                              LABEL _ 'QUIT
                              HELP _ "Quits")
                WIN)
          (SHOWBUTTON (create BUTTON
                              REGION _ (WINDOWPROP WIN 'SHOWREGION)
                              LABEL _ 'Show% Number
                              HELP _ "Displays the texture number for the current shade.")
                WIN)
          (GRAYBOXAREA (fetch (REGION LEFT) of GRIDSPEC)
                (fetch (REGION BOTTOM) of GRIDSPEC)
                (ITIMES (fetch (REGION WIDTH) of GRIDSPEC)
                       BMWIDTH)
                (ITIMES (fetch (REGION HEIGHT) of GRIDSPEC)
                       BMHEIGHT)
                2 BLACKSHADE WIN)
          (RESETGRID.NEW SHADE GRIDSPEC BMWIDTH BMHEIGHT 0 0 WIN)
                                                                   ; GRID GRIDSPEC BMWIDTH BMHEIGHT (QUOTE POINT)
                                                                   ; WIN
          (DSPFILL (WINDOWPROP WIN 'SHADEREGION)
                SHADE
                'TEXTURE WIN])
```

(GRAYBOXAREA

```
  [LAMBDA (X Y WIDTH HEIGHT OUTLINESIZE TEXTURE DS)                ; Edited  1-Sep-87 17:49 by FS
                                                                   (* outlines an area with a gray box.)
    (COND
       ((FIXP OUTLINESIZE))
       ((NULL OUTLINESIZE)
        (SETQ OUTLINESIZE 1))
       (T (\ILLEGAL.ARG OUTLINESIZE)))
    (BLTSHADE (OR TEXTURE BLACKSHADE)
           DS
           (IDIFFERENCE X OUTLINESIZE)
           (IDIFFERENCE Y OUTLINESIZE)
           (IPLUS WIDTH (ITIMES 2 OUTLINESIZE))
           (IPLUS HEIGHT (ITIMES 2 OUTLINESIZE))
           'REPLACE)
    (BLTSHADE (DSPTEXTURE NIL DS)
           DS X Y WIDTH HEIGHT 'REPLACE])
```

(\SHADEBITS

```
  [LAMBDA (BM GRIDSPEC GRIDINTERIOR W BITSWIDE BITSHIGH COLOR)     ; Edited  7-Jun-93 11:43 by sybalsky:mv:envos

          (* cursor is inside the edit grid, so change the bit in the bitmap, change the edit grid and redisplay the bitmap.)

    (PROG (BITSPERPIXEL XPIXEL YPIXEL OTHERCOLOR SHADE OTHERSHADE USECOLOR USESHADE X Y COORDWIN)
          (SETQ BITSPERPIXEL (BITSPERPIXEL BM))
          (SETQ OTHERCOLOR (OPPOSITECOLOR COLOR BITSPERPIXEL))
          (COND
             [(EQ BITSPERPIXEL 1)
              (COND
                 ((EQ COLOR 1)
                  (SETQ SHADE DARKBITSHADE)
                  (SETQ OTHERSHADE WHITESHADE))
                 (T (SETQ SHADE WHITESHADE)
                    (SETQ OTHERSHADE DARKBITSHADE]
             (T (SETQ SHADE COLOR)
                (SETQ OTHERSHADE OTHERCOLOR)))
          (until (MOUSESTATE UP)
             when (AND [NOT (EQ (AND [EQ XPIXEL (SETQ XPIXEL (IMAX 0 (IMIN BITSWIDE (GRIDXCOORD (SETQ X
                                                                                                  (LASTMOUSEX
                                                                                                   W))
                                                                                          GRIDSPEC]
                                     YPIXEL)
                               (SETQ YPIXEL (IMAX 0 (IMIN BITSHIGH (GRIDYCOORD (SETQ Y (LASTMOUSEY W))
                                                                               GRIDSPEC]
                         (INSIDE? GRIDINTERIOR X Y))
             do (COND
                   ((LASTMOUSESTATE LEFT)
                    (SETQ USECOLOR COLOR)
                    (SETQ USESHADE SHADE))
                   (T (SETQ USECOLOR OTHERCOLOR)
                      (SETQ USESHADE OTHERSHADE)))
```

