

File created: 5-Dec-2020 16:27:41 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOM  
S>MEDLEY-35>ROOMS-TEXT.;2

previous date: 17-Aug-90 13:31:54 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-T  
EXT.;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-TEXTCOMS
  ((FILE-ENVIRONMENTS IL:ROOMS-TEXT)
  (IL:P (EXPORT ' (*DEFAULT-TEXT-FONT* MAKE-TEXT DISPLAY-TEXT DEF-TEXT-SHADOWS SET-TEXT-STRING)))
  (IL:STRUCTURES TEXT TEXT-SHADOW)
  (IL:VARIABLES *DEFAULT-TEXT-FONT* DEFAULT-TEXT-FONT--SMALL-SCREEN DEFAULT-TEXT-FONT--LARGE-SCREEN
    SMALL-SCREEN-WIDTH)
  (IL:FUNCTIONS MAKE-TEXT UPDATE-TEXT-CACHES COMPUTE-TEXT-DIMENSIONS MAXIMIZE MINIMIZE DISPLAY-TEXT
    SET-TEXT-STRING SET-DEFAULT-TEXT-FONT)
  (IL:P (SET-DEFAULT-TEXT-FONT))
  (IL:FUNCTIONS
    ;; for back compatibility: buttons & pe's still call these two
    TEXT-%WIDTH TEXT-%HEIGHT)
  (IL:VARIABLES *TEXT-SHADOWS* *TEXT-SHADOW-FACTOR*)
  (IL:FUNCTIONS GET-TEXT-SHADOWS GET-TEXT-SHADOWS-INTERNAL MAKE-TEXT-SHADOWS EXTERNALIZE-TEXT-SHADOWS
    INTERNALIZE-TEXT-SHADOWS INTERNALIZE-TEXT-SHADOWS-INTERNAL)
  ;; a definer for shadows
  (IL:DEFINE-TYPES IL:TEXT-SHADOWS)
  (IL:FUNCTIONS DEF-TEXT-SHADOWS)
  (IL:TEXT-SHADOWS NIL :ARK)))
```

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

```
(EXPORT ' (*DEFAULT-TEXT-FONT* MAKE-TEXT DISPLAY-TEXT DEF-TEXT-SHADOWS SET-TEXT-STRING))
```

```
(DEFSTRUCT
```

```
;;; specifies a bit of text for display
```

```
(TEXT (:CONSTRUCTOR MAKE-TEXT-INTERNAL)
  (:PRINT-FUNCTION (LAMBDA (TEXT STREAM DEPTH)
    (FORMAT STREAM "#<Text ~S>" (TEXT-STRING TEXT))))))
```

```
(STRING "" :TYPE STRING)
```

```
;; the text to print. use SET-TEXT-STRING to change this field.
```

```
(POSITION (MAKE-POSITION 0 0)
  :TYPE CONS)
```

```
;; where to print it
```

```
(ALIGNMENT :LEFT-BOTTOM :TYPE
```

```
;; how to align it
```

```
(MEMBER :LEFT-BOTTOM :LEFT-TOP :CENTER :RIGHT-BOTTOM :RIGHT-TOP))
```

```
;; how to align it relative to POSITION
```

```
(FONT *DEFAULT-TEXT-FONT*)
```

```
;; font to use
```

```
(SHADOWS (MAKE-TEXT-SHADOWS *DEFAULT-TEXT-FONT*)
  :TYPE LIST)
```

```
;; a list of TEXT-SHADOW structures
```

```
;;; caches to speed redisplay
```

```
(%IMAGE NIL :TYPE BITMAP)
(%MASK NIL :TYPE BITMAP))
```

```
(DEFSTRUCT TEXT-SHADOW
```

```
;;; a specification of a call to IL:BITBLT. a list of these is used to achieve special effects when displaying text. the most common effect is that of  
;;; shadowed text, hence the name TEXT-SHADOW.
```

