

File created: 5-Dec-2020 16:35:56 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOM
S>MEDLEY-35>ROOMS-OVERVIEW.;2

previous date: 17-Aug-90 13:23:15 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-O
VERVIEW.;1

Read Table: XCL

Package: ROOMS

Format: XCCS

; Copyright (c) 1987, 1988, 1990, 2020 by Venue & Xerox Corporation. All rights reserved.

```
(IL:RPAQQ IL:ROOMS-OVERVIEWCOMS
  ( (FILE-ENVIRONMENTS IL:ROOMS-OVERVIEW)
    (IL:P (EXPORT ' (RESET-OVERVIEW ROOM-UNWIND-SAVE)
              "ROOMS")
        (REQUIRE "ROOMS")))
  (IL:COMS
    (IL:VARIABLES *OVERVIEW-ROOM*
                  ; the room
    (IL:FUNCTIONS GO-TO-OVERVIEW OV-ENTRY-FN OV-EXIT-FN OV-ROOM-CHANGED-FN OV-SUITE-BODY))
  (IL:COMS
    (IL:VARIABLES *OV-BORDER-SIZE*
                  ; tiling code
    (IL:FUNCTIONS OV-LAYOUT-PLACEMENT-EDITORS OV-ROWS&COLUMNS))
  (IL:COMS
    (IL:VARIABLES *OV-BUTTONS* *OV-SELECTED-BUTTON-WINDOW* *OV-CHANGED?*)
    (IL:FUNCTIONS MAKE-OV-KEYACTION-TABLE OV-WATCH-KEYBOARD OV-SELECT-BUTTON OV-DESELECT-BUTTON
                  OV-OPERATION)
    (IL:VARIABLES *OV-KEYACTION-TABLE*))
  (IL:FUNCTIONS RESET-OVERVIEW)
  (IL:COMS (IL:VARIABLES *ROOM-UNWINDERS*)
    (IL:FUNCTIONS ROOM-UNWIND-SAVE ROOM-UNWIND)
    (EVAL-WHEN (LOAD)
      (IL:P (PUSHNEW ' (RESET-OVERVIEW)
                    *RESET-FORMS* :TEST 'EQUAL)
            (PUSHNEW 'RESET-OVERVIEW *SCREEN-CHANGED-FUNCTIONS*)
            (PUSHNEW 'ROOM-UNWIND *ROOM-EXIT-FUNCTIONS*)
            (PUSHNEW 'OV-ROOM-CHANGED-FN *ROOM-CHANGED-FUNCTIONS*)))))

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

(EXPORT ' (RESET-OVERVIEW ROOM-UNWIND-SAVE)
  "ROOMS")

(REQUIRE "ROOMS")

;; the room

(DEFGLOBALVAR *OVERVIEW-ROOM* NIL)

(DEFUN GO-TO-OVERVIEW ()
  (GO-TO-ROOM *OVERVIEW-ROOM*))

(DEFUN OV-ENTRY-FN (OVERVIEW-ROOM)
  ;; the entry function of the overview room. called whenever we enter the overview room.
  (LET ((KEYBOARD-WATCHER (IL:ADD.PROCESS ' (OV-WATCH-KEYBOARD)
                                          ' IL:KEYACTION *OV-KEYACTION-TABLE* ' IL:RESTARTABLE T)))
    ;; add the keyboard watcher
    (ROOM-UNWIND-SAVE
      ;; make sure it will get deleted when we exit
      (IL:DEL.PROCESS KEYBOARD-WATCHER T))
    ;; make sure it will have the TTY when we enter the overview
    (SETF (ROOM-TTY-PROCESS OVERVIEW-ROOM)
          KEYBOARD-WATCHER))
  ;; place a PE for each room
  (OV-LAYOUT-PLACEMENT-EDITORS (ALL-ROOMS T)
    (INTERNALIZE-REGION (MAKE-REGION :LEFT 0 :BOTTOM 1/4 :WIDTH 1.0 :HEIGHT 3/4)))
  ;; make sure PE's don't get placed again
  (SETF (ROOM-PLACEMENTS OVERVIEW-ROOM)
        (WITH-COLLECTION (DOLIST (PLACEMENT (ROOM-PLACEMENTS OVERVIEW-ROOM))
                                (UNLESS (PLACEMENT-EDITOR-P (IL:WINDOWPROP (PLACEMENT-WINDOW PLACEMENT)
                                                                              :PLACEMENT-EDITOR))
                                  (COLLECT PLACEMENT)))))
  ;; select GO-TO button initially
```