```
               [COND
                  ((SETQ COORDWIN (WINDOWPROP W 'COORDWIN))
                   (CLEARW COORDWIN)
                   (MOVETO 2 4 COORDWIN)
                   (PRINTOUT COORDWIN (IPLUS XPIXEL (WINDOWPROP W 'XOFFSET))
                          ", "
                          (IPLUS YPIXEL (WINDOWPROP W 'YOFFSET]
               (BITMAPBIT BM (IPLUS XPIXEL (WINDOWPROP W 'XOFFSET))
                      (IPLUS YPIXEL (WINDOWPROP W 'YOFFSET))
                      USECOLOR)
               (UPDATE/BM/DISPLAY BM W)
               (SHADEGRIDBOX XPIXEL YPIXEL USESHADE 'REPLACE GRIDSPEC (COND
                                                                         ((NULL (WINDOWPROP W 'GRIDON))
                                                                          0)
                                                                         (T 'POINT))
                      W])
```

(READHOTSPOT

```
[LAMBDA (BM GRIDSPEC GRIDINTERIOR DS)                    ; Edited 10-Jul-92 16:47 by cat
                                                         (* kbr%: "13-Feb-86 15:21")
                                                         (* reads the hotspot from the cursor and sets cursor)
  (UNTILMOUSESTATE UP)
  (PROG (NOWCURSOR XPIXEL YPIXEL DOWNYET? CURSORBM)
        (SETQ NOWCURSOR (CURSOR))
        (CURSORPOSITION (create POSITION
                               XCOORD _ (IPLUS (LEFTOFGRIDCOORD (SETQ XPIXEL (fetch (CURSOR CUHOTSPOTX)
                                                                                    of NOWCURSOR))
                                                                GRIDSPEC)
                                               (IQUOTIENT (fetch (REGION WIDTH) of GRIDSPEC)
                                                          2))
                               YCOORD _ (IPLUS (BOTTOMOFGRIDCOORD (SETQ YPIXEL (fetch (CURSOR CUHOTSPOTY)
                                                                                  of NOWCURSOR))
                                                                  GRIDSPEC)
                                               (IQUOTIENT (fetch (REGION HEIGHT) of GRIDSPEC)
                                                          2)))
                DS)                                      (* SHADEGRIDBOX XPIXEL YPIXEL NOTINUSEGRAY
                                                         (QUOTE REPLACE) GRIDSPEC
                                                         (QUOTE POINT) DS)
        (until (PROGN (BLOCK)
                      (GETMOUSESTATE)
                      (AND DOWNYET? (MOUSESTATE UP)))
          when (INSIDE? GRIDINTERIOR (LASTMOUSEX DS)
                        (LASTMOUSEY DS))
          do [OR DOWNYET? (SETQ DOWNYET? (NOT (EQ LASTMOUSEBUTTONS 0]
                                                         (* COND (XPIXEL (SHADEGRIDBOX XPIXEL YPIXEL
                                                         (EDITBMTEXTURE BM XPIXEL YPIXEL)
                                                         (QUOTE REPLACE) GRIDSPEC
                                                         (QUOTE POINT) DS)))
                                                         (* SHADEGRIDBOX (SETQ XPIXEL
                                                         (GRIDXCOORD (LASTMOUSEX DS) GRIDSPEC))
                                                         (SETQ YPIXEL (GRIDYCOORD (LASTMOUSEY DS)
                                                         GRIDSPEC)) NOTINUSEGRAY (QUOTE REPLACE)
                                                         GRIDSPEC (QUOTE POINT) DS)

          finally (SETQ CURSORBM (BITMAPCREATE 16 16 (BITSPERPIXEL BM)))
                  (BITBLT BM NIL NIL CURSORBM)
                  (CURSOR (CURSORCREATE CURSORBM NIL XPIXEL YPIXEL])
```

(WBOX

```
[LAMBDA (REG THCK TEXTURE DS)                            ; Edited  1-Sep-87 17:52 by FS
                                                         (* Draws a box around REG with bounding lines of THCKness)
  (OR THCK (SETQ THCK 2))
  (BLTSHADE BLACKSHADE DS NIL NIL NIL NIL 'REPLACE REG)
  (BLTSHADE (OR TEXTURE (DSPTEXTURE NIL DS))
         DS
         (IPLUS (fetch (REGION LEFT) of REG)
                THCK)
         (IPLUS (fetch (REGION BOTTOM) of REG)
                THCK)
         (IDIFFERENCE (fetch (REGION WIDTH) of REG)
                (ITIMES 2 THCK))
         (IDIFFERENCE (fetch (REGION HEIGHT) of REG)
                (ITIMES 2 THCK))
         'REPLACE])
```

(\CLEARBM

```
[LAMBDA (BM TXT REG)                                     ; Edited  1-Sep-87 17:53 by FS
  (BLTSHADE (OR TXT WHITESHADE)
         BM NIL NIL NIL NIL 'REPLACE REG])
```

(EDITBMTEXTURE

```
[LAMBDA (BM X Y)                                         (* kbr%: " 9-Jan-86 21:51")
                                                         (* Texture EDITBM should use to represent pixel
                                                         (X . Y) of BM. *)
```

