File created: 5-Dec-2020 16:35:48 **{DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-PLACEMENT-EDITOR.;2**

previous date: 17-Aug-90 13:25:31 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-PLACEMENT-EDITOR.;1

Read Table: XCL

Package: ROOMS

Format: XCCS


```
(IL:RPAQQ IL:ROOMS-PLACEMENT-EDITORCOMS
          ((FILE-ENVIRONMENTS IL:ROOMS-PLACEMENT-EDITOR)
           (IL:P (REQUIRE "ROOMS")
                 (EXPORT '(GET-PE PRINT-PEP-TITLE-STRING)
                         "ROOMS"))
           (IL:STRUCTURES PLACEMENT-EDITOR PE-PLACEMENT)
           (IL:VARIABLES *PLACEMENT-EDITORS* *PE-INHIBIT-REDISPLAY*)
           (IL:FUNCTIONS GET-PE MAKE-PE FIND-PE PE-WINDOW-P PE-CLOSEFN PE-REPAINTFN PE-RESHAPEFN PE-TOTOPFN
                  UPDATE-PE? UPDATE-PE UPDATE-PEPS UPDATE-PEP-SAVE-BITMAP UPDATE-PE-WINDOW PLACED-REGION)
           (IL:COMS                                                    ; display
                  (IL:VARIABLES *DEFAULT-ICON-SIZE* *PE-TITLE-FONT* *PEP-TITLE-FONT* *PEP-INCLUDED-SHADE*
                         *PE-BORDER*)
                  (IL:FUNCTIONS DISPLAY-PEPS SAVE-PEP-IMAGE DISPLAY-PE-TITLE DISPLAY-PE-BORDER DISPLAY-PEP
                         PRINT-PEP-TITLE PRINT-PEP-TITLE-STRING)
                  (IL:VARIABLES *DISPLAY-PE-DEPTH*)
                  (IL:WINDOW-TYPES :PLACEMENT-EDITOR))
           (IL:FUNCTIONS PE-RIGHTBUTTONFN                              ; mouse trackers
                  PE-BUTTONEVENTFN PE-BUTTONEVENTFN-INTERNAL PE-TRACK-MOUSE PE-GETMOUSESTATE
                  PE-TRACK-LEFT-BUTTON PE-TRACK-MIDDLE-BUTTON PEP-GETBOXPOSITION PEP-GETBOXPOSITION-INTERNAL)
           (IL:COMS                                                    ; editing
                  (IL:FUNCTIONS PE-ROOM-CHANGED-FN PEP-SELECTED EXPAND-PLACEMENT PEP-SELECTED-COPY-OR-MOVE
                         PE-ROOM-SELECTED)
                  (EVAL-WHEN (LOAD)
                         (IL:P (PUSHNEW 'PE-ROOM-CHANGED-FN *ROOM-CHANGED-FUNCTIONS*))))
           (IL:GLOBALVARS IL:TINYFONT IL:CROSSHAIRS)))


(DEFINE-FILE-ENVIRONMENT IL:ROOMS-PLACEMENT-EDITOR :COMPILER :COMPILE-FILE
   :PACKAGE "ROOMS"
   :READTABLE "XCL")

(REQUIRE "ROOMS")

(EXPORT '(GET-PE PRINT-PEP-TITLE-STRING)
       "ROOMS")


(DEFSTRUCT (PLACEMENT-EDITOR (:CONC-NAME "PE-")
                             (:CONSTRUCTOR MAKE-PE-INTERNAL)
                             (:PRINT-FUNCTION (LAMBDA (PE STREAM DEPTH)
                                                (FORMAT STREAM "#<Placement Editor for ~S>"
                                                       (PE-ROOM-NAME PE)))))
   (CHANGED? T :TYPE (MEMBER T NIL :PLACEMENTS))
   (ROOM-NAME NIL)
   (SCALE *ONE-TO-ONE* :TYPE SCALE)
   (PEPS NIL :TYPE LIST)
   (WINDOW NIL)
   (TITLE-TEXT NIL :TYPE TEXT)
   (CLIPPING-REGION NIL))


(DEFSTRUCT (PE-PLACEMENT (:CONC-NAME "PEP-")
                         (:PRINT-FUNCTION (LAMBDA (PEP STREAM DEPTH)
                                            (FORMAT STREAM "#<PEP ~O,~O>" (IL:\\HILOC PEP)
                                                   (IL:\\LOLOC PEP)))))
   (PLACEMENT NIL :TYPE PLACEMENT)
   (SCALED-REGION NIL :TYPE REGION)
   (IMMEDIATE? NIL :TYPE (MEMBER T NIL))
   (OPEN? NIL :TYPE (MEMBER T NIL))
   (SAVE-BITMAP NIL)
   (UNSCALED-REGION NIL :TYPE REGION))


(DEFGLOBALVAR *PLACEMENT-EDITORS* (MAKE-HASH-TABLE :TEST 'EQUAL))


(DEFVAR *PE-INHIBIT-REDISPLAY* NIL)


(DEFUN GET-PE (ROOM-NAME &OPTIONAL REGION)

;;; returns the PE for ROOM, creating one if required.  if REGION is provided then the PE will occupy it.

   (LET ((PE (FIND-PE ROOM-NAME)))
```

```
        (IF PE
            ;; this code optimized so there is never more than one redisplay when entering overview
            (LET* ((WINDOW (PE-WINDOW PE))
                   (ICON (WINDOW-ICON WINDOW))
                   SHAPED?)
                  ;; this gets smashed when window closed
                  (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR PE)
                  (LET ((*PE-INHIBIT-REDISPLAY* T))
                       (UN-HIDE-WINDOW WINDOW)
                       (IL:OPENW WINDOW)
                       (IL:TOTOPW WINDOW)
                       (WHEN REGION (SHAPE-WINDOW WINDOW REGION)))
                  (UPDATE-PE? PE)
                  PE)
            (MAKE-PE ROOM-NAME REGION))))
```

```
(DEFUN MAKE-PE (ROOM-NAME &OPTIONAL REGION)
```

