

File created: 5-Dec-2020 16:35:32 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOM
S>MEDLEY-35>ROOMS-INTERACTIVE.;2

previous date: 17-Aug-90 12:47:35 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-I
NTERACTIVE.;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-INTERACTIVECOMS
  ;; mostly portable interactive code (joke?)
  (FILE-ENVIRONMENTS IL:ROOMS-INTERACTIVE)
  (IL:P (EXPORT '(INTERACTIVE-GO-TO-ROOM-NAMED INTERACTIVE-COPY-PLACEMENT INTERACTIVE-MOVE-PLACEMENT))
    (REQUIRE "ROOMS"))
  (IL:VARIABLES *BACKGROUND-ITEM* *MOVE-ITEM* *CLOSE-ITEM*)
  (IL:FUNCTIONS INSTALL-MENU-ITEMS INSTALL-MENU-ITEM)
  (IL:P (PUSHNEW '(INSTALL-MENU-ITEMS)
    *RESET-FORMS* :TEST 'EQUAL))
  (IL:FUNCTIONS INTERACTIVE-CLOSE-WINDOW INTERACTIVE-GO-TO-ROOM INTERACTIVE-GO-TO-OVERVIEW
    INTERACTIVE-GO-TO-ROOM-NAMED INTERACTIVE-EDIT-ROOM EDIT-ROOM INTERACTIVE-EDIT-PLACEMENTS
    INTERACTIVE-INCLUDE-ROOM INTERACTIVE-EXCLUDE-ROOM INTERACTIVE-DELETE-ROOM
    INTERACTIVE-FIND-PLACEMENT INTERACTIVE-COPY-PLACEMENT INTERACTIVE-MOVE-PLACEMENT
    INTERACTIVE-COPY-PLACEMENT-TO-THIS-ROOM INTERACTIVE-MOVE-PLACEMENT-TO-POCKETS
    INTERACTIVE-MOVE-OR-COPY-PLACEMENT INTERACTIVE-RESET SELECT-ROOM INTERACTIVE-MAKE-ROOM
    INTERACTIVE-COPY-ROOM INTERACTIVE-RENAME-ROOM INTERACTIVE-MAKE-DOOR MAKE-DOOR RETRIEVE-WINDOWS
    CHECK-LOST-WINDOWS EVAL-WALK)
  (IL:COMS
    ;; back doors
    (IL:VARIABLES *BACK-DOOR-ROOM-NAME*)
    (IL:FUNCTIONS MAKE-BACK-DOOR BACK-DOOR-ENTRY-FUNCTION)
    (IL:P (PUSHNEW 'BACK-DOOR-ENTRY-FUNCTION *ROOM-ENTRY-FUNCTIONS*)))
  (IL:GLOBALVARS IL:PROMPTWINDOW IL:CROSSHAIRS))
```

;; mostly portable interactive code (joke?)

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

```
(EXPORT '(INTERACTIVE-GO-TO-ROOM-NAMED INTERACTIVE-COPY-PLACEMENT INTERACTIVE-MOVE-PLACEMENT))
(REQUIRE "ROOMS")
```

```
(DEFGLOBALPARAMETER *BACKGROUND-ITEM*
  `("Rooms" '(WITH-BUTTON '(INTERACTIVE-GO-TO-OVERVIEW)
    "Overview" "Enter the overview")
    "Enter the overview"
    (IL:SUBITEMS ("Go to Room" '(WITH-BUTTON '(INTERACTIVE-GO-TO-ROOM :ALLOW-NEW? T)
      "Go to Room" "Go to a room, possibly new.")
      "Go to a room, possibly new.")
      ("Make Room" '(WITH-BUTTON '(INTERACTIVE-MAKE-ROOM)
        "Make Room" "Make a new room.")
        "Make a new room.")
      ("Edit Room" '(WITH-BUTTON '(INTERACTIVE-EDIT-ROOM)
        "Edit Room" "Edit a selected room.")
        "Edit a selected room."
        (IL:SUBITEMS ("Edit This Room" '(WITH-BUTTON '(EDIT-ROOM *CURRENT-ROOM*)
          "Edit This Room" "Edit the current room."
          "Edit a selected room.")
          ("Edit Placements" '(WITH-BUTTON '(INTERACTIVE-EDIT-PLACEMENTS)
            "Edit Placements" "Edit placements of a selected
            room")
            "Edit placements of a selected room")
          ("Exclude Room" '(WITH-BUTTON '(INTERACTIVE-EXCLUDE-ROOM)
            "Exclude Room" "Exclude a room from another."
            "Exclude a room from another."
            (IL:SUBITEMS ("From This Room" '(WITH-BUTTON '(INTERACTIVE-EXCLUDE-ROOM
              *CURRENT-ROOM*)
              "Exclude From This Room"
              "Exclude a room from the current
              room.")
              "Exclude a room from another.")))
          ("Include Room" '(WITH-BUTTON '(INTERACTIVE-INCLUDE-ROOM)
            "Include Room" "Include a room in another."
            "Include a room in another."
            (IL:SUBITEMS ("In This Room" '(WITH-BUTTON '(INTERACTIVE-INCLUDE-ROOM
              *CURRENT-ROOM*)
              "Include In This Room" "Include a
```