```
    (PROG (COLOR SHADE)
          (SETQ COLOR (BITMAPBIT BM X Y))
          (SETQ SHADE (COND
                         ((EQ (BITSPERPIXEL BM)
                              1)
                          (COND
                             ((EQ COLOR 1)
                              DARKBITSHADE)
                             (T WHITESHADE)))
                         (T COLOR)))
          (RETURN SHADE])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD BUTTON (REGION LABEL HELP))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS BITMASK MACRO ((X)
                         (LLSH 1 (IDIFFERENCE 15 X))))

(PUTPROPS UPDATE/BM/DISPLAY MACRO ((BM W)
                                   (BITBLT BM (WINDOWPROP W 'DXOFFSET)
                                          (WINDOWPROP W 'DYOFFSET)
                                          W 0 (WINDOWPROP W 'BMDISPLAYBOTTOM)
                                          (WINDOWPROP W 'BMDISPLAYWIDTH)
                                          1000 NIL 'REPLACE)))
)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQQ DARKBITSHADE 23130)

(RPAQQ NORMALGRIDSQUARE 16)

(RPAQQ NOTINUSEGRAY 42405)

(RPAQQ EDITBMMENU NIL)

(RPAQQ EDITBMWINDOWMENU NIL)

(RPAQQ GRIDSIZEMENU NIL)

(RPAQQ CLICKWAITTIME 250)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DARKBITSHADE NORMALGRIDSQUARE NOTINUSEGRAY EDITBMMENU CLICKWAITTIME)
)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ GRIDTHICKNESS 2)

(RPAQQ MINGRIDSQUARE 8)

(RPAQQ MAXGRIDWIDTH 199)

(RPAQQ MAXGRIDHEIGHT 175)

(RPAQQ BMWINDOWSHADE 33410)

(CONSTANTS (GRIDTHICKNESS 2)
       (MINGRIDSQUARE 8)
       (MAXGRIDWIDTH 199)
       (MAXGRIDHEIGHT 175)
       (BMWINDOWSHADE 33410))
)

(DEFINEQ
```

(SCALEBM

```
  [LAMBDA (SRCEBM SRCEX SRCEY DESTBM DESTX DESTY SRCEWIDTH SRCEHEIGHT XSCALE YSCALE TEMPBM)
                                                                    ; Edited 31-Aug-87 10:40 by FS

    ;; Magnify a bitmap as per EDITBM.  Use smearing algorithm.

    (LET ((DESTWIDTH (BITMAPWIDTH DESTBM))
```

```
          (DESTHEIGHT (BITMAPHEIGHT DESTBM))
          XSTEPS YSTEPS POWER)
```

;; Check parameters, apply defaults

```
(if (NUMBERP SRCEWIDTH)
  else (SETQ SRCEWIDTH (BITMAPWIDTH SRCEBM)))
(if (NUMBERP SRCEHEIGHT)
  else (SETQ SRCEHEIGHT (BITMAPHEIGHT SRCEBM)))
```

;; Save effort by considering min of srce and dest.

```
(SETQ DESTWIDTH (MIN DESTWIDTH (CL:* SRCEWIDTH XSCALE)))
(SETQ DESTHEIGHT (MIN DESTHEIGHT (CL:* SRCEHEIGHT YSCALE)))
(SETQ SRCEWIDTH (MIN SRCEWIDTH (IQUOTIENT DESTWIDTH XSCALE)))
(SETQ SRCEHEIGHT (MIN SRCEHEIGHT (IQUOTIENT DESTHEIGHT YSCALE)))
(if TEMPBM
    then (BLTSHADE WHITESHADE TEMPBM)
  else (SETQ TEMPBM (BITMAPCREATE DESTWIDTH SRCEHEIGHT)))
```

;; CALL EXPANDBM twice, once for each direction, because we have a spare bitmap which makes it run faster than a single call to
;; EXPANDBM would (I think).

;;

;; Do X Direction Smearing.

;; ============

```
(EXPANDBM SRCEBM SRCEX SRCEY SRCEWIDTH SRCEHEIGHT TEMPBM 0 0 DESTWIDTH SRCEHEIGHT XSCALE 1 XSCALE 1)
```

;;

;; Do Y Direction Smearing.

;; ============

```
(EXPANDBM TEMPBM 0 0 DESTWIDTH SRCEHEIGHT DESTBM DESTX DESTY DESTWIDTH DESTHEIGHT 1 YSCALE 1 YSCALE)
```

;;

;; Return the temporary bitmap for recycling purposes.