;;; don't call this. call GET-PE instead. we depend on there only being on PE per room.

```
   (LET* ((WINDOW (IL:CREATEW REGION NIL 0))
          (PE (MAKE-PE-INTERNAL :ROOM-NAME ROOM-NAME :WINDOW WINDOW :TITLE-TEXT (MAKE-TEXT :STRING
                                                                                           (IF (STRINGP ROOM-NAME)
                                                                                               ROOM-NAME
                                                                                               (PRINC-TO-STRING
                                                                                                ROOM-NAME))
                                                                                           :SHADOWS T :FONT
                                                                                           *PE-TITLE-FONT*))))
          (IL:WINDOWPROP WINDOW 'IL:CLOSEFN 'PE-CLOSEFN)
          (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR PE)
          (IL:WINDOWPROP WINDOW 'IL:BUTTONEVENTFN 'PE-BUTTONEVENTFN)
          (IL:WINDOWPROP WINDOW 'IL:RIGHTBUTTONFN 'PE-RIGHTBUTTONFN)
          (IL:WINDOWPROP WINDOW 'IL:REPAINTFN 'PE-REPAINTFN)
          (IL:WINDOWPROP WINDOW 'IL:RESHAPEFN 'PE-RESHAPEFN)
          (IL:WINDOWPROP WINDOW 'IL:TOTOPFN 'PE-TOTOPFN)
          (IL:WINDOWPROP WINDOW 'IL:AFTERMOVEFN 'PE-TOTOPFN)
          (IL:WINDOWPROP WINDOW 'IL:OPENFN 'PE-TOTOPFN)
          (IL:DSPFONT *PEP-TITLE-FONT* WINDOW)
          (SETF (FIND-PE ROOM-NAME)
                PE)
          ;; update things
          (UPDATE-PE-WINDOW PE)
          (UPDATE-PE PE)
          PE))
```

```
(DEFMACRO FIND-PE (ROOM-NAME)
   `(GETHASH ,ROOM-NAME *PLACEMENT-EDITORS*))
```

```
(DEFUN PE-WINDOW-P (WINDOW)
   (PLACEMENT-EDITOR-P (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
```

```
(DEFUN PE-CLOSEFN (WINDOW)
   ;; remove circularity
   (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR NIL))
```

```
(DEFUN PE-REPAINTFN (WINDOW &REST IGNORE)
   (LET ((PE (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
        (DISPLAY-PE-TITLE PE)
        (DISPLAY-PE-BORDER PE)
        (SETF (PE-CHANGED? PE)
              T)
        (UPDATE-PE PE)))
```

```
(DEFUN PE-RESHAPEFN (WINDOW &REST IGNORE)
   (LET ((PE (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
        (UPDATE-PE-WINDOW PE)
        (SETF (PE-CHANGED? PE)
              T)
        (UNLESS *PE-INHIBIT-REDISPLAY* (UPDATE-PE PE))))
```

```
(DEFUN PE-TOTOPFN (WINDOW)
   (IL:TOTOPW WINDOW T)
   (LET ((PE (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
        (WHEN PE
            (DISPLAY-PE-TITLE PE)
            (UNLESS *PE-INHIBIT-REDISPLAY* (UPDATE-PE? PE)))))
```

```
(DEFUN UPDATE-PE? (PE)
   (WHEN (PE-CHANGED? PE)
         (UPDATE-PE PE)))

(DEFUN UPDATE-PE (PE)

;;; update PE per the current state of rooms

   (LET ((*PE-INHIBIT-REDISPLAY* T)
         ;; don't want TOTOP to update too
         (ROOM (ROOM-NAMED (PE-ROOM-NAME PE))))
        (WHEN ROOM
            (LET ((DSP (IL:GETSTREAM (PE-WINDOW PE))))
                 (CASE (PE-CHANGED? PE)
                     (:PLACEMENTS                                    ; incremental redisplay
                        (LET ((OLD-PEPS (PE-PEPS PE)))
                             (UPDATE-PEPS PE OLD-PEPS)
                             (DISPLAY-PEPS PE DSP OLD-PEPS)))
                     (T                                              ; total redisplay
                        (UPDATE-PEPS PE)
                        (PAINT-BACKGROUND ROOM DSP :SCALE (PE-SCALE PE)
                               :NO-TEXT T :CLIPPING-REGION (PE-CLIPPING-REGION PE))
                        (DISPLAY-PEPS PE DSP)))
                 (SETF (PE-CHANGED? PE)
                       NIL)))))

(DEFUN UPDATE-PEPS (PE &OPTIONAL OLD-PEPS)
   ;; build a new list of PEPs for PE.
   ;; the redisplay code depends upon us leaving EQ peps only when the placement hasn't changed.
   (LET ((ROOM (ROOM-NAMED (PE-ROOM-NAME PE))))
        (WHEN ROOM
            (LET ((IMMEDIATE-PLACEMENTS (ROOM-PLACEMENTS ROOM))
                  (SCALE (PE-SCALE PE)))
                 (DO* ((ALL (FIND-PLACEMENTS ROOM))
                       (TAIL ALL (CDR TAIL)))
                      ((NULL TAIL)
                       (SETF (PE-PEPS PE)
                             ALL))
                   (SETF (FIRST TAIL)
                         (LET* ((PLACEMENT (FIRST TAIL))
                                (IMMEDIATE? (MEMBER PLACEMENT IMMEDIATE-PLACEMENTS :TEST 'EQ))
                                (SCALED-REGION (PLACED-REGION PLACEMENT)))
                               (OR (DOLIST (PEP OLD-PEPS)
                                       ;; try to find an old PEP for placement
                                       (WHEN (EQ (PEP-PLACEMENT PEP)
                                                 PLACEMENT)
                                           ;; found it
                                           (WHEN (EQ PEP (FIRST OLD-PEPS))
                                               ;; speed future searches
                                               (POP OLD-PEPS))
                                           (RETURN (WHEN (AND (IF (PEP-IMMEDIATE? PEP)
                                                                  IMMEDIATE?
                                                                  (NOT IMMEDIATE?))
                                                              (EQUAL (PEP-UNSCALED-REGION PEP)
                                                                     SCALED-REGION))
                                                         ; ok to use it
                                                         PEP))))
                                   (UPDATE-PEP-SAVE-BITMAP (MAKE-PE-PLACEMENT :PLACEMENT PLACEMENT :IMMEDIATE?
                                                                 IMMEDIATE? :SCALED-REGION (SCALE-REGION
                                                                                            SCALED-REGION
                                                                                            SCALE)
                                                                 :UNSCALED-REGION SCALED-REGION)))))))))

(DEFUN UPDATE-PEP-SAVE-BITMAP (PEP)
   (LET* ((SCALED-REGION (PEP-SCALED-REGION PEP))
          (SAVE (IL:BITMAPCREATE (REGION-WIDTH SCALED-REGION)
                                 (REGION-HEIGHT SCALED-REGION)))
          (DSP (IL:LOADTIMECONSTANT (IL:DSPCREATE))))
         (IL:DSPDESTINATION SAVE DSP)
         (IL:DSPXOFFSET (- (REGION-LEFT SCALED-REGION))
                DSP)
         (IL:DSPYOFFSET (- (REGION-BOTTOM SCALED-REGION))
                DSP)
         (DISPLAY-PEP PEP DSP)
         (SETF (PEP-SAVE-BITMAP PEP)
               SAVE)
         (SETF (PEP-OPEN? PEP)
```

```
                NIL)
          PEP))

(DEFUN UPDATE-PE-WINDOW (PE)
   (LET* ((WINDOW (PE-WINDOW PE))
          (DSP (IL:GETSTREAM WINDOW))
          (WINDOW-REGION (WINDOW-REGION WINDOW))
          (WINDOW-HEIGHT (REGION-HEIGHT WINDOW-REGION))
          (TEXT (PE-TITLE-TEXT PE))
          (HEIGHT (- WINDOW-HEIGHT (TEXT-%HEIGHT TEXT)))
          (TWICE-BORDER (* *PE-BORDER* 2))
          (CLIPPING-REGION (MAKE-REGION :LEFT *PE-BORDER* :BOTTOM *PE-BORDER* :WIDTH (- (REGION-WIDTH
                                                                                                WINDOW-REGION)
                                                                                        TWICE-BORDER)
                                        :HEIGHT
                                        (- HEIGHT TWICE-BORDER))))
          ;; update scale & clipping region
          (SETF (PE-CLIPPING-REGION PE)
                CLIPPING-REGION)
          (SETF (PE-SCALE PE)
                (MAKE-SCALE CLIPPING-REGION))
          (SETF (TEXT-POSITION TEXT)
                (MAKE-POSITION *PE-BORDER* HEIGHT))
          (DISPLAY-PE-TITLE PE)
          (DISPLAY-PE-BORDER PE)))

(DEFUN PLACED-REGION (PLACEMENT)

;;; returns the region PLACEMENT would occupy on the screen.  for non-shrunken placements this is just the PLACEMENT-REGION, but for shrunken
;;; placements we need to figure what the region of the icon would be.

   (IF (PLACEMENT-SHRUNKEN? PLACEMENT)
       (LET* ((ICON-POSITION (PLACEMENT-ICON-POSITION PLACEMENT))
              (ICON (WINDOW-ICON (PLACEMENT-WINDOW PLACEMENT)))
              (ICON-REGION (IF ICON (WINDOW-REGION ICON))))
             (MAKE-REGION :LEFT (POSITION-X ICON-POSITION)
                    :BOTTOM
                    (POSITION-Y ICON-POSITION)
                    :WIDTH
                    (IF ICON
                        (REGION-WIDTH ICON-REGION)
                        *DEFAULT-ICON-SIZE*)
                    :HEIGHT
                    (IF ICON
                        (REGION-HEIGHT ICON-REGION)
                        *DEFAULT-ICON-SIZE*)))
       (PLACEMENT-REGION PLACEMENT)))

;; display

(DEFVAR *DEFAULT-ICON-SIZE*

;;; when we draw a placement for a non-existant icon, we draw it as a square with this many (scaled) pixels per side.

   75)

(DEFGLOBALVAR *PE-TITLE-FONT* (IL:FONTCREATE 'IL:HELVETICA 36 '(IL:BOLD IL:REGULAR IL:REGULAR)
                                      NIL
                                      'IL:DISPLAY))

(DEFGLOBALPARAMETER *PEP-TITLE-FONT* (IL:FONTCREATE IL:TINYFONT NIL NIL NIL 'IL:DISPLAY))

(DEFGLOBALPARAMETER *PEP-INCLUDED-SHADE* 4680)

(DEFGLOBALPARAMETER *PE-BORDER* 4)

(DEFUN DISPLAY-PEPS (PE DSP &OPTIONAL OLD-PEPS)

;;; displays PE on DSP.  Should be called DISPLAY-PE-INTERNAL.

   (LET ((OLD OLD-PEPS)
         (NEW (PE-PEPS PE)))
        (LOOP
              ;; pop off the EQ peps on the bottom of window stack
              (WHEN (OR (NULL OLD)
                        (NULL NEW)
                        (NOT (EQ (FIRST OLD)
                                 (FIRST NEW))))
```

```
                (RETURN))
            (POP OLD)
            (POP NEW))
```

