File created: 2-Nov-2023 23:48:28 **{WMEDLEY}<lispusers>REGIONMANAGER.;133**

edit by: rmk

changes to: (FNS RM-CREATEW)

previous date: 10-Oct-2023 22:19:05 {WMEDLEY}<lispusers>REGIONMANAGER.;129

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ REGIONMANAGERCOMS
       [;; Typed regions
        [COMS (FNS SET-TYPED-REGIONS GRAB-TYPED-REGION REGISTER-TYPED-REGION REGION-TYPE)
              (FNS RM-CREATEW RM-CLOSEW RM-GETREGION CLOSE-TYPED-W)
              (INITVARS (TYPED-REGIONS))
              (GLOBALVARS TYPED-REGIONS)
              (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS TYPED-REGION))
              (INITRECORDS TYPED-REGION)
              (P (MOVD? 'CREATEW 'CREATEW.ORIG)
                 (MOVD? 'CLOSEW 'CLOSEW.ORIG)
                 (MOVD? 'GETREGION 'GETREGION.ORIG)
                 (MOVD 'RM-CREATEW 'CREATEW)
                 (MOVD 'RM-CLOSEW 'CLOSEW)
                 (MOVD 'RM-GETREGION 'GETREGION]
        ;; Relative regions
        (COMS (FNS RELCREATEREGION RELGETREGION RELCREATEPOSITION)
              (FNS \RELCREATEREGION.REF \RELCREATEREGION.SIZE))
        ;; Composite application construction
        (COMS (FNS RM-ATTACHWINDOW)
              (FNS CLOSEWITH CLOSEWITH.DOIT MOVEWITH MOVEWITH.DOIT)
              (P (MOVD? 'ATTACHWINDOW 'ATTACHWINDOW.ORIG)
                 (MOVD 'RM-ATTACHWINDOW 'ATTACHWINDOW))
              (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS RFIELDDIFF])

;; Typed regions

(DEFINEQ

(SET-TYPED-REGIONS
  [LAMBDA (TYPELISTS REPLACE)                                   ; Edited  2-Jan-2022 16:01 by rmk
                                                                ; Edited 29-Dec-2021 16:17 by rmk
                                                                ; Edited 28-Dec-2021 12:59 by rmk
                                                                ; Edited 27-Nov-2021 08:55 by rmk:
                                                                ; Edited 26-Oct-2021 18:04 by rmk:
    ;; User can pre-initialize a sequence of regions for a given type.  Generally, TYPELISTS is a list of the form
    ;;     ((TYPEATOM1 . REGIONS)...(TYPEATOMn . REGIONS).  Copies of the regions of TYPELIST are added in front of any regions that might
    ;; already be present for that type.  The regions have haslinks to its type and an inuse status indicator.
    ;;
    ;; Convenience cases:
    ;;     TYPEATOM:  Interpreted as ((TYPEATOM)):  No region specified, but regions can accumulate
    ;;
    ;;     (TYPEATOM .REGIONS):  Interpreted as ((TYPEATOM . REGIONS).
    (if (LITATOM TYPELISTS)
        then (SETQ TYPELISTS (CONS (CONS TYPELISTS)))
      elseif (LITATOM (LISTP TYPELISTS))
        then (SETQ TYPELISTS (CONS TYPELISTS)))
    (for TL TYPE REGIONS PREV in TYPELISTS do (SETQ TYPE (CAR TL))
                                              (SETQ REGIONS (CDR TL))
                                              (CL:UNLESS (AND TYPE (LITATOM TYPE)
                                                              (for R in REGIONS always (REGIONP R)))
                                                     (ERROR "Not a TYPED-REGIONS specification" REGIONS))
                                              (SETQ REGIONS (COPY REGIONS))
                                                              ; Not to be confused with any other equal regions.
                                              (if (SETQ PREV (ASSOC TYPE TYPED-REGIONS))
                                                  then [RPLACD PREV (CL:IF REPLACE
                                                                           REGIONS
                                                                           (NCONC REGIONS (CDR PREV)))]
                                                elseif (push TYPED-REGIONS (CONS TYPE REGIONS])

(GRAB-TYPED-REGION
  [LAMBDA (REGION-TYPE MINWIDTH MINHEIGHT)                      ; Edited 10-Oct-2023 13:41 by rmk
                                                                ; Edited 14-Sep-2023 07:30 by rmk
    ;; Returns a REGIONTYPE region that satisfies MINWIDTH and MINHEIGHT, if specified
    (for R in (CDR (ASSOC REGION-TYPE TYPED-REGIONS)) unless (fetch REGION-INUSE of R)
```

```
        when [AND (OR (NULL MINWIDTH)
                      (ILEQ MINWIDTH (fetch WIDTH of R)))
                  (OR (NULL MINHEIGHT)
                      (ILEQ MINHEIGHT (fetch HEIGHT of R]
        do ;; We don't mark it as inuse here, leave that gets done by INSTALL-TYPED-REGION when ownership is given to a window. The only
           ;; downside is that the region could be reallocated before that happens, and 2 window would come up in the same place.
           (RETURN R])
```

(REGISTER-TYPED-REGION

```
  [LAMBDA (REGION REGION-TYPE WINDOW)                                   ; Edited 10-Oct-2023 13:30 by rmk
                                                                        ; Edited 29-Sep-2023 13:33 by rmk
                                                                        ; Edited 14-Sep-2023 10:03 by rmk
```

;; REGION was passed as the REGION argument to the original CREATEW. If that was NIL, CREATEW created its own region, but it didn't do it ;; through GETREGION (=RM.GETREGION) so it hasn't been registered according to the specified type. We set up the arrangements here.

