

File created: 5-Dec-2020 16:35:05 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOM  
S>MEDLEY-35>ROOMS-BUTTONS.;3

previous date: 17-Aug-90 12:33:51 {DSK}<Users>arunwelch>SKYDRIVE>DOCUMENTS>UNIX>LISP>LDE>ROOMS>MEDLEY-35>ROOMS-B  
UTTONS.;2

Read Table: XCL

Package: ROOMS

Format: XCCS

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

```
(IL:RPAQQ IL:ROOMS-BUTTONSCOMS
  ((FILE-ENVIRONMENTS IL:ROOMS-BUTTONS)
  (IL:FILES (IL:SYSLOAD)
    IL:ROOMS-D IL:ROOMS-TEXT IL:ROOMS-BIOS)
  (IL:P (EXPORT ' (BUTTON *DEFAULT-BUTTON-TYPE* DEF-BUTTON-TYPE MAKE-BUTTON BUTTON-PROP
    *BUTTON-HELP-DELAY* *BUTTON-SELECTION-SHADE* MAKE-BUTTON-WINDOW
    SET-BUTTON-WINDOW-TEXT-STRING WITH-BUTTON *DEFAULT-BUTTON-SHADOWS*
    MAKE-EAST-WEST-BITMAP MAKE-NORTH-SOUTH-BITMAP MAKE-NSEW-BITMAP)
    "ROOMS")))
  (IL:COMS
    ; button types
    (IL:DEFINE-TYPES IL:BUTTON-TYPES)
    (IL:STRUCTURES BUTTON-TYPE)
    (IL:VARIABLES *BUTTON-TYPES* *DEFAULT-BUTTON-TYPE*)
    (IL:FUNCTIONS DEF-BUTTON-TYPE BUTTON-TYPE-PROP SELECT-BUTTON-TYPE BUTTON-TYPE-NAMED)
    (IL:SEDI-FORMATS DEF-BUTTON-TYPE))
  (IL:COMS
    ; the button object
    (IL:STRUCTURES BUTTON UPDATED-BUTTON MARGINS)
    (IL:VARIABLES *DEFAULT-BUTTON-SHADOWS*)
    (IL:FUNCTIONS
      ; core code
      MAKE-BUTTON COPY-BUTTON DISPLAY-BUTTON UPDATE-BUTTON SET-BUTTON-TEXT-STRING BUTTON-PROP
    )
    (IL:FUNCTIONS
      ; text
      SET-BUTTON-TEXT-STRING COMPUTE-BUTTON-TEXT-POSITION BUTTON-TEXT-X-COORD
      BUTTON-TEXT-Y-COORD TEXT-FROM-TEXT-FORM)
    (IL:FUNCTIONS
      ; mouse code
      BUTTON-TRACK-MOUSE PERFORM-BUTTON-ACTION EDIT-BUTTON BUTTON-COPY-SELECTED SHADE-BUTTON
      PRINT-BUTTON-HELP))
  (IL:COMS
    ; button windows
    (IL:VARIABLES *BUTTON-HELP-DELAY* *BUTTON-SELECTION-SHADE*)
    (IL:FUNCTIONS MAKE-BUTTON-WINDOW BW-REPAINTFN BW-TOTOPFN BW-BUTTONEVENTFN
      BW-BUTTONEVENTFN-INTERNAL SET-BUTTON-WINDOW-TEXT-STRING MAYBE-RESIZE-BUTTON-WINDOW
      BW-SCREEN-CHANGED-FUNCTION)
    (IL:VARIABLES
      ;; this variable also on ROOMS-CORE, but here so we can be loaded w/o loading all of rooms
      *SCREEN-CHANGED-FUNCTIONS*)
    (IL:P (PUSHNEW 'BW-SCREEN-CHANGED-FUNCTION *SCREEN-CHANGED-FUNCTIONS*)))
  (IL:COMS
    ; button bitmaps
    (IL:STRUCTURES NORTH-SOUTH-BITMAP EAST-WEST-BITMAP NSEW-BITMAP)
    (IL:FUNCTIONS DISPLAY-BUTTON-IMAGE DISPLAY-BUTTON-MASK BUTTON-WIDTH BUTTON-HEIGHT
      BUTTON-BITMAP-BITBLT EW-BITBLT NS-BITBLT NSEW-BITBLT PAINT-REGION))
    ; externalization
    (IL:FUNCTIONS EDIT-BUTTON-WINDOW EXTERNALIZE-BUTTON EXTERNALIZE-FONT)
    (IL:FUNCTIONS WITH-BUTTON)
    (IL:BUTTON-TYPES :DOOR :SHADOWED :TRANSPARENT :PORTHOLE :ARK :ROUND-ARK :STRETCHY-ARK
      :STRETCHY-ROUND-ARK)
    (IL:GLOBALVARS IL:MENUHELDDWAIT)))

(DEFINE-FILE-ENVIRONMENT IL:ROOMS-BUTTONS :COMPILER :COMPILE-FILE
  :PACKAGE (DEFPACKAGE "ROOMS" (:USE "LISP" "XCL")
    (:SHADOW CL:ROOM))
  :READTABLE "XCL")

(IL:FILESLOAD (IL:SYSLOAD)
  IL:ROOMS-D IL:ROOMS-TEXT IL:ROOMS-BIOS)

(EXPORT ' (BUTTON *DEFAULT-BUTTON-TYPE* DEF-BUTTON-TYPE MAKE-BUTTON BUTTON-PROP *BUTTON-HELP-DELAY*
  *BUTTON-SELECTION-SHADE* MAKE-BUTTON-WINDOW SET-BUTTON-WINDOW-TEXT-STRING WITH-BUTTON
  *DEFAULT-BUTTON-SHADOWS* MAKE-EAST-WEST-BITMAP MAKE-NORTH-SOUTH-BITMAP MAKE-NSEW-BITMAP)
  "ROOMS")

;; button types

(DEF-DEFINE-TYPE IL:BUTTON-TYPES "Button types"
  :UNDEFINER (LAMBDA (NAME)
    (REMHASH NAME *BUTTON-TYPES*)))

(DEFSTRUCT BUTTON-TYPE
  NAME
  ;; name of the type
```

IMAGE-BITMAP

:: the background for the text

MASK-BITMAP

:: to allow non-rectangular buttons. should be a bitmap the same size as IMAGE-BITMAP. designates the set of bits of IMAGE-BITMAP which are  
:: the region to be displayed.

(MARGINS (MAKE-MARGINS))

:: a MARGINS record.

PROPS)

(DEFGLOBALVAR \*BUTTON-TYPES\* (MAKE-HASH-TABLE :TEST 'EQ))

(DEFPARAMETER \*DEFAULT-BUTTON-TYPE\* :SHADOWED)

(DEFDEFINER DEF-BUTTON-TYPE IL:BUTTON-TYPES (NAME &REST REST-KEYS &KEY IMAGE MASK (MARGINS (MAKE-MARGINS))  
&ALLOW-OTHER-KEYS)

`(SETF (GETHASH ',NAME \*BUTTON-TYPES\*)  
(MAKE-BUTTON-TYPE :NAME ',NAME :IMAGE-BITMAP ',IMAGE :MASK-BITMAP ',MASK :MARGINS ',MARGINS :PROPS  
, (LET ((PROPS (COPY-LIST REST-KEYS)))  
(DOLIST (KEYWORD '(:IMAGE :MASK :MARGINS))  
(REMF PROPS KEYWORD))  
PROPS))))

(DEFMACRO BUTTON-TYPE-PROP (BUTTON-TYPE PROP &OPTIONAL (NEW-VALUE NIL NEW-VALUE-SUPPLIED))

(IF NEW-VALUE-SUPPLIED  
` (SETF (GETF (BUTTON-TYPE-PROPS ,BUTTON-TYPE)  
,PROP)  
,NEW-VALUE)  
` (GETF (BUTTON-TYPE-PROPS ,BUTTON-TYPE)  
,PROP)))