;; remove image of remaining old peps

```
        (DOLIST (PEP (REVERSE OLD))
            (SAVE-PEP-IMAGE DSP PEP PE))
```

;; display remaining new peps

```
        (DOLIST (PEP NEW)
            (SAVE-PEP-IMAGE DSP PEP PE))))
```

(DEFUN SAVE-PEP-IMAGE (DSP PEP PE)

;;; switch contents of PEP's save & its region of DSP.

```
   (LET* ((REGION (PEP-SCALED-REGION PEP))
          (LEFT (POP REGION))
          (BOTTOM (POP REGION))
          (WIDTH (POP REGION))
          (HEIGHT (POP REGION))
          (CLIPPING (PE-CLIPPING-REGION PE))
          (CLIPPING-LEFT (POP CLIPPING))
          (CLIPPING-BOTTOM (POP CLIPPING))
          (CLIPPING-WIDTH (POP CLIPPING))
          (CLIPPING-HEIGHT (POP CLIPPING))
          (SAVE (PEP-SAVE-BITMAP PEP)))
         (IL:UNINTERRUPTABLY
             (IL:BITBLT SAVE 0 0 DSP LEFT BOTTOM WIDTH HEIGHT 'IL:INPUT 'IL:INVERT NIL CLIPPING)
             (IL:BITBLT DSP (MAX LEFT CLIPPING-LEFT)
                    (MAX BOTTOM CLIPPING-BOTTOM)
                    SAVE 0 0 (MIN WIDTH (- (+ CLIPPING-WIDTH CLIPPING-LEFT)
                                           LEFT))
                    (MIN HEIGHT (- (+ CLIPPING-HEIGHT CLIPPING-BOTTOM)
                                   BOTTOM))
                    'IL:INPUT
                    'IL:INVERT)
             (IL:BITBLT SAVE 0 0 DSP LEFT BOTTOM WIDTH HEIGHT 'IL:INPUT 'IL:INVERT NIL CLIPPING))
         (SETF (PEP-OPEN? PEP)
               (NOT (PEP-OPEN? PEP)))))
```

(DEFUN DISPLAY-PE-TITLE (PE)

```
   (LET* ((WINDOW (PE-WINDOW PE))
          (WINDOW-REGION (WINDOW-REGION WINDOW))
          (DSP (IL:GETSTREAM WINDOW))
          (TEXT (PE-TITLE-TEXT PE))
          (WINDOW-HEIGHT (REGION-HEIGHT WINDOW-REGION))
          (BOTTOM (- WINDOW-HEIGHT (TEXT-%HEIGHT TEXT))))
```

;; blt the background into the title bar

```
         (IL:BITBLT (IL:WINDOWPROP WINDOW 'IL:IMAGECOVERED)
                0 BOTTOM DSP 0 BOTTOM (REGION-WIDTH WINDOW-REGION)
                (- WINDOW-HEIGHT BOTTOM))
         (IF (EQUAL (PE-ROOM-NAME PE)
                    *BACK-DOOR-ROOM-NAME*)
```

;; the title of the back door room gets special shadows

```
             (WHEN (EQ (TEXT-SHADOWS TEXT)
                       T)
                   (SETF (TEXT-SHADOWS TEXT)
                         (MAPLIST #'(LAMBDA (TAIL)
                                            (IF (REST TAIL)
                                                (FIRST TAIL)
                                                (IL:CONSTANT (MAKE-TEXT-SHADOW :SOURCE-TYPE 'IL:MERGE :TEXTURE
                                                                               42405 :OPERATION 'IL:INVERT))))
                                  (GET-TEXT-SHADOWS-INTERNAL *PE-TITLE-FONT*)))
                   (UPDATE-TEXT-CACHES TEXT))
             (UNLESS (EQ (TEXT-SHADOWS TEXT)
                         T)
```

;; used to be back-door room

```
                 (SETF (TEXT-SHADOWS TEXT)
                       T)
                 (UPDATE-TEXT-CACHES TEXT)))
         (DISPLAY-TEXT TEXT DSP)))
```

(DEFUN DISPLAY-PE-BORDER (PE)

```
   (LET* ((WINDOW (PE-WINDOW PE))
          (WINDOW-REGION (WINDOW-REGION WINDOW)))
```

;; draw the window border & clear inside it

```
         (DRAW&FILL-BOX-WITHIN (MAKE-REGION :LEFT 0 :BOTTOM 0 :WIDTH (REGION-WIDTH WINDOW-REGION)
                                            :HEIGHT
                                            (- (REGION-HEIGHT WINDOW-REGION)
```

```
                                        (TEXT-%HEIGHT (PE-TITLE-TEXT PE))))
            (IL:GETSTREAM WINDOW)
            :BORDER-WIDTH
            (FLOOR *PE-BORDER* 2))))
```

(DEFUN **DISPLAY-PEP** (PEP DSP)

;;; displays a PE-PLACEMENT

;; draw a box around the region & fill it if it represents an immediate placement in the room of this PEP.

```
(DRAW&FILL-BOX-WITHIN (PEP-SCALED-REGION PEP)
       DSP :BORDER-WIDTH 1 :SHADE (IF (PEP-IMMEDIATE? PEP)
                                      IL:WHITESHADE
                                      *PEP-INCLUDED-SHADE*))
```

(**PRINT-PEP-TITLE** PEP DSP))

(DEFUN **PRINT-PEP-TITLE** (PEP DSP)

;; print something within the box drawn for PEP on DSP

```
(LET* ((WINDOW-TYPE (WINDOW-TYPE (PLACEMENT-WINDOW (PEP-PLACEMENT PEP))
                                 T)))
      (WHEN WINDOW-TYPE
          (LET ((TITLE (WINDOW-TYPE-PROP WINDOW-TYPE :TITLE)))
```

;; interpret the TITLE property of the window type of the placement this PEP represents

```
               (COND
                  ((NULL TITLE)
```

;; if none specified, just print the name of the type

```
                   (PRINT-PEP-TITLE-STRING (STRING (WINDOW-TYPE-NAME WINDOW-TYPE))
                          (PEP-SCALED-REGION PEP)
                          DSP :NO-TITLE-BAR? (PLACEMENT-SHRUNKEN? (PEP-PLACEMENT PEP))))
                  ((STRINGP TITLE)
```

;; if it's a string, print it

```
                   (PRINT-PEP-TITLE-STRING TITLE (PEP-SCALED-REGION PEP)
                          DSP :NO-TITLE-BAR? (PLACEMENT-SHRUNKEN? (PEP-PLACEMENT PEP))))
                  (T
```

;; otherwise assume it's a function & call it

```
                     (FUNCALL TITLE (PEP-PLACEMENT PEP)
                            (PEP-SCALED-REGION PEP)
                            DSP)))))))
```

(DEFUN **PRINT-PEP-TITLE-STRING** (STRING REGION DSP &KEY (FONT *PEP-TITLE-FONT*)
                                        NO-TITLE-BAR?)