```
    (CL:WHEN REGION-TYPE
        (CL:UNLESS REGION
            (SETQ REGION (WINDOWREGION WINDOW)))
        (LET [(TREGIONLIST (OR (ASSOC REGION-TYPE TYPED-REGIONS)
                               (CAR (PUSH TYPED-REGIONS (CONS REGION-TYPE]
             (CL:UNLESS (MEMB REGION (CDR TREGIONLIST))
                    (NCONC1 TREGIONLIST REGION))
             (replace REGION-INUSE of REGION with T)
             ;; We keep the original separate from the window's region WINDOWPROP so that RM-CLOSEW can update if the user reshapes.
             (WINDOWPROP WINDOW 'TYPED-REGION (CONS REGION-TYPE REGION))
             REGION)])
```

(REGION-TYPE

```
  [LAMBDA (X TYPE)                                                      ; Edited 10-Oct-2023 14:30 by rmk
                                                                        ; Edited 16-Sep-2023 08:41 by rmk
```

;; Value is the type of X if it is a region of type TYPE or a region of any type if TYPE is NIL.

```
    (CL:WHEN (REGIONP X)
        [if TYPE
            then (CL:WHEN (MEMB X (CDR (ASSOC TYPE TYPED-REGIONS)))
                          TYPE)
          else (CAR (find TYPELIST in TYPED-REGIONS suchthat (MEMB X TYPELIST])])

)

(DEFINEQ
```

(RM-CREATEW

```
  [LAMBDA (REGION TITLE BORDERSIZE NOOPENFLG PROPS)                     ; Edited  2-Nov-2023 23:48 by rmk
                                                                        ; Edited 24-Sep-2023 20:38 by rmk
                                                                        ; Edited 14-Sep-2023 22:23 by rmk
                                                                        ; Edited  1-Jan-2022 23:12 by rmk
                                                                        ; Edited 29-Dec-2021 19:25 by rmk
```

;; Generic CREATEW function for managed regions. If REGION-TYPE is specified (as REGION or in PROPS), then we try to find a previous ;; region for that type that is currently unused, create one if needed.

;; We have to bracket the original window creation because the we have to mark that the window uses that region, to put it back in the pool when ;; the window is closed.

;; NOTE: putting the region as the REGION--TYPE property may only be needed for old TEDIT compatibility

```
    (LET [WINDOW REGION-TYPE (RTPROP (LISTGET PROPS 'REGION-TYPE]
         (SETQ REGION-TYPE (if (AND (LITATOM REGION)
                                    REGION)
                               then (PROG1 REGION (SETQ REGION NIL))
                              elseif (LITATOM RTPROP)
                               then RTPROP))
         (CL:UNLESS (OR (REGIONP REGION)
                        (SCREENREGIONP REGION))
             (SETQ REGION (OR (REGIONP RTPROP)
                              (SCREENREGIONP RTPROP))))
         ;; We have REGION-TYPE, but maybe also a region that already has a source. Maybe we should make sure that the source is of that type?
         ;; REGION can also be a screenregion, that falls through.
         (CL:WHEN (AND REGION-TYPE (NULL REGION))
             (SETQ REGION (GRAB-TYPED-REGION REGION-TYPE)))
         (SETQ WINDOW (CREATEW.ORIG REGION TITLE BORDERSIZE NOOPENFLG PROPS))
         (CL:WHEN REGION-TYPE (REGISTER-TYPED-REGION REGION REGION-TYPE WINDOW))
         WINDOW])
```

(RM-CLOSEW

```
  [LAMBDA (WINDOW)                                                      ; Edited 10-Oct-2023 22:11 by rmk
```

;; Makes the window's typed region available for reuse, if the window is marked with a TYPEDREGION.

;; It's possible that the window exists and can be reopened after it has been closed. The glitch in that case is that we may have decided to make ;; the window's region available to another window, and if this window is opened again it will come on top of that other one (if it hasn't moved). Oh ;; well.

;; This replaces the particular typed-region in TYPED-REGIONS with the region that the window ended up with, perhaps after the user reshaped it.