(DEFUN SELECT-BUTTON-TYPE (&OPTIONAL (REASON "Select Button Type"))

:: returns the name of a button type or NIL

(MENU (WITH-COLLECTION  
(DOLIST (TYPE (SORT (WITH-COLLECTION (MAPHASH #'(LAMBDA (NAME TYPE)  
(COLLECT TYPE))  
\*BUTTON-TYPES\*))  
#'STRING-LESSP :KEY #'BUTTON-TYPE-NAME))  
(LET ((NAME (BUTTON-TYPE-NAME TYPE))  
(COLLECT '(, (OR (BUTTON-TYPE-PROP TYPE :SAMPLE-IMAGE)  
:: cache sample images on button type  
(LET\* ((BUTTON (MAKE-BUTTON :TYPE NAME :TEXT (LET ((\*PRINT-CASE\* :CAPITALIZE  
)  
(\*READTABLE\* (  
IL:FIND-READTABLE  
"XCL")))  
(PRINC-TO-STRING NAME))))  
(IMAGE (IL:BITMAPCREATE (BUTTON-WIDTH BUTTON)  
(BUTTON-HEIGHT BUTTON))))  
(DISPLAY-BUTTON BUTTON IMAGE)  
(BUTTON-TYPE-PROP TYPE :SAMPLE-IMAGE IMAGE)  
IMAGE))  
,NAME))))))  
REASON))

(DEFMACRO BUTTON-TYPE-NAMED (TYPE-NAME)

`(GETHASH ',TYPE-NAME \*BUTTON-TYPES\*))

(SEDI:DEF-LIST-FORMAT DEF-BUTTON-TYPE :ARGS (NIL :KEYWORD NIL)

:INDENT (1))

:: the button object

(DEFSTRUCT (BUTTON (:CONSTRUCTOR MAKE-BUTTON-INTERNAL)

(:PRINT-FUNCTION (LAMBDA (BUTTON STREAM DEPTH)

(LET ((TYPE (BUTTON-TYPE BUTTON))  
(TEXT (BUTTON-TEXT BUTTON))  
(FORMAT STREAM "#<~A button ~S>" (TYPECASE TYPE  
(BUTTON-TYPE  
(BUTTON-TYPE-NAME  
TYPE))  
(T TYPE))  
(TYPECASE (BUTTON-TEXT BUTTON)  
(TEXT (TEXT-STRING TEXT))  
(T TEXT))))))

```

                (:COPIER COPY-BUTTON-INTERNAL))
    (TYPE *DEFAULT-BUTTON-TYPE* :TYPE BUTTON-TYPE)
;; a BUTTON-TYPE structure
    (TEXT NIL :TYPE TEXT)
;; a TEXT structure
    (ACTION NIL :TYPE LIST)
;; form to EVAL when this button is pressed
    (HELP-STRING NIL :TYPE STRING)
;; printed when button is held
    (INVERTED? NIL :TYPE (MEMBER T NIL))
;; if true, button image will be inverted
    (%SELECTED? NIL :TYPE (MEMBER T NIL))
;; non-nil when button appears selected
    %MASK %IMAGE
;; caches used by redisplay
    (PROPS NIL :TYPE LIST))

(DEFSTRUCT (UPDATED-BUTTON (:INCLUDE BUTTON)
                            (:PRINT-FUNCTION (LAMBDA (BUTTON STREAM DEPTH)
                                                (FORMAT STREAM "#<Updated ~A button ~S>"
                                                         (LET ((TYPE (BUTTON-TYPE BUTTON)))
                                                             (TYPECASE TYPE
                                                             (BUTTON-TYPE (BUTTON-TYPE-NAME TYPE)
                                                             (T TYPE)))
                                                         (UPDATED-BUTTON-TEXT-FORM BUTTON))))))
    (TEXT-FORM NIL :TYPE T))

(DEFSTRUCT (MARGINS (:TYPE LIST))

;;; defines the region within a button intended for the text. We cannot use a region, as buttons may be stretchable.

    (LEFT 0 :TYPE INTEGER)
    (BOTTOM 0 :TYPE INTEGER)
    (RIGHT 0 :TYPE INTEGER)
    (TOP 0 :TYPE INTEGER))

(DEFVAR *DEFAULT-BUTTON-SHADOWS* NIL
        "Default for :SHADOWS arg to MAKE-BUTTON.\
Overridden when button type has default shadows.")

(DEFUN MAKE-BUTTON (&REST REST-KEYS &KEY (TYPE *DEFAULT-BUTTON-TYPE*)
                  (TEXT NIL TEXT-PROVIDED)
                  (TEXT-FORM NIL TEXT-FORM-PROVIDED)
                  (SHADOWS NIL SHADOWS-PROVIDED)
                  ACTION HELP FONT INVERTED? &ALLOW-OTHER-KEYS)

;;; make & return a button. use MAKE-BUTTON-WINDOW to put this button in a window.

    (LET* ((BUTTON-TYPE (OR (IF (BUTTON-TYPE-P TYPE)
                                TYPE)
                            (BUTTON-TYPE-NAMED TYPE)
                            (ERROR "No button type named ~S exists." TYPE)))
           (TEXT (MAKE-TEXT :STRING (IF (AND (NOT TEXT-PROVIDED)
                                             TEXT-FORM-PROVIDED)
                                         TEXT-FROM-TEXT-FORM TEXT-FORM)
                           TEXT))
           :ALIGNMENT :CENTER :FONT (IF FONT
                                         (IL:FONTCREATE FONT)
                                         *DEFAULT-TEXT-FONT*))
           :SHADOWS
           (IF SHADOWS-PROVIDED
               SHADOWS
               ;; default shadows per button type
               (GETF (BUTTON-TYPE-PROPS BUTTON-TYPE)
                     :DEFAULT-SHADOWS *DEFAULT-BUTTON-SHADOWS*)))
           (BUTTON (APPLY (IF TEXT-FORM-PROVIDED
                             #'MAKE-UPDATED-BUTTON
                             #'MAKE-BUTTON-INTERNAL)
                         :TYPE BUTTON-TYPE :TEXT TEXT :ACTION ACTION :HELP-STRING HELP :INVERTED? INVERTED?
                         :PROPS (LET ((PROPS (COPY-LIST REST-KEYS)))
                                   (DOLIST (KEYWORD '(:TYPE :TEXT :ACTION :HELP :FONT :SHADOWS :TEXT-FORM
                                                       :INVERTED?))
                                       (REMF PROPS KEYWORD))
                                   PROPS)
                         (WHEN TEXT-FORM-PROVIDED

```

```

      `(:TEXT-FORM ,TEXT-FORM)))
    (COMPUTE-BUTTON-TEXT-POSITION BUTTON)
  BUTTON)

```

```

(DEFUN COPY-BUTTON (OLD)
  (LET ((NEW (ETYPESCASE OLD
              (UPDATED-BUTTON (COPY-UPDATED-BUTTON OLD))
              (BUTTON (COPY-BUTTON-INTERNAL OLD))))))
    (SETF (BUTTON-TEXT NEW)
          (COPY-TEXT (BUTTON-TEXT OLD)))
    NEW))

```

```

(DEFUN DISPLAY-BUTTON (BUTTON DSP &KEY NO-UPDATE WIDTH HEIGHT)
  (WHEN (AND (NULL NO-UPDATE)
             (OR WIDTH HEIGHT))
    (ERROR "Illegal to pass WIDTH & HEIGHT unless NO-UPDATE specified"))
  (UNLESS NO-UPDATE (UPDATE-BUTTON BUTTON))
  (LET* ((WIDTH (OR WIDTH (BUTTON-WIDTH BUTTON)))
         (HEIGHT (OR HEIGHT (BUTTON-HEIGHT BUTTON)))
         (TYPE (BUTTON-TYPE BUTTON)))
    (WHEN (OR (BUTTON-TYPE-MASK-BITMAP TYPE)
              (NOT (BUTTON-TYPE-IMAGE-BITMAP TYPE)))
      ;; erase what's in the mask (or if button is transparent)
      (DISPLAY-BUTTON-MASK BUTTON DSP WIDTH HEIGHT))
    ;; paint the image on
    (DISPLAY-BUTTON-IMAGE BUTTON DSP WIDTH HEIGHT)
    (WHEN (BUTTON-%SELECTED? BUTTON)
      ;; rationalize the selection
      (SETF (BUTTON-%SELECTED? BUTTON)
            NIL)
      (SHADE-BUTTON BUTTON DSP))))