```

room in the current room.")
"Include a room in the current room.")))))
("Delete Room" '(WITH-BUTTON '(INTERACTIVE-DELETE-ROOM)
"Delete Room" "Delete a room.")
"Delete a room.")
("NIL "No-op")
("Retrieve Windows" '(WITH-BUTTON '(RETRIEVE-WINDOWS)
"Retrieve Windows" "Retrieve windows lost from all rooms.")
"Retrieve windows lost from all rooms.")
("Suites" '(WITH-BUTTON '(SUITE-MENU)
"Suites" "Save a set of rooms to a file")
"Save a set of rooms to a file"
(IL:SUBITEMS ,@*SUITE-MENU-ITEMS*))
("Make Door" '(INTERACTIVE-MAKE-DOOR :ALLOW-NEW? T)
"Make a door to a room - a button to enter it."
(IL:SUBITEMS ("Make Back Door" '(MAKE-BACK-DOOR)
"Make a back door - a door to the previous room."))))))

```

(DEFPARAMETER ***MOVE-ITEM***

```

'(IL:|Move| 'IL:MOVEW "Moves window by a corner" (IL:SUBITEMS ("Move to another room"
'INTERACTIVE-MOVE-PLACEMENT "Move this
placement to another room"
(IL:SUBITEMS ("Move to pockets"
'
INTERACTIVE-MOVE-PLACEMENT-TO-POCKETS
"Move this placement to
the pocket room")))
("Copy to another room" 'INTERACTIVE-COPY-PLACEMENT
"Copy this placement to another room"
(IL:SUBITEMS ("Copy to this room"
'
INTERACTIVE-COPY-PLACEMENT-TO-THIS-ROOM
"Copy this placement to
this room")))
("Where is?" 'INTERACTIVE-FIND-PLACEMENT "Find which
room this placement is in."))))

```

(DEFPARAMETER ***CLOSE-ITEM*** '(IL:|Close| 'INTERACTIVE-CLOSE-WINDOW "Closes a window"))

```

(DEFUN INSTALL-MENU-ITEMS ()
(INSTALL-MENU-ITEM *BACKGROUND-ITEM* 'IL:|BackgroundMenuCommands| 'IL:|BackgroundMenu|)
(INSTALL-MENU-ITEM *MOVE-ITEM* 'IL:|WindowMenuCommands| 'IL:|WindowMenu|)
(INSTALL-MENU-ITEM *MOVE-ITEM* 'IL:|IconWindowMenuCommands| 'IL:|IconWindowMenu|)
(INSTALL-MENU-ITEM *CLOSE-ITEM* 'IL:|WindowMenuCommands| 'IL:|WindowMenu|)
(INSTALL-MENU-ITEM *CLOSE-ITEM* 'IL:|IconWindowMenuCommands| 'IL:|IconWindowMenu|))

```

(DEFUN **INSTALL-MENU-ITEM** (ITEM ITEMS-VAR MENU-VAR)

```

(LET* ((ITEMS (COPY-TREE (SYMBOL-VALUE ITEMS-VAR)))
(OLD-ENTRY (ASSOC (FIRST ITEM)
ITEMS :TEST 'EQUAL)))
(IF OLD-ENTRY
(SETF (REST OLD-ENTRY)
(REST ITEM))
(NCONC ITEMS (LIST ITEM)))
(SET ITEMS-VAR ITEMS)
;; force the menu to be rebuilt
(SET MENU-VAR 'NIL)))

```

(PUSHNEW '(INSTALL-MENU-ITEMS) *RESET-FORMS* :TEST 'EQUAL)

(DEFUN **INTERACTIVE-CLOSE-WINDOW** (WINDOW &OPTIONAL (FROM-ROOM *CURRENT-ROOM*))

;;; this should probably be called interactive-delete-placement. it's whats called from the window menu & is used by the placement editor.
;;; we need to catch the case where a room has multiple placements and query the user as to which are to be deleted -- all or just the most immediate.