```
    ;; But (WINDOWPROP WINDOW 'REGION) doesn't include the prompt window, if it's there, and (WINDOWREGION WINDOW) would union in all
    ;; of the attached windows (menus etc.)  This code assumes that the promptwindow was taken out of the original region (lots of funky code does
    ;; that), so it unions it back in to the REGION property to reconstruct the original typed-region.  The alternative would be to have the windows region
    ;; copy the original grabbed region and restore only that.  But then we would be ignoring any reshaping adjustments.
    (LET* [CLOSEVAL (TYPEDREGION (WINDOWPROP WINDOW 'TYPED-REGION))
                (REGIONTYPE (CAR TYPEDREGION))
                (TREGION (CDR TYPEDREGION))
                [PWINDOW (WINDOWP (CAR (MKLIST (WINDOWPROP WINDOW 'PROMPTWINDOW]
                [WREGION (CL:IF PWINDOW
                             (UNIONREGIONS (WINDOWPROP WINDOW 'REGION)
                                    (WINDOWPROP PWINDOW 'REGION))
                             (WINDOWPROP WINDOW 'REGION))]
                (TREGIONLIST (AND REGIONTYPE (OR (ASSOC REGIONTYPE TYPED-REGIONS)
                                                 (CAR (PUSH TYPED-REGIONS (CONS REGIONTYPE]
          (CL:WHEN (AND (SETQ CLOSEVAL (CLOSEW.ORIG WINDOW))
                        TYPEDREGION)
              (CL:UNLESS (EQUAL TREGION WREGION)

                  ;; The user reshaped the window after the region was taken from TYPED-REGIONS.  Assume that the new shape is what
                  ;; should be offered when this is recycled. Important to keep the same structure

                  (with REGION TREGION (SETQ LEFT (fetch (REGION LEFT) of WREGION))
                        (SETQ BOTTOM (fetch (REGION BOTTOM) of WREGION))
                        (SETQ WIDTH (fetch (REGION WIDTH) of WREGION))
                        (SETQ HEIGHT (fetch (REGION HEIGHT) of WREGION))))

                ;; Move TREGION to the front so most recently closed will be recycled first

                (CL:WHEN TREGIONLIST
                    (change (CDR TREGIONLIST)
                            (CONS TREGION (DREMOVE TREGION DATUM))))
                (replace REGION-INUSE of TREGION with NIL)
                (WINDOWPROP WINDOW 'TYPED-REGION NIL))
          CLOSEVAL])
```

(RM-GETREGION

```
  [LAMBDA (MINWIDTH MINHEIGHT INITREGION NEWREGIONFN NEWREGIONFNARG INITCORNERS)
                                                                    ; Edited 10-Oct-2023 12:39 by rmk
                                                                    ; Edited 14-Sep-2023 07:50 by rmk
                                                                    ; Edited  1-Jan-2022 21:49 by rmk
    ;; If INITREGION is a type atom:

    ;;   If a region of that type is available, then a (copy) is returned.

    ;;   Otherwise, the user is asked for a new region, that is added to the type list, and again a copy is returned.

    ;; We return a copy because we don't know what will happen to this region, whether it will be changed by future operations (e.g. by a constellation
    ;; operation).  A future retrieval will return the original size and position, and it will then presumably be shrunk in the same way.

    ;;  If INITREGION is not a typeatom, it is passed through to the original GETREGION, and the new region will not be managed.

    (LET (REGION TYPELIST (REGION-TYPE (AND (LITATOM INITREGION)
                                            INITREGION)))
         (SETQ REGION (GRAB-TYPED-REGION REGION-TYPE MINWIDTH MINHEIGHT))
         (CL:UNLESS REGION

             ;; If we found a good one, INITREGIONS must have been a type, and we're done.  Otherwise, run the normal code, but save the new
             ;; region as a new instance if its typed.

             (SETQ REGION (GETREGION.ORIG MINWIDTH MINHEIGHT (CL:IF REGION-TYPE
                                                                    NIL
                                                                    INITREGION)
                                  NEWREGIONFN NEWREGIONFNARG INITCORNERS))
             (CL:WHEN REGION-TYPE

                 ;; A new typed region to add to the list .

                 (NCONC1 [OR (ASSOC REGION-TYPE TYPED-REGIONS)
                             (CAR (PUSH TYPED-REGIONS (CONS REGION-TYPE]
                        REGION)))
       REGION])
```

(CLOSE-TYPED-W

```
  [LAMBDA (TYPE)                                                    ; Edited 14-Sep-2023 07:39 by rmk
                                                                    ; Edited 29-Dec-2021 15:58 by rmk
                                                                    ; Edited 27-Nov-2021 11:50 by rmk:
    ;; Closes all windows whose regions are of type TYPE

    (CL:WHEN TYPE
        (for W R in (OPENWINDOWS) eachtime [SETQ WT (CAR (WINDOWPROP W 'TYPED-REGION]
           when (AND WT (EQMEMB WT TYPE)) do (CLOSEW W)))])

)