;;; prints STRING in the top left corner of REGION if it will fit.

```
   (LET* ((STRING (IF (STRINGP STRING)
                      STRING
                      (PRINC-TO-STRING STRING)))
          (FONT-HEIGHT (IL:FONTHEIGHT FONT))
          (TITLE-Y (- (+ (REGION-BOTTOM REGION)
                         (REGION-HEIGHT REGION))
                      FONT-HEIGHT
                      (IF NO-TITLE-BAR?
                          1
                          0)))
          (STRING-WIDTH (IL:STRINGWIDTH STRING FONT)))
         (WHEN (AND (< STRING-WIDTH (- (REGION-WIDTH REGION)
                                       2))
                    (< FONT-HEIGHT (- (REGION-HEIGHT REGION)
                                      2)))
             (UNLESS NO-TITLE-BAR?
                 (IL:BLTSHADE IL:BLACKSHADE DSP (REGION-LEFT REGION)
                        (1- TITLE-Y)
                        (REGION-WIDTH REGION)
                        FONT-HEIGHT
                        'IL:PAINT))
             (IL:DSPOPERATION (IF NO-TITLE-BAR?
                                  'IL:PAINT
                                  'IL:INVERT)
                    DSP)
             (IL:DSPFONT FONT DSP)
             (IL:MOVETO (+ (REGION-LEFT REGION)
                           (IF NO-TITLE-BAR?
                               (- (FLOOR (REGION-WIDTH REGION)
                                         2)
                                  (FLOOR STRING-WIDTH 2))
                               2))
                    (IF NO-TITLE-BAR?
                        (+ (REGION-BOTTOM REGION)
                           (- (FLOOR (REGION-HEIGHT REGION)
                                     2)
```

```
                              (FLOOR FONT-HEIGHT 2))
                            (IL:FONTDESCENT FONT))
                         (+ TITLE-Y (IL:FONTDESCENT FONT)))
                      DSP)
               (CHECK-TYPE DSP STREAM)
               (IL:\\SOUT STRING DSP)
               (IL:DSPOPERATION 'IL:REPLACE DSP))))


(DEFPARAMETER *DISPLAY-PE-DEPTH* 1
   "Depth to recursivly display placement editors within placement editors")


(DEF-WINDOW-TYPE :PLACEMENT-EDITOR :RECOGNIZER (LAMBDA (WINDOW)
                                                        (PLACEMENT-EDITOR-P (IL:WINDOWPROP WINDOW
                                                                                           :PLACEMENT-EDITOR)))
   :ABSTRACTER (LAMBDA (WINDOW)
                       (LET ((PE (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
                            (WHEN PE
                                `(:REGION ,(EXTERNALIZE-REGION (WINDOW-REGION (PE-WINDOW PE)))
                                        :ROOM-NAME
                                        ,(PE-ROOM-NAME (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR))))))
   :RECONSTITUTER (LAMBDA (PLIST)
                          (PE-WINDOW (GET-PE (GETF PLIST :ROOM-NAME)
                                             (INTERNALIZE-REGION (GETF PLIST :REGION)))))
   :TITLE (LAMBDA (PLACEMENT REGION DSP)
                  (LET ((PE (IL:WINDOWPROP (PLACEMENT-WINDOW PLACEMENT)
                                           :PLACEMENT-EDITOR)))
                       (WHEN PE
                           (PRINT-PEP-TITLE-STRING (LET ((NAME (PE-ROOM-NAME PE)))
                                                     (IF (STRINGP NAME)
                                                         NAME
                                                         (PRINC-TO-STRING NAME)))
                                                   REGION DSP)

#|(when (> *display-pe-depth* 0) (let ((*display-pe-depth* (1- *display-pe-depth*)) (old-scale (pe-scale pe)) (old-peps (pe-peps pe)) (old-clipping-region (pe-clipping-region pe)) (new-clipping-region (make-region :left (+ (region-left region) 1) :bottom (+ (region-bottom region) 1) :width (- (region-width region) 2) :height (- (region-height region) 2 (il:fontheight *pep-title-font*))))) (unwind-protect (progn (setf (pe-scale pe) (make-scale new-clipping-region)) (setf (pe-changed? pe) t) (setf (pe-clipping-region pe) new-clipping-region) (update-peps pe) (il:* il:|;;| "recursively display pictogram") (display-peps pe dsp)) (setf (pe-scale pe) old-scale) (setf (pe-peps pe) old-peps) (setf (pe-clipping-region pe) old-clipping-region)))|#

                           ))))


(DEFUN PE-RIGHTBUTTONFN (WINDOW)
   (UNLESS (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
          (IL:DOWINDOWCOM WINDOW)))


(DEFUN PE-BUTTONEVENTFN (WINDOW)
   (IL:TOTOPW WINDOW)
   (LET ((PE (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
        (IF (IL:INSIDEP (PE-CLIPPING-REGION PE)
                   (IL:LASTMOUSEX WINDOW)
                   (IL:LASTMOUSEY WINDOW))
            (PE-BUTTONEVENTFN-INTERNAL PE WINDOW)
            (UNLESS (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
                (CASE (MENU '(("ReFetch" :RE-FETCH)))
                    (:RE-FETCH (UPDATE-PLACEMENTS)))))))


(DEFUN PE-BUTTONEVENTFN-INTERNAL (PE WINDOW)
   (LET ((WINDOW WINDOW)
         (PE PE))
        (LOOP (WHEN (AND (PLACEMENT-EDITOR-P PE)
                         (PE-TRACK-MOUSE PE WINDOW))
                   (RETURN))
              (PE-GETMOUSESTATE)
              (UNLESS (IL:LASTMOUSESTATE (OR IL:LEFT IL:MIDDLE))
                     (RETURN))
              (SETQ WINDOW (IL:WHICHW))