(OV-SELECT-BUTTON (GETHASH :ENTER *OV-BUTTONS*))

(DEFUN OV-EXIT-FN (OVERVIEW-ROOM)
(WHEN *OV-SELECTED-BUTTON-WINDOW* (OV-DESELECT-BUTTON *OV-SELECTED-BUTTON-WINDOW*)))

(DEFUN OV-ROOM-CHANGED-FN (ROOM REASON)
;; called whenever a room is changed
(WHEN (AND (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
(NOT (EQ ROOM *OVERVIEW-ROOM*)))
;; when we're in the overview
(CASE REASON
((:CREATED :DELETED)
;; have to re-layout placement editors
(MAPHASH #'(LAMBDA (NAME PE)
(WHEN (IL:OPENWP (PE-WINDOW PE))
(IL:CLOSEW (PE-WINDOW PE))))
PLACEMENT-EDITORS)
;; hack: signal OV-WATCH-KEYBOARD process that re-layout is required. this makes multiple deletes & adds appear as one event.
(SETQ *OV-CHANGED?* T))))))

(DEFUN OV-SUITE-BODY ()
\(:VERSION 0)
(:WINDOW :SPACE-BAR :TYPE :BUTTON :TEXT
, (LET ((FILLER (MAKE-STRING (- (FLOOR (/ IL:SCREENWIDTH (* (IL:CHARWIDTH (CHAR-CODE #\Space)
DEFAULT-TEXT-FONT)
6))))
4)
:INITIAL-ELEMENT #\Space)))
(CONCATENATE 'STRING FILLER "GO TO" FILLER))
:SHADOWS :ARK :TYPE :STRETCHY-ARK :HELP "GO TO mode - selected rooms will be entered" :ACTION
OV-SELECT-BUTTON :OV-BUTTON :ENTER :PROTECTED? T)
(:WINDOW :EDIT :TYPE :BUTTON :TEXT "EDIT" :SHADOWS :ARK :TYPE :ARK :HELP "EDIT mode - selected rooms will
be edited" :ACTION OV-SELECT-BUTTON :OV-BUTTON :EDIT :PROTECTED? T)
(:WINDOW :EXPAND :TYPE :BUTTON :TEXT "EXPAND" :SHADOWS :ARK :TYPE :ARK :HELP "EDIT mode - selected rooms
will be edited" :ACTION OV-SELECT-BUTTON :OV-BUTTON :EXPAND :PROTECTED? T)
(:WINDOW :MOVE :TYPE :BUTTON :TEXT "MOVE" :SHADOWS :ARK :TYPE :ARK :HELP "MOVE mode - selected placements
will be moved, rooms renamed" :ACTION OV-SELECT-BUTTON :OV-BUTTON :MOVE :PROTECTED? T)
(:WINDOW :COPY :TYPE :BUTTON :TEXT "COPY" :SHADOWS :ARK :TYPE :ARK :HELP "COPY mode - selected rooms &
placements will be copied" :ACTION OV-SELECT-BUTTON :OV-BUTTON :COPY :PROTECTED? T)
(:WINDOW :DELETE :TYPE :BUTTON :TEXT "DELETE" :SHADOWS :ARK :TYPE :ARK :HELP "DELETE mode - selected rooms
& placements will be deleted" :ACTION OV-SELECT-BUTTON :OV-BUTTON :DELETE :PROTECTED? T)
(:WINDOW :PROMPT-WINDOW :TYPE :PROMPT-WINDOW)
(:ROOM "Overview" :PLACEMENTS ((:PROMPT-WINDOW :REGION (0 3/16 1.0 1/16)
:BORDER 0 :SHADE 65535 :OPERATION IL:INVERT :TITLE NIL :FONT
(IL:HELVETICA 12 (IL:BOLD IL:REGULAR IL:REGULAR)))
(:SPACE-BAR :REGION (1/3 1/60 100 100))
(:EXPAND :REGION (7/8 1/60 100 100))
,@(CASE (IL:MACHINETYPE)
(IL:MAIKO '(:DELETE :REGION (26 1/60 100 100))
(:COPY :REGION (26 1/20 100 100))
(:MOVE :REGION (26 1/12 100 100))
(:EDIT :REGION (125 1/20 100 100))))
(T '(:EDIT :REGION (1/40 1/60 100 100))
(:MOVE :REGION (1/40 1/20 100 100))
(:COPY :REGION (1/40 1/12 100 100))
(:DELETE :REGION (1/40 7/60 100 100))))))
:INCLUSIONS T :BACKGROUND ((:WHOLE-SCREEN 25500 :BORDER 2)
(:REGION (0 0 1.0 3/16)
:SHADE 31710 :BORDER 2)
(:TEXT "Rooms Overview" :POSITION (0.5 . 1/8)
:ALIGNMENT :CENTER)
(:TEXT "TM" :POSITION (0.47 . 1/7)
:ALIGNMENT :CENTER :FONT (:EVAL IL:BIGFONT))
(:TEXT "Copyright (c) Envos Corporations, 1988; Patent Pending" :POSITION
(0.5 . 5/64)
:ALIGNMENT :CENTER :FONT (:EVAL IL:BIGFONT)))
:BEFORE-ENTRY-FUNCTIONS
(OV-ENTRY-FN)
:BEFORE-EXIT-FUNCTIONS
(OV-EXIT-FN)))