(RPAQ? TYPED-REGIONS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TYPED-REGIONS)
)
```

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(HASHLINK TYPED-REGION (REGION-INUSE REGION-INUSE-HASH))
)

(DECLARE%: EVAL@COMPILE

(GLOBALVARS REGION-INUSE-HASH)
)

(SETUPHASHARRAY 'REGION-INUSE-HASH NIL)
)

(DECLARE%: EVAL@COMPILE

(GLOBALVARS REGION-INUSE-HASH)
)

(SETUPHASHARRAY 'REGION-INUSE-HASH NIL)

(MOVD? 'CREATEW 'CREATEW.ORIG)

(MOVD? 'CLOSEW 'CLOSEW.ORIG)

(MOVD? 'GETREGION 'GETREGION.ORIG)

(MOVD 'RM-CREATEW 'CREATEW)

(MOVD 'RM-CLOSEW 'CLOSEW)

(MOVD 'RM-GETREGION 'GETREGION)
```

;; Relative regions

```
(DEFINEQ
```

(**RELCREATEREGION**

```
  [LAMBDA (WIDTH HEIGHT CORNERX CORNERY REFX REFY ONSCREEN)
```

; Edited 27-Jan-2022 13:23 by rmk
; Edited 25-Jan-2022 15:29 by rmk
; Edited 23-Jan-2022 21:18 by rmk
; Edited 12-Jan-2022 17:50 by rmk
; Edited 30-Dec-2021 20:54 by rmk
; Edited 27-Dec-2021 15:54 by rmk

;; The region is oriented so that he REFX and REFY are at the corner named by CORNERX/Y.

;; Creates a WIDTH-HEIGHT region relative to the CORNER and REF parameters.

;; CORNERX and CORNERY default to LEFT and BOTTOM.

;; REFX and REFY default to the current cursor screen coordinates. Otherwise,

;; REFX is a position and REFY is NIL: REFX and REFY are extracted from the position

;; Positive integers: absolute screen coordinates

;; (region spec) or (window spec) pairs: coordinates relative to the region or the window's region

;; Spec can name the X/Y endpoints (e.g. LEFT/0 or RIGHT/1) or a floating point proportion of the distance on the relevant dimension (e.g. .5= the midpoint.

;; If ONSCREEN, the width or height is adjusted so that the corner opposite to the fixed corner is always visible.

;;

;; The arguments can be given as a list to be spread out, so that region relative region specifications can be passed through intermediate functions.
;; The test here is not very tight, if it is incorrect the recursive call will fail.

```
    (IF (AND (LISTP WIDTH)
             (NOT (REGIONP WIDTH))
             (NULL HEIGHT)
             (IGREATERP (LENGTH WIDTH)
                    3))
```

THEN ;; If less than 3, presumably a relative width

```
        (APPLY (FUNCTION RELCREATEREGION)
               WIDTH)
```

ELSE ;; Resolve the width and height, if based on a region or window

```
        (SETQ WIDTH (\RELCREATEREGION.SIZE WIDTH 'X))
        (SETQ HEIGHT (\RELCREATEREGION.SIZE HEIGHT 'Y))
```

;; Resolve the corner

```
        (CL:UNLESS CORNERX
            (SETQ CORNERX 'LEFT))
        (CL:UNLESS CORNERY
            (SETQ CORNERY 'BOTTOM))
        (CL:WHEN (AND (LISTP CORNERX)
                      (NULL CORNERY))
            (SETQ CORNERY (CADR CORNERX))
            (SETQ CORNERX (CAR CORNERX)))
```

```
    ;; Resolve the reference point
    [IF (AND (POSITIONP REFX)
             (NULL REFY))
        THEN (SETQ REFY (FETCH (POSITION YCOORD) OF REFX))
             (SETQ REFX (FETCH (POSITION XCOORD) OF REFX))
      ELSE (GETMOUSESTATE)
           (SETQ REFX (\RELCREATEREGION.REF REFX 'X))
           (SETQ REFY (\RELCREATEREGION.REF REFY 'Y]
    ;; Align the new-region corner with the reference point
    (LET* ((LEFT REFX)
           (BOTTOM REFY)
           (RIGHT (IPLUS LEFT WIDTH))
           (TOP (IPLUS BOTTOM HEIGHT)))
          (CL:WHEN (EQ 'RIGHT CORNERX)
              (SETQ RIGHT LEFT)
              (SETQ LEFT (IDIFFERENCE LEFT WIDTH)))
          (CL:WHEN (EQ 'TOP CORNERY)
              (SETQ TOP BOTTOM)
              (SETQ BOTTOM (IDIFFERENCE BOTTOM HEIGHT)))
          (CL:WHEN ONSCREEN                                    ; Keep the region on the screen.  Not clear whether we should
                                                               ; keep the width and height and just move the left and bottom.
                                                               ; Here we allow some shrinkage
              (CL:WHEN (ILESSP LEFT 0)
                  (ADD WIDTH (IMIN 100 LEFT))
                  (SETQ LEFT 0))
              (CL:WHEN (ILESSP BOTTOM 0)
                  (ADD HEIGHT (IMIN 100 BOTTOM))
                  (SETQ BOTTOM 0))
              (CL:WHEN (IGREATERP RIGHT SCREENWIDTH)
                  (ADD WIDTH (IDIFFERENCE SCREENWIDTH RIGHT)))
              (CL:WHEN (IGREATERP TOP SCREENHEIGHT)
                  (ADD HEIGHT (IDIFFERENCE SCREENHEIGHT TOP))))
          (CREATEREGION LEFT BOTTOM WIDTH HEIGHT])
```