```

```

(DEFUN UPDATE-BUTTON (BUTTON DSP)
  ;; should really be called BUTTON-NEEDS-REDISPLAY?
  (WHEN (UPDATED-BUTTON-P BUTTON)
    ;; set the text string of WINDOW's BUTTON to the value of its TEXT-FORM.
    (LET ((NEW-TEXT-STRING (TEXT-FROM-TEXT-FORM (UPDATED-BUTTON-TEXT-FORM BUTTON)
                                              DSP BUTTON)))
      (UNLESS (EQUAL NEW-TEXT-STRING (BUTTON-TEXT BUTTON))
        ;; optimization: don't bother if string is same
        (SET-BUTTON-TEXT-STRING BUTTON NEW-TEXT-STRING)
        ;; return T if things have changed
        (RETURN-FROM UPDATE-BUTTON T))))
  ;; a null image cache means button needs redisplay
  (NULL (BUTTON-%IMAGE BUTTON)))

```

```

(DEFUN SET-BUTTON-TEXT-STRING (BUTTON STRING)
  ;; does everything but redisplay
  (SET-TEXT-STRING (BUTTON-TEXT BUTTON)
                  STRING)
  (COMPUTE-BUTTON-TEXT-POSITION BUTTON)
  ;; clear caches
  (SETF (BUTTON-%MASK BUTTON)
        NIL)
  (SETF (BUTTON-%IMAGE BUTTON)
        NIL))

```

```

(DEFMACRO BUTTON-PROP (BUTTON PROP &OPTIONAL (NEW-VALUE NIL NEW-VALUE-SUPPLIED))
  (IF NEW-VALUE-SUPPLIED
    ` (SETF (GETF (BUTTON-PROPS ,BUTTON)
                 ,PROP)
            ,NEW-VALUE)
    ` (GETF (BUTTON-PROPS ,BUTTON)
            ,PROP)))

```

```

(DEFUN SET-BUTTON-TEXT-STRING (BUTTON STRING)
  ;; does everything but redisplay

```

```
(SET-TEXT-STRING (BUTTON-TEXT BUTTON)
  STRING)
( COMPUTE-BUTTON-TEXT-POSITION BUTTON)
;; clear caches
(SETF (BUTTON-%MASK BUTTON)
  NIL)
(SETF (BUTTON-%IMAGE BUTTON)
  NIL))
```

```
(DEFUN COMPUTE-BUTTON-TEXT-POSITION (BUTTON)
  (SETF (TEXT-POSITION (BUTTON-TEXT BUTTON))
    (MAKE-POSITION (BUTTON-TEXT-X-COORD BUTTON)
      (BUTTON-TEXT-Y-COORD BUTTON))))
```

```
(DEFUN BUTTON-TEXT-X-COORD (BUTTON)
  (LET ((TEXT (BUTTON-TEXT BUTTON))
    (MARGINS (BUTTON-TYPE-MARGINS (BUTTON-TYPE BUTTON))))
    (ECASE (TEXT-ALIGNMENT TEXT)
      (:CENTER (+ (MARGINS-LEFT MARGINS)
        (FLOOR (MAX (TEXT-%WIDTH TEXT)
          (- (BUTTON-WIDTH BUTTON)
            (MARGINS-LEFT MARGINS)
            (MARGINS-RIGHT MARGINS))))
          2)))
      ((:LEFT-BOTTOM :LEFT-TOP) (MARGINS-LEFT MARGINS))
      ((:RIGHT-BOTTOM :RIGHT-TOP) (MARGINS-RIGHT MARGINS)))))
```

```
(DEFUN BUTTON-TEXT-Y-COORD (BUTTON)
  (LET ((TEXT (BUTTON-TEXT BUTTON))
    (MARGINS (BUTTON-TYPE-MARGINS (BUTTON-TYPE BUTTON))))
    (ECASE (TEXT-ALIGNMENT TEXT)
      (:CENTER (+ (MARGINS-BOTTOM MARGINS)
        (FLOOR (MAX (TEXT-%HEIGHT TEXT)
          (- (BUTTON-HEIGHT BUTTON)
            (MARGINS-BOTTOM MARGINS)
            (MARGINS-TOP MARGINS))))
          2)))
      ((:LEFT-BOTTOM :RIGHT-BOTTOM) (MARGINS-BOTTOM MARGINS))
      ((:LEFT-TOP :RIGHT-TOP) (MARGINS-TOP MARGINS)))))
```

```
(DEFUN TEXT-FROM-TEXT-FORM (TEXT-FORM &OPTIONAL DSP BUTTON)
```

;; return the text string for an updated button in WINDOW.

```
(TYPECASE TEXT-FORM
  (LIST (EVAL TEXT-FORM))
  ;; note: when an updated button is first created this is called with WINDOW=NIL. text form functions are required to handle this condition
  ;; gracefully.
  (T (FUNCALL TEXT-FORM DSP BUTTON))))
```

```
(DEFUN BUTTON-TRACK-MOUSE (BUTTON DSP)
```

;; a mouse key has gone down in BUTTON. watch the mouse with button shaded 'til either the key goes up or the mouse leaves BUTTON. if key went up then perform button action & return true.

```
(LET ((REGION (MAKE-REGION :LEFT 0 :BOTTOM 0 :WIDTH (BUTTON-WIDTH BUTTON)
  :HEIGHT
  (BUTTON-HEIGHT BUTTON)))
  (TIMER (IL:SETUPTIMER *BUTTON-HELP-DELAY*)))
  (UNWIND-PROTECT
    (PROGN (SHADE-BUTTON BUTTON DSP :REGION REGION)
      (LOOP (IL:GETMOUSESTATE)
        (UNLESS (IL:INSIDEP REGION (IL:LASTMOUSEX DSP)
          (IL:LASTMOUSEY DSP))
          (RETURN)))
        (UNLESS (IL:LASTMOUSESTATE (OR IL:LEFT IL:MIDDLE))
          (PERFORM-BUTTON-ACTION BUTTON DSP)
          ;; return true if we performed action
          (RETURN T))
        (WHEN (AND TIMER (IL:TIMEREXPIRED? TIMER))
          (SETQ TIMER NIL)
          (PRINT-BUTTON-HELP BUTTON))))
    (SHADE-BUTTON BUTTON DSP :REGION REGION :DESELECT T))))
```

```
(DEFUN PERFORM-BUTTON-ACTION (BUTTON DSP)
```

```
(LET ((ACTION (BUTTON-ACTION BUTTON)))
  (TYPECASE ACTION
    (LIST (EVAL ACTION))
```

(T (FUNCALL ACTION DSP BUTTON))))

```

(DEFUN EDIT-BUTTON (BUTTON)
  (IL:ALLOW.BUTTON.EVENTS)
  (LET* ((EXTERNAL-FORM (EXTERNALIZE-BUTTON BUTTON T))
        (COPY (COPY-TREE EXTERNAL-FORM))
        (EDITED (WITH-PROFILE (FIND-PROFILE "XCL")
                              (IL:EDITE EXTERNAL-FORM NIL (TEXT-STRING (BUTTON-TEXT BUTTON))
                              NIL NIL :CLOSE-ON-COMPLETION))))
    (IF (EQUAL EDITED COPY)
        BUTTON
        (APPLY #'MAKE-BUTTON EDITED))))

```

```

(DEFUN BUTTON-COPY-SELECTED (BUTTON)
  (IF (FBOUNDP 'MAKE-BIO)
      ;; if ROOMS-BIO is loaded
      (LET* ((DESTINATION (IL:WFORMDS (IL:PROCESS.TTY (IL:TTY.PROCESS))))
            (COPYINSERTFN (AND DESTINATION (IL:WINDOWPROP DESTINATION 'IL:COPYINSERTFN))))
          ;; fake IL:COPYINSERT, but instead of punting to IL:BKSYSBUF punt to copying the window
          (IF COPYINSERTFN
              (FUNCALL COPYINSERTFN (MAKE-BIO (COPY-BUTTON BUTTON))
                                    DESTINATION)
              (MAKE-BUTTON-WINDOW (COPY-BUTTON BUTTON)))
          (MAKE-BUTTON-WINDOW (COPY-BUTTON BUTTON)))

```

```

(DEFUN SHADE-BUTTON (BUTTON DSP &KEY (REGION (MAKE-REGION :LEFT 0 :BOTTOM 0 :WIDTH (BUTTON-WIDTH BUTTON)
                                                         :HEIGHT (BUTTON-HEIGHT BUTTON)))
                    DESELECT)

```

;; called when mouse key down in BUTTON.