```

(LET ((MAIN-WINDOW (MAIN-WINDOW WINDOW))
(WINDOW-TO-CLOSE WINDOW))
(WHEN (AND (NOT (ICON? WINDOW))
(NOT (EQ WINDOW MAIN-WINDOW)))
;; it's an attached window
(LET ((PASS-TO-MAIN-COMS (IL:WINDOWPROP WINDOW 'IL:PASSTOMAINCOMS)))
;; have to simulate IL:DOATTACHEDWINDOWCOM
(UNLESS (OR (EQ PASS-TO-MAIN-COMS T)
(MEMBER 'IL:CLOSEW PASS-TO-MAIN-COMS :TEST 'EQ))
;; this window closes locally
(CLOSE-WINDOW WINDOW)

```

```

(RETURN-FROM INTERACTIVE-CLOSE-WINDOW))
(SETQ WINDOW-TO-CLOSE MAIN-WINDOW))
(LET ((ROOMS (FIND-ROOMS-CONTAINING MAIN-WINDOW)))
  ;; note: this needs to run fairly quickly, so we don't call UPDATE-PLACEMENTS.
  (IF (NULL ROOMS)
    ;; new window -- just close it
    (CLOSE-WINDOW WINDOW-TO-CLOSE)
    (CASE (IF (AND (ENDP (REST ROOMS))
                  (FIND-PLACEMENT MAIN-WINDOW FROM-ROOM))
      ;; we're looking at the only placement
      (IF (EQ FROM-ROOM (FIRST ROOMS))
        ;; it's an immediate placement - just delete it
        :ALL
        ;; it's inherited - get confirmation
        (IF (CONFIRM "This placement is in the included room ~S.~%Are you sure you want
                    to delete it?" (ROOM-NAME (FIRST ROOMS)))
          :ALL)
        (MENU ' ("All placements" :ALL)
              ("Just this placement" :THIS))
          "Delete?" "This window has placements in more than one room"))
      (:ALL (LET ((HIDDEN? (WINDOW-HIDDEN? MAIN-WINDOW)))
        ;; note whether window was hidden & make it not
        (WHEN HIDDEN? (UN-HIDE-WINDOW MAIN-WINDOW))
        ;; try to close visible part
        (CLOSE-WINDOW (IF (SHRUNKEN? MAIN-WINDOW)
                          (WINDOW-ICON MAIN-WINDOW)
                          MAIN-WINDOW))
        (IF (AND HIDDEN? (OR (IL:OPENWP MAIN-WINDOW)
                              (IL:OPENWP (WINDOW-ICON MAIN-WINDOW))))
          ;; if close failed & window was hidden before, then re-hide it
          (HIDE-WINDOW MAIN-WINDOW)
          ;; otherwise go ahead & delete all its placements
          (DOLIST (ROOM ROOMS)
            (LET ((PLACEMENT (FIND-PLACEMENT-IN-ROOM MAIN-WINDOW ROOM)))
              (WHEN PLACEMENT (DELETE-PLACEMENT PLACEMENT ROOM))))))
        (:THIS (MULTIPLE-VALUE-BIND (PLACEMENT IN-ROOM)
          (FIND-PLACEMENT MAIN-WINDOW FROM-ROOM)
          (WHEN PLACEMENT (DELETE-PLACEMENT PLACEMENT IN-ROOM)))
        ;; don't actually close -- just hide it
        (HIDE-WINDOW MAIN-WINDOW)
        (SETQ PLACEMENT (FIND-PLACEMENT MAIN-WINDOW *CURRENT-ROOM*))
        (WHEN PLACEMENT
          ;; we now inherit it from somewhere else
          (PLACE-PLACEMENT PLACEMENT))))))))))

(DEFUN INTERACTIVE-GO-TO-ROOM (&KEY ROOM ALLOW-NEW?)
  (LET ((NAME (IF ROOM
                  (ROOM-NAME ROOM)
                  (SELECT-ROOM :ALLOW-NEW? ALLOW-NEW? :REASON "Go to room" :NAME-ONLY? T))))
    (WHEN NAME
      (WITH-BUTTON '(INTERACTIVE-GO-TO-ROOM-NAMED ' ,NAME)
        NAME
        (FORMAT NIL "Go to room named ~S." NAME))))))