```
;; offset for this BLT
```

```
(DX 0 :TYPE INTEGER)
(DY 0 :TYPE INTEGER)
```

```
;; args to IL:BITBLT
```

(SOURCE-TYPE 'IL:INPUT :TYPE (MEMBER IL:INPUT IL:INVERT IL:TEXTURE IL:MERGE))
(OPERATION 'IL:PAINT :TYPE (MEMBER IL:PAINT IL:REPLACE IL:ERASE IL:INVERT))
(TEXTURE 0 :TYPE TEXTURE))

(DEFVAR \*DEFAULT-TEXT-FONT\* NIL)

(DEFGLOBALPARAMETER DEFAULT-TEXT-FONT--SMALL-SCREEN (IL:FONTCREATE 'IL:HELVETICA 10 'IL:BOLD))

(DEFGLOBALPARAMETER DEFAULT-TEXT-FONT--LARGE-SCREEN (IL:FONTCREATE 'IL:HELVETICA 18 'IL:BOLD))

(DEFGLOBALPARAMETER SMALL-SCREEN-WIDTH 1400)

(DEFUN MAKE-TEXT (&KEY STRING (POSITION (MAKE-POSITION 0 0))
(ALIGNMENT :LEFT-BOTTOM)
(FONT \*DEFAULT-TEXT-FONT\*)
SHADOWS)

:: check args

(UNLESS (IL:POSITIONP POSITION)
(ERROR "~S not a position" POSITION))
(ECASE ALIGNMENT
((:LEFT-BOTTOM :LEFT-TOP :CENTER :RIGHT-BOTTOM :RIGHT-TOP) ))
(CHECK-TYPE FONT FONT)
(LET ((TEXT (MAKE-TEXT-INTERNAL :STRING STRING :POSITION POSITION :ALIGNMENT ALIGNMENT :FONT FONT :SHADOWS
(INTERNALIZE-TEXT-SHADOWS SHADOWS))))

:: fill in the caches

(UPDATE-TEXT-CACHES TEXT)
TEXT))

(DEFUN UPDATE-TEXT-CACHES (TEXT)

(LET\* ((FONT (TEXT-FONT TEXT))
(STRING-WIDTH (IL:STRINGWIDTH (TEXT-STRING TEXT)
FONT))
(FONT-HEIGHT (IL:FONTHEIGHT FONT))
(TEMP-BITMAP (IL:BITMAPCREATE STRING-WIDTH FONT-HEIGHT)))
(LET ((DSP (IL:LOADTIMECONSTANT (IL:DSPCREATE))))

:: first put string into a temporary bitmap

(IL:DSPDESTINATION TEMP-BITMAP DSP)
(IL:DSPFONT FONT DSP)
(IL:MOVETO 0 (IL:FONTDESCENT FONT)
DSP)
(PRINC (TEXT-STRING TEXT)
DSP))

(MULTIPLE-VALUE-BIND (WIDTH HEIGHT X-OFFSET Y-OFFSET)

(COMPUTE-TEXT-DIMENSIONS TEXT STRING-WIDTH)

(LET\* ((OLD-IMAGE (TEXT-%IMAGE TEXT))
(IMAGE (IF (AND OLD-IMAGE (= HEIGHT (IL:BITMAPHEIGHT OLD-IMAGE))
(= WIDTH (IL:BITMAPWIDTH OLD-IMAGE))))

:: OK to re-use bitmap

(PROGN (IL:BLTSHADE IL:WHITESHAE OLD-IMAGE 0 0 WIDTH HEIGHT)
OLD-IMAGE)
(IL:BITMAPCREATE WIDTH HEIGHT)))

(SHADOWS (GET-TEXT-SHADOWS TEXT))

(OLD-MASK (TEXT-%MASK TEXT))

(MASK (WHEN (CDR SHADOWS) ; don't need mask for simple shadows

(IF (AND OLD-MASK (= HEIGHT (IL:BITMAPHEIGHT OLD-MASK))
(= WIDTH (IL:BITMAPWIDTH OLD-MASK))))

:: OK to re-use bitmap

(PROGN (IL:BLTSHADE IL:WHITESHAE OLD-MASK 0 0 WIDTH HEIGHT)
OLD-MASK)
(IL:BITMAPCREATE WIDTH HEIGHT))))

(DOLIST (SHADOW (GET-TEXT-SHADOWS TEXT))

(IL:BITBLT TEMP-BITMAP 0 0 IMAGE (+ (TEXT-SHADOW-DX SHADOW)
X-OFFSET)

(+ (TEXT-SHADOW-DY SHADOW)
Y-OFFSET)

STRING-WIDTH FONT-HEIGHT (TEXT-SHADOW-SOURCE-TYPE SHADOW)

(TEXT-SHADOW-OPERATION SHADOW)

(TEXT-SHADOW-TEXTURE SHADOW))

(WHEN MASK

(IL:BITBLT TEMP-BITMAP 0 0 MASK (+ X-OFFSET (TEXT-SHADOW-DX SHADOW))
(+ Y-OFFSET (TEXT-SHADOW-DY SHADOW))

STRING-WIDTH FONT-HEIGHT 'IL:SOURCE 'IL:PAINT)))

(SETF (TEXT-%IMAGE TEXT)

IMAGE)

(SETF (TEXT-%MASK TEXT)

MASK)

IMAGE)))

(DEFUN COMPUTE-TEXT-DIMENSIONS (TEXT STRING-WIDTH)

::: compute & return width, height & offsets of TEXT, taking shadows into consideration.