;; DESELECT? tells the intention of the call.

;; see also DISPLAY-BUTTON

```

(LET ((MASK (BUTTON-%MASK BUTTON))
      (SELECTED? (BUTTON-%SELECTED? BUTTON)))
    (WHEN (EQ DESELECT SELECTED?)
        ;; invert MASK with *BUTTON-SELECTION-SHADE*
        (IL:BITBLT MASK NIL NIL DSP 0 0 (REGION-WIDTH REGION)
                  (REGION-HEIGHT REGION)
                  (IF (NULL MASK)
                      'IL:TEXTURE
                      'IL:MERGE)
                  'IL:INVERT *BUTTON-SELECTION-SHADE*))
        ;; toggle SELECTED? bit
        (SETF (BUTTON-%SELECTED? BUTTON)
              (NOT SELECTED?))))

```

```

(DEFUN PRINT-BUTTON-HELP (BUTTON)
  (NOTIFY-USER (OR (BUTTON-HELP-STRING BUTTON)
                  "No help provided for this button.)))

```

;; button windows

(DEFGLOBALVAR \***BUTTON-HELP-DELAY**\* IL:MENUHELDWAIT)

(DEFPARAMETER \***BUTTON-SELECTION-SHADE**\* 32768)

```

(DEFUN MAKE-BUTTON-WINDOW (BUTTON &OPTIONAL POSITION)
  (LET* ((WIDTH (BUTTON-WIDTH BUTTON))
        (HEIGHT (BUTTON-HEIGHT BUTTON))
        (POSITION (OR (IL:POSITIONP POSITION)
                      (IL:GETBOXPOSITION WIDTH HEIGHT)))
        (WINDOW (IL:CREATEW (IL:CREATEREGION (POSITION-X POSITION)
                                             (POSITION-Y POSITION)
                                             WIDTH HEIGHT)
                            NIL 0)))
    (IL:WINDOWPROP WINDOW 'BUTTON BUTTON)
    (IL:WINDOWPROP WINDOW 'IL:BUTTONEVENTFN 'BW-BUTTONEVENTFN)
    (IL:WINDOWPROP WINDOW 'IL:AFTERMOVEFN 'BW-REPAINTFN)
    (IL:WINDOWPROP WINDOW 'IL:OPENFN 'BW-REPAINTFN)
    (IL:WINDOWPROP WINDOW 'IL:TOTOPFN 'BW-TOTOPFN)
    (IL:WINDOWPROP WINDOW 'IL:REPAINTFN 'BW-REPAINTFN)
    (IL:WINDOWPROP WINDOW 'IL:RESHAPEFN 'IL:DON\T)
    (IL:WINDOWPROP WINDOW 'IL:SHRINKFN 'IL:DON\T)

```

```

(WHEN (BUTTON-PROP BUTTON :PROTECTED?)
 (IL:WINDOWPROP WINDOW 'IL:RIGHTBUTTONFN 'IL:TOTOPW))
(BW-REPAINTFN WINDOW)
WINDOW))

```

```

(DEFUN BW-REPAINTFN (WINDOW &REST REST &KEY NO-UPDATE)
 (DECLARE (IGNORE REST))
 (IL:TOTOPW WINDOW T)
 (LET* ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON))
 (DSP (IL:WINDOWPROP WINDOW 'IL:DSP))
 (TYPE (BUTTON-TYPE BUTTON)))
 (UNLESS NO-UPDATE
 (WHEN (UPDATED-BUTTON-P BUTTON)
 (UPDATE-BUTTON BUTTON)))
 (LET ((WIDTH (BUTTON-WIDTH BUTTON))
 (HEIGHT (BUTTON-HEIGHT BUTTON)))
 (MAYBE-RESIZE-BUTTON-WINDOW WINDOW BUTTON WIDTH HEIGHT)
 (IF (AND (BUTTON-TYPE-IMAGE-BITMAP TYPE)
 (NOT (BUTTON-TYPE-MASK-BITMAP TYPE)))
 ;; OK to clear - don't care what's behind
 (IL:CLEARW WINDOW)
 ;; copy what's behind the window through
 (IL:BITBLT (IL:WINDOWPROP WINDOW 'IL:IMAGECOVERED)
 0 0 DSP 0 0 WIDTH HEIGHT 'IL:INPUT 'IL:REPLACE))
 (DISPLAY-BUTTON BUTTON DSP :NO-UPDATE T :WIDTH WIDTH :HEIGHT HEIGHT))))))

```

```

(DEFUN BW-TOTOPFN (WINDOW)
 ;; called when window is un-hidden or brought to top
 (LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON))
 (WHEN (BUTTON-P BUTTON)
 (WHEN (OR ;; needs redisplay because of update
 (UPDATE-BUTTON BUTTON)
 ;; or it has a mask & needs background copied through
 (BUTTON-TYPE-MASK-BITMAP (BUTTON-TYPE BUTTON))
 ;; or it has no mask and no image, i.e. it's transparent & needs background copied through.
 (NULL (BUTTON-TYPE-IMAGE-BITMAP (BUTTON-TYPE BUTTON))))
 (BW-REPAINTFN WINDOW :NO-UPDATE T))))))

```

```

(DEFUN BW-BUTTONEVENTFN (WINDOW)
 (LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON))
 (IF (IL:MOUSESTATE IL:MIDDLE)
 (COND
 ((BUTTON-PROP BUTTON :PROTECTED?)
 (BW-BUTTONEVENTFN-INTERNAL BUTTON WINDOW))
 ((EDIT-KEY-DOWN-P)
 (EDIT-BUTTON-WINDOW BUTTON WINDOW))
 ((COPY-KEY-DOWN-P)
 (BUTTON-COPY-SELECTED BUTTON))
 ((MOVE-KEY-DOWN-P)
 (IL:MOVEW WINDOW))
 ((DELETE-KEY-DOWN-P)
 (IF (FBOUNDP 'INTERACTIVE-CLOSE-WINDOW)
 (INTERACTIVE-CLOSE-WINDOW WINDOW)
 (CLOSE-WINDOW WINDOW)))
 ((HELP-KEY-DOWN-P)
 (PRINT-BUTTON-HELP BUTTON))
 (T (BW-BUTTONEVENTFN-INTERNAL BUTTON WINDOW))))
 (BW-BUTTONEVENTFN-INTERNAL BUTTON WINDOW))))))

```

```

(DEFUN BW-BUTTONEVENTFN-INTERNAL (WINDOW BUTTON)
 (LET ((WINDOW WINDOW)
 (BUTTON BUTTON))
 (LOOP (WHEN (BUTTON-P BUTTON)
 (IL:TOTOPW WINDOW)
 (WHEN (BUTTON-TRACK-MOUSE BUTTON WINDOW)
 (WHEN (UPDATE-BUTTON BUTTON)
 ;; button's action caused it to need redisplay
 (BW-REPAINTFN WINDOW :NO-UPDATE T))
 (RETURN)))
 (UNLESS (IL:MOUSESTATE (OR IL:LEFT IL:MIDDLE))
 (RETURN))
 (SETQ WINDOW (IL:WHICHW))
 (SETQ BUTTON (WHEN WINDOW
 (IL:WINDOWPROP WINDOW 'BUTTON))))))

```

(DEFUN **SET-BUTTON-WINDOW-TEXT-STRING** (WINDOW STRING)

;; note: this does everything but the redisplay.

(LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON)))
 (**SET-BUTTON-TEXT-STRING** BUTTON STRING)
 (**MAYBE-RESIZE-BUTTON-WINDOW** WINDOW BUTTON)))

(DEFUN **MAYBE-RESIZE-BUTTON-WINDOW** (WINDOW BUTTON &OPTIONAL (WIDTH (**BUTTON-WIDTH** BUTTON))
 (HEIGHT (**BUTTON-HEIGHT** BUTTON)))

(LET ((OLD-REGION (WINDOW-REGION WINDOW)))
 (UNLESS (AND (= WIDTH (REGION-WIDTH OLD-REGION))
 (= HEIGHT (REGION-HEIGHT OLD-REGION)))
 (UNWIND-PROTECT
 (PROGN (IL:TOTOPW WINDOW T)
 (IL:WINDOWPROP WINDOW 'IL:RESHAPEFN 'IL:NILL)
 (IL:SHAPEW1 WINDOW (MAKE-REGION :LEFT (REGION-LEFT OLD-REGION)
 :BOTTOM
 (REGION-BOTTOM OLD-REGION)
 :WIDTH WIDTH :HEIGHT HEIGHT))
 ;; return true if we shaped
 T)
 (IL:WINDOWPROP WINDOW 'IL:RESHAPEFN 'IL:DON\'T))))))

(DEFUN **BW-SCREEN-CHANGED-FUNCTION** ()

(LET ((OLD-DEFAULT-FONT \*DEFAULT-TEXT-FONT\*)
 (NEW-DEFAULT-FONT (SET-DEFAULT-TEXT-FONT)))
 (UNLESS (EQ OLD-DEFAULT-FONT NEW-DEFAULT-FONT)
 (DOLIST (WINDOW (ALL-WINDOWS T))
 (LET ((BUTTON (IL:WINDOWPROP WINDOW 'BUTTON)))
 (WHEN (AND (BUTTON-P BUTTON)
 (EQ OLD-DEFAULT-FONT (TEXT-FONT (BUTTON-TEXT BUTTON))))
 ;; upgrade buttons with default font
 (SETF (TEXT-FONT (BUTTON-TEXT BUTTON))
 NEW-DEFAULT-FONT)
 (UPDATE-TEXT-CACHES (BUTTON-TEXT BUTTON))
 (**COMPUTE-BUTTON-TEXT-POSITION** BUTTON)
 ;; force redisplay
 (SETF (BUTTON-%IMAGE BUTTON)
 NIL)
 (SETF (BUTTON-%MASK BUTTON)
 NIL)))))))

(DEFGLOBALVAR **\*SCREEN-CHANGED-FUNCTIONS\*** (LIST '%INTERNALIZE-ALL-PLACEMENTS))

(PUSHNEW 'BW-SCREEN-CHANGED-FUNCTION \*SCREEN-CHANGED-FUNCTIONS\*)

;; button bitmaps

(DEFSTRUCT (**NORTH-SOUTH-BITMAP** (:CONC-NAME "NS-BITMAP-"))
 NORTH
 CENTER
 SOUTH)

(DEFSTRUCT (**EAST-WEST-BITMAP** (:CONC-NAME "EW-BITMAP-"))
 EAST
 CENTER
 WEST)

(DEFSTRUCT **NSEW-BITMAP**
 NORTH
 NW
 NE
 SOUTH
 SW
 SE
 EAST
 CENTER
 WEST)

(DEFUN **DISPLAY-BUTTON-IMAGE** (BUTTON DSP WIDTH HEIGHT)

(LET ((CACHED-IMAGE (BUTTON-%IMAGE BUTTON))
 (INVERTED? (BUTTON-INVERTED? BUTTON))
 (MASK? (BUTTON-%MASK BUTTON)))
 (UNLESS CACHED-IMAGE
 (SETQ CACHED-IMAGE (IL:BITMAPCREATE WIDTH HEIGHT))
 (LET ((TYPE-IMAGE (BUTTON-TYPE-IMAGE-BITMAP (BUTTON-TYPE BUTTON))))



```

(WHEN TYPE-IMAGE (BUTTON-BITMAP-BITBLT TYPE-IMAGE CACHED-IMAGE WIDTH HEIGHT)))
(DISPLAY-TEXT (BUTTON-TEXT BUTTON)
  CACHED-IMAGE)
(SETF (BUTTON-%IMAGE BUTTON)
  CACHED-IMAGE))
(IL:BITBLT CACHED-IMAGE 0 0 DSP 0 0 WIDTH HEIGHT (IF (AND INVERTED? (NOT MASK?))
  'IL:INVERT
  'IL:SOURCE)

(IF (AND INVERTED? MASK?)
  'IL:INVERT
  'IL:PAINT)))

```

```

(DEFUN DISPLAY-BUTTON-MASK (BUTTON DSP WIDTH HEIGHT)
  (LET ((CACHED-MASK (BUTTON-%MASK BUTTON))
    (UNLESS CACHED-MASK
      (SETQ CACHED-MASK (IL:BITMAPCREATE WIDTH HEIGHT))
      (LET ((TYPE-MASK (BUTTON-TYPE-MASK-BITMAP (BUTTON-TYPE BUTTON)))
        (WHEN TYPE-MASK (BUTTON-BITMAP-BITBLT TYPE-MASK CACHED-MASK WIDTH HEIGHT)))
        (DISPLAY-TEXT (BUTTON-TEXT BUTTON)
          CACHED-MASK :MASK-ONLY T)
        (SETF (BUTTON-%MASK BUTTON)
          CACHED-MASK))
        (IL:BITBLT CACHED-MASK 0 0 DSP 0 0 WIDTH HEIGHT 'IL:SOURCE (IF (BUTTON-INVERTED? BUTTON)
          'IL:PAINT
          'IL:ERASE))))))

```

```

(DEFUN BUTTON-WIDTH (BUTTON)
  (LET* ((BUTTON-TYPE (BUTTON-TYPE BUTTON))
    (MARGINS (BUTTON-TYPE-MARGINS BUTTON-TYPE))
    (BITMAP (BUTTON-TYPE-IMAGE-BITMAP BUTTON-TYPE))
    (TEXT-WIDTH (TEXT-%WIDTH (BUTTON-TEXT BUTTON)))
    (ETYPESCASE BITMAP
      (BITMAP (IL:BITMAPWIDTH BITMAP))
      (NULL TEXT-WIDTH)
      (NORTH-SOUTH-BITMAP (IL:BITMAPWIDTH (NS-BITMAP-NORTH BITMAP)))
      ((OR NSEW-BITMAP EAST-WEST-BITMAP) (LET* ((WIDTH (+ TEXT-WIDTH (MARGINS-LEFT MARGINS)
        (MARGINS-RIGHT MARGINS))
        (EAST-WIDTH (IL:BITMAPWIDTH (TYPECASE BITMAP
          (NSEW-BITMAP
            (NSEW-BITMAP-EAST
              BITMAP))
            (EAST-WEST-BITMAP
              (EW-BITMAP-EAST BITMAP
                ))))
          (CENTER-WIDTH (IL:BITMAPWIDTH (TYPECASE BITMAP
            (NSEW-BITMAP
              (NSEW-BITMAP-CENTER
                BITMAP))
            (EAST-WEST-BITMAP
              (EW-BITMAP-CENTER
                BITMAP))))))
          (WEST-WIDTH (IL:BITMAPWIDTH (TYPECASE BITMAP
            (NSEW-BITMAP
              (NSEW-BITMAP-WEST
                BITMAP))
            (EAST-WEST-BITMAP
              (EW-BITMAP-WEST BITMAP
                ))))
          ))))
      ;; we could use WIDTH directly but we'd rather tile in an even number of CENTER bitmaps, in case
      ;; it's a pattern that needs to blend with the EAST and WEST.
      (MAX (+ WIDTH (- CENTER-WIDTH (MOD (- WIDTH EAST-WIDTH
        WEST-WIDTH)
        CENTER-WIDTH)))
        (+ EAST-WIDTH WEST-WIDTH))))))

```

```

(DEFUN BUTTON-HEIGHT (BUTTON)
  (LET* ((BUTTON-TYPE (BUTTON-TYPE BUTTON))
    (MARGINS (BUTTON-TYPE-MARGINS BUTTON-TYPE))
    (BITMAP (BUTTON-TYPE-IMAGE-BITMAP BUTTON-TYPE))
    (TEXT-HEIGHT (TEXT-%HEIGHT (BUTTON-TEXT BUTTON)))
    (ETYPESCASE BITMAP
      (BITMAP (IL:BITMAPHEIGHT BITMAP))
      (NULL TEXT-HEIGHT)
      (EAST-WEST-BITMAP (IL:BITMAPHEIGHT (EW-BITMAP-EAST BITMAP)))
      ((OR NSEW-BITMAP NORTH-SOUTH-BITMAP) (LET* ((HEIGHT (+ TEXT-HEIGHT (MARGINS-BOTTOM MARGINS)
        (MARGINS-TOP MARGINS))
        (NORTH-HEIGHT (IL:BITMAPHEIGHT
          (TYPECASE BITMAP
            (NSEW-BITMAP (NSEW-BITMAP-NORTH
              BITMAP))
            (NORTH-SOUTH-BITMAP (NS-BITMAP-NORTH
              BITMAP))))))
          (CENTER-HEIGHT (IL:BITMAPHEIGHT

```

```

(TYPECASE BITMAP
  (NSEW-BITMAP (NSEW-BITMAP-CENTER
    BITMAP))
  (NORTH-SOUTH-BITMAP (
    NS-BITMAP-CENTER
    BITMAP))))
(SOUTH-HEIGHT (IL:BITMAPHEIGHT
  (TYPECASE BITMAP
    (NSEW-BITMAP (NSEW-BITMAP-SOUTH
      BITMAP))
    (NORTH-SOUTH-BITMAP (NS-BITMAP-SOUTH
      BITMAP))))))

```

;; we could use HEIGHT directly but we'd rather tile in an even number of CENTER bitmaps, in case  
;; it's a pattern that needs to blend with the EAST and WEST.

```

(MAX (+ HEIGHT (- CENTER-HEIGHT
  (MOD (- HEIGHT NORTH-HEIGHT
    SOUTH-HEIGHT)
    CENTER-HEIGHT)))
  (+ NORTH-HEIGHT SOUTH-HEIGHT))))

```

```

(DEFUN BUTTON-BITMAP-BITBLT (BITMAP DESTINATION WIDTH HEIGHT)
  (ETYPECASE BITMAP
    (BITMAP (IL:BITBLT BITMAP 0 0 DESTINATION 0 0 WIDTH HEIGHT))
    (EAST-WEST-BITMAP (EW-BITBLT (EW-BITMAP-WEST BITMAP)
      (EW-BITMAP-CENTER BITMAP)
      (EW-BITMAP-EAST BITMAP)
      DESTINATION WIDTH 0))
    (NORTH-SOUTH-BITMAP (NS-BITBLT (NS-BITMAP-SOUTH BITMAP)
      (NS-BITMAP-CENTER BITMAP)
      (NS-BITMAP-NORTH BITMAP)
      DESTINATION HEIGHT 0))
    (NSEW-BITMAP (NSEW-BITBLT BITMAP DESTINATION WIDTH HEIGHT))))

```

```

(DEFUN EW-BITBLT (WEST CENTER EAST DESTINATION WIDTH BOTTOM)
  (LET* ((WEST-WIDTH (IL:BITMAPWIDTH WEST))
    (CENTER-WIDTH (IL:BITMAPWIDTH CENTER))
    (EAST-WIDTH (IL:BITMAPWIDTH EAST))
    (EAST-LEFT (- WIDTH EAST-WIDTH))
    (HEIGHT (IL:BITMAPHEIGHT CENTER)))
    ;; blt the west bitmap down the left
    (IL:BITBLT WEST 0 0 DESTINATION 0 BOTTOM WEST-WIDTH HEIGHT)
    (WHEN (> EAST-LEFT WEST-WIDTH)
      ;; blt in one copy of center
      (IL:BITBLT CENTER 0 0 DESTINATION WEST-WIDTH BOTTOM CENTER-WIDTH HEIGHT)
      (DO* ((WIDTH CENTER-WIDTH (+ WIDTH WIDTH))
        (LEFT (+ WEST-WIDTH WIDTH)
          (+ WEST-WIDTH WIDTH)))
          ((>= LEFT EAST-LEFT)))
        ;; blt the center bitmap across the middle
        (IL:BITBLT DESTINATION WEST-WIDTH BOTTOM DESTINATION LEFT BOTTOM (MIN WIDTH (- EAST-LEFT LEFT))
          HEIGHT))
        ;; blt the east bitmap on the right end
        (IL:BITBLT EAST 0 0 DESTINATION EAST-LEFT BOTTOM EAST-WIDTH HEIGHT)))

```

```

(DEFUN NS-BITBLT (SOUTH CENTER NORTH DESTINATION HEIGHT LEFT &OPTIONAL (DO-ENDS? T))
  (LET* ((SOUTH-HEIGHT (IL:BITMAPHEIGHT SOUTH))
    (CENTER-HEIGHT (IL:BITMAPHEIGHT CENTER))
    (NORTH-HEIGHT (IL:BITMAPHEIGHT NORTH))
    (NORTH-BOTTOM (- HEIGHT NORTH-HEIGHT))
    (WIDTH (IL:BITMAPWIDTH CENTER)))
    (WHEN DO-ENDS?
      ;; blt the south bitmap across the bottom
      (IL:BITBLT SOUTH 0 0 DESTINATION LEFT 0 WIDTH SOUTH-HEIGHT))
      (WHEN (> NORTH-BOTTOM SOUTH-HEIGHT)
        ;; blt in one copy of center
        (IL:BITBLT CENTER 0 0 DESTINATION LEFT SOUTH-HEIGHT WIDTH CENTER-HEIGHT)
        (DO* ((HEIGHT CENTER-HEIGHT (+ HEIGHT HEIGHT))
          (BOTTOM (+ SOUTH-HEIGHT HEIGHT)
            (+ SOUTH-HEIGHT HEIGHT)))
            ((>= BOTTOM NORTH-BOTTOM)))
          ;; blt the center bitmap up the middle
          (IL:BITBLT DESTINATION LEFT SOUTH-HEIGHT DESTINATION LEFT BOTTOM WIDTH (MIN HEIGHT
            (- NORTH-BOTTOM
              BOTTOM))))
        (WHEN DO-ENDS?
          ;; blt the north bitmap across the top

```

(IL:BITBLT NORTH 0 0 DESTINATION LEFT NORTH-BOTTOM WIDTH NORTH-HEIGHT)))

```

(DEFUN NSEW-BITBLT (NSEW-BITMAP DESTINATION WIDTH HEIGHT)
  (LET* ((SW (NSEW-BITMAP-SW NSEW-BITMAP))
         (SE (NSEW-BITMAP-SE NSEW-BITMAP))
         (NW (NSEW-BITMAP-NW NSEW-BITMAP))
         (NE (NSEW-BITMAP-NE NSEW-BITMAP))
         (NORTH-BOTTOM (- HEIGHT (IL:BITMAPHEIGHT NW)))
         (EAST-LEFT (- WIDTH (IL:BITMAPWIDTH SE))))
    ;; across the bottom
    (EW-BITBLT SW (NSEW-BITMAP-SOUTH NSEW-BITMAP)
                SE DESTINATION WIDTH 0)
    ;; across the top
    (EW-BITBLT NW (NSEW-BITMAP-NORTH NSEW-BITMAP)
                NE DESTINATION WIDTH NORTH-BOTTOM)
    ;; up the left
    (NS-BITBLT SW (NSEW-BITMAP-WEST NSEW-BITMAP)
                NW DESTINATION HEIGHT 0 NIL)
    ;; up the right
    (NS-BITBLT SE (NSEW-BITMAP-EAST NSEW-BITMAP)
                NE DESTINATION HEIGHT EAST-LEFT NIL)
    ;; tile the center
    (PAINT-REGION DESTINATION (LET ((LEFT (IL:BITMAPWIDTH SW))
                                   (BOTTOM (IL:BITMAPHEIGHT SW)))
                                (MAKE-REGION :LEFT LEFT :BOTTOM BOTTOM :WIDTH (- EAST-LEFT LEFT)
                                             :HEIGHT
                                             (- NORTH-BOTTOM BOTTOM)))
                  (NSEW-BITMAP-CENTER NSEW-BITMAP))))

```

```

(DEFUN PAINT-REGION (DESTINATION REGION SHADE &OPTIONAL CLIPPING-REGION)
  ;; fill REGION of DESTINATION with SHADE
  (TYPECASE SHADE
    (BITMAP
     ;; tile the bitmap within REGION
     (LET* ((REGION-LEFT (REGION-LEFT REGION))
            (REGION-BOTTOM (REGION-BOTTOM REGION))
            (REGION-WIDTH (REGION-WIDTH REGION))
            (REGION-HEIGHT (REGION-HEIGHT REGION))
            (BITMAP-WIDTH (IL:BITMAPWIDTH SHADE))
            (BITMAP-HEIGHT (IL:BITMAPHEIGHT SHADE))
            (REGION-RIGHT (+ REGION-LEFT REGION-WIDTH))
            (REGION-TOP (+ REGION-BOTTOM REGION-HEIGHT))
            (CLIPPING-REGION (IF CLIPPING-REGION
                               (IL:INTERSECTREGIONS CLIPPING-REGION REGION)
                               REGION)))
        ;; blt in one copy in lower left corner
        (IL:BITBLT SHADE 0 0 DESTINATION REGION-LEFT REGION-BOTTOM BITMAP-WIDTH BITMAP-HEIGHT NIL NIL
                   NIL CLIPPING-REGION)
        ;; blt across bottom, doubling size each time
        (LET ((LEFT BITMAP-WIDTH))
            (LOOP (WHEN (>= LEFT REGION-RIGHT)
                  (RETURN))
                 (IL:BITBLT DESTINATION REGION-LEFT REGION-BOTTOM DESTINATION (+ LEFT REGION-LEFT)
                           REGION-BOTTOM LEFT BITMAP-HEIGHT NIL NIL NIL CLIPPING-REGION)
                 (SETF LEFT (+ LEFT LEFT))))
        ;; blt up, doubling size each time
        (LET ((BOTTOM BITMAP-HEIGHT))
            (LOOP (WHEN (>= BOTTOM REGION-TOP)
                  (RETURN))
                 (IL:BITBLT DESTINATION REGION-LEFT REGION-BOTTOM DESTINATION REGION-LEFT
                           (+ REGION-BOTTOM BOTTOM)
                           REGION-WIDTH BOTTOM NIL NIL NIL CLIPPING-REGION)
                 (SETF BOTTOM (+ BOTTOM BOTTOM))))))
    (TEXTURE
     ;; squirt the texture onto the screen within REGION
     (IL:BLTSHADE SHADE DESTINATION (REGION-LEFT REGION)
                  (REGION-BOTTOM REGION)
                  (REGION-WIDTH REGION)
                  (REGION-HEIGHT REGION)
                  NIL CLIPPING-REGION)))

```

;; externalization

```
(DEFUN EDIT-BUTTON-WINDOW (BUTTON WINDOW)
  (UNLESS (IL:WINDOWPROP WINDOW 'BUTTON-BEING-EDITED)
    (UNWIND-PROTECT
      (PROGN (IL:WINDOWPROP WINDOW 'BUTTON-BEING-EDITED T)
        (LET ((NEW-BUTTON (EDIT-BUTTON BUTTON)))
          (UNLESS (EQ BUTTON NEW-BUTTON)
            (IL:WINDOWPROP WINDOW 'BUTTON NEW-BUTTON)
            (BW-REPAINTFN WINDOW))))
        (IL:REMWINDOWPROP WINDOW 'BUTTON-BEING-EDITED))))))
```

```
(DEFUN EXTERNALIZE-BUTTON (BUTTON &OPTIONAL VERBOSE)
```

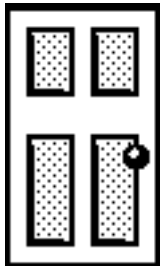
;;; returns a property list to which MAKE-BUTTON can be applied

```
(LET* ((TEXT (BUTTON-TEXT BUTTON))
  (TYPE (BUTTON-TYPE BUTTON))
  (TYPE-NAME (BUTTON-TYPE-NAME TYPE))
  (SHADOWS (TEXT-SHADOWS TEXT))
  (FONT (TEXT-FONT TEXT))
  (INVERTED? (BUTTON-INVERTED? BUTTON)))
  ` (,@ (ETYPECASE BUTTON
    (UPDATED-BUTTON ` (:TEXT-FORM , (UPDATED-BUTTON-TEXT-FORM BUTTON))
      (BUTTON ` (:TEXT , (TEXT-STRING TEXT))))
    :ACTION
    , (COPY-TREE (BUTTON-ACTION BUTTON))
    :HELP
    , (BUTTON-HELP-STRING BUTTON)
    ,@ (WHEN (OR VERBOSE (NOT (EQ FONT *DEFAULT-TEXT-FONT*)))
      (LIST :FONT (EXTERNALIZE-FONT FONT)))
    ,@ (WHEN (OR VERBOSE (NOT (EQUAL SHADOWS (GETF (BUTTON-TYPE-PROPS TYPE)
      :DEFAULT-SHADOWS *DEFAULT-BUTTON-SHADOWS*))))
      (LIST :SHADOWS (EXTERNALIZE-TEXT-SHADOWS SHADOWS)))
    ,@ (WHEN (OR (NULL TYPE-NAME)
      VERBOSE
      (NOT (EQUAL TYPE-NAME *DEFAULT-BUTTON-TYPE*)))
      (LIST :TYPE (IF (NULL TYPE-NAME)
        TYPE
        TYPE-NAME)))
    ,@ (WHEN (OR VERBOSE INVERTED?)
      (LIST :INVERTED? INVERTED?))
    ,@ (COPY-TREE (BUTTON-PROPS BUTTON))))))
```

```
(DEFUN EXTERNALIZE-FONT (FONT)
  (LIST (IL:FONTPROP FONT 'IL:FAMILY)
    (IL:FONTPROP FONT 'IL:SIZE)
    (IL:FONTPROP FONT 'IL:FACE)))
```

```
(DEFUN WITH-BUTTON (ACTION TEXT HELP)
  (IF (COPY-KEY-DOWN-P)
    (PROG1 NIL
      (MAKE-BUTTON-WINDOW (MAKE-BUTTON :TYPE *DEFAULT-BUTTON-TYPE* :TEXT TEXT :HELP HELP :ACTION ACTION)
        TEXT HELP))
    (EVAL ACTION)))
```

```
(DEF-BUTTON-TYPE :DOOR :IMAGE
```




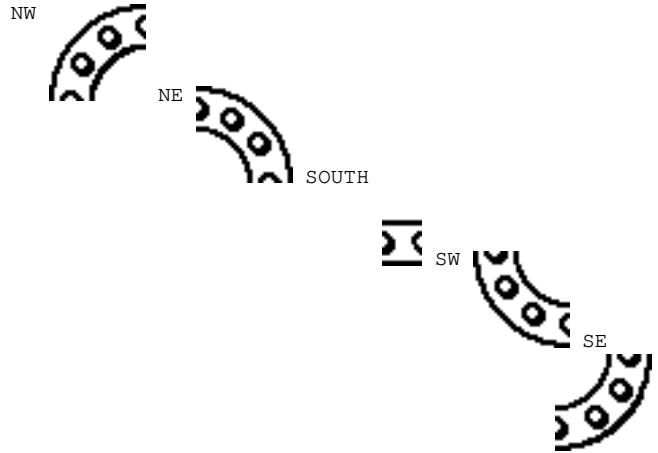
```
:MASK NIL
:MARGINS (2 18 3 2)
:DEFAULT-SHADOWS NIL)
```




```
(DEF-BUTTON-TYPE :SHADOWED :IMAGE
  #S(NSEW-BITMAP NORTH ■ NW ■ NE ■ SOUTH ! SW ■ SE ■ EAST ■ CENTER ■ WEST ■)
  :MASK #S(NSEW-BITMAP NORTH ■ NW ■ NE ■ SOUTH ! SW ■ SE ■ EAST ■ CENTER ■ WEST ■)
  :MARGINS (3 5 7 3))
```

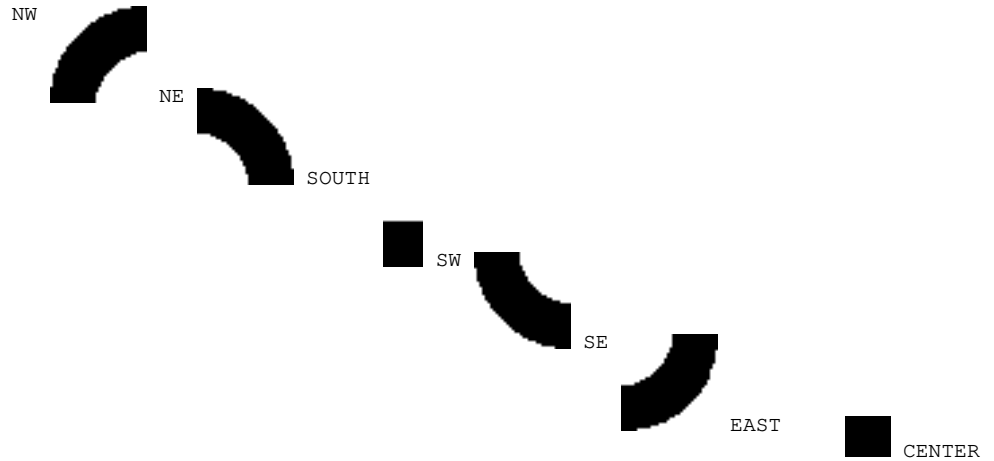
```
(DEF-BUTTON-TYPE :TRANSPARENT :IMAGE NIL
  :MASK NIL
  :MARGINS (0 0 0 0))
```


:DEFAULT-SHADOWS T)

(DEF-BUTTON-TYPE :PORTHOLE :IMAGE #S (NSEW-BITMAP NORTH )





EAST  CENTER  
WEST  )  
:MASK #S (NSEW-BITMAP NORTH )





















WEST  )  
:MARGINS (17 17 17 17)  
:DEFAULT-SHADOWS T)

(DEF-BUTTON-TYPE :ARK :IMAGE  
#S (NORTH-SOUTH-BITMAP NORTH  CENTER  SOUTH  )  
:MARGINS (4 5 5 6)  
:DEFAULT-SHADOWS :ARK)

(DEF-BUTTON-TYPE :ROUND-ARK :IMAGE   
:MASK   
:MARGINS (5 4 5 2)  
:DEFAULT-SHADOWS :ARK)

(DEF-BUTTON-TYPE :STRETCHY-ARK :IMAGE  
#S (NSEW-BITMAP NORTH  NW  NE  SOUTH  SW  SE  EAST  CENTER  WEST  )  
:MARGINS (6 6 6 6)  
:DEFAULT-SHADOWS :ARK)

```
(DEF-BUTTON-TYPE :STRETCHY-ROUND-ARK :IMAGE
#S(NSEW-BITMAP NORTH  NW  NE  SOUTH  SW  SE  EAST  CENTER  WEST )
:MASK #S(NSEW-BITMAP NORTH  NW  NE  SOUTH  SW  SE  EAST  CENTER  WEST )
:MARGINS (4 1 4 2)
:DEFAULT-SHADOWS :ARK)

(IL:DECLARE\ : IL:DOEVAL@COMPILE IL:DONTCOPY
(IL:GLOBALVARS IL:MENUHELDWAIT)
)

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

---

**FUNCTION INDEX**

BUTTON-BITMAP-BITBLT	10	COPY-BUTTON	4	NSEW-BITBLT	11
BUTTON-COPY-SELECTED	6	DISPLAY-BUTTON	4	PAINT-REGION	11
BUTTON-HEIGHT	9	DISPLAY-BUTTON-IMAGE	8	PERFORM-BUTTON-ACTION	5
BUTTON-TEXT-X-COORD	5	DISPLAY-BUTTON-MASK	9	PRINT-BUTTON-HELP	6
BUTTON-TEXT-Y-COORD	5	EDIT-BUTTON	6	SELECT-BUTTON-TYPE	2
BUTTON-TRACK-MOUSE	5	EDIT-BUTTON-WINDOW	12	SET-BUTTON-TEXT-STRING	4
BUTTON-WIDTH	9	EW-BITBLT	10	SET-BUTTON-WINDOW-TEXT-STRING	8
BW-BUTTONEVENTFN	7	EXTERNALIZE-BUTTON	12	SHADE-BUTTON	6
BW-BUTTONEVENTFN-INTERNAL	7	EXTERNALIZE-FONT	12	TEXT-FROM-TEXT-FORM	5
BW-REPAINTFN	7	MAKE-BUTTON	3	UPDATE-BUTTON	4
BW-SCREEN-CHANGED-FUNCTION	8	MAKE-BUTTON-WINDOW	6	WITH-BUTTON	12
BW-TOTOPFN	7	MAYBE-RESIZE-BUTTON-WINDOW	8		
COMPUTE-BUTTON-TEXT-POSITION	5	NS-BITBLT	10		

---

**BUTTON-TYPE INDEX**

:ARK	13	:PORTHOLE	13	:SHADOWED	12	:STRETCHY-ROUND-ARK	14
:DOOR	12	:ROUND-ARK	13	:STRETCHY-ARK	13	:TRANSPARENT	12

---

**STRUCTURE INDEX**

BUTTON	2	EAST-WEST-BITMAP	8	NORTH-SOUTH-BITMAP	8	UPDATED-BUTTON	3
BUTTON-TYPE	1	MARGINS	3	NSEW-BITMAP	8		

---

**VARIABLE INDEX**

*BUTTON-HELP-DELAY*	6	*BUTTON-TYPES*	2	*DEFAULT-BUTTON-TYPE*	2
*BUTTON-SELECTION-SHADE*	6	*DEFAULT-BUTTON-SHADOWS*	3	*SCREEN-CHANGED-FUNCTIONS*	8

---

**MACRO INDEX**

BUTTON-PROP	4	BUTTON-TYPE-NAMED	2	BUTTON-TYPE-PROP	2
-------------	---	-------------------	---	------------------	---

---

**FILE-ENVIRONMENT INDEX**

IL:ROOMS-BUTTONS	1
------------------	---

---

**SEDIT-FORMAT INDEX**

DEF-BUTTON-TYPE	2
-----------------	---

---

**DEFINER INDEX**

DEF-BUTTON-TYPE	2
-----------------	---

---

**DEFINE-TYPE INDEX**

IL:BUTTON-TYPES	1
-----------------	---

---