File created: 26-Jun-88 00:05:44 **{NEWTON:EUROPARC:RX}<LOVSTRAND>LISP>LYRIC>SOLID-MOVEW.;23**

changes to: (VARS SOLID-MOVEWCOMS)
(FNS SOLID-MOVEW \SOLID-MOVEW-CLOSEW-WATCHER)

previous date: 8-May-88 01:58:13 {NEWTON:EUROPARC:RX}<LOVSTRAND>LISP>LYRIC>SOLID-MOVEW.;21

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1988 by Xerox Corporation.  All rights reserved.

(RPAQQ SOLID-MOVEWCOMS
       ((INITVARS (*SOLID-MOVEW-FLAG* 15000)
              (*SOLID-MOVEW-SHADOW* T)
              (*SOLID-MOVEW-SHADOW-SHADE* 42405)
              (*SOLID-MOVEW-GRIDDING* NIL)
              (*SOLID-MOVEW-CASHING* T)
              (\SOLID-LAST-SAVMAP)
              (\SOLID-LAST-TMPMAP))
        (FNS MAYBE-SOLID-MOVEW SOLID-MOVEW \SOLID-MOVEW-CLOSEW-WATCHER)
        [APPENDVARS (GAINSPACEFORMS ((OR \SOLID-LAST-SAVMAP \SOLID-LAST-TMPMAP)
                                     "discard SOLID-MOVEW cached bitmaps"
                                     (PROGN (SETQ \SOLID-LAST-SAVMAP)
                                            (SETQ \SOLID-LAST-TMPMAP]
        [P (MOVD? (FUNCTION MOVEW)
                  (FUNCTION ORIGINAL-MOVEW))
           (CL:UNLESS [AND (BOUNDP 'LDFLG)
                           (FMEMB LDFLG '(PROP ALLPROP]
                 (MOVD (FUNCTION MAYBE-SOLID-MOVEW)
                       (FUNCTION MOVEW)))]
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              SOLID-MOVEW)
        (GLOBALVARS *SOLID-MOVEW-FLAG* *SOLID-MOVEW-SHADOW* *SOLID-MOVEW-SHADOW-SHADE* *SOLID-MOVEW-GRIDDING*
               *SOLID-MOVEW-CASHING* \SOLID-LAST-SAVMAP \SOLID-LAST-TMPMAP)))

(RPAQ? *SOLID-MOVEW-FLAG* 15000)

(RPAQ? *SOLID-MOVEW-SHADOW* T)

(RPAQ? *SOLID-MOVEW-SHADOW-SHADE* 42405)

(RPAQ? *SOLID-MOVEW-GRIDDING* NIL)

(RPAQ? *SOLID-MOVEW-CASHING* T)

(RPAQ? \SOLID-LAST-SAVMAP )

(RPAQ? \SOLID-LAST-TMPMAP )

(DEFINEQ

(MAYBE-SOLID-MOVEW
  [LAMBDA (WINDOW POSorX Y)                                  ; Edited  8-May-88 00:50 by Magic Squirrel
    (with REGION (WINDOWREGION WINDOW)
          (if (COND
                [(EQ *SOLID-MOVEW-FLAG* 'ICON)               ; Only move solidly if window is an icon.
                 (OR (WINDOWPROP WINDOW 'ICONFOR)
                     (WINDOWPROP WINDOW 'ICONIMAGE]
                [(POSITIONP *SOLID-MOVEW-FLAG*)              ; Only move solidly if window contains less than specified
                                                             ; number of bits.
                 (AND (ILEQ WIDTH (fetch XCOORD of *SOLID-MOVEW-FLAG*))
                      (ILEQ HEIGHT (fetch YCOORD of *SOLID-MOVEW-FLAG*]
                ((NUMBERP *SOLID-MOVEW-FLAG*)                ; Only move solidly if window is smaller than specified width x
                                                             ; height.
                 (ILEQ (ITIMES WIDTH HEIGHT)
                       *SOLID-MOVEW-FLAG*))
                (T                                           ; Move solidly if *SOLID-MOVEW-FLAG* says so.
                   *SOLID-MOVEW-FLAG*))
              then (SOLID-MOVEW WINDOW POSorX Y)
             else (ORIGINAL-MOVEW WINDOW POSorX Y])