(DEFUN INTERACTIVE-GO-TO-OVERVIEW ()
  (UPDATE-PLACEMENTS)
  (GO-TO-ROOM *OVERVIEW-ROOM* :BAGGAGE (SELECT-BAGGAGE)
    :NO-UPDATE T))

(DEFUN INTERACTIVE-GO-TO-ROOM-NAMED (NAME)
  (LET ((ROOM (ROOM-NAMED NAME)))
    (IF ROOM
      (PROGN (UPDATE-PLACEMENTS *CURRENT-ROOM*)
        (GO-TO-ROOM ROOM :BAGGAGE (SELECT-BAGGAGE)
          :NO-UPDATE T))
      (NOTIFY-USER "No room named ~S exists!" NAME))))

(DEFUN INTERACTIVE-EDIT-ROOM ()
  (LET ((NAME (SELECT-ROOM :REASON "Edit" :NAME-ONLY? T)))
    (WHEN NAME
      (WITH-BUTTON '(EDIT-ROOM (ROOM-NAMED ' ,NAME))
        (FORMAT NIL "Edit ~A" NAME)

```

(FORMAT NIL "Edit room named ~S." NAME))))

(DEFUN EDIT-ROOM (ROOM)

(LET* ((ROOM (COND
(AND (ROOM-P ROOM)
(ROOM-NAMED (ROOM-NAME ROOM)))
ROOM)
(ROOM-NAMED ROOM))
(T (NOTIFY-USER "Can't edit room ~S" ROOM)
(RETURN-FROM EDIT-ROOM))))
(EXTERNAL-FORM `(:INCLUSIONS ,(COPY-TREE (ROOM-INCLUSIONS ROOM))
:BACKGROUND
,(COPY-TREE (BACKGROUND-EXTERNAL-FORM (ROOM-BACKGROUND ROOM)))
,@(COPY-TREE (ROOM-PROPS ROOM))))))
(WITH-PROFILE (FIND-PROFILE "XCL")
(IL:EDITE EXTERNAL-FORM NIL (ROOM-NAME ROOM)
'IL:|Expression|
#' (LAMBDA (&REST IGNORE)
;; in case ROOM has been redefined
(SETQ ROOM (ROOM-NAMED (ROOM-NAME ROOM)))
(SETF (ROOM-BACKGROUND ROOM)
(MAKE-BACKGROUND (COPY-TREE (GETF EXTERNAL-FORM :BACKGROUND))))
(WHEN (IN-ROOM? ROOM)
(UPDATE-PLACEMENTS))
(SETF (ROOM-INCLUSIONS ROOM)
(COPY-TREE (GETF EXTERNAL-FORM :INCLUSIONS)))
(LET ((PROPS (COPY-LIST EXTERNAL-FORM))
(DOLIST (PROP '(:INCLUSIONS :BACKGROUND))
(REMF PROPS PROP))
(SETF (ROOM-PROPS ROOM)
(COPY-TREE PROPS)))
(ROOM-CHANGED ROOM :EDITED))
'(:DONTWAIT))))))

(DEFUN INTERACTIVE-EDIT-PLACEMENTS ()

(LET ((NAME (SELECT-ROOM :REASON "Edit Placements" :NAME-ONLY? T)))
(WHEN NAME
(WITH-BUTTON `(GET-PE ',NAME)
(FORMAT NIL "Edit ~A's Placements" NAME)
(FORMAT NIL "Edit the placements of ~S." NAME))))))

(DEFUN INTERACTIVE-INCLUDE-ROOM (&OPTIONAL IN-ROOM)

(LET* ((ALL-ROOMS (ALL-ROOMS T))
(ROOM (OR IN-ROOM (SELECT-ROOM :ALLOW-NEW? T :REASON "Include in ..." :FROM-ROOMS ALL-ROOMS))))
(WHEN ROOM
(UNLESS (LISTP (ROOM-INCLUSIONS ROOM))
(RETURN-FROM INTERACTIVE-INCLUDE-ROOM (NOTIFY-USER "Can't add inclusions to ~S." ROOM)))
(LET ((INCLUSION (SELECT-ROOM :ALLOW-NEW? T :REASON (FORMAT NIL "Include in ~A" (ROOM-NAME ROOM))
:FROM-ROOMS
(REMOVE ROOM ALL-ROOMS))))
(WHEN INCLUSION
(WHEN (MEMBER (ROOM-NAME INCLUSION)
(ROOM-INCLUSIONS ROOM)
:TEST
'EQUAL)
(RETURN-FROM INTERACTIVE-INCLUDE-ROOM (NOTIFY-USER "~S is already included in ~S"
(ROOM-NAME INCLUSION)
(ROOM-NAME ROOM))))
(UPDATE-PLACEMENTS)
(WHEN (AND (EQUAL (BACKGROUND-EXTERNAL-FORM (ROOM-BACKGROUND INCLUSION))
`(:TEXT ,(ROOM-NAME INCLUSION)))
(EQUAL (BACKGROUND-EXTERNAL-FORM (ROOM-BACKGROUND ROOM))
`(:TEXT ,(ROOM-NAME ROOM))))
;; feature: when both names are in default position we delete name of included room s.t. they don't overwrite.
(SETF (ROOM-BACKGROUND INCLUSION)
(MAKE-BACKGROUND `(:TEXT ,")))
(ROOM-CHANGED INCLUSION :EDITED))
(PUSH (ROOM-NAME INCLUSION)
(ROOM-INCLUSIONS ROOM))
(ROOM-CHANGED ROOM :EDITED)
(NOTIFY-USER "Included ~S in ~S." (ROOM-NAME INCLUSION)
(ROOM-NAME ROOM)
T))))))

(DEFUN INTERACTIVE-EXCLUDE-ROOM (&OPTIONAL FROM-ROOM)

(LET ((ROOM (OR FROM-ROOM (SELECT-ROOM :REASON "Exclude from ..."))))
(WHEN ROOM
(UNLESS (CONSP (ROOM-INCLUSIONS ROOM))
(RETURN-FROM INTERACTIVE-EXCLUDE-ROOM (NOTIFY-USER "~S has no inclusions." ROOM)))
(LET ((INCLUSION (MENU (ROOM-INCLUSIONS ROOM)
(FORMAT NIL "Exclude from ~A" (ROOM-NAME ROOM))))))