(RELGETREGION

```
  [LAMBDA (WIDTH HEIGHT CORNERX CORNERY REFX REFY MINSIZE)      ; Edited 27-Jan-2022 13:24 by rmk
                                                               ; Edited 25-Jan-2022 15:30 by rmk
                                                               ; Edited 23-Jan-2022 21:20 by rmk
                                                               ; Edited 28-Dec-2021 23:13 by rmk
                                                               ; Edited 10-Dec-2021 10:15 by rmk
    ;; Prompts for a relative region as created by RELCREATEREGION. Initially the anchored corner is fixed and the cursor is moved to the diagonally
    ;; opposite corner.  If MINSIZE, the WIDTH and HEIGHT are taken to be the minimums that are acceptable, modulo the fact that the opposite
    ;; corner is guaranteed to be visibleand, the size of the ghost region can only grow.  If not MINSIZE, we also allow the user to shrink the ghost
    ;; region.
    (COND
       ((AND (LISTP WIDTH)
             (NOT (REGIONP WIDTH))
             (NULL HEIGHT)
             (IGREATERP (LENGTH WIDTH)
                    3))
        (APPLY (FUNCTION RELGETREGION)
               WIDTH))
       (T (CL:WHEN (AND (LISTP CORNERX)
                        (NULL CORNERY))
              (SETQ CORNERY (CADR CORNERX))
              (SETQ CORNERX (CAR CORNERX)))
          (CL:UNLESS CORNERX
              (SETQ CORNERX 'LEFT))
          (CL:UNLESS CORNERY
              (SETQ CORNERY 'BOTTOM))
          (LET* ((REGION (OR (REGIONP WIDTH)
                             (RELCREATEREGION WIDTH HEIGHT CORNERX CORNERY REFX REFY T)))
                 (BASEX (FETCH (REGION LEFT) OF REGION))
                 (BASEY (FETCH (REGION BOTTOM) OF REGION))
                 (RWIDTH (FETCH (REGION WIDTH) OF REGION))
                 (RHEIGHT (FETCH (REGION HEIGHT) OF REGION))
                 (OPPX (IPLUS BASEX RWIDTH))
                 (OPPY (IPLUS BASEY RHEIGHT)))
                ;; Default parameters assume the anchor is  (LEFT BOTTOM)
                (CL:WHEN (EQ 'RIGHT CORNERX)
                       (SWAP BASEX OPPX))
                (CL:WHEN (EQ 'TOP CORNERY)
                       (SWAP BASEY OPPY))
                (\CURSORPOSITION OPPX OPPY)
                (CL:UNLESS MINSIZE                             ; No minimum size constraint
                    (SETQ RWIDTH NIL)
                    (SETQ RHEIGHT NIL))
                (GETREGION RWIDTH RHEIGHT REGION NIL NIL (LIST BASEX BASEY OPPX OPPY])
```

(RELCREATEPOSITION

```
  [LAMBDA (REFX REFY)                                          ; Edited 23-Jan-2022 17:08 by rmk
```

```
    (CREATEPOSITION (\RELCREATEREGION.REF REFX 'X)
           (\RELCREATEREGION.REF REFY 'Y])

)

(DEFINEQ
```

(\RELCREATEREGION.REF

```
  [LAMBDA (REF WHICH)                                        ; Edited 27-Feb-2022 08:43 by rmk
                                                             ; Edited 23-Jan-2022 20:20 by rmk
                                                             ; Edited  2-Jan-2022 11:01 by rmk
```

;; REF can be NIL, an absolute screen position, the atom SCREEN, or a list of (anchor fraction adjustment) where anchor can be a region, window, ;; or the atom SCREEN, fraction can be a number or atoms LEFT/RIGHT/BOTTOM/TOP as apropriate.

```
                                                             ; Edited 30-Dec-2021 17:49 by rmk
    (LET (ANCHOR VAL SIZE FRACTION SPEC (BASE 0))
```

;; Would be nice if the screen had a region

```
         (IF (NULL REF)
             THEN (CL:IF (EQ WHICH 'X)
                         LASTMOUSEX
                         LASTMOUSEY)
           ELSEIF (AND (FIXP REF)
                       (NOT (MINUSP REF)))
             THEN REF
           ELSEIF (EQ REF 'SCREEN)
             THEN
```

;; LEFT and BOTTOM are 0

```
                  0
           ELSEIF (EQ REF 'TTY)
             THEN (CL:IF (EQ WHICH 'X)
                         (FETCH (REGION LEFT) OF (WINDOWPROP (WFROMDS T)
                                                             'REGION))
                         (FETCH (REGION BOTTOM) OF (WINDOWPROP (WFROMDS T)
                                                               'REGION)))
           ELSEIF [AND (LISTP REF)
                       (SETQ ANCHOR (OR (REGIONP (CAR REF))
                                        (AND (WINDOWP (CAR REF))
                                             (WINDOWPROP (CAR REF)
                                                         'REGION))
                                        (AND (EQ (CAR REF)
                                                 'SCREEN)
                                             'SCREEN)
                                        (AND (EQ (CAR REF)
                                                 'TTY)
                                             (WINDOWPROP (WFROMDS T)
                                                         'REGION]
             THEN (SETQ SPEC (CDR REF))
                  [IF (EQ WHICH 'X)
                      THEN (IF (EQ ANCHOR 'SCREEN)
                               THEN (SETQ SIZE SCREENWIDTH)
                             ELSE (SETQ BASE (FETCH (REGION LEFT) OF ANCHOR))
                                  (SETQ SIZE (FETCH (REGION WIDTH) OF ANCHOR)))
                           (SETQ FRACTION (SELECTQ (CAR SPEC)
                                                   ((NIL LEFT)
                                                        0)
                                                   (RIGHT 1)
                                                   (CAR SPEC)))
                    ELSE (IF (EQ ANCHOR 'SCREEN)
                             THEN (SETQ SIZE SCREENHEIGHT)
                           ELSE (SETQ BASE (FETCH (REGION BOTTOM) OF ANCHOR))
                                (SETQ SIZE (FETCH (REGION HEIGHT) OF ANCHOR)))
                         (SETQ FRACTION (SELECTQ (CAR SPEC)
                                                 ((NIL BOTTOM)
                                                      0)
                                                 (TOP 1)
                                                 (CAR SPEC]
                  [SETQ VAL (IPLUS BASE (ROUND (TIMES FRACTION SIZE]
                  (CL:WHEN (CADR SPEC)
                      (ADD VAL (CADR SPEC)))
                  VAL
           ELSE (\ILLEGAL.ARG REF])
```

(\RELCREATEREGION.SIZE

```
  [LAMBDA (PARAM WHICH)                                      ; Edited 27-Feb-2022 08:40 by rmk
                                                             ; Edited  2-Jan-2022 11:00 by rmk
                                                             ; Edited 30-Dec-2021 17:51 by rmk
```

;; PARAM can be FIXP or (region anchor adjustment) which determine size relative to the region.

```
    (LET (VAL ANCHOR SPEC)
         (IF (FIXP PARAM)
           ELSEIF [SETQ ANCHOR (OR (REGIONP PARAM)
                                   (AND (WINDOWP PARAM)
                                        (WINDOWREGION PARAM]
             THEN (CL:IF (EQ WHICH 'X)
                         (FETCH WIDTH OF ANCHOR)
                         (FETCH HEIGHT OF ANCHOR))
```

```
          ELSEIF (LISTP PARAM)
            THEN (IF (SETQ ANCHOR (OR (REGIONP (CAR PARAM))
                                      (AND (WINDOWP (CAR PARAM))
                                           (WINDOWPROP (CAR PARAM)
                                                  'REGION))
                                      (AND (EQ (CAR PARAM)
                                               'SCREEN)
                                           'SCREEN)
                                      (AND (EQ (CAR PARAM)
                                               'TTY)
                                           (WINDOWPROP (WFROMDS T)
                                                  'REGION))
                                      (\ILLEGAL.ARG PARAM)))
                     THEN [SETQ VAL (CL:IF (EQ WHICH 'X)
                                        (CL:IF (EQ ANCHOR 'SCREEN)
                                            SCREENWIDTH
                                            (FETCH WIDTH OF ANCHOR))
                                        (CL:IF (EQ ANCHOR 'SCREEN)
                                            SCREENHEIGHT
                                            (FETCH HEIGHT OF ANCHOR)))]
                          (SETQ SPEC (CDR PARAM))
                          (CL:WHEN (CAR SPEC)
                              (SETQ VAL (ROUND (TIMES (CAR SPEC)
                                                      VAL))))
                          (CL:WHEN (CADR SPEC)
                              (ADD VAL (CADR SPEC)))
                          VAL)
          ELSEIF (EQ PARAM 'SCREEN)
            THEN (CL:IF (EQ WHICH 'X)
                        SCREENWIDTH
                        SCREENHEIGHT)
          ELSEIF (EQ PARAM 'TTY)
            THEN (CL:IF (EQ WHICH 'X)
                        (FETCH (REGION WIDTH) OF (WINDOWREGION (WFROMDS T)
                                                               'REGION))
                        (FETCH (REGION HEIGHT) OF (WINDOWREGION (WFROMDS T)
                                                                'REGION)))
          ELSE (\ILLEGAL.ARG PARAM])

)

;; Composite application construction

(DEFINEQ

(RM-ATTACHWINDOW
  [LAMBDA (WINDOWTOATTACH MAINWINDOW EDGE POSITIONONEDGE WINDOWCOMACTION TAKEFROMCENTRAL)
                                                                ; Edited 29-Dec-2021 09:36 by rmk
                                                                ; Edited 28-Nov-2021 16:10 by rmk:
    ;; MAINWINDOW may not be the central window, could be attached to an attachment.

    ;; If the central window is under construction, we shrink it down so that the new attachment fits within the original footprint of the central window and
    ;; all of its previous attachments.

    ;; This addresses the common situation where the user provides a region for the central window and the constellation of windows that will surround
    ;; it, and the whole constellation is supposed to stay within that original bounding box, even as new attachments (promptwindows, menus...) are
    ;; tacked on.

    ;;

    ;; A second extension:  If WINDOWCOMACTION is a list, smash it into the PASSTOMAINCOMS.  ATTACHWINDOW.ORIG only allows a few
    ;; atomic-value options.

    (LET (MIN (CENTRALWINDOW (CENTRALWINDOW MAINWINDOW))
              CENTRALREGION NEWALLREGION ORIGALLREGION NEWCENTRALREGION VAL)
         (CL:WHEN (OR TAKEFROMCENTRAL (WINDOWPROP CENTRALWINDOW 'UNDERCONSTRUCTION))
             (SETQ ORIGALLREGION (ATTACHEDWINDOWREGION CENTRALWINDOW))
             (SETQ CENTRALREGION (WINDOWREGION CENTRALWINDOW)))
         (SETQ VAL (ATTACHWINDOW.ORIG WINDOWTOATTACH MAINWINDOW EDGE POSITIONONEDGE WINDOWCOMACTION))
         (CL:WHEN ORIGALLREGION
             (SETQ NEWALLREGION (ATTACHEDWINDOWREGION CENTRALWINDOW))
             (CL:UNLESS (EQUAL ORIGALLREGION NEWALLREGION)
                 ;; Something changed, presumably the total region expanded, so something has to shrink to stay within the original region.  We
                 ;; want to shrink the main window only, keeping everything else as it was.  Hopefully, previously attached windows that wanted a
                 ;; fixed size on the relevant dimension have a MINSIZE that won't let them shrink.  And hopefully the central window does allow
                 ;; shrinking, otherwise nothing happens.

                 ;; It also could be that the region hasn't changed, if the new window hides in the shadow of a previously attached one.

                 (SETQ NEWCENTRALREGION (SELECTQ EDGE
                                            (LEFT (CREATE REGION USING CENTRALREGION LEFT _
                                                      (PLUS (FETCH (REGION LEFT) OF
                                                                   CENTRALREGION
                                                            )
                                                            (RFIELDDIFF LEFT ORIGALLREGION
                                                                   NEWALLREGION))
                                                      WIDTH _ (DIFFERENCE (FETCH (REGION
                                                                                     WIDTH)
                                                                                 OF CENTRALREGION
```

```
                                                                                    )
                                                                                (RFIELDDIFF WIDTH
                                                                                      NEWALLREGION
                                                                                      ORIGALLREGION))))
                                                                (RIGHT (CREATE REGION USING CENTRALREGION WIDTH _
                                                                                        (DIFFERENCE (FETCH (REGION WIDTH)
                                                                                                    OF CENTRALREGION)
                                                                                                (RFIELDDIFF WIDTH NEWALLREGION
                                                                                                      ORIGALLREGION))))
                                                                (TOP (CREATE REGION USING CENTRALREGION HEIGHT _
