File created: 2-Mar-87 16:07:23 **{ERIS}<LISPUSERS>LYRIC>BACKGROUNDIMAGES.;6**

changes to: (FNS BACKGROUND.SETUP)
(VARS BACKGROUNDIMAGESCOMS)

previous date: 11-Feb-87 21:26:26 {ERIS}<LISPUSERS>LYRIC>BACKGROUNDIMAGES.;5

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ BACKGROUNDIMAGESCOMS
       [

;;; Enables you to load interesting backgrounds.  Simplest entry is just to call (BACKGROUND.SETUP).

         (FNS BACKGROUND.SETUP BACKGROUND.FILES BACKGROUND.FILE BACKGROUND.FETCH BACKGROUND.SHORTNAME
              BACKGROUND.MODE BACKGROUND.SHADE)
         (FNS BACKGROUND.CENTER BACKGROUND.REFLECT BACKGROUND.TILE BACKGROUND.LESS)
         (INITVARS (BACKGROUNDS NIL)
                (BACKGROUND.MODE 'CENTER)
                (BACKGROUND.SHADE 34850))
         (GLOBALVARS BACKGROUNDS BackgroundMenuCommands LISPUSERSDIRECTORIES BACKGROUND.MODE BACKGROUND.SHADE)
         [ADDVARS (GAINSPACEFORMS ((LISTP BACKGROUNDS)
                                   "Delete saved background bitmaps"
                                   (SETQ BACKGROUNDS NIL]
         (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                               (NLAML)
                                                                               (LAMA])


;;; Enables you to load interesting backgrounds.  Simplest entry is just to call (BACKGROUND.SETUP).

(DEFINEQ

(BACKGROUND.SETUP
  [LAMBDA (NAMES)                                            ; Edited  2-Mar-87 15:57 by Stansbury

;;; Background decoration.  Puts stuff on the background menu that will let you stick up fun backgrounds on the screen.

    (LET
     [(IMAGES (if (LISTP NAMES)
                  then NAMES
                else (BACKGROUND.FILES NAMES]
     (if (LISTP IMAGES)
         then
          (FILESLOAD (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
                 BACKGROUNDMENU)
          (BkgMenu.remove.item 'Background)
          [BkgMenu.add.item
           `(Background '(CHANGEBACKGROUND BACKGROUND.SHADE)
                    "Change background"
                    (SUBITEMS [Change '(CHANGEBACKGROUND BACKGROUND.SHADE)
                                     "Change background"
                                     ,(CONS 'SUBITEMS
                                             (for IMAGE in IMAGES
                                                collect (LET ((NAME (if (LISTP IMAGE)
                                                                        then (CAR IMAGE)
                                                                      else IMAGE))
                                                              (FILENAME (if (LISTP IMAGE)
                                                                            then (CDR IMAGE)
                                                                          else NIL)))
                                                             (LIST NAME `'(BACKGROUND.FETCH (QUOTE %, NAME)
                                                                                 (QUOTE %, FILENAME)
                                                                                 BACKGROUND.MODE)
                                                                   (CONCAT "Change background to " NAME]
                              [Mode '(PROMPTPRINT (BACKGROUND.MODE))
                                    "Change mode of applying background images"
                                    (SUBITEMS [Center '(BACKGROUND.MODE 'CENTER]
                                            [Tile '(BACKGROUND.MODE 'TILE]
                                            (Reflect '(BACKGROUND.MODE 'REFLECT]
                              (Shade '(BACKGROUND.SHADE (EDITSHADE BACKGROUND.SHADE))
                                     "Change the default background shade"]
          (BkgMenu.fixup)
          T
       else NIL])

(BACKGROUND.FILES
  [LAMBDA (WHICH)                                            ; Edited 11-Feb-87 20:29 by Stansbury
```

```
;;; Returns a list of names of press files which contain background-sized images

    (for filename in [SELECTQ WHICH
                        (T ;; Find all images on all lispusersdirectories

                            (for DIR in LISPUSERSDIRECTORIES bind IMAGES first (SETQ IMAGES NIL)
                               do (for image in (APPEND (FILDIR (PACK* DIR "background-*.bitmap"))
                                                        (FILDIR (PACK* DIR "background-*.press")))
                                     do (pushnew IMAGES image))
                               finally (RETURN (SORT IMAGES))))
                        (PROGN ;; Find just the clump of images on the first lispusersdirectory that has any images on it.  (Useful because
                               ;; usually images will be on just one lispusersdirectory.)