              (SETQ PE (WHEN WINDOW (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR))))))


(DEFUN PE-TRACK-MOUSE (PE WINDOW)
   (LET ((*TIMER* (IL:SETUPTIMER *BUTTON-HELP-DELAY*)))
        (DECLARE (SPECIAL *TIMER*))
        (COND
           ((IL:LASTMOUSESTATE (IL:ONLY IL:LEFT))
            (LET ((SELECTED-PEP (PE-TRACK-LEFT-BUTTON PE)))
                 (WHEN SELECTED-PEP
                     (PEP-SELECTED SELECTED-PEP PE)
                     ;; return true when something's been selected
                     T)))
           ((IL:LASTMOUSESTATE IL:MIDDLE)
            (LET ((ROOM-SELECTED? (PE-TRACK-MIDDLE-BUTTON PE)))
```

```
                (WHEN ROOM-SELECTED?
                    (PE-ROOM-SELECTED PE)
                    ;; return true when something's been selected
                    T))))))

(DEFUN PE-GETMOUSESTATE ()
   ;; call GETMOUSESTATE, and if the mouse hasn't moved, block.  This is in case we're in the Overview, so the keyboard watcher will run.
   (LET ((X IL:LASTMOUSEX)
         (Y IL:LASTMOUSEY))
        (IL:GETMOUSESTATE)
        (WHEN (AND (= X IL:LASTMOUSEX)
                   (= Y IL:LASTMOUSEY))
              (IL:BLOCK))))

(DEFUN PE-TRACK-LEFT-BUTTON (PE)
   (DECLARE (SPECIAL *TIMER*))
   (LET ((SELECTED-PEP NIL)
         (CLIPPING-REGION (PE-CLIPPING-REGION PE))
         (WINDOW (PE-WINDOW PE))
         ;; need PEPS in top down order to handle occlusion correctly
         (PEPS (REVERSE (PE-PEPS PE))))
        (MACROLET ((INVERT-SELECTED-PEP NIL `(LET ((REGION (PEP-SCALED-REGION SELECTED-PEP)))
                                                  (IL:BLTSHADE IL:BLACKSHADE WINDOW (REGION-LEFT REGION)
                                                         (REGION-BOTTOM REGION)
                                                         (REGION-WIDTH REGION)
                                                         (REGION-HEIGHT REGION)
                                                         'IL:INVERT CLIPPING-REGION))))
                  (LOOP (UNLESS (AND (IL:LASTMOUSESTATE (IL:ONLY IL:LEFT))
                                     (IL:INSIDEP CLIPPING-REGION (IL:LASTMOUSEX WINDOW)
                                                 (IL:LASTMOUSEY WINDOW)))
                            (RETURN (WHEN SELECTED-PEP
                                        ;; restore the display
                                        (INVERT-SELECTED-PEP)
                                        (WHEN (AND (NOT (IL:LASTMOUSESTATE IL:MIDDLE))
                                                   (IL:INSIDEP CLIPPING-REGION (IL:LASTMOUSEX WINDOW)
                                                               (IL:LASTMOUSEY WINDOW)))
                                            ;; return the PEP if there's one selected & the button event fn should be exited.
                                            SELECTED-PEP))))
                            (DOLIST (PEP PEPS (WHEN SELECTED-PEP ;; look down the occlusion stack
                                ;; bottomed out -- must be in the background
                                ;; have to de-select selection
                                (INVERT-SELECTED-PEP)
                                (SETQ SELECTED-PEP NIL)))
                                (WHEN (IL:INSIDEP (PEP-SCALED-REGION PEP)
                                                  (IL:LASTMOUSEX WINDOW)
                                                  (IL:LASTMOUSEY WINDOW))
                                    ;; we're in a PEP
                                    (UNLESS (EQ PEP SELECTED-PEP)
                                        ;; it's a new selection
                                        (WHEN SELECTED-PEP
                                            ;; unselect the current selection
                                            (INVERT-SELECTED-PEP))
                                        ;; select the new one
                                        (SETQ SELECTED-PEP PEP)
                                        (INVERT-SELECTED-PEP))
                                    ;; pop to the outer loop
                                    (RETURN)))
                            (WHEN (AND *TIMER* (IL:TIMEREXPIRED? *TIMER*))
                                (NOTIFY-USER "Use LEFT to select placements, MIDDLE Rooms.")
                                (IL:TOTOPW WINDOW T)
                                (SETQ *TIMER* NIL))
                            (PE-GETMOUSESTATE)))))

(DEFUN PE-TRACK-MIDDLE-BUTTON (PE)
   (DECLARE (SPECIAL *TIMER*))
   (LET* ((WINDOW (PE-WINDOW PE))
          (REGION (WINDOW-REGION WINDOW)))
         (UNWIND-PROTECT
             (PROGN (IL:INVERTW WINDOW)
                    (LOOP (UNLESS (IL:INSIDEP REGION IL:LASTMOUSEX IL:LASTMOUSEY)
                                  ;; return NIL if mouse leaves clipping region
```

```
                            (RETURN NIL))
                        (UNLESS (IL:LASTMOUSESTATE IL:MIDDLE)
                            ;; return true iff room was selected
                            (RETURN (NOT (IL:LASTMOUSESTATE (OR IL:LEFT IL:RIGHT)))))
                        (WHEN (AND *TIMER* (IL:TIMEREXPIRED? *TIMER*))
                            (NOTIFY-USER "Use LEFT to select placements, MIDDLE Rooms.")
                            (IL:TOTOPW WINDOW T)
                            (SETQ *TIMER* NIL))
                        (PE-GETMOUSESTATE)))
            (IL:INVERTW WINDOW))))
```