```
TEMPBM])
```

(BLTPATTERN

```
[LAMBDA (SRCE SX SY SW SH DEST DX DY DW DH OPER TEMPBM)        ; Edited 8-Dec-88 18:52 by SHIH
```

;; Fills region of Destination with tiles of Source region, using operation. If Temporary bitmap is provided, use it for optimal performance (this is
;; because bitmaps are much faster to paint than other destinations, e.g. windows).

```
(PROG (W H RX RW TW TH)
      (if (NULL SW)
          then (SETQ SW (BITMAPWIDTH SRCE)))
      (if (NULL SH)
          then (SETQ SH (BITMAPHEIGHT SRCE)))
 ;;
      (if (NULL OPER)
          then (SETQ OPER 'REPLACE))                           ; IRM says OPER defaults to replace
      [if TEMPBM
          then ;; Temp bitmap is only useful if its larger than pattern.
               (SETQ TW (BITMAPWIDTH TEMPBM))
               (SETQ TH (BITMAPHEIGHT TEMPBM))
               (if [OR (AND (<= SW (BITMAPWIDTH SRCE))
                            (<= SH (BITMAPHEIGHT SRCE))
                            (>= TW SW)
                            (>= TH SH))
                       (AND (NEQ OPER 'REPLACE)
                            (>= TW (BITMAPWIDTH SRCE))
                            (>= TH (BITMAPHEIGHT SRCE]
                   then (BLTPATTERN.REPLACEDISPLAY SRCE SX SY SW SH TEMPBM 0 0 TW TH)
                        ;; Allow code to fall through using TEMPBM as source area.
                        (SETQ SRCE TEMPBM)
                        (SETQ SX 0)
                        (SETQ SY 0)
                        [SETQ SW (MAX SW (ITIMES SW (IQUOTIENT TW SW]
                        (SETQ SH (MAX SH (ITIMES SH (IQUOTIENT TH SH]
      (if (AND (EQ OPER 'REPLACE)
               (<= SW (BITMAPWIDTH SRCE))
               (<= SH (BITMAPHEIGHT SRCE))
               (OR (BITMAPP DEST)
                   (WINDOWP DEST)))
          then (BLTPATTERN.REPLACEDISPLAY SRCE SX SY SW SH DEST DX DY DW DH)
        else ;; Even if operation is REPLACE, don't know if destination is inexpensively readable (e.g. Interpress stream. SO, this is the
             ;; general case here.
             (BLTPATTERN.GENERIC SRCE SX SY SW SH DEST DX DY DW DH OPER])
```

(BLTPATTERN.REPLACEDISPLAY

```
[LAMBDA (SRCE SX SY SW SH DEST DX DY DW DH)                    ; Edited 8-Dec-88 16:28 by SHIH
```

;; This routine only replaces the destination with the source, and assumes the destination itself can be easily read from and blt'ed to.

```
    ;; Put initial bitmap into destination.  Source should not be within destination area, otherwise it will be overwritten.
    (LET (RX RY RW RH W H)                                          ; R's are remaining area.
         (SETQ W (MIN SW DW))
         (SETQ H (MIN SH DH))
         ;; Algorithm below whites out extraneous area.  General bltpattern routine leaves overlap areas *alone*, so this routine is not consistent
         ;; when specified-size > source-size (general routine shouldnt come here if so).
         (BLTSHADE WHITESHADE DEST DX DY W H 'REPLACE)
         (BITBLT SRCE SX SY DEST DX DY W H NIL 'REPLACE)
         (SETQ RX (+ DX W))
         (SETQ RW (- DW W))
         ;; Now power up until width is full.
         (while (> RW 0) do (SETQ W (MIN SW RW))
                            (BITBLT DEST DX DY DEST RX DY W H NIL 'REPLACE)
                            (SETQ RW (- RW W))                      ; Reduce remaining width
                            (SETQ RX (+ RX W))                      ; Set next starting position
                            (SETQ SW (+ SW SW))                     ; Can now use 2x area.
                            )
        ;;
         (SETQ RY (+ DY H))
         (SETQ RH (- DH H))
         (SETQ SH H)
         (SETQ W DW)
         ;; Now power up until height is full.
         (while (> RH 0) do (SETQ H (MIN SH RH))
                            (BITBLT DEST DX DY DEST DX RY W H NIL 'REPLACE)
                            (SETQ RH (- RH H))                      ; Reduce remaining width
                            (SETQ RY (+ RY H))                      ; Set next starting position
                            (SETQ SH (+ SH SH))                     ; Can now use 2x area.
                            ])
```

(BLTPATTERN.GENERIC

```
  [LAMBDA (SRCE SX SY SW SH DEST DX DY DW DH OPER)                  ; Edited  8-Dec-88 16:51 by SHIH
    ;; Generically repeat pattern from srce over dest.
    (LET (W H RX RW TW TH)
         (if (NULL SW)
             then (SETQ SW (BITMAPWIDTH SRCE)))
         (if (NULL SH)
             then (SETQ SH (BITMAPHEIGHT SRCE)))
         (while (> DH 0) do (SETQ H (MIN SH DH))
                            ;;
                            (SETQ RW DW)
                            (SETQ RX DX)
                            ;;
                            ;; Fill rows
                            ;;
                            (while (> RW 0) do (SETQ W (MIN SW RW))
                                               (BITBLT SRCE SX SY DEST RX DY W H NIL OPER)
                                               (SETQ RW (- RW W))
                                               (SETQ RX (+ RX W)))
                            ;;
                            (SETQ DH (- DH H))
                            (SETQ DY (+ DY H])

)

(DEFINEQ
```

(EXPANDBITMAP

```
  [LAMBDA (BITMAP WIDTHFACTOR HEIGHTFACTOR)                         ; Edited  2-Sep-87 17:49 by FS
    ;; Returns a new bitmap which is WidthFactor and HeightFactor bigger.
    ;; FS:  This slow piece of code has been replaced with a much faster, general one, EXPAND.I
    (LET (WIDTH HEIGHT BITSPERPIXEL NEWWIDTH NEWHEIGHT NEWX NEWY NEWBITMAP)
         (OR WIDTHFACTOR (SETQ WIDTHFACTOR 1))
         (OR HEIGHTFACTOR (SETQ HEIGHTFACTOR 1))
         (SETQ HEIGHT (fetch (BITMAP BITMAPHEIGHT) of BITMAP))
         (SETQ WIDTH (fetch (BITMAP BITMAPWIDTH) of BITMAP))
         (SETQ BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP))
         (SETQ NEWWIDTH (ITIMES WIDTHFACTOR WIDTH))
         (SETQ NEWHEIGHT (ITIMES HEIGHTFACTOR HEIGHT))
         (SETQ NEWBITMAP (BITMAPCREATE NEWWIDTH NEWHEIGHT BITSPERPIXEL))
         ;; OLD code commented out here.
                                                                    (* LET NIL (* Expand in x-direction.
                                                                    *) (SETQ NEWX 0) (for X from 0 to
                                                                    (SUB1 WIDTH) do (for I from 1 to WIDTHFACTOR do
                                                                    (BITBLT BITMAP X 0 NEWBITMAP NEWX 0 1 HEIGHT
```

```
                                                                    (QUOTE INPUT) (QUOTE REPLACE))
                                                                    (add NEWX 1))) (* Expand in y-direction.
                                                                    *) (SETQ NEWY (SUB1 NEWHEIGHT))
                                                                    (for Y from (SUB1 HEIGHT) to 0 by -1 do
                                                                    (for I from 1 to HEIGHTFACTOR do
                                                                    (BITBLT NEWBITMAP 0 Y NEWBITMAP 0 NEWY NEWWIDTH
      1 (QUOTE INPUT) (QUOTE REPLACE)) (add NEWY -1))))
      (EXPANDBM BITMAP 0 0 WIDTH HEIGHT NEWBITMAP 0 0 NEWWIDTH NEWHEIGHT WIDTHFACTOR HEIGHTFACTOR
            WIDTHFACTOR HEIGHTFACTOR)
     NEWBITMAP])
```

(EXPANDBM

```
  [LAMBDA (SRCEBM SRCEX SRCEY SRCEW SRCEH DESTBM DESTX DESTY DESTW DESTH XSCALE YSCALE XSPACE YSPACE)
                                                                    ; Edited 28-Aug-87 19:00 by FS
```

;; Expands a region of SrceBM by X&Y scale into a region of DestBM, spaced Xspace by YSpace apart (space must be larger than scale). SrceBM
;; cannot be the same bitmap as DestBM. The entire region inside DestBM is cleared.

```
(PROG (XSTEPS YSTEPS POWER)
```

;; Check parameters, apply defaults

```
      (if (NUMBERP SRCEX)
        else (SETQ SRCEX 0))
      (if (NUMBERP SRCEY)
        else (SETQ SRCEY 0))
      (if (NUMBERP SRCEW)
        else (SETQ SRCEW (BITMAPWIDTH SRCEBM)))
      (if (NUMBERP SRCEH)
        else (SETQ SRCEH (BITMAPHEIGHT SRCEBM)))
      (if (NUMBERP DESTX)
        else (SETQ SRCEX 0))
      (if (NUMBERP DESTY)
        else (SETQ SRCEY 0))
```

;; Save effort by considering min of srce and dest.

```
      [SETQ DESTW (IMIN DESTW (CL:* SRCEW (IMAX XSCALE XSPACE]
      [SETQ DESTH (IMIN DESTH (CL:* SRCEH (IMAX YSCALE YSPACE]
      [SETQ SRCEW (IMIN SRCEW (+ 1 (IQUOTIENT DESTW (IMAX XSCALE XSPACE]
      [SETQ SRCEH (IMIN SRCEH (+ 1 (IQUOTIENT DESTH (IMAX YSCALE YSPACE]
      (BLTSHADE WHITESHADE DESTBM DESTX DESTY DESTW DESTH)
      (if (AND (EQL XSPACE 1)
               (EQL YSPACE 1))
          then (BITBLT SRCEBM SRCEX SRCEY DESTBM DESTX DESTY SRCEW SRCEH)
               (RETURN DESTBM))
```

;;

;; Do X Direction Smearing.

;; ============

;; Spread out bitmap by spacefactor. Start from far side to avoid overwrite (if srce = dest)

```
      (if (EQL XSPACE 1)
          then ;; Don't fill destination, instead use srce in YSmear loop.
               ;; (BITBLT SRCEBM SRCEX SRCEY DESTBM DESTX DESTY SRCEW SRCEH)

        else ;; Spread out bitmap by spacefactor. Start from far side to avoid overwrite (if srce = dest)
             (for I from (SUB1 SRCEW) to 0 by -1 do (BITBLT SRCEBM (+ SRCEX I)
                                                            SRCEY DESTBM (+ DESTX (CL:* I XSPACE))
                                                            DESTY 1 SRCEH)))
```

;; Now smear by scalefactor. Each step smears out a power of two. LSH is in ucode.

```
      [if (EQL XSCALE 1)
        else (SETQ POWER 1)
             (while (<= POWER (LSH XSCALE -1)) do ;; In the X direction, only need to blt SRCEH bits high, and must shorten W to
                                                  ;; remain within DESTW
                                                  (BITBLT DESTBM DESTX DESTY DESTBM (+ DESTX POWER)
                                                         DESTY
                                                         (- DESTW POWER)
                                                         SRCEH NIL 'PAINT)
                                                  (SETQ POWER (+ POWER POWER)))
             ;; Clean up for non power of two.
             (if (ZEROP (- XSCALE POWER))
               else (BITBLT DESTBM DESTX DESTY DESTBM (+ DESTX (- XSCALE POWER))
                            DESTY
                            (- DESTW (- XSCALE POWER))
                            SRCEH NIL 'PAINT]
```

;;

;; Do Y Direction Smearing.

;; ============

;; Spread out bitmap by spacefactor. Start from far side to avoid overwrite (if srce = dest)

```
      [if (EQL YSPACE 1)
```

```
            else (if (EQL XSPACE 1)
                     then ;; Didn't need to paint in destination, so can avoid second loop by blting from SRCBM instead of DESTBM.
                          (for J from (SUB1 SRCEH) to 0 by -1 do (BITBLT SRCEBM SRCEX (+ SRCEY J)
                                                                         DESTBM DESTX (+ DESTY (CL:* J YSPACE))
                                                                         DESTW 1))
                   else (for J from (SUB1 SRCEH) to 0 by -1 do (BITBLT DESTBM DESTX (+ DESTY J)
                                                                       DESTBM DESTX (+ DESTY (CL:* J YSPACE))
                                                                       DESTW 1))
                        ;; Since we reused DESTBM, parts of the dest have bits in them but shouldn't.  So, clear them.
                        (for J from 0 to SRCEH by YSPACE do (BLTSHADE WHITESHADE DESTBM DESTX (+ DESTY J 1)
                                                                      DESTW
                                                                      (SUB1 YSPACE]
      ;; Now smear correctly.  Each step smears out a power of two.  LSH is in ucode.
          [if (EQL YSCALE 1)
            else (SETQ POWER 1)
                 (while (<= POWER (LSH YSCALE -1)) do (BITBLT DESTBM DESTX DESTY DESTBM DESTX (+ DESTY POWER)
                                                              DESTW
                                                              (- DESTH POWER)
                                                              NIL
                                                              'PAINT)
                                                      (SETQ POWER (+ POWER POWER)))
                 ;; Clean up for non power of two.
                 (if (ZEROP (- YSCALE POWER))
                   else (BITBLT DESTBM DESTX DESTY DESTBM DESTX (+ DESTY (- YSCALE POWER))
                                DESTW DESTH NIL 'PAINT]
      ;;
      ;; Return the temporary bitmap for recycling purposes.
       DESTBM])
```

(SHRINKBITMAP

```
  [LAMBDA (BITMAP WIDTHFACTOR HEIGHTFACTOR DESTINATIONBITMAP)    (* hdj "18-Feb-86 14:23")
    (LET* [(BITSPP (BITSPERPIXEL BITMAP))
           (WFACTOR (OR WIDTHFACTOR 4))
           (HFACTOR (OR HEIGHTFACTOR 1))
           (HEIGHT (BITMAPHEIGHT BITMAP))
           (WIDTH (BITMAPWIDTH BITMAP))
           (SCRATCH (BITMAPCREATE WIDTH (IQUOTIENT HEIGHT HFACTOR)
                                  BITSPP))
           (DESTINATION (OR DESTINATIONBITMAP (BITMAPCREATE (IQUOTIENT WIDTH WFACTOR)
                                                            (IQUOTIENT HEIGHT HFACTOR)
                                                            BITSPP]
          [if (AND (EQP WFACTOR 1)
                   (EQP HFACTOR 1))
              then (BITBLT BITMAP NIL NIL DESTINATION)
            else (BLTSHADE 0 DESTINATION)
                 (for Y from 0 to (SUB1 HEIGHT) do (BITBLT BITMAP 0 Y SCRATCH 0 (IQUOTIENT Y HFACTOR)
                                                           WIDTH 1 'INPUT 'PAINT))
                 (for X from 0 to (SUB1 WIDTH) do (BITBLT SCRATCH X 0 DESTINATION (IQUOTIENT X WFACTOR)
                                                          0 1 HEIGHT 'INPUT 'PAINT]
          DESTINATION])
```

(\FAST4BIT

```
  [LAMBDA (A B N MAP)                                            (* kbr%: "16-May-85 17:14")
                                                                 (* DECLARATIONS%: (BLOCKRECORD NIBBLE
                                                                 ((N1 BITS 4) (N2 BITS 4) (N3 BITS 4)
                                                                 (N4 BITS 4))))
    (bind AW (I _ 0) for J from 0 do (SETQ AW (\ADDBASE A J))
                                     (OR (IGREATERP N I)
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (NIBBLE N1) of AW)))
                                     (OR (IGREATERP N (add I 1))
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (NIBBLE N2) of AW)))
                                     (OR (IGREATERP N (add I 1))
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (NIBBLE N3) of AW)))
                                     (OR (IGREATERP N (add I 1))
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (NIBBLE N4) of AW)))
                                     (add I 1])

)

(CL:DEFUN ROTATE-BITMAP (SOURCE)
   "rotates the bitmap SOURCE by 90 degrees clockwise, returning a new bitmap"

;;; This must be compiled to work
    ;; Rotate a bitmap by 90 degrees clockwise.  Uses pilotbitblt hackery for maximum speed and confusion for the reader.
```

```
   (LET* ((SOURCE-HEIGHT (BITMAPHEIGHT SOURCE))
          (DESTINATION (BITMAPCREATE SOURCE-HEIGHT (BITMAPWIDTH SOURCE)))
          )
          ;; The ROTATE-BBT table maps scanlines of the SOURCE bitmap into columns of the DESTINATION bitmap.  The topmost scanline
          ;; (lowest address) maps into the rightmost column of the destination.  We proceed from top to bottom in the source, and from right to left
          ;; in the destination.  Refer to the Mesa PrincOps document for a description of Pilot BitBLT, and see also the declaration for the
          ;; PILOTBBT datatype.
          (ROTATE-BBT (create PILOTBBT
                             PBTDISJOINT _ T                          ; the bitmaps are separate
                             PBTDEST _ (ffetch (BITMAP BITMAPBASE) of DESTINATION)
                                                                      ; set the destination (held constant)
                             PBTSOURCE _ (ffetch (BITMAP BITMAPBASE) of SOURCE)
                                                                      ; set the source (incremented by 1 scanline per iteration)
                             PBTDESTBPL _ (UNFOLD (ffetch (BITMAP BITMAPRASTERWIDTH) of DESTINATION)
                                                  BITSPERWORD)        ; the destination is this many bits between scanlines
                             PBTSOURCEBPL _ 1                         ; move 1 bit of each source scanline per 1 scanline of the
                                                                      ; destination
                             PBTSOURCEBIT _ 0                         ; start at the first bit of each source scanline (held constant)
                             PBTDESTBIT _ (BITMAPWIDTH DESTINATION)
                                                                      ; start putting data into the destination on the right edge
                                                                      ; (pre-decremented)
                             PBTFLAGS _ 0                             ; replace mode (paint might be faster)
                             PBTHEIGHT _ (BITMAPHEIGHT DESTINATION)
                                                                      ; how high the destination is
                             PBTWIDTH _ 1                             ; how wide the destination stripe is
                             ))
          (SOURCE-WORD-WIDTH (ffetch (BITMAP BITMAPRASTERWIDTH) of SOURCE)))
         (for I from 1 to SOURCE-HEIGHT do (add (ffetch (PILOTBBT PBTDESTBIT) of ROTATE-BBT)
                                                -1)
                                           (\PILOTBITBLT ROTATE-BBT 0)

                                           ;; the line below is slower than need be, but works when the source crosses a segment.  A faster
                                           ;; way (which breaks on a segment cross) is to say

                                              ;;  (|add| (|ffetch| (PILOTBBT PBTSOURCELO) |of| ROTATE-BBT) SOURCE-WORD-WIDTH)

                                              (FREPLACE (PILOTBBT PBTSOURCE) OF ROTATE-BBT
                                                 WITH (\ADDBASE (FFETCH (PILOTBBT PBTSOURCE) OF ROTATE-BBT)
                                                                SOURCE-WORD-WIDTH)))
         DESTINATION))


(CL:DEFUN ROTATE-BITMAP-LEFT (SOURCE)
   "rotates the bitmap SOURCE by 90 degrees counter-clockwise, returning a new bitmap"

;;; This must be compiled to work

   ;; Rotate a bitmap by 90 degrees counter-clockwise.  Uses pilotbitblt hackery for maximum speed and confusion for the reader.
   (LET* ((SOURCE-WIDTH (BITMAPWIDTH SOURCE))
          (DESTINATION (BITMAPCREATE (BITMAPHEIGHT SOURCE)
                                     SOURCE-WIDTH))
          )
          ;; The ROTATE-BBT table maps columns of the SOURCE bitmap into rows of the DESTINATION bitmap.  The rightmost column maps into
          ;; the topmost row(lowest address) of the destination.  We proceed from right to left in the source, and from top to bottom in the
          ;; destination.  Refer to the Mesa PrincOps document for a description of Pilot BitBLT, and see also the declaration for the PILOTBBT
          ;; datatype.
          (ROTATE-BBT (CREATE PILOTBBT
                             PBTDISJOINT _ T                          ; the bitmaps are separate
                             PBTDEST _ (FFETCH (BITMAP BITMAPBASE) OF DESTINATION)
                                                                      ; set the destination (held constant)
                             PBTSOURCE _ (FFETCH (BITMAP BITMAPBASE) OF SOURCE)
                                                                      ; set the source
                             PBTDESTBPL _ 1                           ; the destination is this many bits between scanlines
                             PBTSOURCEBPL _ (UNFOLD (FFETCH (BITMAP BITMAPRASTERWIDTH) OF SOURCE)
                                                    BITSPERWORD)      ; move a scanline at a time.
                             PBTSOURCEBIT _ (BITMAPWIDTH SOURCE)      ; start getting data at the right edge of the source
                             PBTDESTBIT _ 0                           ; start putting data into the destination on the left edge
                             PBTFLAGS _ 0                             ; replace mode (paint might be faster)
                             PBTHEIGHT _ (BITMAPHEIGHT SOURCE)        ; how high the stripe is
                             PBTWIDTH _ 1                             ; how wide the destination stripe is
                             ))
          (DEST-WORD-WIDTH (FFETCH (BITMAP BITMAPRASTERWIDTH) OF DESTINATION)))
         (FOR I FROM 1 TO SOURCE-WIDTH DO (add (FFETCH (PILOTBBT PBTSOURCEBIT) OF ROTATE-BBT)
                                               -1)
                                          (\PILOTBITBLT ROTATE-BBT 0)

                                          ;; the line below is slower than need be, but works when the source crosses a segment.  A faster
                                          ;; way (which breaks on a segment cross) is to say

                                             ;;  (|add| (|ffetch| (PILOTBBT PBTSOURCELO) |of| ROTATE-BBT)
                                             ;; SOURCE-WORD-WIDTH)

                                             (FREPLACE (PILOTBBT PBTDEST) OF ROTATE-BBT
                                                WITH (\ADDBASE (FFETCH (PILOTBBT PBTDEST) OF ROTATE-BBT)
                                                               DEST-WORD-WIDTH)))
         DESTINATION))

(PUTPROPS HLDISPLAY FILETYPE CL:COMPILE-FILE)

(READVARS-FROM-STRINGS '(\4BITEXPANSIONTABLE)
```

```
       "({Y16 SMALLPOSP 0 0 15 240 255 3840 3855 4080 4095 61440 61455 61680 61695 65280 65295 65520 65535 })
       ")

(PUTPROPS HLDISPLAY COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1900 1988 1989 1990
                                    1992 1993 1994))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

PROPERTY INDEX

CONSTANT INDEX

RECORD INDEX