;; tiling code

(DEFGLOBALPARAMETER *OV-BORDER-SIZE* 10)

(DEFUN OV-LAYOUT-PLACEMENT-EDITORS (ROOMS CONTAINING-REGION)

;; layout placement editors for ROOMS in rows & columns within SCREEN-REGION, attempting to use screen space as best as possible

```
(WHEN ROOMS
  (LET* ((N-ROOMS (LENGTH ROOMS))
         (TITLE-FONT-HEIGHT (IL:FONTHEIGHT *PE-TITLE-FONT*))
         (TITLE-HEIGHT
          ;; height of title including shadows
          (+ TITLE-FONT-HEIGHT 1 (CEILING TITLE-FONT-HEIGHT *TEXT-SHADOW-FACTOR*))))
    (MULTIPLE-VALUE-BIND (ROWS COLUMNS TILE-WIDTH TILE-HEIGHT SCALE)
      (OV-ROWS&COLUMNS N-ROOMS (REGION-WIDTH CONTAINING-REGION)
        (REGION-HEIGHT CONTAINING-REGION)
        *OV-BORDER-SIZE* TITLE-HEIGHT)
      (LET* ((WIDTH (FLOOR (* IL:SCREENWIDTH SCALE)))
            (HEIGHT (FLOOR (* IL:SCREENHEIGHT SCALE)))
            (X-OFFSET (+ (REGION-LEFT CONTAINING-REGION)
                        *OV-BORDER-SIZE*
                        ;; center within borders
                        (FLOOR (- TILE-WIDTH WIDTH
                                2))))
            (Y-OFFSET (+ (REGION-BOTTOM CONTAINING-REGION)
                        *OV-BORDER-SIZE*
                        ;; center within borders
                        (FLOOR (- TILE-HEIGHT HEIGHT
                                2))))))
        (DO* ((ROOMS ROOMS (REST ROOMS))
              (ROOM (FIRST ROOMS)
                    (FIRST ROOMS))
              (COLUMN 0 (MOD (1+ COLUMN)
                             COLUMNS))
              (ROW (1- ROWS)
                   (IF (= COLUMN 0)
                       (1- ROW)
                       ROW)))
          ((ENDP ROOMS))
            (GET-PE (ROOM-NAME ROOM)
                    (MAKE-REGION :LEFT (+ (* COLUMN TILE-WIDTH)
                                           (* COLUMN *OV-BORDER-SIZE*)
                                           X-OFFSET)
                                :BOTTOM
                                (+ (* ROW TILE-HEIGHT)
                                   (* ROW TITLE-HEIGHT)
                                   Y-OFFSET)
                                :WIDTH WIDTH :HEIGHT (+ HEIGHT TITLE-HEIGHT)))))))))
```