                               (for DIR in LISPUSERSDIRECTORIES
                                  do (LET [(images (APPEND (FILDIR (PACK* DIR "background-*.bitmap"))
                                                           (FILDIR (PACK* DIR "background-*.press"]
                                          (if (LISTP images)
                                              then (RETURN images]
       collect (CONS (BACKGROUND.SHORTNAME filename)
                     filename])

(BACKGROUND.FILE
  [LAMBDA (NAME)                                                  ; Edited 11-Feb-87 20:29 by Stansbury

;;; Finds the file containing a press encoding of the named background.

    (for DIR in LISPUSERSDIRECTORIES do (LET ((BITMAP.FILENAME (PACKFILENAME 'DIRECTORY DIR 'NAME
                                                                         (CONCAT "background-" NAME)
                                                                         'EXTENSION "bitmap"))
                                              (PRESS.FILENAME (PACKFILENAME 'DIRECTORY DIR 'NAME (CONCAT
                                                                                                  "backgrou
nd-" NAME)
                                                                         'EXTENSION "press")))
                                             (if (INFILEP BITMAP.FILENAME)
                                                 then (RETURN BITMAP.FILENAME)
                                               elseif (INFILEP PRESS.FILENAME)
                                                 then (RETURN PRESS.FILENAME])

(BACKGROUND.FETCH
  [LAMBDA (NAME FILENAME MODE)                                    ; Edited 11-Feb-87 20:30 by Stansbury

;;; Puts up the specified background.  If it is cached, just grabs it off the cache;  else reads the press file off the server, translates it into a bitmap, slams it
;;; up, and caches it.

    (LET ((BITMAP (LISTGET BACKGROUNDS NAME)))
         [if (NOT (BITMAPP BITMAP))
             then ;; Find background: either off a Lisp bitmap file, or off an old Press file

                  (CLRPROMPT)
                  (PRINTOUT PROMPTWINDOW "Fetching background " NAME " ... ")
                  (if (NULL FILENAME)
                      then (SETQ FILENAME (BACKGROUND.FILE NAME)))
                  (if (OR (NULL FILENAME)
                          (NOT (INFILEP FILENAME)))
                      then (PROMPTPRINT "Background " FILENAME " not available.")
                    else (if (PRESSFILEP FILENAME)
                             then (FILESLOAD (SYSLOAD FROM VALUEOF LISPUSERSDIRECTORIES)
                                         BITMAPFNS)
                                  (SETQ BITMAP (READPRESS FILENAME))
                           else (LET [(STREAM (OPENSTREAM FILENAME 'INPUT]
                                     (SETQ BITMAP (HREAD STREAM))
                                     (CLOSEF STREAM)))
                         (PRINTOUT PROMPTWINDOW "done." T)

                         ;; Cache it (before modifying it)

                         (if (LISTP BACKGROUNDS)
                             then (LISTPUT BACKGROUNDS NAME BITMAP)
                           else (SETQ BACKGROUNDS (LIST NAME BITMAP]

         ;; Adjust bitmap and apply to background of screen

         (PRINTOUT PROMPTWINDOW "Adjusting background ... ")
         (SETQ BITMAP (SELECTQ MODE
                          (TILE (BACKGROUND.TILE BITMAP))
                          (REFLECT (BACKGROUND.REFLECT BITMAP))
                          ((NIL CENTER)
                               (BACKGROUND.CENTER BITMAP))
                          (\ILLEGAL.ARG MODE)))
         (CHANGEBACKGROUND BITMAP)
         (PRINTOUT PROMPTWINDOW "done." T)
         BITMAP])

(BACKGROUND.SHORTNAME
  [LAMBDA (IMAGE)                                                 ; Edited 11-Feb-87 20:30 by Stansbury
```

;;; Parses the IMAGE file name to find the short name of a background. IMAGE file names are of the form
;;; {server}<directory>SHORTNAME-background.press

```
    (MKATOM (L-CASE (LET [(LONGNAME (FILENAMEFIELD IMAGE 'NAME]
                         (SUBSTRING LONGNAME (LET ((start (STRPOS "-" LONGNAME)))
                                                  (if (FIXP start)
                                                      then (ADD1 start)
                                                    else start))
                                NIL))
                    T])
```

(**BACKGROUND.MODE**
```
  [LAMBDA (NEWVAL)                                            ; Edited 11-Feb-87 20:42 by Stansbury
```

;;; Finds the value of or resets the background image applying mode.

```
    (if (NULL NEWVAL)
        then BACKGROUND.MODE
      else (SELECTQ NEWVAL
                ((CENTER TILE REFLECT)
                     (PROG1 BACKGROUND.MODE (SETQ BACKGROUND.MODE NEWVAL)))
                (\ILLEGAL.ARG NEWVAL])
```

(**BACKGROUND.SHADE**
```
  [LAMBDA (NEW-SHADE)                                         ; Edited 11-Feb-87 21:26 by Stansbury
```

;;; returns the old value of the default background shade. Also, if new-shade is a texture, makes it be the new default background shade.

```
    (if (NULL NEW-SHADE)
        then BACKGROUND.SHADE
      elseif (TEXTUREP NEW-SHADE)
        then (PROG1 BACKGROUND.SHADE (SETQ BACKGROUND.SHADE NEW-SHADE))
      else (\ILLEGAL.ARG NEW-SHADE])

)

(DEFINEQ
```

(**BACKGROUND.CENTER**
```
  [LAMBDA (BITMAP)                                            ; Edited 11-Feb-87 21:12 by Stansbury
```

;;; Returns a new bitmap the size of the screen which has the argument bitmap centered in it and a gray border. This will center the bitmap on the
;;; screen, regardless of the screen size.

```
    (LET ((NEWBITMAP (BITMAPCREATE SCREENWIDTH SCREENHEIGHT 1))
          (X (QUOTIENT (DIFFERENCE SCREENWIDTH (BITMAPWIDTH BITMAP))
                     2))
          (Y (QUOTIENT (DIFFERENCE SCREENHEIGHT (BITMAPHEIGHT BITMAP))
                     2)))
         (BLTSHADE BACKGROUND.SHADE NEWBITMAP)
         (BITBLT BITMAP 1 1 NEWBITMAP X Y)
         NEWBITMAP])
```

(**BACKGROUND.REFLECT**
```
  [LAMBDA (BITMAP)                                            ; Edited 11-Feb-87 20:56 by Stansbury
```

;;; Centers BITMAP on a screen-sized bitmap and tiles the remaining space with reflections of BITMAP