(SOLID-MOVEW
  [LAMBDA (WINDOW POSorX Y)                                  ; Edited 26-Jun-88 00:04 by Lennart
    (DECLARE (GLOBALVARS LAMBDASPLST)
          (LOCALVARS . T))
    (PROG (left bottom width height shadowWidth shadowHeight savMap tmpMap x y savX savY tmpX tmpY intX intY
                savWidth savHeight tmpWidth tmpHeight intWidth intHeight image mask shade windows screen moveFns
                oldCursor buttonWait result wxOff wyOff cxOff cyOff firstTime)
      ;; Punt if position already is given
```

```
    (if POSorX
        then (RETURN (ORIGINAL-MOVEW WINDOW POSorX Y)))
```

;; Make sure this window is moveable

```
    (SETQ moveFns (MKLIST (fetch (WINDOW MOVEFN) of WINDOW)))
    (if (FMEMB (CAR moveFns)
               LAMBDASPLST)
        then (SETQ moveFns (LIST moveFns)))
    (if (FMEMB 'DON'T moveFns)
        then (PROMPTPRINT "Can't move this window.")
             (RETURN))
```

;; Optional shadowing

```
    [SETQ shadowWidth (ABS (COND
                              ((POSITIONP *SOLID-MOVEW-SHADOW*)
                               (fetch (POSITION XCOORD) of *SOLID-MOVEW-SHADOW*))
                              ((NUMBERP *SOLID-MOVEW-SHADOW*)
                               *SOLID-MOVEW-SHADOW*)
                              (*SOLID-MOVEW-SHADOW* 3)
                              (T 0]
    [SETQ shadowHeight (ABS (COND
                               ((POSITIONP *SOLID-MOVEW-SHADOW*)
                                (fetch (POSITION YCOORD) of *SOLID-MOVEW-SHADOW*))
                               ((NUMBERP *SOLID-MOVEW-SHADOW*)
                                *SOLID-MOVEW-SHADOW*)
                               (*SOLID-MOVEW-SHADOW* 3)
                               (T 0]
```

;; Set window dependent parameters.

```
    (with REGION (WINDOWREGION WINDOW)                    ; This will be the union of the window's region and all its attached
                                                          ; windows.
        (SETQ left LEFT)
        (SETQ bottom BOTTOM)
        (SETQ width WIDTH)
        (SETQ height HEIGHT))
    (SETQ savX left)
    (SETQ savY (IDIFFERENCE bottom shadowHeight))
    (SETQ savWidth (IPLUS width shadowWidth))
    (SETQ savHeight (IPLUS height shadowHeight))
    (SETQ tmpWidth (ITIMES savWidth 2))
    (SETQ tmpHeight (ITIMES savHeight 2))
```

;; Create temporary data structures.