```
(LET* ((SHADOWS (GET-TEXT-SHADOWS TEXT))
      (MAX-DX (MAXIMIZE (SHADOW SHADOWS)
                        (TEXT-SHADOW-DX SHADOW)))
      (MIN-DX (MINIMIZE (SHADOW SHADOWS)
                        (TEXT-SHADOW-DX SHADOW)))
      (MAX-DY (MAXIMIZE (SHADOW SHADOWS)
                        (TEXT-SHADOW-DY SHADOW)))
      (MIN-DY (MINIMIZE (SHADOW SHADOWS)
                        (TEXT-SHADOW-DY SHADOW))))
      (VALUES
        ;; width
        (+ STRING-WIDTH MAX-DX (- MIN-DX))
        ;; height
        (+ (IL:FONTHEIGHT (TEXT-FONT TEXT))
           MAX-DY
           (- MIN-DY))
        ;; x-offset
        (- MIN-DX)
        ;; y-offset
        (- MIN-DY))))
```

(DEFMACRO MAXIMIZE ((VAR LIST) (FORM))

```
`(LET ((SI::$MAX-VALUE$ NIL)
      (SI::$VALUE$ NIL))
  (DOLIST (,VAR ,LIST SI::$MAX-VALUE$)
    (SETQ SI::$VALUE$ ,FORM)
    (UNLESS (AND SI::$MAX-VALUE$ (> SI::$MAX-VALUE$ SI::$VALUE$))
      (SETQ SI::$MAX-VALUE$ SI::$VALUE$))))
```

(DEFMACRO MINIMIZE ((VAR LIST) (FORM))

```
`(LET* ((SI::$MIN-VALUE$ NIL)
      (SI::$VALUE$ NIL))
  (DOLIST (,VAR ,LIST SI::$MIN-VALUE$)
    (SETQ SI::$VALUE$ ,FORM)
    (UNLESS (AND SI::$MIN-VALUE$ (< SI::$MIN-VALUE$ SI::$VALUE$))
      (SETQ SI::$MIN-VALUE$ SI::$VALUE$))))
```

(DEFUN DISPLAY-TEXT (TEXT DESTINATION &KEY SCALE MASK-ONLY)

::: print TEXT, a TEXT structure, to DESTINATION, a valid destination for IL:BITBLT.

```
(LET* ((POSITION (TEXT-POSITION TEXT))
      (ALIGNMENT (TEXT-ALIGNMENT TEXT))
      (IMAGE (TEXT-%IMAGE TEXT))
      (WIDTH (IL:BITMAPWIDTH IMAGE))
      (HEIGHT (IL:BITMAPHEIGHT IMAGE))
      (SCALED-X (IF SCALE
                    (SCALE-X (POSITION-X POSITION)
                              SCALE)
                    (POSITION-X POSITION)))
      (SCALED-Y (IF SCALE
                    (SCALE-Y (POSITION-Y POSITION)
                              SCALE)
                    (POSITION-Y POSITION)))
      (X-COORD (ECASE ALIGNMENT
                 ((:LEFT-BOTTOM :LEFT-TOP) SCALED-X)
                 ((:RIGHT-BOTTOM :RIGHT-TOP) (- SCALED-X WIDTH))
                 (:CENTER (- SCALED-X (FLOOR WIDTH 2)))))
      (Y-COORD (CASE ALIGNMENT
                ((:LEFT-BOTTOM :RIGHT-BOTTOM) SCALED-Y)
                ((:LEFT-TOP :RIGHT-TOP) (- SCALED-Y HEIGHT))
                (:CENTER (- SCALED-Y (FLOOR HEIGHT 2)))))
      (MASK (TEXT-%MASK TEXT)))
  (WHEN MASK ; erase the mask
    (IL:BITBLT MASK 0 0 DESTINATION X-COORD Y-COORD WIDTH HEIGHT 'IL:INPUT (IF MASK-ONLY
                                          'IL:PAINT
                                          'IL:ERASE))))
  (UNLESS MASK-ONLY ; paint in the image
    (IL:BITBLT IMAGE 0 0 DESTINATION X-COORD Y-COORD WIDTH HEIGHT 'IL:INPUT 'IL:PAINT))))
```

