File created: 23-Dec-86 12:31:59 **{QV}<NOTECARDS>1.3L>LIBRARY>SEDIT-IMAGEOBJ.;2**

changes to: (VARS SEDIT-IMAGEOBJCOMS)
(FUNCTIONS _EditNode _EditSelection _ImageObj)

previous date: 18-Nov-86 17:49:45 {PHYLUM}<DEFGROUPS>NOTECARDS>SEDIT-IMAGEOBJ.;16

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ SEDIT-IMAGEOBJCOMS
       ((* File created by MILLER)
        (FNS ExtendEditENVtoHandleImageObj UNWIND-PROTECT \\initialize.imageObject \\linearize.imageObject
             \\parse..imageObject \\pretty.print.to.tedit \\set.all.widths \\set.selection.imageObject)
        (GLOBALVARS \\type.imageObject)
        (FUNCTIONS _EditNode _EditSelection _ImageObj)
                                                                    ; really 'defmacro's
        (P (\\initialize.imageObject))
        (VARS \\right.margin.in.points.default)))
```

(* * File created by MILLER)

```
(DEFINEQ
```

(ExtendEditENVtoHandleImageObj

```
  [LAMBDA (EditEnv)                                                 (* MarkM " 6-Nov-86 20:04")
```

;; Given an existing EditENV, extend it to handle Image Objects correctly (as an ImageHost) if it doesn't already know about them

```
    (LET ((parseInfo (fetch ParseInfo of EditEnv)))
         (OR (LISTGET parseInfo 'IMAGEOBJ)
             (LISTPUT parseInfo 'IMAGEOBJ '\\parse..imageObject])
```

(UNWIND-PROTECT

```
  [LAMBDA (value after)                                             (* MarkM " 6-Nov-86 20:06")
    value])
```

(\\initialize.imageObject

```
  [LAMBDA NIL                                                       ; Edited 23-Dec-86 11:59 by kirk:
```

;; Extend the standard editing environment to know to create an imageObject EditNode when it sees an imageObject

```
    (SETQ \\type.imageObject (create EditNodeType using \\type.unknown Name _ 'imageObject Linearize _
                                                        '\\linearize.imageObject SetSelection _
                                                        '\\set.selection.imageObject))
    (ExtendEditENVtoHandleImageObj (if (BOUNDP '\\lisp.edit.environment)
                                       then \\lisp.edit.environment
                                     elseif (BOUNDP '*EDIT-ENVIRONMENT*)
                                       then *EDIT-ENVIRONMENT*
                                     elseif (BOUNDP '\\standard.env)
                                       then \\standard.env
                                     else (SHOULDNT "can't find the standard editing env"])
```

(\\linearize.imageObject

```
  [LAMBDA (node context index)                                      (* MarkM " 6-Nov-86 20:12")
```

;; The linearize method for imageObjects. Since what I need to do is output a bitmap, I create the appropriate size of bitmap, create a display
;; stream for it, and ask the imageObject to display itself into that display stream.

```
    (LET* ((imageObj (fetch Structure of node))
           (imageBox (PROGN
```

;; The NILs for CurrentX & RightMargin are consistent with Sketch & Grapher's use of ImageObjects, although,
;; SEdit could actually pass these. Feel free to fix it to do so

```
                     (_ImageObj
                      imageObj IMAGEBOXFN NIL NIL)))
           (xSize (fetch (IMAGEBOX XSIZE) of imageBox))
           (ySize (fetch (IMAGEBOX YSIZE) of imageBox))
           (yDesc (fetch (IMAGEBOX YDESC) of imageBox))
           (bitmap (BITMAPCREATE xSize ySize))
           (displayStream (DSPCREATE bitmap)))
          (DSPYOFFSET yDesc displayStream)
```

;; The NILs for ImageStreamType & HostStream are consistent with Sketch & Grapher's use of ImageObjects

```
          (_ImageObj
           imageObj DISPLAYFN displayStream NIL NIL)
          (\\output.bitmap context (CONS yDesc bitmap])
```

(\\parse..imageObject