```
    (if *SOLID-MOVEW-CASHING*
        then (LET ((lastSavWidth 0)
                   (lastSavHeight 0)
                   (lastTmpWidth 0)
                   (lastTmpHeight 0))
                  (CL:WHEN (BITMAPP \SOLID-LAST-SAVMAP)
                      (SETQ lastSavWidth (BITMAPWIDTH \SOLID-LAST-SAVMAP))
                      (SETQ lastSavHeight (BITMAPHEIGHT \SOLID-LAST-SAVMAP)))
                  (CL:WHEN (BITMAPP \SOLID-LAST-TMPMAP)
                      (SETQ lastTmpWidth (BITMAPWIDTH \SOLID-LAST-TMPMAP))
                      (SETQ lastTmpHeight (BITMAPHEIGHT \SOLID-LAST-TMPMAP)))
                  (CL:UNLESS (AND (IGEQ lastSavWidth savWidth)
                                  (IGEQ lastSavHeight savHeight))
                      (SETQ \SOLID-LAST-SAVMAP (BITMAPCREATE (IMAX savWidth lastSavWidth)
                                                             (IMAX savHeight lastSavHeight))))
                  (CL:UNLESS (AND (IGEQ lastTmpWidth tmpWidth)
                                  (IGEQ lastTmpHeight tmpHeight))
                      (SETQ \SOLID-LAST-TMPMAP (BITMAPCREATE (IMAX tmpWidth lastTmpWidth)
                                                             (IMAX tmpHeight lastTmpHeight))))
                  (SETQ savMap \SOLID-LAST-SAVMAP)
                  (SETQ tmpMap \SOLID-LAST-TMPMAP))
      else (SETQ savMap (BITMAPCREATE savWidth height))
           (SETQ tmpMap (BITMAPCREATE tmpWidth tmpHeight)))
    [SETQ windows (CONS WINDOW (ATTACHEDWINDOWS WINDOW 'MOVEW]
    (SETQ screen (fetch (WINDOW SCREEN SCDESTINATION) of WINDOW))
    (with REGION (fetch (WINDOW REG) of WINDOW)
        (SETQ wxOff (IDIFFERENCE LEFT savX))
        (SETQ wyOff (IDIFFERENCE BOTTOM savY)))
    [if (CDR windows)
        then                                              ; Tricky case, lots of windows -- form the union image & mask.
                                                          ; (no caching here yet)
             (SETQ image (BITMAPCREATE width height))
             (SETQ mask (BITMAPCREATE width height))
             [for w in windows do (with REGION (fetch (WINDOW REG) of w)
                                       (if (WINDOWPROP w 'ICONMASK)
                                           then (BITBLT (WINDOWPROP w 'ICONIMAGE)
                                                        0 0 image (IDIFFERENCE LEFT left)
                                                        (IDIFFERENCE BOTTOM bottom))
                                                (BITBLT (WINDOWPROP w 'ICONMASK)
                                                        0 0 mask (IDIFFERENCE LEFT left)
                                                        (IDIFFERENCE BOTTOM bottom))
                                         else (BITBLT screen LEFT BOTTOM image (IDIFFERENCE LEFT left)
                                                      (IDIFFERENCE BOTTOM bottom)
                                                      WIDTH HEIGHT)
```

```
                                        (BLTSHADE BLACKSHADE mask (IDIFFERENCE LEFT left)
                                               (IDIFFERENCE BOTTOM bottom)
                                               WIDTH HEIGHT]
    else                                                    ; Easy case, just one window.
         (SETQ image (OR (WINDOWPROP WINDOW 'IMAGE)
                         (WINDOWPROP WINDOW 'ICONIMAGE)
                         (fetch (WINDOW SAVE) of WINDOW)))
         [SETQ mask (OR (WINDOWPROP WINDOW 'MASK)
                        (WINDOWPROP WINDOW 'ICONMASK]
         (SETQ shade (WINDOWPROP WINDOW 'SHADEIMAGE]
    (TOTOPW WINDOW)
    (BITBLT screen savX savY savMap 0 0 savWidth savHeight)
                                                            ; Save screen image around attached windows (concave
                                                            ; corners' case).
    (for w in windows when (OPENWP w) do (UNINTERRUPTABLY
                                        (\INTERNALTOTOPW w)
                                                            ; Make sure the window is softly on top...
                                        (with REGION (fetch (WINDOW REG) of w)
                                             (BITBLT (fetch (WINDOW SAVE) of w)
                                                    0 0 savMap (IDIFFERENCE LEFT savX)
                                                    (IDIFFERENCE BOTTOM savY))
                                                            ; Save the screen image behind the window.
                                             (BITBLT screen LEFT BOTTOM (fetch (WINDOW SAVE)
                                                                               of w)
                                                    0 0)
                                                            ; Put a copy of the window's image in the SAVE map so that
                                                            ; closing it won't produce any flickering.

                                             )
                                        (\CLOSEW1 w))  ; Then softly close the target window to make it "movable".