                                                                                       (DIFFERENCE (FETCH (REGION HEIGHT)
                                                                                                   OF CENTRALREGION)
                                                                                               (RFIELDDIFF HEIGHT NEWALLREGION
                                                                                                     ORIGALLREGION))))
                                                                (BOTTOM (CREATE REGION USING CENTRALREGION BOTTOM _
                                                                                       (PLUS (FETCH (REGION BOTTOM)
                                                                                                  OF CENTRALREGION)
                                                                                             (RFIELDDIFF BOTTOM ORIGALLREGION
                                                                                                   NEWALLREGION))
                                                                                      HEIGHT _ (DIFFERENCE
                                                                                                 (FETCH (REGION HEIGHT)
                                                                                                    OF CENTRALREGION)
                                                                                                 (RFIELDDIFF HEIGHT
                                                                                                       NEWALLREGION
                                                                                                       ORIGALLREGION))))
                                                                (SHOULDNT)))
                                ;; We want to reshape only the central window. We detach the new (just attached) window, do the shrinking, then reattach. If
                                ;; other attached windows get reshaped, that's par for the course.  Presumably they are specified as fixed on the relevant
                                ;; dimension, or the user doesn't care.

                                ;; Maybe this little wrinkle is solving a non-problem--if the user cares about whether or not the new window will shrink, now or with
                                ;; later reshaping, then he should have specified its own minsize property.

                                ;; On the otherhand, maybe we should remove all of the SHAPEW's (or but in DONT)  in the PASSTOMAIN coms of all the
                                ;; windows attached directly or indirectly to the central window, do the reshaping, and then restore.

                                (DETACHWINDOW WINDOWTOATTACH MAINWINDOW)
                                (SHAPEW CENTRALWINDOW NEWCENTRALREGION)

                                ;; Now reattach the new window

                                (SETQ VAL (ATTACHWINDOW.ORIG WINDOWTOATTACH MAINWINDOW EDGE POSITIONONEDGE WINDOWCOMACTION))

                                ;; This is a little error check for debugging, to catch cases where there might be interactions with other interfering strategies.  If the
                                ;; new window turned out to be bigger on the relevant dimension than the original set up, then we simply have to relax.

                                ;; If the new window is bigger than the original region on the other dimenion dimension, then we have to relax our requirement.
                                ;; We use ATTACHEDWINDOWREGION in case the new window is already a conglomerate.

                                (CL:UNLESS (OR (EQUAL ORIGALLREGION (ATTACHEDWINDOWREGION CENTRALWINDOW))
                                               (SELECTQ EDGE
                                                    ((TOP BOTTOM)
                                                         (GEQ (FETCH (REGION WIDTH) OF (ATTACHEDWINDOWREGION WINDOWTOATTACH
                                                                                                             'REGION))
                                                              (FETCH (REGION WIDTH) OF ORIGALLREGION)))
                                                    ((LEFT RIGHT)
                                                         (GEQ (FETCH (REGION HEIGHT) OF (ATTACHEDWINDOWREGION WINDOWTOATTACH
                                                                                                              'REGION))
                                                              (FETCH (REGION HEIGHT) OF ORIGALLREGION)))
                                                    NIL))
                                           (HELP ORIGALLREGION (ATTACHEDWINDOWREGION MAINWINDOW)))
                                (CL:WHEN (LISTP WINDOWCOMACTION)

                                    ;; Maybe this should be done in the ORIG function--an oversight?

                                    (WINDOWPROP WINDOWTOATTACH 'PASSTOMAINCOMS WINDOWCOMACTION))))
                        VAL])

)

(DEFINEQ

(CLOSEWITH
  [LAMBDA (CHILDREN PARENT)                                  ; Edited 28-Jan-2022 23:51 by rmk
    [FOR C ONE INSIDE CHILDREN WHEN (AND C (SETQ C (WFROMDS C))) DO (SETQ ONE T)
                                                              (WINDOWADDPROP PARENT 'CLOSECHILDREN C)
       FINALLY (CL:WHEN ONE
                    (WINDOWADDPROP PARENT 'CLOSEFN (FUNCTION CLOSEWITH.DOIT)))]
    PARENT])

(CLOSEWITH.DOIT
  [LAMBDA (PARENT)                                           ; Edited 28-Jan-2022 17:54 by rmk
    (FOR C IN (WINDOWPROP PARENT 'CLOSECHILDREN) WHEN (OPENWP C) DO (CLOSEW C))
    (WINDOWPROP PARENT 'CLOSECHILDREN NIL)
    PARENT])

(MOVEWITH
  [LAMBDA (CHILDREN PARENT)                                  ; Edited 28-Jan-2022 23:43 by rmk
    [FOR C ONE INSIDE CHILDREN WHEN (AND C (SETQ C (WFROMDS C))) DO (SETQ ONE T)
                                                              (WINDOWADDPROP PARENT 'MOVECHILDREN C)
```

```
       FINALLY (CL:WHEN ONE
                    (WINDOWADDPROP PARENT 'MOVEFN (FUNCTION MOVEWITH.DOIT)))]
    PARENT])
```

(**MOVEWITH.DOIT**

```
  [LAMBDA (PARENT NEWPOS)                                    ; Edited 28-Jan-2022 22:34 by rmk
    [FOR C (DELTA _ (PTDIFFERENCE NEWPOS (WINDOWPOSITION PARENT))) IN (WINDOWPROP PARENT 'MOVECHILDREN)
       DO (MOVEW C (PTPLUS DELTA (WINDOWPOSITION C]
    PARENT])

)

(MOVD? 'ATTACHWINDOW 'ATTACHWINDOW.ORIG)

(MOVD 'RM-ATTACHWINDOW 'ATTACHWINDOW)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS RFIELDDIFF MACRO ((FIELD R1 R2)
                            (DIFFERENCE (FETCH (REGION FIELD) OF R1)
                                   (FETCH (REGION FIELD) OF R2))))
)
)
```

FUNCTION INDEX

MACRO INDEX

RECORD INDEX

VARIABLE INDEX