```
{MEDLEY}<rooms>ROOMS-TEXT.;1
```

```
(DEFUN SET-TEXT-STRING (TEXT STRING)
```

```
;; call this to change the string of a TEXT object
```

```
(SETF (TEXT-STRING TEXT)
      STRING)
```

```
;; update all caches
```

```
(UPDATE-TEXT-CACHES TEXT)
```

```
;; return the string
```

```
STRING)
```

```
(DEFUN SET-DEFAULT-TEXT-FONT ()
```

```
;; called when screen size changes
```

```
(FLET ((DEFAULT-FONT (SCREEN-WIDTH)
                    (IF (> SCREEN-WIDTH SMALL-SCREEN-WIDTH)
                        DEFAULT-TEXT-FONT--LARGE-SCREEN
                        DEFAULT-TEXT-FONT--SMALL-SCREEN)))
```

```
;; if user hasn't changed *DEFAULT-TEXT-FONT* then set it proportional to the screen size.
```

```
(IF (OR (NULL *DEFAULT-TEXT-FONT*)
        (EQ (DEFAULT-FONT (REGION-WIDTH OLD-WHOLESIZE))
            *DEFAULT-TEXT-FONT*))
    (SETQ *DEFAULT-TEXT-FONT* (DEFAULT-FONT IL:SCREENWIDTH))
    *DEFAULT-TEXT-FONT*))
```

```
(SET-DEFAULT-TEXT-FONT)
```

```
(DEFMACRO TEXT-%WIDTH (TEXT)
```

```
`(IL:BITMAPWIDTH (TEXT-%IMAGE ,TEXT))
```

```
(DEFMACRO TEXT-%HEIGHT (TEXT)
```

```
`(IL:BITMAPHEIGHT (TEXT-%IMAGE ,TEXT))
```

```
(DEFGLOBALVAR *TEXT-SHADOWS* (MAKE-HASH-TABLE :TEST 'EQ)
```

```
"Cache of default shadows indexed by font.")
```

```
(DEFPARAMETER *TEXT-SHADOW-FACTOR* 10
```

```
"Text shadows will use the inverse of this number to determine what fraction of the font size should be shadow.")
```

```
(DEFUN GET-TEXT-SHADOWS (TEXT)
```

```
(LET ((SHADOWS (TEXT-SHADOWS TEXT)))
```

```
(ETYPECASE SHADOWS
```

```
((MEMBER T) (GET-TEXT-SHADOWS-INTERNAL (TEXT-FONT TEXT)))
```

```
(SYMBOL
```

```
;; user defined shadows
```

```
(LET ((INTERNAL (GETHASH SHADOWS *TEXT-SHADOWS*)))
```

```
(OR INTERNAL (ERROR "No text shadows named ~S" SHADOWS))))
```

```
(CONS SHADOWS)))
```

```
(DEFUN GET-TEXT-SHADOWS-INTERNAL (FONT)
```

```
;; cache default shadows per font
```

```
(OR (GETHASH FONT *TEXT-SHADOWS*)
```

```
(SETF (GETHASH FONT *TEXT-SHADOWS*)
```

```
(MAKE-TEXT-SHADOWS FONT)))
```

```
(DEFUN MAKE-TEXT-SHADOWS (FONT &OPTIONAL (FACTOR *TEXT-SHADOW-FACTOR*))
```

```
(LIST (LET ((DEPTH (CEILING (IL:FONTHEIGHT FONT)
```

```
FACTOR)))
```

```
(MAKE-TEXT-SHADOW :DX DEPTH :DY (- DEPTH)
```

```
:OPERATION
```

```
'IL:PAINT))
```

```
(MAKE-TEXT-SHADOW :DY 1)
```

```
(MAKE-TEXT-SHADOW :DX 1)
```

```
(MAKE-TEXT-SHADOW :DY -1)
```

```
(MAKE-TEXT-SHADOW :DX -1)
```

```
(MAKE-TEXT-SHADOW :OPERATION 'IL:ERASE)))
```

```
(DEFUN EXTERNALIZE-TEXT-SHADOWS (SHADOWS)
```

```
(ETYPECASE SHADOWS
```

```
(SYMBOL SHADOWS)
```

```
(CONS (MAPCAR #'(LAMBDA (SHADOW)
```

```

      (LIST :DX (TEXT-SHADOW-DX SHADOW)
            :DY
            (TEXT-SHADOW-DY SHADOW)
            :OPERATION
            (TEXT-SHADOW-OPERATION SHADOW)
            :SOURCE-TYPE
            (TEXT-SHADOW-SOURCE-TYPE SHADOW)
            :TEXTURE
            (TEXT-SHADOW-TEXTURE SHADOW))
    SHADOWS)))

```