(DEFUN **OV-ROWS&COLUMNS** (N WIDTH HEIGHT BORDER TITLE-HEIGHT)

;;; compute the optimal (in terms of use of screen space) tiling for n tiles within WIDTH & HEIGHT with the constraint that each tile must preserve the
 ;;; screen aspect ratio.

;;; returns 5 values: 1. the number of rows; 2. the number of columns; 3. the tile width; 4. the tile height and 5. the scale factor to use.

```
(LET ((ROWS 0)
      (MAX-SCALE 0)
      (TILE-WIDTH-AT-MAX-SCALE TILE-HEIGHT-AT-MAX-SCALE))
  (LOOP ;; go through each possible tiling & maximize the scale we'd have to use at that tiling. with a little algebra we could probably find a
        ;; formula which directly gave us this maximum, but this code is plenty fast & easy to understand, so why bother?
        (INCF ROWS)
        (LET* ((COLUMNS (CEILING N ROWS))
              ;; there's one more border than rows & columns, but the same number of titles as rows.
              (X-BORDERS (* (1+ COLUMNS)
                            BORDER))
              (Y-BORDERS (+ BORDER (* ROWS TITLE-HEIGHT)
                             BORDER)))
              ;; subtract off the borders from the available space
              (USEFUL-WIDTH (- WIDTH X-BORDERS))
              (USEFUL-HEIGHT (- HEIGHT Y-BORDERS))
              ;; divide up the useful space
              (TILE-WIDTH (/ USEFUL-WIDTH COLUMNS))
              (TILE-HEIGHT (/ USEFUL-HEIGHT ROWS))
              ;; calculate the scale w.r.t the screen dimensions
              (X-SCALE (/ TILE-WIDTH IL:SCREENWIDTH))
              (Y-SCALE (/ TILE-HEIGHT IL:SCREENHEIGHT))
              ;; in order to preserve aspect ratio the X & Y scales must be the same. we must chose the lesser so we stay within the tile.
              ;; we'll center within the tile when we actually lay things out.
              (SCALE (MIN X-SCALE Y-SCALE)))
          ;; scale will smoothly increase until it reaches it maximum value, then decrease. we return the previous value as soon as it
          ;; begins to decrease.
```

```
(WHEN (< SCALE MAX-SCALE)
  (RETURN (VALUES (1- ROWS)
                  (CEILING N (1- ROWS))
                  (FLOOR TILE-WIDTH-AT-MAX-SCALE)
                  (FLOOR TILE-HEIGHT-AT-MAX-SCALE)
                  MAX-SCALE)))
  (SETF MAX-SCALE SCALE)
  (SETF TILE-WIDTH-AT-MAX-SCALE TILE-WIDTH)
  (SETF TILE-HEIGHT-AT-MAX-SCALE TILE-HEIGHT))))
```

:: buttons

```
(DEFGLOBALVAR *OV-BUTTONS* (MAKE-HASH-TABLE :TEST 'EQ))
```

```
(DEFGLOBALVAR *OV-SELECTED-BUTTON-WINDOW* NIL)
```

```
(DEFGLOBALVAR *OV-CHANGED?* NIL)
```

```
(DEFUN MAKE-OV-KEYACTION-TABLE ())
```