```
    (LET* ((WIDTH (BITMAPWIDTH BITMAP))
           (HEIGHT (BITMAPHEIGHT BITMAP))
           (MAXWIDTH (TIMES 3 WIDTH))
           (MAXHEIGHT (TIMES 2 HEIGHT))
           (TOO.SMALL (OR (GREATERP SCREENWIDTH MAXWIDTH)
                          (GREATERP SCREENHEIGHT MAXHEIGHT)))
           (NEWBITMAP (BITMAPCREATE (if TOO.SMALL
                                        then MAXWIDTH
                                      else SCREENWIDTH)
                             (if TOO.SMALL
                                 then MAXHEIGHT
                               else SCREENHEIGHT)
                             1))
           (X (IQUOTIENT (DIFFERENCE (BITMAPWIDTH NEWBITMAP)
                                WIDTH)
                     2))
           (Y (if (GREATERP HEIGHT (BITMAPHEIGHT NEWBITMAP))
                  then (IQUOTIENT (DIFFERENCE (BITMAPHEIGHT NEWBITMAP)
                                         HEIGHT)
                              2)
                else 0)))
          ;; Stick original bitmap in middle
          (BITBLT BITMAP NIL NIL NEWBITMAP X Y)
          (if (OR (GREATERP SCREENWIDTH WIDTH)
                  (GREATERP SCREENHEIGHT HEIGHT))
                then ;; Build reflections
                     (LET ((HORIZ (BITMAPCREATE WIDTH HEIGHT 1))
                           (VERT (BITMAPCREATE WIDTH HEIGHT 1))
                           (HORIZ.VERT (BITMAPCREATE WIDTH HEIGHT 1)))
                          (for I from 0 to (SUB1 WIDTH) do (BITBLT BITMAP I 0 HORIZ (DIFFERENCE (SUB1 WIDTH)
                                                                                              I)
                                                                  0 1 HEIGHT))
                          (for I from 0 to (SUB1 HEIGHT) do (BITBLT BITMAP 0 I VERT 0 (DIFFERENCE (SUB1 HEIGHT)
                                                                                                I)
                                                                   WIDTH 1))
                          (for I from 0 to (SUB1 HEIGHT) do (BITBLT HORIZ 0 I HORIZ.VERT 0 (DIFFERENCE (SUB1 HEIGHT)
                                                                                                     I)
                                                                   WIDTH 1))
                          ;; Upper left hand corner
                          (BITBLT HORIZ.VERT NIL NIL NEWBITMAP (DIFFERENCE X WIDTH)
                                 (PLUS Y HEIGHT))
                          ;; Above, center
                          (BITBLT VERT NIL NIL NEWBITMAP X (PLUS Y HEIGHT))
                          ;; Upper right hand corner
                          (BITBLT HORIZ.VERT NIL NIL NEWBITMAP (PLUS X WIDTH)
                                 (PLUS Y HEIGHT))
                          ;; left
                          (BITBLT HORIZ NIL NIL NEWBITMAP (DIFFERENCE X WIDTH)
                                 Y)
                          ;; Right
                          (BITBLT HORIZ NIL NIL NEWBITMAP (PLUS X WIDTH)
                                 Y)
                          ;; If resulting reflected bitmap is still too small, recurse till it gets as big as the screen.
                          (if TOO.SMALL
                              then (BACKGROUND.REFLECT NEWBITMAP)
                            else NEWBITMAP))
              else NEWBITMAP])
```

```
(BACKGROUND.TILE
  [LAMBDA (BITMAP)                                                  (* hts%: " 1-Apr-86 18:13")
    (bind (NEWBITMAP _ (BITMAPCREATE SCREENWIDTH SCREENHEIGHT 1)) for LEFT from (BACKGROUND.LESS SCREENWIDTH
                                                                                       (BITMAPWIDTH BITMAP))
       by (BITMAPWIDTH BITMAP) to SCREENWIDTH do (for BOTTOM from (if (GREATERP (BITMAPHEIGHT BITMAP)
                                                                             SCREENHEIGHT)
                                                                    then (BACKGROUND.LESS SCREENHEIGHT
                                                                                (BITMAPHEIGHT BITMAP))
                                                                  else 0)
                                                    by (BITMAPHEIGHT BITMAP) to SCREENHEIGHT
                                                    do (BITBLT BITMAP NIL NIL NEWBITMAP LEFT BOTTOM))
       finally (RETURN NEWBITMAP])
```

```
(BACKGROUND.LESS
  [LAMBDA (BOXSIZE IMAGESIZE)                                       ; Edited 11-Feb-87 20:56 by Stansbury

;;; Tells where you have to start drawing to end up with a centered, tiled image

    (bind START first (SETQ START (ADD1 (QUOTIENT (DIFFERENCE BOXSIZE IMAGESIZE)
                                                 2)))
       until (LEQ START 1) do (add START (MINUS IMAGESIZE)) finally (RETURN START])

)

(RPAQ? BACKGROUNDS NIL)

(RPAQ? BACKGROUND.MODE 'CENTER)

(RPAQ? BACKGROUND.SHADE 34850)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS BACKGROUNDS BackgroundMenuCommands LISPUSERSDIRECTORIES BACKGROUND.MODE BACKGROUND.SHADE)
)

(ADDTOVAR GAINSPACEFORMS ((LISTP BACKGROUNDS)
                          "Delete saved background bitmaps"
                          (SETQ BACKGROUNDS NIL)))

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )
(ADDTOVAR LAMA )
)

(PUTPROPS BACKGROUNDIMAGES COPYRIGHT ("Xerox Corporation" 1986 1987))
```

FUNCTION INDEX

VARIABLE INDEX