(DEFUN **PEP-GETBOXPOSITION** (PE PEP)

;;; called when a placement is MOVE or COPY selected to get the new position for the placement.

;;; returns two values - a position and a PE - or NIL

```
    (LET* ((OLD-CURSOR (IL:CURSOR))
           (OLD-TTY (IL:TTY.PROCESS)))
          (UNWIND-PROTECT
              (PROGN (IL:CURSOR IL:CROSSHAIRS)
                     (IL:TTY.PROCESS (IL:THIS.PROCESS))
                     (IL:CURSORPOSITION (MAKE-POSITION (REGION-LEFT (PEP-SCALED-REGION PEP))
                                                       (REGION-BOTTOM (PEP-SCALED-REGION PEP)))
                            (PE-WINDOW PE))
                     (DO* ((PLACED-REGION (PLACED-REGION (PEP-PLACEMENT PEP)))
                           (HEIGHT (REGION-HEIGHT PLACED-REGION))
                           (WIDTH (REGION-WIDTH PLACED-REGION))
                           (WINDOW NIL
                                   ;; the window the cursor is over (if any)
                                   (IL:WHICHW))
                           (PE NIL
                               ;; the PE the cursor is over (if any)
                               (IF (AND WINDOW (PE-WINDOW-P WINDOW))
                                   (IL:WINDOWPROP WINDOW :PLACEMENT-EDITOR)))
                           (POSITION NIL
                                   ;; the selected position within PE (if any)
                                   (IF PE
                                       (LET ((CLIPPING-REGION (PE-CLIPPING-REGION PE))
                                             (SCALE (PE-SCALE PE)))
                                            (IF (IL:INSIDEP CLIPPING-REGION (IL:LASTMOUSEX WINDOW)
                                                            (IL:LASTMOUSEY WINDOW))
                                                ;; have to rescale box for each PE
                                                (PEP-GETBOXPOSITION-INTERNAL (SCALE-WIDTH WIDTH SCALE)
                                                       (SCALE-HEIGHT HEIGHT SCALE)
                                                       (PE-CLIPPING-REGION PE)
                                                       WINDOW)
                                                (IL:GETMOUSESTATE)))
                                       (IL:GETMOUSESTATE))))
                          ((OR POSITION (IL:LASTMOUSESTATE (OR IL:LEFT IL:MIDDLE IL:RIGHT)))
                           (WHEN POSITION
                               (VALUES (UN-SCALE-POSITION POSITION (PE-SCALE PE))
                                       PE)))))
              (IL:CURSOR OLD-CURSOR)
              (IL:TTY.PROCESS OLD-TTY))))
```

(DEFUN **PEP-GETBOXPOSITION-INTERNAL** (WIDTH HEIGHT CLIPPING-REGION WINDOW)

;;; track a box WIDTH by HEIGHT within CLIPPING-REGION in WINDOW. if a button goes down return the X,Y position. if cursor goes outside
;;; CLIPPING-REGION then return NIL.

```
    (LET* ((DSP (IL:GETSTREAM WINDOW))
           (OLD-OPERATION (IL:DSPOPERATION NIL DSP))
           (LAST-X (IL:LASTMOUSEX DSP))
           (LAST-Y (IL:LASTMOUSEY DSP)))
          (MACROLET ((INVERT-BOX NIL `(IL:DRAWGRAYBOX LAST-X LAST-Y (+ LAST-X WIDTH)
                                                      (+ LAST-Y HEIGHT)
                                                      DSP IL:GRAYSHADE)))
                (UNWIND-PROTECT
                    (PROGN (IL:TOTOPW WINDOW)
                           (IL:DSPOPERATION 'IL:INVERT DSP)
                           (INVERT-BOX)
                           (TAGBODY LOOP (UNLESS (IL:INSIDEP CLIPPING-REGION LAST-X LAST-Y)
                                                 (RETURN-FROM PEP-GETBOXPOSITION-INTERNAL))
                                    (WHEN (IL:MOUSESTATE (OR IL:LEFT IL:MIDDLE IL:RIGHT))
                                        (RETURN-FROM PEP-GETBOXPOSITION-INTERNAL (MAKE-POSITION LAST-X LAST-Y)))
                                    (UNLESS (AND (= (IL:LASTMOUSEX DSP)
                                                    LAST-X)
                                                 (= (IL:LASTMOUSEY DSP)
                                                    LAST-Y))
```

```
                                    (IL:UNINTERRUPTABLY
                                        ;; un-draw old
                                        (INVERT-BOX)
                                        (SETQ LAST-X (IL:LASTMOUSEX DSP))
                                        (SETQ LAST-Y (IL:LASTMOUSEY DSP))
                                        ;; re-draw new
                                        (INVERT-BOX)))
                                (GO LOOP)))
                    (INVERT-BOX)
                    (IL:DSPOPERATION OLD-OPERATION DSP)))))
```

;; editing

```
(DEFUN PE-ROOM-CHANGED-FN (ROOM REASON)
   (WHEN (EQ REASON :DELETED)
       ;; if ROOM has been deleted then delete the placement editor
       (LET ((PE (FIND-PE (ROOM-NAME ROOM))))
            (WHEN PE
                (LET ((WINDOW (PE-WINDOW PE)))
                     ;; delete the placement editor
                     (UN-HIDE-WINDOW WINDOW)
                     (CLOSE-WINDOW WINDOW)
                     (REMHASH (ROOM-NAME ROOM)
                              *PLACEMENT-EDITORS*)))))
   (LET ((INCLUDERS (ROOM-INCLUDERS ROOM T)))
        ;; ensure PE's for all rooms which include ROOM will be redisplayed
        (DOLIST (INCLUDER INCLUDERS)
            (LET ((PE (FIND-PE (ROOM-NAME INCLUDER))))
                 (WHEN PE
                     ;; otherwise mark it as needing update
                     (UNLESS (EQ (PE-CHANGED? PE)
                                 T)
                         ;; OK to upgrade :PLACEMENTS to T, but not vice versa
                         (SETF (PE-CHANGED? PE)
                               (IF (EQ REASON :PLACEMENTS)
                                   REASON
                                   T)))
                     (LET ((WINDOW (PE-WINDOW PE)))
                          (WHEN (AND (IL:OPENWP WINDOW)
                                     (NOT (WINDOW-HIDDEN? WINDOW)))
                              ;; update the PE if it's visible
                              (UPDATE-PE PE))))))))
```

```
(DEFUN PEP-SELECTED (PEP PE)
```