                                    )
    (CL:UNWIND-PROTECT
        [first [SETQ oldCursor (CURSOR (CONSTANT (CURSORCREATE (BITMAPCREATE 0 0]
               (SETQ cxOff (IDIFFERENCE LASTMOUSEX savX))
               (SETQ cyOff (IDIFFERENCE LASTMOUSEY savY))
               (\CURSORPOSITION savX savY)
               (GETMOUSESTATE)
               (SETQ buttonWait T)
               (SETQ firstTime T) eachtime (GETMOUSESTATE)
                                  (SETQ x LASTMOUSEX)
                                  (SETQ y LASTMOUSEY)
         while (if (LASTMOUSESTATE UP)
                   then buttonWait
                 else (SETQ buttonWait NIL)
                      (LASTMOUSESTATE (NOT UP)))
         when (OR (NEQ savX x)
                  (NEQ savY y)
                  firstTime)
         do (SETQ firstTime NIL)
            (if (AND *SOLID-MOVEW-GRIDDING* (FMEMB (FUNCTION ICONW.MOVEFN)
                                                   moveFns))
                then ;; Handle gridded icons here.
                     (with POSITION (ICONW.MOVEFN WINDOW (CREATEPOSITION x y))
                           (SETQ x XCOORD)
                           (SETQ y YCOORD)))
            (COND
               ((OR (IGEQ (ABS (IDIFFERENCE x savX))
                          savWidth)
                    (IGEQ (ABS (IDIFFERENCE y savY))
                          savHeight))
                ;; This jump is large enough not to have the window intersect with itself.
                (BITBLT savMap 0 0 screen savX savY savWidth savHeight)
                (UNINTERRUPTABLY
                    (BITBLT screen x y savMap 0 0 savWidth savHeight)
                    (SETQ savX x)
                    (SETQ savY y))
                (BITBLT screen x y tmpMap 0 0 tmpWidth tmpHeight)
                (if mask
                    then (AND *SOLID-MOVEW-SHADOW* (BITBLT mask 0 0 tmpMap shadowWidth 0 width height
                                                           'MERGE
                                                           'PAINT *SOLID-MOVEW-SHADOW-SHADE*))
                         (BITBLT mask 0 0 tmpMap 0 shadowHeight width height NIL 'ERASE)
                         (BITBLT image 0 0 tmpMap 0 shadowHeight width height NIL 'PAINT)
                  else (AND *SOLID-MOVEW-SHADOW* (BLTSHADE *SOLID-MOVEW-SHADOW-SHADE* tmpMap shadowWidth
                                                           0 width height 'PAINT))
                       (BITBLT image 0 0 tmpMap 0 shadowHeight width height))
                (AND shade (BITBLT shade 0 0 tmpMap 0 shadowHeight width height NIL 'PAINT))
                (BITBLT tmpMap 0 0 screen x y tmpWidth tmpHeight))
               (T ;; The new image intersects with the old.
                  (SETQ tmpX (IMIN x savX))
                  (SETQ tmpY (IMIN y savY))
                  (SETQ intX (IDIFFERENCE x tmpX))
                  (SETQ intY (IDIFFERENCE y tmpY))
```