```
(WHEN INCLUSION
  (UPDATE-PLACEMENTS)
  (SETF (ROOM-INCLUSIONS ROOM)
        (REMOVE INCLUSION (ROOM-INCLUSIONS ROOM :TEST 'EQUAL)))
  (ROOM-CHANGED ROOM :EDITED)
  (NOTIFY-USER "~S is no longer included in ~S." INCLUSION (ROOM-NAME ROOM)
  T))))
```

```
(DEFUN INTERACTIVE-DELETE-ROOM (&OPTIONAL ROOM)
  (FLET ((DELETE? (ROOM)
          (WHEN (AND ROOM (CONFIRM " Delete room ~S? (will close windows)" (ROOM-NAME ROOM)))
            (DELETE-ROOM ROOM))))
    (LET ((ROOMS (ROOMS-NOT-IN-ANY-SUITE T))
          (IF ROOM
            (IF (MEMBER ROOM ROOMS :TEST 'EQ)
                (DELETE? ROOM)
                (NOTIFY-USER "Delete ~S from suite ~S before deleting" (ROOM-NAME ROOM)
                             (FIND-SUITE-CONTAINING (ROOM-NAME ROOM))))
            (IF ROOMS
                (DELETE? (SELECT-ROOM :REASON "Delete" :FROM-ROOMS ROOMS))
                (NOTIFY-USER "All rooms belong to some suite."))))))
```

```
(DEFUN INTERACTIVE-FIND-PLACEMENT (WINDOW)
  (LET ((WINDOW (MAIN-WINDOW WINDOW))
        (UPDATE-PLACEMENTS)
        (NOTIFY-USER "This placement is in ~S." (ROOM-NAME (MULTIPLE-VALUE-BIND (PLACEMENT ROOM)
                                                                                (FIND-PLACEMENT WINDOW)
                                                                                ROOM))))))
```

```
(DEFUN INTERACTIVE-COPY-PLACEMENT (WINDOW &OPTIONAL ROOM-NAME)
  (UN-HIDE-WINDOW WINDOW)
  (LET ((NAME (OR ROOM-NAME (SELECT-ROOM :REASON "Copy this placement to" :ALLOW-NEW? T :NAME-ONLY? T)))
        (WHEN NAME (INTERACTIVE-MOVE-OR-COPY-PLACEMENT WINDOW NAME T))))
```