```
  [LAMBDA (structure context mode)                                    (* MarkM " 6-Nov-86 20:12")
    ;; Parse an image object by creating and installing a node of the corresponding type
    (\\build.node structure context \\type.imageObject mode)
    (LET* ((new.node (fetch CurrentNode of context))
           (imageBox (_ImageObj
                       structure IMAGEBOXFN))
           (width (fetch (IMAGEBOX XSIZE) of imageBox)))
          ;; What is this?  (it comes from \\parse..gap) (replace LinearForm of new.node with (CONS (fetch LinearItem of structure) (fetch SelfLink of
          ;; new.node)))
          (\\set.all.widths new.node width])
```

(**pretty.print.to.tedit**

```
  [LAMBDA (structure tedit width.in.points)                           (* MarkM " 6-Nov-86 20:13")
    ;; Pretty prints into a TEdit window, process, or stream.  If the tedit argument isn't supplied, a new text stream is created.  If width.in.points is NIL,
    ;; then we try to use the width of the TEdit paraLooks or window.  Returns the textStream.
    (LET* [(textStream (if tedit
                           then (TEXTSTREAM tedit)
                         else (OPENTEXTSTREAM)))
           (textObj (TEXTOBJ textStream))
           (textWindow (fetch SELWINDOW of textObj))
           (paraLooks (TEDIT.GET.PARALOOKS textStream NIL))
           (leftMargin (LISTGET paraLooks '1STLEFTMARGIN))
           (rightMargin (LISTGET paraLooks 'RIGHTMARGIN]
          (if (EQP rightMargin 0)
              then (SETQ rightMargin (if textWindow
                                         then (WINDOWPROP textWindow 'WIDTH)
                                       else \\right.margin.in.points.default)))
          (OR width.in.points (SETQ width.in.points (DIFFERENCE rightMargin leftMargin)))
          (TEDIT.INSERT textStream " ")
          (TEDIT.SETSEL textStream (DIFFERENCE (TEDIT.GETPOINT textStream)
                                               1)
                0)
          (\\pretty.print structure textStream (TIMES MICASPERPT width.in.points))
          textStream])
```

(**set.all.widths**

```
  [LAMBDA (node width)                                                (* MarkM " 6-Nov-86 20:13")
    ;; Sets all the widths of node to width.  Should be used by \\parse..gap
    (replace InlineWidth of node with width)
    (replace PreferredWidth of node with width)
    (replace MinWidth of node with width)
    (replace PreferredLLength of node with width)
    (replace MinLLength of node with width)
    (replace ActualWidth of node with width)
    (replace ActualLLength of node with width])
```

(**set.selection.imageObject**

```
  [LAMBDA (selection context node index offset item type)             (* MarkM " 6-Nov-86 20:14")
    ;; Someone may have just buttoned inside the image object.  If so, pass it the button event and (perhaps) select it.
    ;; Calling \\set.selection.me is our way of doing (run-super)
    (if (FIXP offset)
        then ;; The NILs for Selection, RelX, RelY, Window, HostStream, and Button are consistent with Sketch's use of ImageObjects (as is the fact
             ;; that we pass the window as the WindowStream)
             (SELECTQ (_ImageObj
                        (fetch Structure of node)
                        BUTTONEVENTINFN
                        (fetch DisplayWindow of context)
                        NIL NIL NIL NIL NIL NIL)
                  (NIL (\\set.selection.me selection context node))
                  (DON'T                                              ; ignore it
                         NIL)
                  (CHANGED (\\note.change node context)
                           (\\set.selection.me selection context node))
                  (SHOULDNT "unknown return value"))
      else (\\set.selection.me selection context node])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \\type.imageObject)
)


(DEFMACRO _EditNode (self selector &REST args)
   `(APPLY* [fetch ,selector of (fetch NodeType of (fetch SelectNode of ,self]
            ,@(CONS self args)))
```

```
(DEFMACRO _EditSelection (self selector &REST args)
   `(APPLY* [fetch ,selector of (fetch NodeType of (fetch SelectNode of ,self]
            ,@(CONS self args)))


(DEFMACRO _ImageObj (self selector &REST args)
   `(APPLY* (IMAGEOBJPROP ,self ',selector)
            ,@(CONS self args)))
```

;; really 'defmacro's

```
(\\initialize.imageObject)

(RPAQQ \\right.margin.in.points.default 300)

(PUTPROPS SEDIT-IMAGEOBJ COPYRIGHT ("Xerox Corporation" 1986))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX