File created: 17-Feb-2021 13:49:06   **{DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>WINDOWSCROLL.;2**

changes to: (FNS SCROLL.HANDLER)

previous date: 16-Feb-94 12:36:43   {DSK}<Users>kaplan>Local>medley3.5>git-medley>sources>WINDOWSCROLL.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ WINDOWSCROLLCOMS
       [                                                              ; Scrolling stuff
        (FNS SCROLLW SCROLLBYREPAINTFN ADJUSTOFFSETS CREATESCROLLINGW IN/SCROLL/BAR? RELDSPXOFFSET RELDSPYOFFSET
             SCROLL.HANDLER \SCROLL.HANDLER.DOIT \DECODE.EXTENT.USE \UPDATE.EXTENT.IMAGE
             EXTENDPASTHORIZBOUNDARIES EXTENDPASTVERTBOUNDARIES REDISPLAYW FILLWITHBACKGROUND UPDATE/SCROLL/REG
             WTODSX WTODSY WXOFFSET WYOFFSET BITMAPSCROLLFN SCROLLBITMAP REDISPLAYBITMAP ULREDISPLAYBITMAP
             EXTENDEXTENT WIDTHIFWINDOW HEIGHTIFWINDOW)
                                                                      ; this function should be on LLDISPLAY but Ron has it checked
                                                                      ; out. Move it later - rrb.
        (FNS \DSPUNTRANSFORMREGION)
        (CURSORS VertScrollCursor ScrollUpCursor ScrollDownCursor HorizScrollCursor ScrollLeftCursor
               ScrollRightCursor VertThumbCursor HorizThumbCursor WAITINGCURSOR)
        (GLOBALVARS \LastInWindow VertScrollCursor ScrollUpCursor ScrollDownCursor ScrollLeftCursor
               ScrollRightCursor HorizScrollCursor)
        (INITVARS (SCROLLBARWIDTH 24)
               (SCROLLWAITTIME 100)
               (SCROLLBARSHADE 32800)
               (WAITBEFORESCROLLTIME 750)
               (WAITBETWEENSCROLLTIME 100))
        (DECLARE%: DONTEVAL@LOAD DOCOPY (ADDVARS (GLOBALVARS SCROLLBARWIDTH SCROLLWAITTIME SCROLLBARSHADE
                                                        WAITBEFORESCROLLTIME WAITBETWEENSCROLLTIME WAITINGCURSOR
                                                        ])
```

;; Scrolling stuff

```
(DEFINEQ

(SCROLLW
  [LAMBDA (WINDOW DX DY CONTINUOUSFLG)                                ; Edited 16-Feb-94 11:58 by nilsson

    ;; scrolls a window by DX in the X direction and DY in the Y direction.  If CONTINUOUSFLG is non-NIL, this is part of a continuous scroll so that the
    ;; window scrolling function can decide for example to scroll a constant smount.

    (\CHECKCARET WINDOW)
    (APPLY* (OR (fetch SCROLLFN of WINDOW)
                (FUNCTION SCROLLBYREPAINTFN))
           WINDOW DX DY CONTINUOUSFLG])

(SCROLLBYREPAINTFN
  [LAMBDA (WINDOW XDELTA YDELTA CONTINUOUSFLG)                        ; Edited 16-Feb-94 12:26 by nilsson

    ;; standard scrolling function that scrolls by blting existing bits and then calling the windows repaintfn to repaint the newly exposed bits.

    ;; changed 23-jul-86 to treat the part of the window that is coming from off screen as needing to be repainted.

    (PROG ((DSP (WINDOWPROP WINDOW 'DSP))
           (EXTENT (WINDOWPROP WINDOW 'EXTENT))
           (EXTENTUSE (WINDOWPROP WINDOW 'SCROLLEXTENTUSE))
           X CRHEIGHT CRWIDTH CRLEFT CRBOTTOM WHOLEHEIGHT WHOLEWIDTH XEXTENTUSE YEXTENTUSE ONSCREENREG ONSLEFT
           ONSBOTTOM ONSWIDTH ONSHEIGHT)
          (SETQ X (DSPCLIPPINGREGION NIL DSP))
          (SETQ CRLEFT (fetch (REGION LEFT) of X))
          (SETQ CRBOTTOM (fetch (REGION BOTTOM) of X))
          (SETQ CRWIDTH (fetch (REGION WIDTH) of X))
          (SETQ CRHEIGHT (fetch (REGION HEIGHT) of X))
          [AND EXTENT (SELECTQ EXTENTUSE
                          (NIL                                        ; original scrolling mode.
                               (SETQ XEXTENTUSE 'LIMIT)
                               (SETQ YEXTENTUSE '+))
                          ((T + - +- LIMIT)
                              (SETQ XEXTENTUSE (SETQ YEXTENTUSE EXTENTUSE)))
                          (-+ (SETQ XEXTENTUSE (SETQ YEXTENTUSE '+-)))
                          (COND
                             [(LISTP EXTENTUSE)                       ; CAR is X spec, CDR is Y spec
                              (SETQ XEXTENTUSE (\DECODE.EXTENT.USE (CAR EXTENTUSE)))
                              (SETQ YEXTENTUSE (\DECODE.EXTENT.USE (CDR EXTENTUSE]
                             (T                                       ; unknown value, default to T
                                (SETQ XEXTENTUSE (SETQ YEXTENTUSE T]

;;; calculate the amount to be moved in X
```

```
          [COND
             ((FLOATP XDELTA)                                      ; thumb scroll, XDELTA gives the fraction of the way from the left
                                                                   ; margin the cursor was.
              (COND
                 [(AND EXTENT (NEQ (fetch (REGION WIDTH) of EXTENT)
                                   -1))
                  (PROG (OLDX NEWX)

                   ;; if there is an extent, calculate a value of XDELTA that moves to the proper place.  If there is not, Don't do anything.

                        [SETQ NEWX (IPLUS (fetch (REGION LEFT) of EXTENT)
                                          (FIXR (FTIMES XDELTA (IDIFFERENCE (fetch (REGION WIDTH) of EXTENT)
                                                                      CRWIDTH]
                        (SETQ OLDX (WXOFFSET NIL DSP))
                        (SETQ XDELTA (IDIFFERENCE OLDX NEWX]
                 (T (SETQ XDELTA 0]
          [COND
             (CONTINUOUSFLG

                    ;; if continuous set it scroll by the linefeed height {no particularly good reason why the linefeed height but why not}.

                    (COND
                       ((EQ XDELTA 0))
                       [(IGREATERP XDELTA 0)                       ; linefeed height is normally negative.
                        (SETQ XDELTA (IMINUS (DSPLINEFEED NIL DSP]
                       (T (SETQ XDELTA (DSPLINEFEED NIL DSP]

;;; calculate the amount to be moved in Y

          [COND
             ((FLOATP YDELTA)                                      ; thumb scroll, YDELTA gives the fraction of the way from the top
                                                                   ; margin the cursor was.
              (COND
                 [(AND EXTENT (NEQ (fetch (REGION HEIGHT) of EXTENT)
                                   -1))
                  (PROG (OLDY NEWY)

                   ;; if there is an extent, calculate a value of YDELTA that moves to the proper place.  If there is not, Don't do anything.

                        (SETQ NEWY (IPLUS (FIXR (FTIMES (FDIFFERENCE 1.0 YDELTA)
                                                        (IDIFFERENCE (fetch (REGION HEIGHT) of EXTENT)
                                                               CRHEIGHT)))
                                          (fetch (REGION BOTTOM) of EXTENT)))
                        (SETQ OLDY (WYOFFSET NIL DSP))
                        (SETQ YDELTA (IDIFFERENCE OLDY NEWY]
                 (T (SETQ YDELTA 0]
          [COND
             (CONTINUOUSFLG                                        ; if continuous set it scroll by the linefeed height
                    (COND
                       ((EQ YDELTA 0))
                       [(IGREATERP YDELTA 0)                       ; linefeed height is normally negative.
                        (SETQ YDELTA (IMINUS (DSPLINEFEED NIL DSP]
                       (T (SETQ YDELTA (DSPLINEFEED NIL DSP]
          (COND
             [[NOT (SUBREGIONP (fetch (SCREEN SCREGION) of (fetch (WINDOW SCREEN) of WINDOW))
                          (WINDOWPROP WINDOW 'REGION]              ; reduce clipping region to be that part of the window that is on
                                                                   ; the screen.
              (COND
                 ([SETQ ONSCREENREG (INTERSECTREGIONS X (\DSPUNTRANSFORMREGION (fetch (SCREEN SCREGION)
                                                                                  of (fetch (WINDOW SCREEN)
                                                                                        of WINDOW))
                                                               (fetch IMAGEDATA of DSP]
                                                                   ; note what part of the region is on the screen too.
                  (SETQ ONSLEFT (fetch (REGION LEFT) of ONSCREENREG))
                  (SETQ ONSBOTTOM (fetch (REGION BOTTOM) of ONSCREENREG))
                  (SETQ ONSWIDTH (fetch (REGION WIDTH) of ONSCREENREG))
                  (SETQ ONSHEIGHT (fetch (REGION HEIGHT) of ONSCREENREG)))
                 (T                                                ; whole image is off the screen.  Just move the coordinates.
                    (WXOFFSET XDELTA DSP)
                    (WYOFFSET YDELTA DSP)
                    (RETURN]
             (T (SETQ ONSLEFT CRLEFT)
                (SETQ ONSBOTTOM CRBOTTOM)
                (SETQ ONSWIDTH CRWIDTH)
                (SETQ ONSHEIGHT CRHEIGHT)))

       ;; only one of XDELTA or YDELTA should be non-zero but do both anyway.  When both can be non-zero, this code should avoid calling the
       ;; repaintfn on the part of the object that is scrolled on by X but then scrolled off by Y.
                                                                   ; do X first because in the common case of printing it is faster to
                                                                   ; do it first.
          (COND
             ((AND (NEQ XDELTA 0)
                   (COND
                      ((AND EXTENT (NEQ XEXTENTUSE T)
                            (NEQ (fetch (REGION WIDTH) of EXTENT)
                                 -1))                              ; use the extent to limit the scrolling.
                                                                   ; for now limit right extent to right of window ETC.  ie keep it
                                                                   ; always visible.
                       (SETQ XDELTA (IMIN (IDIFFERENCE CRLEFT (IDIFFERENCE (fetch (REGION LEFT) of EXTENT)
                                                                     (SELECTQ XEXTENTUSE
                                                                         ((+- +)
                                                                 ; if X is allowed to go off to right move effective left of extent.
                                                                               CRWIDTH)
                                                                          0)))
                                          (IMAX (IDIFFERENCE (IPLUS CRLEFT CRWIDTH)
                                                        (PLUS (fetch (REGION PRIGHT) of EXTENT)
                                                              (SELECTQ XEXTENTUSE
                                                                  ((- +-)
                                                                      CRWIDTH)
                                                                  0)))
                                                XDELTA)))        ; make sure it is still not 0
                       (NEQ XDELTA 0))
                      (T T)))
              (BITBLT WINDOW ONSLEFT ONSBOTTOM WINDOW (IPLUS XDELTA ONSLEFT)
                     ONSBOTTOM ONSWIDTH ONSHEIGHT 'INPUT 'REPLACE)
              (WXOFFSET XDELTA DSP)
              (SETQ ONSLEFT (IDIFFERENCE ONSLEFT XDELTA))
              (REDISPLAYW WINDOW (COND
                                     ((IGREATERP XDELTA 0)       ; moving to right, create new region on left for repaintfn
                                      (CREATEREGION ONSLEFT ONSBOTTOM (IMIN XDELTA ONSWIDTH)
                                             ONSHEIGHT))
                                     (T                          ; moving to left.
                                        (CREATEREGION (IMAX (IPLUS ONSLEFT ONSWIDTH XDELTA)
                                                            ONSLEFT)
                                               ONSBOTTOM
                                               (IMIN (IMINUS XDELTA)
                                                     ONSWIDTH)
                                               ONSHEIGHT)))
                     T)))
          (COND
             ((AND (NEQ YDELTA 0)
                   (COND
                      ((AND EXTENT (NEQ YEXTENTUSE T)
                            (NEQ (fetch (REGION HEIGHT) of EXTENT)
                                 -1))                            ; limit amount by the extent
                       (SETQ YDELTA (IMIN (IDIFFERENCE CRBOTTOM (IDIFFERENCE (fetch (REGION BOTTOM) of EXTENT)
                                                                        (SELECTQ YEXTENTUSE
                                                                            ((+- +)
                                                                 ; if Y is allowed to go off to top, move effective bottom of extent.
                                                                                 CRHEIGHT)
                                                                            0)))
                                          (IMAX (IDIFFERENCE (IPLUS CRBOTTOM CRHEIGHT)
                                                        (PLUS (fetch (REGION PTOP) of EXTENT)
                                                              (SELECTQ YEXTENTUSE
                                                                  ((- +-)
                                                                      CRHEIGHT)
                                                                  0)))
                                                YDELTA)))        ; make sure its still not 0
                       (NEQ YDELTA 0))
                      (T T)))                                    ; move the current image if any of it is still in view.
              (BITBLT WINDOW ONSLEFT ONSBOTTOM WINDOW ONSLEFT (IPLUS YDELTA ONSBOTTOM)
                     ONSWIDTH ONSHEIGHT 'INPUT 'REPLACE)
              (WYOFFSET YDELTA DSP)                              ; use X as pointer to bottom in scrolled clipping region.
              (SETQ X (IDIFFERENCE ONSBOTTOM YDELTA))
              (REDISPLAYW WINDOW [COND
                                     ((IGREATERP YDELTA 0)       ; moving up.
                                      (CREATEREGION ONSLEFT X ONSWIDTH (IMIN YDELTA ONSHEIGHT)))
                                     (T                          ; moving down, fill in top
                                        (CREATEREGION ONSLEFT (IMAX (IPLUS ONSHEIGHT X YDELTA)
                                                                    X)
                                               ONSWIDTH
                                               (IMIN (IMINUS YDELTA)
                                                     ONSHEIGHT]
                     T)))
          (RETURN])
```

```
(ADJUSTOFFSETS
  [LAMBDA (WINDOW XDELTA YDELTA)                                 ; Edited 16-Feb-94 12:27 by nilsson
    (PROG [(DSP (WINDOWPROP WINDOW 'DSP]                         ; determine the change in offsets caused by the scroll.  and
                                                                 ; redisplay the graph.
          (WYOFFSET YDELTA DSP)
          (WXOFFSET XDELTA DSP)
          (RETURN])
```

```
(CREATESCROLLINGW
  [LAMBDA (TITLE BORDER)                                         ; Edited 16-Feb-94 12:27 by nilsson
    (WINDOWPROP (CREATEW NIL TITLE BORDER)
           'SCROLLFN
           (FUNCTION SCROLLBYREPAINTFN])
```

```
(IN/SCROLL/BAR?
  [LAMBDA (WINDOW X Y)                                           ; Edited 16-Feb-94 12:27 by nilsson
                                                                 ; is X, Y in the scroll bar for WINDOW?
    (AND (fetch SCROLLFN of WINDOW)
         (NOT (WINDOWPROP WINDOW 'NOSCROLLBARS))
         (COND
            ((INSIDE? (fetch REG of WINDOW)
                    X Y)

             ;; if it is inside the window, it is not in its scroll bar.  This handles case where window is near left or bottom edge.

             NIL)
            [(INSIDE? (fetch (WINDOW VERTSCROLLREG) of WINDOW)
                    X Y)
             (PROG [(EXTENT (WINDOWPROP WINDOW 'EXTENT))
                    (EXTENTUSE (WINDOWPROP WINDOW 'SCROLLEXTENTUSE]
                   (RETURN (COND
                              [(OR (NOT EXTENT)
                                   (EQ (fetch (REGION WIDTH) of EXTENT)
                                       -1)
                                   (NOT EXTENTUSE)
                                   (NEQ (COND
                                           ((LISTP EXTENTUSE)
                                            (\DECODE.EXTENT.USE (CDR EXTENTUSE)))
                                           (T (\DECODE.EXTENT.USE EXTENTUSE)))
                                        'LIMIT]
                              (T (EXTENDPASTVERTBOUNDARIES (DSPCLIPPINGREGION NIL WINDOW)
                                        EXTENT]
            ((INSIDE? (fetch (WINDOW HORIZSCROLLREG) of WINDOW)
                    X Y)

             ;; if there is an extent, make sure it is past the current view boundaries.  -1 is used to mark an unknown width, treat it as if EXTENT
             ;; wasn't given.

             (PROG [(EXTENT (WINDOWPROP WINDOW 'EXTENT))
                    (EXTENTUSE (WINDOWPROP WINDOW 'SCROLLEXTENTUSE]
                   (RETURN (COND
                              [(OR (NOT EXTENT)
                                   (EQ (fetch (REGION WIDTH) of EXTENT)
                                       -1)
                                   (NEQ (COND
                                           ((LISTP EXTENTUSE)
                                            (\DECODE.EXTENT.USE (CAR EXTENTUSE)))
                                           (T (\DECODE.EXTENT.USE EXTENTUSE)))
                                        'LIMIT]
                              (T (EXTENDPASTHORIZBOUNDARIES (DSPCLIPPINGREGION NIL WINDOW)
                                        EXTENT])
```

```
(RELDSPXOFFSET
  [LAMBDA (DX DISPLAYSTREAM)                                     ; Edited 16-Feb-94 12:28 by nilsson
                                                                 ; relative offsetting function.
    (DSPXOFFSET (IPLUS DX (DSPXOFFSET NIL DISPLAYSTREAM))
           DISPLAYSTREAM])

(RELDSPYOFFSET
  [LAMBDA (DY DISPLAYSTREAM)                                     ; Edited 16-Feb-94 12:28 by nilsson
                                                                 ; relative offsetting function.
    (DSPYOFFSET (IPLUS DY (DSPYOFFSET NIL DISPLAYSTREAM))
           DISPLAYSTREAM])

(SCROLL.HANDLER
  [LAMBDA (WINDOW)                                               ; Edited 17-Feb-2021 13:48 by rmk:
                                                                 ; Edited 16-Feb-94 12:29 by nilsson

    ;; cursor has moved into scroll region.  region of a window that has a scrollfn and has been IN/SCROLL/BAR?  Handle interaction to determine type
    ;; of scroll, if any, desired.                               ; returns non-NIL if scrolling was applicable.
    (PROG (SCROLLREG SCROLLW BUTTON DIRECTION SCROLLCURSOR LEFTCURSOR RIGHTCURSOR MIDDLECURSOR TIMEDOWN
                 CONTINUOUSSCROLL? TIMEIN TIMEINTIMER)           ; create a window as the easiest thing to do.  Fairly inefficient.

     ;; if the main window is not open, it was probably closed before we got control here.  Don't do anything.

          (OR (OPENWP WINDOW)
              (RETURN))
          (GETMOUSESTATE)
          (COND
             ((AND (INSIDE? (SETQ SCROLLREG (fetch (WINDOW VERTSCROLLREG) of WINDOW))
                          LASTMOUSEX LASTMOUSEY)
                   (PROGN (DISMISS SCROLLWAITTIME)
                          (GETMOUSESTATE)
                          (INSIDE? SCROLLREG LASTMOUSEX LASTMOUSEY)))
              [COND
                 ((SETQ SCROLLW (fetch (WINDOW VERTSCROLLWINDOW) of WINDOW))
                                                                 ; if there is one already, reopen it.
                  (OPENW SCROLLW))
                 ((SETQ SCROLLW (replace (WINDOW VERTSCROLLWINDOW) of WINDOW with (CREATEW SCROLLREG NIL 2)))

                  ;; RMK:  So that the scroll bar is recognizable and connected (unreferenced) to its scrollee window

                  (WINDOWPROP SCROLLW 'VERTICALSCROLLBARFOR (LOC WINDOW]
              (SETQ DIRECTION 'VERT)
              (SETQ SCROLLCURSOR VertScrollCursor)
              (SETQ LEFTCURSOR ScrollUpCursor)
              (SETQ RIGHTCURSOR ScrollDownCursor)
              (SETQ MIDDLECURSOR VertThumbCursor))
```

```
            ((AND (INSIDE? (SETQ SCROLLREG (fetch (WINDOW HORIZSCROLLREG) of WINDOW))
                         LASTMOUSEX LASTMOUSEY)
                  (PROGN (DISMISS SCROLLWAITTIME)
                         (GETMOUSESTATE)
                         (INSIDE? SCROLLREG LASTMOUSEX LASTMOUSEY)))
             [COND
                ((SETQ SCROLLW (fetch (WINDOW HORIZSCROLLWINDOW) of WINDOW))
                                                            ; if there is one already, reopen it.
                 (OPENW SCROLLW))
                ((SETQ SCROLLW (replace (WINDOW HORIZSCROLLWINDOW) of WINDOW with (CREATEW SCROLLREG NIL 2)))
                 (WINDOWPROP SCROLLW 'HORIZONTALSCROLLBARFOR (LOC WINDOW]
             (SETQ DIRECTION 'HORIZ)
             (SETQ SCROLLCURSOR HorizScrollCursor)
             (SETQ LEFTCURSOR ScrollLeftCursor)
             (SETQ MIDDLECURSOR HorizThumbCursor)
             (SETQ RIGHTCURSOR ScrollRightCursor))
            (T                                              ; moved out quickly
               (RETURN NIL)))
         (\UPDATE.EXTENT.IMAGE SCROLLW DIRECTION WINDOW)

   ;; set up the timer for when to bring the window to the top.  This gives the user a chance to notice that the scroll bar has come up and get out of it
   ;; if it was unintentional.

         (SETQ TIMEIN (SETUPTIMER 1200))
         (RETURN (RESETFORM (CURSOR SCROLLCURSOR)
                        (PROG NIL
                          LP  (GETMOUSESTATE)
                              (COND
                                 ((NOT (OPENWP WINDOW))            ; the user closed the window, quit.
                                  (CLOSEW SCROLLW)
                                  (SETQ \LastInWindow NIL)
                                  (RETURN T)))
                              (COND
                                 ((AND TIMEIN (TIMEREXPIRED? TIMEIN))

                                  ;; after a little while, bring the window to the top.  This avoids bringing it up if nothing is happening.

                                  (SETQ TIMEIN NIL)
                                  (TOTOPW WINDOW)))
                              (COND
                                 ((NOT (INSIDE? SCROLLREG LASTMOUSEX LASTMOUSEY))
                                                                   ; if cursor is no longer in scroll region quit.
                                  (CLOSEW SCROLLW)

                                  ;; if the mouse is in the window, set last in window so window will get control again.  If it is outside, don't
                                  ;; set it so that the cursoroutfn for WINDOW will get called.

                                  (AND (INSIDE? (WINDOWPROP WINDOW 'REGION)
                                              LASTMOUSEX LASTMOUSEY)
                                       (SETQ \LastInWindow NIL))
                                  (RETURN T)))                     ; bring the scroll window to the top so that it will be visible.
                              (TOTOPW SCROLLW)
                              [COND
                                 [(LASTMOUSESTATE UP)              ; no buttons down;  if there was one down, take action;
                                                                   ; otherwise, wait for one to go down.
                                  (COND
                                     (BUTTON (COND
                                                (CONTINUOUSSCROLL?
                                                                   ; were continuously scrolling, stop it.
                                                       (SETQ CONTINUOUSSCROLL? NIL))
                                                (T (\SCROLL.HANDLER.DOIT WINDOW BUTTON DIRECTION SCROLLREG
                                                          LASTMOUSEX LASTMOUSEY)
                                                   (\UPDATE.EXTENT.IMAGE SCROLLW DIRECTION WINDOW)))
                                            (CURSOR SCROLLCURSOR)
                                            (SETQ BUTTON)          ; if a button went up, reset the timedown for scrolling.
                                            (SETQ TIMEDOWN)
                                            (SETQ CONTINUOUSSCROLL? NIL))
                                     (T (BLOCK]
                                 [(LASTMOUSESTATE (OR LEFT RIGHT))
                                  (COND
                                     ((AND (LASTMOUSESTATE LEFT)
                                           (NEQ BUTTON 'LEFT))     ; LEFT button just when down.
                                      (SETQ BUTTON 'LEFT)
                                      (SETQ TIMEDOWN (CLOCK 0))
                                      (CURSOR LEFTCURSOR))
                                     ((AND (LASTMOUSESTATE RIGHT)
                                           (NEQ BUTTON 'RIGHT))    ; RIGHT button just when down.
                                      (SETQ BUTTON 'RIGHT)
                                      (SETQ TIMEDOWN (CLOCK 0))
                                      (CURSOR RIGHTCURSOR))
                                     ((AND CONTINUOUSSCROLL? (\CLOCKGREATERP TIMEDOWN WAITBETWEENSCROLLTIME))
                                                                   ; button is still down, keep scrolling.
                                                                   ; note time before calling scroll fn so time to display is included in
                                                                   ; the wait time.
                                      (SETQ TIMEDOWN (\CLOCK0 TIMEDOWN))
                                      (\SCROLL.HANDLER.DOIT WINDOW BUTTON DIRECTION SCROLLREG LASTMOUSEX
                                             LASTMOUSEY T)
                                      (\UPDATE.EXTENT.IMAGE SCROLLW DIRECTION WINDOW))
                                     ((\CLOCKGREATERP TIMEDOWN WAITBEFORESCROLLTIME)
                                                                   ; has enough time past to start continuous scroll?
                                      (SETQ CONTINUOUSSCROLL? T]
                                 ((LASTMOUSESTATE MIDDLE)
                                  (COND
                                     ((NEQ BUTTON 'MIDDLE)       ; MIDDLE button just when down.
                                      (SETQ BUTTON 'MIDDLE)      ; don't keep track of time down for middle buttons.
                                      (CURSOR MIDDLECURSOR))
                                     (T NIL]
                              (GO LP])
```

(**SCROLL.HANDLER.DOIT**

```
  [LAMBDA (WINDOW BUTTON DIRECTION SCROLLREGION XPOS YPOS CONTINUOUS?)
                                                                        ; Edited 16-Feb-94 12:29 by nilsson

    ;; decodes how far to scroll given that the button was let up at position XPOS YPOS in the scroll region SCROLLREGION.

    (ERSETQ (PROG ((WBORDER (WINDOWPROP WINDOW 'BORDER))
                   LFT TOP (SIZEOFORIGIN 8))                           ; correct for the border on the window so that it never moves
                                                                        ; more than the amount that is seen.
                  (SETQ LFT (IPLUS WBORDER (fetch (REGION LEFT) of SCROLLREGION)))
                  (SETQ TOP (IDIFFERENCE (fetch (REGION TOP) of SCROLLREGION)
                                   WBORDER))
                  (RETURN (SCROLLW WINDOW
                                  (COND
                                     ((EQ DIRECTION 'HORIZ)
                                      (SELECTQ BUTTON
                                          (LEFT                         ; always scroll at least 1
                                                (IMIN (IDIFFERENCE LFT XPOS)
                                                      1))
                                          (RIGHT                        ; correct for border in window.
                                                 (IMAX (IDIFFERENCE XPOS LFT)
                                                       1))
                                          (MIDDLE [COND
                                                     ((IGREATERP (IPLUS LFT SIZEOFORIGIN)
                                                              XPOS)
                                                                        ; make a portion of the left of the scroll bar indicate left edge of
                                                                        ; doc since it is a common case.
                                                      0.0)
                                                     (T (MIN 1.0 (MAX 0.0 (FQUOTIENT (IDIFFERENCE XPOS
                                                                                              (IPLUS LFT
                                                                                                     SIZEOFORIGIN)
                                                                                              )
                                                                                 (IDIFFERENCE
                                                                                  (fetch (REGION WIDTH)
                                                                                     of SCROLLREGION)
                                                                                  (IPLUS 4 SIZEOFORIGIN])
                                          (SHOULDNT)))
                                     (T 0))
                                  (COND
                                     ((EQ DIRECTION 'VERT)
                                      (SELECTQ BUTTON
                                          (LEFT                         ; always scroll at least 1
                                                (IMAX (IDIFFERENCE TOP YPOS)
                                                      1))
                                          (RIGHT (IMIN (IDIFFERENCE YPOS TOP)
                                                       -1))
                                          (MIDDLE [COND
                                                     ((IGREATERP YPOS (IDIFFERENCE TOP SIZEOFORIGIN))
                                                                        ; make a portion of the top of the scroll bar indicate top edge of
                                                                        ; doc since it is a common case.
                                                      0.0)
                                                     (T (MIN 1.0 (MAX 0.0 (FQUOTIENT (IDIFFERENCE (IDIFFERENCE
                                                                                                     TOP
                                                                                                     SIZEOFORIGIN)
                                                                                              YPOS)
                                                                                 (IDIFFERENCE
                                                                                  (fetch (REGION HEIGHT)
                                                                                     of SCROLLREGION)
                                                                                  (IPLUS 4 SIZEOFORIGIN])
                                          (SHOULDNT)))
                                     (T 0))
                                  CONTINUOUS?])
```

(**DECODE.EXTENT.USE**

```
  [LAMBDA (EXTENTUSE)                                                   ; Edited 16-Feb-94 12:30 by nilsson

;;; decodes an indicator of how the extent should be used to limit scrolling.

    (SELECTQ EXTENTUSE
         (NIL 'LIMIT)
         ((LIMIT T + - +-)
              EXTENTUSE)
         (-+ '+-)
         T])
```

(**UPDATE.EXTENT.IMAGE**

```
  [LAMBDA (SCROLLBARW DIRECTION SCROLLINGW)                        ; Edited 16-Feb-94 12:32 by nilsson
                                                                   ; paints the appropriate grey region in the scrolling bar window.
    (CLEARW SCROLLBARW)
    (PROG [(EXTENT (WINDOWPROP SCROLLINGW 'EXTENT]
          (OR EXTENT (RETURN NIL))
          (COND
             [(EQ DIRECTION 'VERT)
              (PROG (GRAYHEIGHT GRAYBOTTOM SCROLLWIDTH SCROLLHEIGHT (WINREGION (DSPCLIPPINGREGION NIL SCROLLINGW
                                                                                                    ))
                           (SCROLLREGION (DSPCLIPPINGREGION NIL SCROLLBARW))
                           WINHEIGHT
                           (EXHEIGHT (fetch (REGION HEIGHT) of EXTENT)))
                                                                   ; -1 is used to mark an extent of unknown height.  If height is 0,
                                                                   ; return also.
                    (OR (GREATERP EXHEIGHT 0)
                        (RETURN))
                    (SETQ SCROLLWIDTH (fetch (REGION WIDTH) of SCROLLREGION))
                    (SETQ SCROLLHEIGHT (fetch (REGION HEIGHT) of SCROLLREGION))
                    (SETQ WINHEIGHT (fetch (REGION HEIGHT) of WINREGION))
                    [SETQ GRAYHEIGHT (IMAX 2 (IMIN SCROLLHEIGHT (IQUOTIENT (ITIMES WINHEIGHT SCROLLHEIGHT)
                                                                       EXHEIGHT]
                    (SETQ GRAYBOTTOM (IDIFFERENCE (IDIFFERENCE SCROLLHEIGHT
                                                         (IQUOTIENT (ITIMES SCROLLHEIGHT
                                                                           (IDIFFERENCE (fetch (REGION TOP)
                                                                                           of EXTENT)
                                                                                  (fetch (REGION TOP)
                                                                                     of WINREGION)))
                                                                EXHEIGHT))
                                            GRAYHEIGHT))
                    (BITBLT NIL NIL NIL SCROLLBARW 0 GRAYBOTTOM SCROLLWIDTH GRAYHEIGHT 'TEXTURE 'REPLACE
                           BLACKSHADE)
                    (BITBLT NIL NIL NIL SCROLLBARW 1 (IPLUS GRAYBOTTOM 2)
                           (IDIFFERENCE SCROLLWIDTH 2)
                           (IDIFFERENCE GRAYHEIGHT 4)
                           'TEXTURE
                           'REPLACE
                           (OR (TEXTUREP SCROLLBARSHADE)
                               32800]
             ((EQ DIRECTION 'HORIZ)
              (PROG (GRAYWIDTH GRAYLEFT SCROLLWIDTH SCROLLHEIGHT (WINREGION (DSPCLIPPINGREGION NIL SCROLLINGW))
                           (SCROLLREGION (DSPCLIPPINGREGION NIL SCROLLBARW))
                           WINWIDTH
                           (EXWIDTH (fetch (REGION WIDTH) of EXTENT)))
                                                                   ; -1 is used to mark an EXTENT of unknown width.  If width is
                                                                   ; zero, return too.
                    (AND (GREATERP 0 EXWIDTH)
                         (RETURN))
                    (SETQ SCROLLWIDTH (fetch (REGION WIDTH) of SCROLLREGION))
                    (SETQ SCROLLHEIGHT (fetch (REGION HEIGHT) of SCROLLREGION))
                    (SETQ WINWIDTH (fetch (REGION WIDTH) of WINREGION))
                    (SETQ GRAYWIDTH (IMIN SCROLLWIDTH (IQUOTIENT (ITIMES WINWIDTH SCROLLWIDTH)
                                                             EXWIDTH)))
                    (SETQ GRAYLEFT (IQUOTIENT (ITIMES WINWIDTH (IDIFFERENCE (fetch (REGION LEFT) of WINREGION)
                                                                      (fetch (REGION LEFT) of EXTENT)))
                                          EXWIDTH))
                    (BITBLT NIL NIL NIL SCROLLBARW GRAYLEFT 0 GRAYWIDTH SCROLLHEIGHT 'TEXTURE 'REPLACE
                           BLACKSHADE)
                    (BITBLT NIL NIL NIL SCROLLBARW (IPLUS GRAYLEFT 2)
                           1
                           (IDIFFERENCE GRAYWIDTH 4)
                           (IDIFFERENCE SCROLLHEIGHT 2)
                           'TEXTURE
                           'REPLACE
                           (OR (TEXTUREP SCROLLBARSHADE)
                               32800])
```

(EXTENDPASTHORIZBOUNDARIES

```
  [LAMBDA (VIEW EXTENT)                                            ; Edited 16-Feb-94 12:32 by nilsson
                                                                   ; does VIEW entirely cover the hoizontal dimensions of
                                                                   ; EXTENT?

    (OR (IGREATERP (fetch (REGION LEFT) of VIEW)
               (fetch (REGION LEFT) of EXTENT))
        (IGREATERP (fetch (REGION RIGHT) of EXTENT)
               (fetch (REGION RIGHT) of VIEW])
```

(EXTENDPASTVERTBOUNDARIES

```
  [LAMBDA (VIEW EXTENT)                                            ; Edited 16-Feb-94 12:33 by nilsson
                                                                   ; does VIEW entirely cover the vertical dimensions of EXTENT?
    (OR (IGREATERP (fetch (REGION BOTTOM) of VIEW)
               (fetch (REGION BOTTOM) of EXTENT))
        (IGREATERP (fetch (REGION TOP) of EXTENT)
               (fetch (REGION TOP) of VIEW])
```

(REDISPLAYW

```
  [LAMBDA (WINDOW REGION ALWAYSFLG)
    (WINDOWOP 'REDISPLAYFN (fetch (WINDOW SCREEN) of WINDOW)
           WINDOW REGION ALWAYSFLG])
```

(**FILLWITHBACKGROUND**

```
  [LAMBDA (WIN REG)                                          ; Edited 16-Feb-94 12:33 by nilsson
                                                             ; fills a window with its background.  This is the default window
                                                             ; repainting function.
    (DSPFILL REG (DSPTEXTURE NIL WIN)
           'REPLACE
           (WINDOWPROP WIN 'DSP])
```

(**UPDATE/SCROLL/REG**

```
  [LAMBDA (WINDOW)                                           ; Edited 16-Feb-94 12:34 by nilsson
                                                             ; updates the scroll region field of the WINDOW
    (COND
       ((fetch (WINDOW SCROLLFN) of WINDOW)
        (PROG ((IMAGEREG (fetch (WINDOW REG) of WINDOW)))    ; kill the cache for the scroll region.
              (COND
                 ((fetch (WINDOW VERTSCROLLWINDOW) of WINDOW)
                  (CLOSEW (fetch (WINDOW VERTSCROLLWINDOW) of WINDOW))
                  (replace (WINDOW VERTSCROLLWINDOW) of WINDOW with NIL)))
              (COND
                 ((fetch (WINDOW HORIZSCROLLWINDOW) of WINDOW)
                  (CLOSEW (fetch (WINDOW HORIZSCROLLWINDOW) of WINDOW))
                  (replace (WINDOW HORIZSCROLLWINDOW) of WINDOW with NIL)))
              [replace (WINDOW VERTSCROLLREG) of WINDOW with (create REGION
                                                                    LEFT _ (IMAX 0 (IDIFFERENCE
                                                                                    (fetch (REGION LEFT)
                                                                                       of IMAGEREG)
                                                                                    SCROLLBARWIDTH))
                                                                    BOTTOM _ (fetch (REGION BOTTOM) of IMAGEREG)
                                                                    WIDTH _ SCROLLBARWIDTH
                                                                    HEIGHT _
                                                                    (IPLUS (fetch (REGION HEIGHT) of IMAGEREG)
                                                                           (COND
                                                                              [(fetch (WINDOW WTITLE) of WINDOW)
                                                                               (DSPLINEFEED
                                                                                NIL
                                                                                (fetch (SCREEN SCTITLEDS)
                                                                                   of (fetch (WINDOW SCREEN)
                                                                                         of WINDOW]
                                                                              (T 0]
              (replace (WINDOW HORIZSCROLLREG) of WINDOW with (create REGION
                                                                     LEFT _ (fetch (REGION LEFT) of IMAGEREG)
                                                                     BOTTOM _ (IMAX 0 (IDIFFERENCE
                                                                                       (fetch (REGION BOTTOM)
                                                                                          of IMAGEREG)
                                                                                       SCROLLBARWIDTH))
                                                                     WIDTH _ (fetch (REGION WIDTH) of IMAGEREG)
                                                                     HEIGHT _ SCROLLBARWIDTH])
```

(**WTODSX**

```
  [LAMBDA (WX WINDOW)                                        ; Edited 16-Feb-94 12:34 by nilsson
    ;; converts from the window natural coordinates which have 0,0 at lower left corner of the window and the displaystreams coordinates.
    (IPLUS WX (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL (fetch DSP of WINDOW])
```

(**WTODSY**

```
  [LAMBDA (WY WINDOW)                                        ; Edited 16-Feb-94 12:34 by nilsson
    ;; converts from the window natural coordinates which have 0,0 at lower left corner of the window and the displaystreams coordinates.
    (IPLUS WY (fetch (REGION BOTTOM) of (DSPCLIPPINGREGION NIL (fetch DSP of WINDOW])
```

(**WXOFFSET**

```
  [LAMBDA (DX WINDOW)                                        ; Edited 16-Feb-94 12:26 by nilsson
    ;; offsets a displaystream by a given delta but leaves its clipping region where it was.  Used for offsetting display streams under window.
    (PROG [CR (DS (OR (DISPLAYSTREAMP (\OUTSTREAMARG WINDOW))
                      (\ILLEGAL.ARG WINDOW]
          (SETQ CR (DSPCLIPPINGREGION NIL DS))
          (RETURN (PROG1 (fetch (REGION LEFT) of CR)
                      (COND
                         ((NUMBERP DX)
                          (DSPXOFFSET (IPLUS DX (DSPXOFFSET NIL DS))
                                 DS)
                          (add (fetch (REGION LEFT) of CR)
                               (IMINUS DX))                  ; recall DSPCLIPPINGREGION to update dependent fields in
                                                             ; DS.
                          (DSPCLIPPINGREGION CR DS))))])
```

```
(WYOFFSET
  [LAMBDA (DY WINDOW)                                                 ; Edited 16-Feb-94 12:26 by nilsson

    ;; offsets a displaystream by a given delta but leaves its clipping region where it was.  Used for offsetting display streams under window.

    (PROG [CR (DS (OR (DISPLAYSTREAMP (\OUTSTREAMARG WINDOW))
                      (\ILLEGAL.ARG WINDOW]
          (SETQ CR (DSPCLIPPINGREGION NIL DS))
          (RETURN (PROG1 (fetch (REGION BOTTOM) of CR)
                         (COND
                            ((NUMBERP DY)
                             (DSPYOFFSET (IPLUS DY (DSPYOFFSET NIL DS))
                                    DS)
                             (add (fetch (REGION BOTTOM) of CR)
                                  (IMINUS DY))                        ; recall DSPCLIPPINGREGION to update dependent fields in
                                                                      ; DS.
                             (DSPCLIPPINGREGION CR DS))))])

(BITMAPSCROLLFN
  [LAMBDA (WINDOW XDELTA YDELTA)                                      ; Edited 16-Feb-94 12:34 by nilsson
                                                                      ; scrolls a bitmap under a window
    (SCROLLBITMAP (WINDOWPROP WINDOW 'BITMAP)
           WINDOW XDELTA YDELTA])

(SCROLLBITMAP
  [LAMBDA (BITMAP WINDOW XDELTA YDELTA)                               ; Edited 16-Feb-94 12:35 by nilsson
                                                                      ; scrolls a bitmap under a window.
    (PROG ((DSP (WINDOWPROP WINDOW 'DSP))
           REGION)
          (COND
             ((NOT (type? BITMAP BITMAP))
              (RETURN)))
          (SETQ REGION (DSPCLIPPINGREGION NIL DSP))                   ; determine the change in offsets caused by the scroll.
          (WYOFFSET (IMAX (IMIN (fetch (REGION BOTTOM) of REGION)
                                YDELTA)
                          (IDIFFERENCE (fetch (REGION HEIGHT) of REGION)
                                 (fetch (BITMAP BITMAPHEIGHT) of BITMAP)))
                 DSP)
          (WXOFFSET (IMAX (IMIN (fetch (REGION LEFT) of REGION)
                                XDELTA)
                          (IDIFFERENCE (fetch (REGION WIDTH) of REGION)
                                 (fetch (BITMAP BITMAPWIDTH) of BITMAP)))
                 DSP)                                                 ; stuff new image over old
          (BITBLT BITMAP 0 0 DSP])

(REDISPLAYBITMAP
  [LAMBDA (BITMAP WINDOW)                                             ; Edited 16-Feb-94 12:35 by nilsson

    ;; blts a bitmap into a window so that the lower left corner of the bitmap is in the lower left corner of the window.

    (OR (type? BITMAP BITMAP)
        (ERROR "ILLEGAL ARG" BITMAP))
    (PROG ((DSP (WINDOWPROP WINDOW 'DSP))
           WREGION)
          (SETQ WREGION (DSPCLIPPINGREGION NIL DSP))
          (RETURN (BITBLT BITMAP 0 0 DSP (fetch (REGION LEFT) of WREGION)
                         (fetch (REGION BOTTOM) of WREGION])

(ULREDISPLAYBITMAP
  [LAMBDA (BITMAP WNEW)                                               ; Edited 16-Feb-94 12:35 by nilsson

    ;; blts a bitmap into a window so that the upper left corner of the bitmap is in the upper left corner of the window.

    (OR (type? BITMAP BITMAP)
        (ERROR "ILLEGAL ARG" BITMAP))
    (PROG ((DSP (WINDOWPROP WNEW 'DSP))
           REGION)
          (SETQ REGION (DSPCLIPPINGREGION NIL DSP))
          (RETURN (BITBLT BITMAP 0 0 DSP (fetch (REGION LEFT) of REGION)
                         (IDIFFERENCE (IPLUS (fetch (REGION BOTTOM) of REGION)
                                             (fetch (REGION HEIGHT) of REGION))
                                (fetch BITMAPHEIGHT of BITMAP])

(EXTENDEXTENT
  [LAMBDA (WINDOW INCLUDEREGION)                                      ; Edited 16-Feb-94 12:35 by nilsson
                                                                      ; destructively changes the EXTENT region of a WINDOW to
                                                                      ; include INCLUDEREGION
    (PROG [(EXTENT (WINDOWPROP WINDOW 'EXTENT]
          (RETURN (COND
                     (EXTENT (EXTENDREGION EXTENT INCLUDEREGION))
                     (T (WINDOWPROP WINDOW 'EXTENT (create REGION using INCLUDEREGION])

(WIDTHIFWINDOW
  [LAMBDA (INTERIORWIDTH BORDER)                                      ; Edited 16-Feb-94 12:35 by nilsson
```

```
                                                            ; returns the exterior width of a window with interior dimension
                                                            ; INTERIORWIDTH
    (IPLUS INTERIORWIDTH (ITIMES 2 (OR BORDER WBorder])

(HEIGHTIFWINDOW
  [LAMBDA (INTERIORHEIGHT TITLEFLG BORDER SCREEN)           ; Edited 16-Feb-94 12:36 by nilsson
                                                            ; returns the exterior height of a window which has interior height
                                                            ; dimension INTERIORHEIGHT
    (SETQ SCREEN (\INSURESCREEN SCREEN))
    (IPLUS INTERIORHEIGHT (COND
                             [TITLEFLG (IMINUS (DSPLINEFEED NIL (fetch (SCREEN SCTITLEDS) of SCREEN]
                             (T 0))
           (ITIMES 2 (OR BORDER WBorder])
)

;; this function should be on LLDISPLAY but Ron has it checked out. Move it later - rrb.

(DEFINEQ

(\DSPUNTRANSFORMREGION
  [LAMBDA (REGION DISPLAYDATA)                              (* rmk%: "30-AUG-83 13:19")
                                                            (* translates a region from destination coordinates into display
                                                            stream coordinates.)
    (CREATEREGION (\DSPUNTRANSFORMX (fetch (REGION LEFT) of REGION)
                         DISPLAYDATA)
           (\DSPUNTRANSFORMY (fetch (REGION BOTTOM) of REGION)
                  DISPLAYDATA)
           (fetch (REGION WIDTH) of REGION)
           (fetch (REGION HEIGHT) of REGION])
)

(RPAQ VertScrollCursor (CURSORCREATE '

                            'NIL 7 15))

(RPAQ ScrollUpCursor (CURSORCREATE '

                            'NIL 7 15))

(RPAQ ScrollDownCursor (CURSORCREATE '

                            'NIL 7 15))

(RPAQ HorizScrollCursor (CURSORCREATE '

                            'NIL 7 5))

(RPAQ ScrollLeftCursor (CURSORCREATE '

                            'NIL 8 5))

(RPAQ ScrollRightCursor (CURSORCREATE '

                            'NIL 7 5))

(RPAQ VertThumbCursor (CURSORCREATE '

                            'NIL 6 8))

(RPAQ HorizThumbCursor (CURSORCREATE '

                            'NIL 6 6))

(RPAQ WAITINGCURSOR (CURSORCREATE '

                            'NIL 7 8))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \LastInWindow VertScrollCursor ScrollUpCursor ScrollDownCursor ScrollLeftCursor ScrollRightCursor
       HorizScrollCursor)
)

(RPAQ? SCROLLBARWIDTH 24)

(RPAQ? SCROLLWAITTIME 100)

(RPAQ? SCROLLBARSHADE 32800)

(RPAQ? WAITBEFORESCROLLTIME 750)
```

```
(RPAQ? WAITBETWEENSCROLLTIME 100)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(ADDTOVAR GLOBALVARS SCROLLBARWIDTH SCROLLWAITTIME SCROLLBARSHADE WAITBEFORESCROLLTIME WAITBETWEENSCROLLTIME
                            WAITINGCURSOR)
)

(PUTPROPS WINDOWSCROLL COPYRIGHT ("Venue & Xerox Corporation" 1986 1990 1993 1994 2021))
```

{MEDLEY}<sources>WINDOWSCROLL.;1 28-Jun-2024 18:34:03
-- Listed on 30-Jun-2024 13:16:27 --

FUNCTION INDEX

VARIABLE INDEX