;;; called when a placement is selected in PE

```
   (LET ((OP (COND
                ((COPY-KEY-DOWN-P)
                 :COPY)
                ((MOVE-KEY-DOWN-P)
                 :MOVE)
                ((DELETE-KEY-DOWN-P)
                 :DELETE)
                ((EXPAND-KEY-DOWN-P)
                 :EXPAND)
                (T (OV-OPERATION)))))
        (CASE OP
            ((:COPY :MOVE) (PEP-SELECTED-COPY-OR-MOVE OP PEP PE))
            (:DELETE (INTERACTIVE-CLOSE-WINDOW (PLACEMENT-WINDOW (PEP-PLACEMENT PEP))
                             (ROOM-NAMED (PE-ROOM-NAME PE))))
            (:EXPAND (EXPAND-PLACEMENT (PEP-PLACEMENT PEP)))
            (T (NOTIFY-USER "Use a modifier (eg. COPY, MOVE or DELETE)")))))
```

```
(DEFUN EXPAND-PLACEMENT (PLACEMENT)
   (LET ((WINDOW (PLACEMENT-WINDOW PLACEMENT)))
        (IF (WINDOW-HIDDEN? WINDOW)
            (PROGN (NOTIFY-USER "Click LEFT when finished")
                   (PLACE-PLACEMENT PLACEMENT)
                   (LOOP (WHEN (IL:MOUSESTATE IL:LEFT)
                               (RETURN)))
                   (HIDE-WINDOW WINDOW))
            (IL:FLASHWINDOW (IF (SHRUNKEN? WINDOW)
                                (WINDOW-ICON WINDOW)
                                WINDOW)))))
```

```
(DEFUN PEP-SELECTED-COPY-OR-MOVE (OP PEP PE)
   (MULTIPLE-VALUE-BIND (DESTINATION-POS DESTINATION-PE)
       (PEP-GETBOXPOSITION PE PEP)
     (IF DESTINATION-POS
         (LET* ((OLD-PLACEMENT (PEP-PLACEMENT PEP))
                (NEW-PLACEMENT (COPY-PLACEMENT OLD-PLACEMENT)))
               ;; adjust the position of the new placement
               (IF (PLACEMENT-SHRUNKEN? NEW-PLACEMENT)
                   (SETF (PLACEMENT-ICON-POSITION NEW-PLACEMENT)
                         DESTINATION-POS)
                   (SETF (PLACEMENT-REGION NEW-PLACEMENT)
                         (MAKE-REGION :LEFT (POSITION-X DESTINATION-POS)
                                :BOTTOM
                                (POSITION-Y DESTINATION-POS)
                                :WIDTH
                                (REGION-WIDTH (PLACEMENT-REGION NEW-PLACEMENT))
                                :HEIGHT
                                (REGION-HEIGHT (PLACEMENT-REGION NEW-PLACEMENT)))))
               ;; do the move/copy
               (LET ((PE-ROOM (ROOM-NAMED (PE-ROOM-NAME PE)))
                     (DESTINATION-ROOM (ROOM-NAMED (PE-ROOM-NAME DESTINATION-PE))))
                    (MULTIPLE-VALUE-BIND (PLACEMENT SOURCE-ROOM)
                        ;; find the room this placement is due to


                        (FIND-PLACEMENT (PLACEMENT-WINDOW OLD-PLACEMENT)
                               PE-ROOM)
                        (ECASE OP
                            (:MOVE
                               (IF (EQ PE DESTINATION-PE)
                                   ;; Allow inherited placements to be moved in place -- w/o moving them to the room they're visible in.
                                   (SETQ DESTINATION-ROOM SOURCE-ROOM)
                                   (UNLESS (EQ DESTINATION-ROOM SOURCE-ROOM)
                                       ;; We don't bother deleting first when source & destination are same, as we know ADD-PLACEMENT
                                       ;; will delete the old & we only want to redisplay once
                                       (DELETE-PLACEMENT PLACEMENT SOURCE-ROOM)))
                               (ADD-PLACEMENT NEW-PLACEMENT DESTINATION-ROOM))
                            (:COPY (ADD-PLACEMENT NEW-PLACEMENT DESTINATION-ROOM))))))
         (NOTIFY-USER "Invalid destination."))))

(DEFUN PE-ROOM-SELECTED (PE)

;;; called when a room is selected in PE

   (LET ((ROOM (ROOM-NAMED (PE-ROOM-NAME PE)))
         (OP (COND
                ((IL:KEYDOWNP 'IL:SPACE)
                 :ENTER)
                ((EDIT-KEY-DOWN-P)
                 :EDIT)
                ((COPY-KEY-DOWN-P)
                 :COPY)
                ((MOVE-KEY-DOWN-P)
                 :MOVE)
                ((DELETE-KEY-DOWN-P)
                 :DELETE)
                (T (IF (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
                       (OV-OPERATION)
                       :ENTER)))))
        (CASE OP
            (:EDIT (EDIT-ROOM ROOM))
            (:COPY (INTERACTIVE-COPY-ROOM ROOM))
            (:MOVE (INTERACTIVE-RENAME-ROOM ROOM))
            (:DELETE (INTERACTIVE-DELETE-ROOM ROOM))
            (:ENTER (GO-TO-ROOM ROOM))
            (T (NOTIFY-USER "Use a modifier (eg. COPY, DELETE or GO TO)."))))))

(EVAL-WHEN (LOAD)

(PUSHNEW 'PE-ROOM-CHANGED-FN *ROOM-CHANGED-FUNCTIONS*)
)

(IL:DECLARE\: IL:DOEVAL@COMPILE IL:DONTCOPY

(IL:GLOBALVARS IL:TINYFONT IL:CROSSHAIRS)
)

(IL:PUTPROPS IL:ROOMS-PLACEMENT-EDITOR IL:COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1990 2020))
```

FUNCTION INDEX

VARIABLE INDEX

STRUCTURE INDEX

WINDOW-TYPE INDEX

MACRO INDEX

FILE-ENVIRONMENT INDEX