```
(DEFUN INTERACTIVE-MOVE-PLACEMENT (WINDOW &OPTIONAL ROOM-NAME)
  (UN-HIDE-WINDOW WINDOW)
  (LET ((NAME (OR ROOM-NAME (SELECT-ROOM :REASON "Move this placement to" :ALLOW-NEW? T :NAME-ONLY? T)))
        (WHEN NAME (INTERACTIVE-MOVE-OR-COPY-PLACEMENT WINDOW NAME NIL))))
```

```
(DEFUN INTERACTIVE-COPY-PLACEMENT-TO-THIS-ROOM (WINDOW)
  (INTERACTIVE-MOVE-OR-COPY-PLACEMENT WINDOW (ROOM-NAME *CURRENT-ROOM*
  T))
```

```
(DEFUN INTERACTIVE-MOVE-PLACEMENT-TO-POCKETS (WINDOW)
  (IF *POCKET-ROOM-NAME*
      (INTERACTIVE-MOVE-OR-COPY-PLACEMENT WINDOW *POCKET-ROOM-NAME* NIL)
      (NOTIFY-USER "There is no pocket room.")))
```

```
(DEFUN INTERACTIVE-MOVE-OR-COPY-PLACEMENT (WINDOW TO-ROOM-NAMED COPY?)
  (LET ((WINDOW (MAIN-WINDOW WINDOW))
        (TO-ROOM (OR (ROOM-NAMED TO-ROOM-NAMED)
                     (PROGN (NOTIFY-USER "There is no room named ~S." TO-ROOM-NAMED)
                             NIL))))
    (WHEN TO-ROOM
      (UPDATE-PLACEMENTS)
      (MULTIPLE-VALUE-BIND (PLACEMENT FROM-ROOM)
                          (FIND-PLACEMENT WINDOW)
                          (COND
                           ((EQ FROM-ROOM TO-ROOM)
                            (NOTIFY-USER "This placement is already in ~S." (ROOM-NAME FROM-ROOM))
                            :NOOP)
                           (T (MOVE-PLACEMENT PLACEMENT FROM-ROOM TO-ROOM COPY?)
                              (NOTIFY-USER "~A this placement from ~S to ~S." (IF COPY?
                                                                                "Copied"
                                                                                "Moved")
                                           (ROOM-NAME FROM-ROOM)
                                           TO-ROOM-NAMED)
                              T))))))
```

```
(DEFUN INTERACTIVE-RESET ()
  (WHEN (CONFIRM "Reset Rooms? (Will lose windows.)")
        (RESET)))
```

```
(DEFUN SELECT-ROOM (&KEY ALLOW-NEW? NAME-ONLY? (FROM-ROOMS (ALL-ROOMS T))
                  (REASON "Select Room"))
  (LET ((ITEMS (WITH-COLLECTION (DOLIST (ROOM FROM-ROOMS)
                                       (COLLECT ` (, (ROOM-NAME ROOM)
                                                ', ROOM)
                                       T))))
```

```

                ITEMS))
            (WHEN ALLOW-NEW?
              (COLLECT '("<new room>" :NEW))))))
(IF ITEMS
  (LET* ((CHOICE (MENU ITEMS REASON))
         (ROOM (IF (AND ALLOW-NEW? (EQ CHOICE :NEW))
                   (INTERACTIVE-MAKE-ROOM)
                   (CHOICE))))
        (WHEN ROOM
          (IF NAME-ONLY?
              (ROOM-NAME ROOM)
              (ROOM)))
        (PROGN (NOTIFY-USER "No rooms!")
                NIL))))

```

```

(DEFUN INTERACTIVE-MAKE-ROOM ()
  (LET ((NAME (PROMPT-USER "Name:" "Type name of new room (CR to abort).")))
    (WHEN NAME
      (IF (ROOM-NAMED NAME)
          (NOTIFY-USER "A room named ~S already exists. Aborted." NAME)
          (MAKE-ROOM NAME))))))

```

```

(DEFUN INTERACTIVE-COPY-ROOM (&OPTIONAL ROOM)
  (LET ((ROOM (OR ROOM (SELECT-ROOM :REASON "Copy"))))
    (WHEN ROOM
      (LET ((NAME (PROMPT-USER "New Name:" "Copying room ~S." (ROOM-NAME ROOM))))
        (WHEN NAME
          (IF (ROOM-NAMED NAME)
              (NOTIFY-USER "A room named ~S already exists." NAME)
              (PROGN (COPY-ROOM ROOM NAME)
                     (NOTIFY-USER "Copied room ~S to ~S." (ROOM-NAME ROOM)
                                   NAME))))))))))

```

```

(DEFUN INTERACTIVE-RENAME-ROOM (&OPTIONAL ROOM)
  (LET ((ROOM (OR ROOM (SELECT-ROOM :REASON "Rename"))))
    (WHEN ROOM
      (LET ((NAME (PROMPT-USER "New Name:" "Renaming room ~S." (ROOM-NAME ROOM))))
        (WHEN NAME
          (IF (ROOM-NAMED NAME)
              (NOTIFY-USER "A room named ~S already exists." NAME)
              (PROGN (RENAME-ROOM ROOM NAME)
                     (NOTIFY-USER "Renamed room ~S to be ~S." (ROOM-NAME ROOM)
                                   NAME))))))))))

```

```

(DEFUN INTERACTIVE-MAKE-DOOR (&KEY ALLOW-NEW?)
  (LET ((NAME (SELECT-ROOM :NAME-ONLY? T :ALLOW-NEW? ALLOW-NEW?)))
    (WHEN NAME
      (LET ((BUTTON-TYPE (SELECT-BUTTON-TYPE)))
        (WHEN BUTTON-TYPE (MAKE-DOOR :ROOM-NAME NAME :BUTTON-TYPE BUTTON-TYPE))))))

```

```

(DEFUN MAKE-DOOR (&KEY ROOM-NAME (BUTTON-TYPE *DEFAULT-BUTTON-TYPE*)
                 POSITION)
  (MAKE-BUTTON-WINDOW (MAKE-BUTTON :TEXT ROOM-NAME :ACTION \
    (INTERACTIVE-GO-TO-ROOM-NAMED
      , (IF (CONSTANTP ROOM-NAME)
            ROOM-NAME
            (LIST 'QUOTE ROOM-NAME))
      :HELP
      (FORMAT NIL "Go to room named ~S" ROOM-NAME)
      :TYPE BUTTON-TYPE)
    POSITION))

```

```

(DEFUN RETRIEVE-WINDOWS ()

```

;;; un-hide all lost windows, telling the user what you've done.

```

  (LET ((LOST-WINDOWS (LOST-WINDOWS)))
    (IF LOST-WINDOWS
      (PROGN (DOLIST (WINDOW LOST-WINDOWS)
                   (UN-HIDE-WINDOW WINDOW))
             (NOTIFY-USER "~S window(s) retrieved." (LENGTH LOST-WINDOWS)))
      (NOTIFY-USER "All windows are in some room.))))

```

```

(DEFUN CHECK-LOST-WINDOWS ()
  (LET ((LOST-WINDOWS (LOST-WINDOWS)))
    (WHEN LOST-WINDOWS
      (NOTIFY-USER "~D lost window(s). Try \"Retrieve Windows\"." (LENGTH LOST-WINDOWS))))))

```

```

(DEFUN EVAL-WALK (EXPRESSION)

```

;; an inverted evaluator: expressions are implicitly quoted unless wrapped in :EVAL. Only conses when it must, i.e. structure w/o EVALs in it will be

:: shared.

```
(IF (CONSP EXPRESSION)
  (IF (AND (CONSP (FIRST EXPRESSION))
    (EQ (FIRST (FIRST EXPRESSION))
      :EVAL))
    (CONS (EVAL (SECOND (FIRST EXPRESSION)))
      (EVAL-WALK (REST EXPRESSION)))
    (LET* ((OLD-FIRST (FIRST EXPRESSION))
      (OLD-REST (REST EXPRESSION))
      (NEW-FIRST (EVAL-WALK OLD-FIRST))
      (NEW-REST (EVAL-WALK OLD-REST)))
      (IF (AND (EQ OLD-FIRST NEW-FIRST)
        (EQ OLD-REST NEW-REST))
        EXPRESSION
        (CONS NEW-FIRST NEW-REST))))
    EXPRESSION))
```

:: back doors

```
(DEFGLOBALVAR *BACK-DOOR-ROOM-NAME* NIL)
```

```
(DEFUN MAKE-BACK-DOOR (&KEY POSITION BUTTON-TYPE)
  (MAKE-BUTTON-WINDOW (MAKE-BUTTON :TEXT-FORM ' (SYMBOL-VALUE ' *BACK-DOOR-ROOM-NAME*)
    :ACTION
      ' (INTERACTIVE-GO-TO-ROOM-NAMED *BACK-DOOR-ROOM-NAME*)
    :TYPE
      (OR BUTTON-TYPE :DOOR)
    :HELP "Go to the previous room." :INVERTED? T)
    POSITION))
```

```
(DEFUN BACK-DOOR-ENTRY-FUNCTION (ENTERING-ROOM)
```

;;; called whenever we enter a room

;;; maintains the value of *BACK-DOOR-ROOM-NAME* to be the name of the last named room we were in before the current room.

```
(LET* ((LEAVING-ROOM *CURRENT-ROOM*)
  (LEAVING-NAME (ROOM-NAME LEAVING-ROOM))
  (ENTERING-NAME (ROOM-NAME ENTERING-ROOM)))
  (UNLESS *BACK-DOOR-ROOM-NAME*
    ;; bootstrapping
    (SETQ *BACK-DOOR-ROOM-NAME* LEAVING-NAME))
  (WHEN (NOT (EQUAL ENTERING-NAME LEAVING-NAME))
    ;; ignore screen refreshes
    (IF (ROOM-NAMED LEAVING-NAME)
      (IF (ROOM-NAMED ENTERING-NAME)
        ;; simple case - going between named rooms
        (SETQ *BACK-DOOR-ROOM-NAME* LEAVING-NAME)
        (PROGN ;; when entering an un-named room from a named room we save the current back door on the room we're entering
          ;; & update the global back door
          (ROOM-PROP ENTERING-ROOM :BACK-DOOR *BACK-DOOR-ROOM-NAME*)
          (SETQ *BACK-DOOR-ROOM-NAME* LEAVING-NAME)))
      (IF (ROOM-NAMED ENTERING-NAME)
        ;; entering a named room from an unnamed one
        (WHEN (EQUAL *BACK-DOOR-ROOM-NAME* ENTERING-NAME)
          ;; if popping back to room we came from then restore back door we saved upon entering. global will be correct, making
          ;; passage through un-named rooms transparent.
          (SETQ *BACK-DOOR-ROOM-NAME* (ROOM-PROP LEAVING-ROOM :BACK-DOOR)))
        ;; going between un-named rooms we just pass along the saved back door, & don't update the global
        (ROOM-PROP ENTERING-ROOM :BACK-DOOR (ROOM-PROP LEAVING-ROOM :BACK-DOOR))))))
```

```
(PUSHNEW 'BACK-DOOR-ENTRY-FUNCTION *ROOM-ENTRY-FUNCTIONS*)
```

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

```
(IL:GLOBALVARS IL:PROMPTWINDOW IL:CROSSHAIRS)
)
```

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

FUNCTION INDEX

BACK-DOOR-ENTRY-FUNCTION	7	INTERACTIVE-GO-TO-OVERVIEW	3
CHECK-LOST-WINDOWS	6	INTERACTIVE-GO-TO-ROOM	3
EDIT-ROOM	4	INTERACTIVE-GO-TO-ROOM-NAMED	3
EVAL-WALK	6	INTERACTIVE-INCLUDE-ROOM	4
INSTALL-MENU-ITEM	2	INTERACTIVE-MAKE-DOOR	6
INSTALL-MENU-ITEMS	2	INTERACTIVE-MAKE-ROOM	6
INTERACTIVE-CLOSE-WINDOW	2	INTERACTIVE-MOVE-OR-COPY-PLACEMENT	5
INTERACTIVE-COPY-PLACEMENT	5	INTERACTIVE-MOVE-PLACEMENT	5
INTERACTIVE-COPY-PLACEMENT-TO-THIS-ROOM	5	INTERACTIVE-MOVE-PLACEMENT-TO-POCKETS	5
INTERACTIVE-COPY-ROOM	6	INTERACTIVE-RENAME-ROOM	6
INTERACTIVE-DELETE-ROOM	5	INTERACTIVE-RESET	5
INTERACTIVE-EDIT-PLACEMENTS	4	MAKE-BACK-DOOR	7
INTERACTIVE-EDIT-ROOM	3	MAKE-DOOR	6
INTERACTIVE-EXCLUDE-ROOM	4	RETRIEVE-WINDOWS	6
INTERACTIVE-FIND-PLACEMENT	5	SELECT-ROOM	5

VARIABLE INDEX

BACK-DOOR-ROOM-NAME ...	7	*BACKGROUND-ITEM*	1	*CLOSE-ITEM*	2	*MOVE-ITEM*	2
---------------------------	---	-------------------------	---	--------------------	---	-------------------	---

FILE-ENVIRONMENT INDEX

IL:ROOMS-INTERACTIVE	1
----------------------------	---