;;; make keyaction table for overview

;; want to get users' mods to shift, ctrl & meta, but don't want users' interrupts

```
(LET ((TABLE (IL:KEYACTIONTABLE IL:\\DEFAULTKEYACTION)))
  (DECLARE (GLOBAL IL:\\DEFAULTKEYACTION))
```

;; install default interrupts

```
(IL:INTERRUPTCHAR T NIL NIL TABLE)
```

;; we need delete key & don't care about type ahead, so remove delete interrupt so screen doesn't flash

```
(IL:INTERRUPTCHAR (CHAR-CODE #\Rubout)
  NIL NIL TABLE)
TABLE))
```

```
(DEFUN OV-WATCH-KEYBOARD ())
```

;;; added as process when in overview.

(LOOP ;; watch the keyboard

```
(LET* ((KEY (COND
  ((EDIT-KEY-DOWN-P) :EDIT)
  ((MOVE-KEY-DOWN-P) :MOVE)
  ((COPY-KEY-DOWN-P) :COPY)
  ((DELETE-KEY-DOWN-P) :DELETE)
  ((EXPAND-KEY-DOWN-P) :EXPAND)
  ((IL:KEYDOWNP 'IL:SPACE) :ENTER))))
  (WHEN KEY
```

```
(LET ((BUTTON-WINDOW (GETHASH KEY *OV-BUTTONS*)))
  (UNLESS (EQ BUTTON-WINDOW *OV-SELECTED-BUTTON-WINDOW*)
```

;; select the button

```
(OV-SELECT-BUTTON BUTTON-WINDOW :REDISPLAY T))))
```

;; watch for room creation/deletion signal. this flag is set by OV-ROOM-CHANGED-FN. doing this in a separate process makes
;; DELETE-SUITE only re-layout once. this hack depends upon non-preemptive scheduling.

```
(WHEN *OV-CHANGED?*
```

;; have to re-layout placement editors

```
(SETQ *OV-CHANGED?* NIL)
```

(OV-LAYOUT-PLACEMENT-EDITORS (ALL-ROOMS T)

```
(INTERNALIZE-REGION (MAKE-REGION :LEFT 0 :BOTTOM 1/4 :WIDTH 1.0 :HEIGHT 3/4))))
```

;; clear any type ahead

```
(IL:\\CLEARSYSBUF)
```

;; don't want to cycle too fast, else chords won't be sticky

```
(IL:BLOCK 50))
```

```
(DEFUN OV-SELECT-BUTTON (WINDOW)
```

(DECLARE (IGNORE REST))

;;; called when one of the overview buttons is selected

```
(UNLESS (EQ WINDOW *OV-SELECTED-BUTTON-WINDOW*))
```

```
(WHEN *OV-SELECTED-BUTTON-WINDOW*
  ;; first unselect the previously selected button
  (OV-DESELECT-BUTTON *OV-SELECTED-BUTTON-WINDOW*))
;; mark us as the selected button
(LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON)))
  (SETQ *OV-SELECTED-BUTTON-WINDOW* WINDOW)
  (SETF (BUTTON-INVERTED? BUTTON)
    T)
  (IL:REDISPLAYW WINDOW)))
```

```
(DEFUN OV-DESELECT-BUTTON (WINDOW)
  (LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON)))
    (SETF (BUTTON-INVERTED? BUTTON)
      NIL)
    (SETQ *OV-SELECTED-BUTTON-WINDOW* NIL)
    (IL:REDISPLAYW WINDOW)))
```

```
(DEFUN OV-OPERATION ())
```

;;; call this to find out what key is down in the overview

```
(AND *OV-SELECTED-BUTTON-WINDOW* (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
  (BUTTON-PROP (IL:WINDOWPROP *OV-SELECTED-BUTTON-WINDOW* 'BUTTON)
    :OV-BUTTON))
```

```
(DEFGLOBALPARAMETER *OV-KEYACTION-TABLE*)
```

```
(DEFUN RESET-OVERVIEW ()
  (SETQ *OV-KEYACTION-TABLE* (MAKE-OV-KEYACTION-TABLE))
  (WHEN *OVERVIEW-ROOM*
    ;; clean up existing overview
    (DELETE-ROOM *OVERVIEW-ROOM*))
  (LET ((ROOM-NAMED-OVERVIEW (ROOM-NAMED "Overview")))
    ;; make an un-named room from the description in *OVERVIEW-SUITE-BODY*
    (UNWIND-PROTECT
      (PROGN (INSTALL-SUITE-BODY (COPY-TREE (OV-SUITE-BODY)))
        (SETQ *OVERVIEW-ROOM* (ROOM-NAMED "Overview")))
      (IF ROOM-NAMED-OVERVIEW
        (SETF (ROOM-NAMED "Overview")
          ROOM-NAMED-OVERVIEW)
        (REMHASH "Overview" *ROOMS*)))
      (WHEN (EQ *CURRENT-ROOM* *OVERVIEW-ROOM*)
        ;; re-tile Overview to get rid of pe for Overview
        (OV-ROOM-CHANGED-FN NIL :DELETED))
      (DOLIST (PLACEMENT (ROOM-PLACEMENTS *OVERVIEW-ROOM*))
        (LET ((BUTTON (IL:WINDOWPROP (PLACEMENT-WINDOW PLACEMENT)
          'BUTTON)))
          (WHEN BUTTON
            ;; save pointers to buttons in *OV-BUTTONS*
            (SETF (GETHASH (BUTTON-PROP BUTTON :OV-BUTTON)
              *OV-BUTTONS*)
              (PLACEMENT-WINDOW PLACEMENT))))))
    *OVERVIEW-ROOM*))
```

```
(DEFGLOBALVAR *ROOM-UNWINDERS* NIL)
```

```
(DEFMACRO ROOM-UNWIND-SAVE (&BODY BODY)
  `(PUSH #'(LAMBDA NIL ,@BODY)
    *ROOM-UNWINDERS*))
```

```
(DEFUN ROOM-UNWIND (ROOM)
  (DECLARE (IGNORE ROOM))
  (DOLIST (UNWINDER (PROG1 *ROOM-UNWINDERS* (SETQ *ROOM-UNWINDERS* NIL)))
    (FUNCALL UNWINDER)))
```

```
(EVAL-WHEN (LOAD)
```

```
(PUSHNEW ' (RESET-OVERVIEW)
  *RESET-FORMS* :TEST 'EQUAL)
```

```
(PUSHNEW ' RESET-OVERVIEW *SCREEN-CHANGED-FUNCTIONS*)
```

```
(PUSHNEW ' ROOM-UNWIND *ROOM-EXIT-FUNCTIONS*)
```

{MEDLEY}<rooms>ROOMS-OVERVIEW.;1

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

(IL:PUTPROPS **IL:ROOMS-OVERVIEW IL:COPYRIGHT** ("Venue & Xerox Corporation" 1987 1988 1990 2020))

FUNCTION INDEX

GO-TO-OVERVIEW	1	OV-LAYOUT-PLACEMENT-EDITORS	2	OV-SUITE-BODY	2
MAKE-OV-KEYACTION-TABLE	4	OV-OPERATION	5	OV-WATCH-KEYBOARD	4
OV-DESELECT-BUTTON	5	OV-ROOM-CHANGED-FN	2	RESET-OVERVIEW	5
OV-ENTRY-FN	1	OV-ROWS&COLUMNS	3	ROOM-UNWIND	5
OV-EXIT-FN	2	OV-SELECT-BUTTON	4		

VARIABLE INDEX

OV-BORDER-SIZE	2	*OV-KEYACTION-TABLE*	5	*ROOM-UNWINDERS*	5
OV-BUTTONS	4	*OV-SELECTED-BUTTON-WINDOW*	4		
OV-CHANGED?	4	*OVERVIEW-ROOM*	1		

MACRO INDEX

ROOM-UNWIND-SAVE	5
------------------------	---

FILE-ENVIRONMENT INDEX

IL:ROOMS-OVERVIEW	1
-------------------------	---
