

File created: 17-Aug-87 14:00:19 {DSK}<LISPPFILES>DEV>NEATICONS.;2

changes to: (IL:ADVISE IL:SHRINKW)
(IL:VARS IL:NEATICONSCOMS)

previous date: 17-Aug-87 13:51:56 {DSK}<LISPPFILES>DEV>NEATICONS.;1

Read Table: XCL

Package: NEATICONS

Format: XCCS

; Copyright (c) 1967, 1986, 1987 by Quintus Computer Systems, Inc. All rights reserved.

(IL:RPAQQ **IL:NEATICONSCOMS**

;; This file makes sure that all icons (shrunk windows) created after this file is loaded are neat. This means that when an icon is near
;; another window it is neatly placed NEATICONS:DEFAULT-SPACING (defaults to 5) pixels away, or the edge of the screen flushed with
;; the edge of the screen, or if one of its edges is near the corresponding edge of another window, the edges will be perfectly aligned. It's
;; a lot easier to understand what this means by trying it. An icon will be moved at most NEATICONS:DEFAULT-TOLERANCE (defaults
;; to 100) pixels horizontally or vertically in order to make it neat. So if you put an icon in the middle of nowhere, it will stay there.

;; This is done by advising SHRINKW to make newly generated icons neat. The function NEATICONS:NEATEN.WINDOW, when applied
;; to a window, will make that window always neat. So after loading this file, newly created icons will always be neat; and users can
;; make any window neat by calling (NEATICONS:NEATEN window). Note that existing icons can be made neat by expanding and
;; re-shrinking them.

;; Exported variables and functions

(IL:VARIABLES DEFAULT-SPACING DEFAULT-TOLERANCE)
(IL:FUNCTIONS NEATEN UNNEATEN)

;; Private stuff

(IL:ADVISE IL:SHRINKW)
(IL:FUNCTIONS BETWEEN MIN-ABSOLUTE MIN-SUM-SQUARES NEAT-POSITION)
(IL:PROP IL:MAKEFILE-ENVIRONMENT IL:NEATICONS)))

;; This file makes sure that all icons (shrunk windows) created after this file is loaded are neat. This means that when an icon is near another window
;; it is neatly placed NEATICONS:DEFAULT-SPACING (defaults to 5) pixels away, or the edge of the screen flushed with the edge of the screen, or if
;; one of its edges is near the corresponding edge of another window, the edges will be perfectly aligned. It's a lot easier to understand what this means
;; by trying it. An icon will be moved at most NEATICONS:DEFAULT-TOLERANCE (defaults to 100) pixels horizontally or vertically in order to make it
;; neat. So if you put an icon in the middle of nowhere, it will stay there.

;; This is done by advising SHRINKW to make newly generated icons neat. The function NEATICONS:NEATEN.WINDOW, when applied to a window,
;; will make that window always neat. So after loading this file, newly created icons will always be neat; and users can make any window neat by calling
;; (NEATICONS:NEATEN window). Note that existing icons can be made neat by expanding and re-shrinking them.

;; Exported variables and functions

(DEFGLOBALPARAMETER **DEFAULT-SPACING** 5
"Number of pixels between neat icons.")

(DEFGLOBALPARAMETER **DEFAULT-TOLERANCE** 100
"How far an icon will be moved to be neat.")

(DEFUN **NEATEN** (&OPTIONAL (WINDOW (IL:WHICHW)))
"Makes WINDOW (default: (WHICHW)) always neatly aligned with nearby windows"

;;; Makes WINDOW neat, i.e., makes it neatly aligned on the screen, and makes sure that wherever it is moved, it will remain neatly placed. Also makes
;;; sure that any existing MOVEFN on the window will still get called when the window is moved, and that function has the final decision as the window's
;;; actual new position. WINDOW defaults to (WHICHW)

;;; For more on what it means to be neat, and how a neat position for a window is determined, see NEATICONS::NEAT-POSITION

(|if| (IL:WINDOWP WINDOW)
|then| (LET ((OLDMOVEFN (IL:WINDOWPROP WINDOW 'IL:MOVEFN 'NEAT-POSITION)))
(|if| (NOT (IL:EQMEMB 'NEAT-POSITION OLDMOVEFN)) ; If it's not already neat ...
|then| (IL:WINDOWPROP WINDOW 'USERMOVEFN OLDMOVEFN)
(IL:RELMOVEW WINDOW '(0 . 0))) ; invokes NEAT-POSITION to neaten WINDOW
WINDOW)))

(DEFUN **UNNEATEN** (&OPTIONAL (WINDOW (IL:WHICHW)))
"Makes WINDOW (default: (WHICHW)) a normal, non-neat window"

;;; undoes the effect of NEATICONS:NEATEN. WINDOW becomes a normal, sloppy, window. WINDOW defaults to (WHICHW)

(|if| (NOT (IL:WINDOWP WINDOW))
|then| (IL:ERROR "Not a window" WINDOW))
(IL:WINDOWPROP WINDOW 'IL:MOVEFN (IL:WINDOWPROP WINDOW 'USERMOVEFN NIL))
WINDOW)

;; Private stuff

(REINSTALL-ADVISE 'IL:SHRINKW :AROUND '(:LAST (LET ((IL:ICON IL:*)
(NEATEN IL:ICON)

IL:ICON)))

(IL:READWISE IL:SHRINKW)

```

(DEFMACRO BETWEEN (X LOWER UPPER)
  "X is between LOWER and UPPER?"
  `(LET ((XVALUE ,X))
    (AND (>= XVALUE ,LOWER)
      (<= XVALUE ,UPPER))))

```

```

(DEFMACRO MIN-ABSOLUTE (ARG1 ARG2 &OPTIONAL ARG3)
  "Returns whichever arg has the smallest absolute value"
  `(LET* ((ARG1-VALUE ,ARG1)
    (ARG2-VALUE ,ARG2)
    (BEST-OF-TWO `(|if| (< (ABS ARG2-VALUE)
      (ABS ARG1-VALUE))
    |then| ARG2-VALUE
    |else| ARG1-VALUE)))
    , (|if| ARG3
      |then| `(LET ((ARG3-VALUE ,ARG3))
        (|if| (< (ABS ARG3-VALUE)
          (ABS BEST-OF-TWO))
          |then| ARG3-VALUE
          |else| BEST-OF-TWO))
        |else| `BEST-OF-TWO)))

```

```

(DEFMACRO MIN-SUM-SQUARES (&REST PAIRS)

```

```

;;; (min-sum-squares (deltax-1 deltax-2) (deltay-1 deltax-2) ...)

```

```

;;; returns the (x,y) pair (2 values) that have the smallest deltax^2 + deltax^2

```

```

  `(PROG ((BEST-DX , (CAAR PAIRS))
    (BEST-DY , (CADAR PAIRS))
    (BEST-SUMSQ TEMP-SUMSQ)
    (SETQ BEST-SUMSQ (+ (* BEST-DX BEST-DX)
      (* BEST-DY BEST-DY)))
    , @ (|for| PR |in| (CDR PAIRS))
      |collect| `(|if| (< (SETQ TEMP-SUMSQ (+ (* , (CAR PR)
        , (CAR PR))
        (* , (CADR PR)
        , (CADR PR)))))
        BEST-SUMSQ)
      |then| (SETQ BEST-SUMSQ TEMP-SUMSQ)
        (SETQ BEST-DX , (CAR PR))
        (SETQ BEST-DY , (CADR PR))))
    (RETURN (VALUES BEST-DX BEST-DY)))

```

```

(DEFUN NEAT-POSITION (WINDOW-TO-MOVE TENTATIVE-POSITION &OPTIONAL (TOLERANCE DEFAULT-TOLERANCE)
  (SPACING DEFAULT-SPACING))

```

```

  "Returns the position nearest to TENTATIVE-POSITION that is neat."

```

```

  (|bind| ;; Variables describing the window we're moving and its new place:

```

```

    (USERMOVEFN _ (IL:WINDOWPROP WINDOW-TO-MOVE 'USERMOVEFN))
    (MYREG _ (IL:WINDOWPROP WINDOW-TO-MOVE 'IL:REGION))
    (MYLEFT _ (|fetch| (IL:POSITION IL:XCOORD) |of| TENTATIVE-POSITION))
    (MYBOTTOM _ (|fetch| (IL:POSITION IL:YCOORD) |of| TENTATIVE-POSITION))
    MYWIDTH MYHEIGHT MYRIGHT

```

```

    ;; These describe the region WINDOW-TO-MOVE can be placed within and still meet the constraints imposed by TOLERANCE:

```

```

    MYTOP MINLEFT MAXRIGHT MINBOTTOM MAXTOP

```

```

    ;; Variables to keep track of the best new place we've found so far:

```

```

    BEST-DELTAX BEST-DELTAY BEST-CORNER-DELTAX BEST-CORNER-DELTAY CORNER-DELTAX-WINDOW CORNER-DELTAY-WINDOW

```

```

    ;; Variables holding information about each window in turn:

```

```

    REGION LEFT RIGHT BOTTOM TOP

```

```

    ;; When we're all done, these hold information needed to compute the final value:

```

```

    BEST-POSITION USER-MOVE-VALUE |first| (SETQ MYWIDTH (|fetch| (IL:REGION IL:WIDTH) |of| MYREG))
      (SETQ MYHEIGHT (|fetch| (IL:REGION IL:HEIGHT) |of| MYREG))
      (SETQ MYRIGHT (+ MYLEFT MYWIDTH -1))
      (SETQ MYTOP (+ MYBOTTOM MYHEIGHT -1))
      (SETQ MINLEFT (- MYLEFT TOLERANCE))
      (SETQ MAXRIGHT (+ MYRIGHT TOLERANCE))
      (SETQ MINBOTTOM (- MYBOTTOM TOLERANCE))
      (SETQ MAXTOP (+ MYTOP TOLERANCE))

```

```

    ;; First guess at best position is nearest corner of the screen

```

```

    (SETQ BEST-CORNER-DELTAX (SETQ BEST-DELTAX
      (MIN-ABSOLUTE (- IL:SCREENWIDTH
        MYRIGHT 1)
        (- MYLEFT))))

```

```

                                (SETQ BEST-CORNER-DELTAY (SETQ BEST-DELTAY
                                                                (MIN-ABSOLUTE (- IL:SCREENHEIGHT
                                                                MYTOP 1)
                                                                (- MYBOTTOM))))
|for| WINDOW |in| (IL:OPENWINDOWS) |unless| (EQ WINDOW WINDOW-TO-MOVE)
|do| (SETQ REGION (IL:WINDOWPROP WINDOW 'IL:REGION))
      (SETQ LEFT (|fetch| (IL:REGION IL:LEFT) |of| REGION))
      (SETQ RIGHT (|fetch| (IL:REGION IL:RIGHT) |of| REGION))
      (SETQ BOTTOM (|fetch| (IL:REGION IL:BOTTOM) |of| REGION))
      (SETQ TOP (|fetch| (IL:REGION IL:TOP) |of| REGION))
      MYLEFT-LEFT
      (LET ((LEFT-MYLEFT (- LEFT MYLEFT))
            (LEFT-MYRIGHT (- (- LEFT MYRIGHT)
                              SPACING))
            (RIGHT-MYLEFT (+ (- RIGHT MYLEFT)
                              SPACING))
            (RIGHT-MYRIGHT (- RIGHT MYRIGHT))
            (BOTTOM-MYBOTTOM (- BOTTOM MYBOTTOM))
            (BOTTOM-MYTOP (- (- BOTTOM MYTOP)
                              SPACING))
            (TOP-MYBOTTOM (+ (- TOP MYBOTTOM)
                              SPACING))
            (TOP-MYTOP (- TOP MYTOP)))
        ;; First, see if we can align with a corner of a window
        (|if| (AND (OR (BETWEEN BOTTOM MINBOTTOM MAXTOP)
                      (BETWEEN TOP MINBOTTOM MAXTOP))
                  (OR (BETWEEN LEFT MINLEFT MAXRIGHT)
                      (BETWEEN RIGHT MINLEFT MAXRIGHT))))
          |then| (MULTIPLE-VALUE-SETQ (BEST-CORNER-DELTAX BEST-CORNER-DELTAY)
                                     (MIN-SUM-SQUARES (BEST-CORNER-DELTAX BEST-CORNER-DELTAY)
                                     (LEFT-MYRIGHT TOP-MYTOP)
                                     (LEFT-MYRIGHT BOTTOM-MYBOTTOM)
                                     (RIGHT-MYLEFT BOTTOM-MYBOTTOM)
                                     (RIGHT-MYLEFT TOP-MYTOP)
                                     (LEFT-MYLEFT BOTTOM-MYTOP)
                                     (LEFT-MYLEFT TOP-MYBOTTOM)
                                     (RIGHT-MYRIGHT BOTTOM-MYTOP)
                                     (RIGHT-MYRIGHT TOP-MYBOTTOM)))
          ;; Now see if we can align with a side of a window
          (|if| (OR (BETWEEN MYBOTTOM BOTTOM TOP)
                   (BETWEEN MYTOP BOTTOM TOP))
            |then| (SETQ BEST-DELTAX (MIN-ABSOLUTE BEST-DELTAX LEFT-MYRIGHT RIGHT-MYLEFT))
            (|if| (OR (BETWEEN MYLEFT LEFT RIGHT)
                     (BETWEEN MYRIGHT LEFT RIGHT))
              |then| (SETQ BEST-DELTAY (MIN-ABSOLUTE BEST-DELTAY BOTTOM-MYTOP TOP-MYBOTTOM)))
            |finally| (|if| (AND (IL:WINDOWP CORNER-DELTAX-WINDOW)
                              (EQ CORNER-DELTAX-WINDOW CORNER-DELTAY-WINDOW))
              |then| ;; we might be putting my window in the corner of another window. This code is meant to prevent the window from getting
                    ;; thrown on top of another window by preventing it from aligning two of its edges with the two corresponding edges of
                    ;; another window. But it doesn't work very well.
                    (|if| (<= (+ (* BEST-DELTAX BEST-DELTAX)
                                (* BEST-CORNER-DELTAY BEST-CORNER-DELTAY))
                          (+ (* BEST-CORNER-DELTAX BEST-CORNER-DELTAX)
                              (* BEST-DELTAY BEST-DELTAY)))
                      |then| (SETQ BEST-DELTAY BEST-CORNER-DELTAY)
                      |else| (SETQ BEST-DELTAX BEST-CORNER-DELTAX))
                    |else| (SETQ BEST-DELTAX (MIN-ABSOLUTE BEST-DELTAX BEST-CORNER-DELTAX))
                          (SETQ BEST-DELTAY (MIN-ABSOLUTE BEST-DELTAY BEST-CORNER-DELTAX)))
                    (SETQ BEST-POSITION (|create| IL:POSITION
                                                IL:XCOORD _ (|if| (<= (ABS BEST-DELTAX)
                                                                    TOLERANCE)
                                                                |then| (+ MYLEFT BEST-DELTAX)
                                                                |else| MYLEFT)
                                                IL:YCOORD _ (|if| (<= (ABS BEST-DELTAY)
                                                                    TOLERANCE)
                                                                |then| (+ MYBOTTOM BEST-DELTAY)
                                                                |else| MYBOTTOM))))
              ; find result of any other MOVEFNs
              (SETQ USER-MOVE-VALUE
                (|if| (NULL USERMOVEFN)
                  |then| NIL
                  |elseif| (EQ USERMOVEFN 'IL:DON'T)
                  |then| 'IL:DON'T
                  |elseif| (LISTP USERMOVEFN)
                  |then| (|bind| (VAL _ BEST-POSITION) |for| FN |in| USERMOVEFN |until| (EQ VAL 'IL:DON'T)
                               |unless| (EQ FN 'NEAT-POSITION) |do| (SETQ VAL (OR (FUNCALL FN WINDOW-TO-MOVE VAL)
                                                                           VAL))
                               |finally| (RETURN VAL)))
                  |elseif| (AND (SYMBOL-FUNCTION USERMOVEFN)
                               (IL:NEQ USERMOVEFN 'NEAT-POSITION))
                  |then| (FUNCALL USERMOVEFN WINDOW-TO-MOVE BEST-POSITION)
                  |else| NIL))
                (RETURN (|if| (OR (EQ USER-MOVE-VALUE 'IL:DON'T)
                                (IL:POSITIONP USER-MOVE-VALUE))
                  |then| USER-MOVE-VALUE
                  |else| (RETURN VAL))

```

|else| BEST-POSITION)))))

```
(IL:PUTPROPS IL:NEATCONS IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE
(PROG1
  (DEFPACKAGE "NEATCONS" (:USE "LISP" "XCL")
    (:IMPORT |bind| |first| |for| |in| |until|
              |unless| |do| |collect| |finally|
              |if| |then| |else| |elseif| |create|
              |fetch| |of|))
    (EXPORT (MAPCAR #'(IL:LAMBDA (STRING)
      (INTERN STRING
        "NEATCONS"))
        '("DEFAULT-SPACING"
          "DEFAULT-TOLERANCE"
          "NEATEN" "UNNEATEN"
          "USERMOVEFN"))
        "NEATCONS"))))
(IL:PUTPROPS IL:NEATCONS IL:COPYRIGHT ("Quintus Computer Systems, Inc" 1967 1986 1987))
```

FUNCTION INDEX

NEAT-POSITION	2	NEATEN	1	UNNEATEN	1
---------------------	---	--------------	---	----------------	---

MACRO INDEX

BETWEEN	2	MIN-ABSOLUTE	2	MIN-SUM-SQUARES	2
---------------	---	--------------------	---	-----------------------	---

VARIABLE INDEX

DEFAULT-SPACING	1	DEFAULT-TOLERANCE	1
-----------------------	---	-------------------------	---

PROPERTY INDEX

IL:NEATICONS	4
--------------------	---

ADVICE INDEX

IL:SHRINKW	1
------------------	---