```

(DEFUN INTERNALIZE-TEXT-SHADOWS (SHADOWS)
  (ETYPESCASE SHADOWS
    (SYMBOL
      ;; named shadows -- handled by GET-TEXT-SHADOWS
      SHADOWS)
    (CONS
      ;; explitley specified shadows
      (INTERNALIZE-TEXT-SHADOWS-INTERNAL SHADOWS))))

```

```

(DEFUN INTERNALIZE-TEXT-SHADOWS-INTERNAL (SHADOWS)
  (MAPCAR #'(LAMBDA (SHADOW)
    (IF (TEXT-SHADOW-P SHADOW)
      SHADOW
      ;; parse shadow from property list
      (LET ((DX (GETF SHADOW :DX 0))
            (DY (GETF SHADOW :DY 0))
            (OPERATION (GETF SHADOW :OPERATION 'IL:PAINT))
            (SOURCE-TYPE (GETF SHADOW :SOURCE-TYPE 'IL:INPUT))
            (TEXTURE (GETF SHADOW :TEXTURE 0)))
        ;; check the types (defstruct won't)
        (CHECK-TYPE DX INTEGER)
        (CHECK-TYPE DY INTEGER)
        (CHECK-TYPE OPERATION (MEMBER IL:PAINT IL:REPLACE IL:ERASE IL:INVERT))
        (CHECK-TYPE SOURCE-TYPE (MEMBER IL:INPUT IL:INVERT IL:TEXTURE IL:MERGE))
        (CHECK-TYPE TEXTURE TEXTURE)
        ;; make a shadow
        (MAKE-TEXT-SHADOW :DX DX :DY DY :OPERATION OPERATION :SOURCE-TYPE SOURCE-TYPE
                          :TEXTURE TEXTURE))))
    SHADOWS))

```

;; a definer for shadows

```

(DEF-DEFINE-TYPE IL:TEXT-SHADOWS "Text Shadows"
  :UNDEFINER (LAMBDA (NAME)
    (REMHASH NAME *TEXT-SHADOWS*)))
(DEFDEFINER DEF-TEXT-SHADOWS IL:TEXT-SHADOWS (NAME &REST EXTERNAL-FORM)
  `(SETF (GETHASH ',NAME *TEXT-SHADOWS*)
    (INTERNALIZE-TEXT-SHADOWS-INTERNAL ',EXTERNAL-FORM)))

```

```

(DEF-TEXT-SHADOWS NIL NIL)

```

```

(DEF-TEXT-SHADOWS :ARK (:OPERATION IL:ERASE)
  (:DX -1 :DY 1))

```

```

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

```

---

**FUNCTION INDEX**

COMPUTE-TEXT-DIMENSIONS .....	3	GET-TEXT-SHADOWS-INTERNAL .....	4	MAKE-TEXT-SHADOWS .....	4
DISPLAY-TEXT .....	3	INTERNALIZE-TEXT-SHADOWS .....	5	SET-DEFAULT-TEXT-FONT .....	4
EXTERNALIZE-TEXT-SHADOWS .....	4	INTERNALIZE-TEXT-SHADOWS-INTERNAL .....	5	SET-TEXT-STRING .....	4
GET-TEXT-SHADOWS .....	4	MAKE-TEXT .....	2	UPDATE-TEXT-CACHES .....	2

---

**VARIABLE INDEX**

*DEFAULT-TEXT-FONT* .....	2	*TEXT-SHADOWS* .....	4	DEFAULT-TEXT-FONT--SMALL-SCREEN ..	2
*TEXT-SHADOW-FACTOR* .....	4	DEFAULT-TEXT-FONT--LARGE-SCREEN ..	2	SMALL-SCREEN-WIDTH .....	2

---

**MACRO INDEX**

MAXIMIZE .....	3	MINIMIZE .....	3	TEXT-%HEIGHT .....	4	TEXT-%WIDTH .....	4
----------------	---	----------------	---	--------------------	---	-------------------	---

---

**STRUCTURE INDEX**

IL:* .....	1	TEXT-SHADOW .....	1
------------	---	-------------------	---

---

**TEXT-SHADOW INDEX**

:ARK .....	5
------------	---

---

**DEFINER INDEX**

DEF-TEXT-SHADOWS .....	5
------------------------	---

---

**DEFINE-TYPE INDEX**

IL:TEXT-SHADOWS .....	5
-----------------------	---

---

**FILE-ENVIRONMENT INDEX**

IL:ROOMS-TEXT .....	1
---------------------	---

---