```
                        (SETQ intWidth (IPLUS (ABS (IDIFFERENCE x savX))
                                              savWidth))
                        (SETQ intHeight (IPLUS (ABS (IDIFFERENCE y savY))
                                               savHeight))
                        (BITBLT screen tmpX tmpY tmpMap 0 0 intWidth intHeight)
                        (BITBLT savMap 0 0 tmpMap (IDIFFERENCE savX tmpX)
                               (IDIFFERENCE savY tmpY)
                               savWidth savHeight)
                        (UNINTERRUPTABLY
                            (BITBLT tmpMap intX intY savMap 0 0 savWidth savHeight)
                            (SETQ savX x)
                            (SETQ savY y))
                        (if mask
                            then (AND *SOLID-MOVEW-SHADOW* (BITBLT mask 0 0 tmpMap (IPLUS intX shadowWidth)
                                                                   intY width height 'MERGE 'PAINT
                                                                   *SOLID-MOVEW-SHADOW-SHADE*))
                                 (BITBLT mask 0 0 tmpMap intX (IPLUS intY shadowHeight)
                                        width height NIL 'ERASE)
                                 (BITBLT image 0 0 tmpMap intX (IPLUS intY shadowHeight)
                                        width height NIL 'PAINT)
                          else (AND *SOLID-MOVEW-SHADOW* (BLTSHADE *SOLID-MOVEW-SHADOW-SHADE* tmpMap
                                                                   (IPLUS intX shadowWidth)
                                                                   intY width height 'PAINT))
                               (BITBLT image 0 0 tmpMap intX (IPLUS intY shadowHeight)
                                      width height))
                        (AND shade (BITBLT shade 0 0 tmpMap intX (IPLUS intY shadowHeight)
                                           width height NIL 'PAINT))
                        (BITBLT tmpMap 0 0 screen tmpX tmpY intWidth intHeight]
                ;; Finally cleanup before we exit, ie. restore cursor, put original bits back on the screen, open the windows again...
                (\CURSORPOSITION (IPLUS savX cxOff)
                       (IPLUS savY cyOff))
                (CURSOR oldCursor)
                ;; Blink, blink...
                (BITBLT savMap 0 0 screen savX savY savWidth savHeight)
                ;; Only move if the operation was completed, ie not if user aborted by hitting ^E etc
                (CL:WHEN (LASTMOUSESTATE UP)
                    (WINDOWADDPROP WINDOW 'CLOSEFN '\SOLID-MOVEW-CLOSEW-WATCHER)
                    (SETQ result (ORIGINAL-MOVEW WINDOW (IPLUS x wxOff)
                                                 (IPLUS y wyOff)))
                    (WINDOWDELPROP WINDOW 'CLOSEFN '\SOLID-MOVEW-CLOSEW-WATCHER))
                (if (WINDOWPROP WINDOW 'SOLID-CLIENT-CLOSED)
                    then ;; Some side effect of the (original) movew explicitly closed the window -- don't reopen it.
                         (WINDOWPROP WINDOW 'SOLID-CLIENT-CLOSED NIL)
                  else (MAPC (REVERSE windows)
                             #'\OPENW1))
                (DOUSERFNS (WINDOWPROP WINDOW 'TOTOPFN)
                       WINDOW))
            (RETURN result])
```

(\SOLID-MOVEW-CLOSEW-WATCHER

```
  [LAMBDA (WINDOW)                                          ; Edited 25-Jun-88 23:58 by Lennart
    (WINDOWPROP WINDOW 'SOLID-CLIENT-CLOSED T])

)

[APPENDTOVAR GAINSPACEFORMS ((OR \SOLID-LAST-SAVMAP \SOLID-LAST-TMPMAP)
                             "discard SOLID-MOVEW cached bitmaps"
                             (PROGN (SETQ \SOLID-LAST-SAVMAP)
                                    (SETQ \SOLID-LAST-TMPMAP]

(MOVD? (FUNCTION MOVEW)
       (FUNCTION ORIGINAL-MOVEW))

(CL:UNLESS [AND (BOUNDP 'LDFLG)
                (FMEMB LDFLG '(PROP ALLPROP]
    (MOVD (FUNCTION MAYBE-SOLID-MOVEW)
          (FUNCTION MOVEW)))

(PUTPROPS SOLID-MOVEW FILETYPE :COMPILE-FILE)

(PUTPROPS SOLID-MOVEW MAKEFILE-ENVIRONMENT (:READTABLE "INTERLISP" :PACKAGE "INTERLISP"))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS *SOLID-MOVEW-FLAG* *SOLID-MOVEW-SHADOW* *SOLID-MOVEW-SHADOW-SHADE* *SOLID-MOVEW-GRIDDING*
       *SOLID-MOVEW-CASHING* \SOLID-LAST-SAVMAP \SOLID-LAST-TMPMAP)
)

(PUTPROPS SOLID-MOVEW COPYRIGHT ("Xerox Corporation" 1988))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX