File created: 14-Mar-2021 20:40:30 **{DSK}<Users>kaplan>Local>medley3.5>git-medley>library>GRAPHER.;5**

changes to: (VARS GRAPHERCOMS)

previous date: 14-May-2018 10:24:38 {DSK}<Users>kaplan>Local>medley3.5>git-medley>library>GRAPHER.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1983-1994, 2018, 2021 by Venue & Xerox Corporation.

(RPAQQ GRAPHERCOMS
       [(COMS                                                    ; Graph Editing
             (FNS ADD/AND/DISPLAY/LINK APPLYTOSELECTEDNODE CALL.MOVENODEFN CHANGE.NODEFONT.SIZE
                  DEFAULT.ADDNODEFN DELETE/AND/DISPLAY/LINK DISPLAY/NAME DISPLAYGRAPH DISPLAYLINK
                  DISPLAYLINK/BT DISPLAYLINK/LR DISPLAYLINK/RL DISPLAYLINK/TB DISPLAYNODE ERASE/GRAPHNODE
                  DISPLAYNODE DISPLAYNODELINKS DRAW/GRAPHNODE/BORDER DRAWAREABOX EDITADDLINK EDITADDNODE
                  EDITAPPLYTOLINK EDITCHANGEFONT EDITCHANGELABEL EDITDELETELINK EDITDELETENODE EDITGRAPH
                  EDITGRAPH1 EDITGRAPH2 EDITMOVENODE EDITTOGGLEBORDER EDITTOGGLELABEL FILL/GRAPHNODE/LABEL
                  FIX/SCALE FLIPNODE FONTNAMELIST FROMLINKS GETNODEFROMID GN/BOTTOM GN/LEFT GN/RIGHT GN/TOP
                  GRAPHADDLINK GRAPHADDNODE GRAPHBUTTONEVENTFN GRAPHCHANGELABEL GRAPHDELETELINK GRAPHDELETENODE
                  GRAPHEDITCOMMANDFN GRAPHEDITEVENTFN GRAPHER/CENTERPRINTINAREA GRAPHERPROP
                  GRAPHNODE/BORDER/WIDTH GRAPHREGION HARDCOPYGRAPH INTERSECT/REGIONP/LBWH
                  INVERTED/GRAPHNODE/BORDER INVERTED/SHADE/FOR/GRAPHER LAYOUT/POSITION LINKPARAMETERS MAX/RIGHT
                  MAX/TOP MEASUREGRAPHNODE MEMBTONODES MIN/BOTTOM MIN/LEFT MOVENODE NODECREATE NODELST/AS/MENU
                  NODEREGION PRINTDISPLAYNODE PROMPTINWINDOW READ/NODE REDISPLAYGRAPH REMOVETONODES
                  RESET/NODE/BORDER RESET/NODE/LABELSHADE SCALE/GRAPH SCALE/GRAPHNODE/BORDER SCALE/TONODES
                  SET/LABEL/SIZE SET/LAYOUT/POSITION SHOWGRAPH SIZE/GRAPH/WINDOW TOGGLE/DIRECTEDFLG
                  TOGGLE/SIDESFLG TOLINKS TRACKCURSOR TRACKNODE TRANSGRAPH)
             [P (* Was MODERNIZE loaded before?)
                (CL:WHEN (GETD 'MODERNWINDOW.SETUP)
                    (MODERNWINDOW.SETUP 'APPLYTOSELECTEDNODE))]
             ;; Support for EDITSUBGRAPH and EDITREGION
             (FNS EDITMOVEREGION EDITMOVESUBTREE NOT.TRACKCURSOR RECURSIVE.COLLECTDESCENDENTS MOVEDESCENDENTS
                  COLLECT.CHILD.NODES CREATE.NEW.NODEPOSITION GETBOXPOSITION.FROMINITIALREGION
                  COLLECTDESCENDENTS))
       (COMS                                                   ; functions for finding larger and smaller fonts
             (FNS NEXTSIZEFONT DECREASING.FONT.LIST SCALE.FONT)
             [DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (DECREASING.FONT.LIST (DECREASING.FONT.LIST]
             (GLOBALVARS DECREASING.FONT.LIST))
                                                               ; functions for LAYOUTGRAPH And LAYOUTLATTICE
       (FNS BRH/LAYOUT BRH/LAYOUT/DAUGHTERS BRH/OFFSET BRHC/INTERTREE/SPACE BRHC/LAYOUT BRHC/LAYOUT/DAUGHTERS
            BRHC/LAYOUT/TERMINAL BRHC/OFFSET BRHL/LAYOUT BRHL/LAYOUT/DAUGHTERS BRHL/MOVE/RIGHT
            BROWSE/LAYOUT/HORIZ BROWSE/LAYOUT/HORIZ/COMPACTLY BROWSE/LAYOUT/LATTICE BRV/OFFSET
            EXTEND/TRANSITION/CHAIN FOREST/BREAK/CYCLES INIT/NODES/FOR/LAYOUT INTERPRET/MARK/FORMAT
            LATTICE/BREAK/CYCLES LAYOUTFOREST LAYOUTGRAPH LAYOUTLATTICE LAYOUTSEXPR LAYOUTSEXPR1
            MARK/GRAPH/NODE NEW/INSTANCE/OF/GRAPHNODE RAISE/TRANSITION/CHAIN REFLECT/GRAPH/DIAGONALLY
            REFLECT/GRAPH/HORIZONTALLY REFLECT/GRAPH/VERTICALLY SWITCH/NODE/HEIGHT/WIDTH)
       (CONSTANTS (LINKPARAMS 'Link% Parameters))
       [VARS (DEFAULT.GRAPH.NODEBORDER)
             (DEFAULT.GRAPH.NODEFONT)
             (DEFAULT.GRAPH.NODELABELSHADE)
             (ScalableLinkParameters '(LINEWIDTH))
             (CACHE/NODE/LABEL/BITMAPS)
             (NODEBORDERWIDTH 1)
             (GRAPH/HARDCOPY/FORMAT '(MODE PORTRAIT PAGENUMBERS T TRANS NIL]
       [INITVARS (DEFAULT.GRAPH.WINDOWSIZE (LIST (TIMES SCREENWIDTH 0.7)
                                                 (TIMES SCREENHEIGHT 0.4)))
                 (EDITGRAPHMENUCOMMANDS '((Move% Node 'MOVENODE "Moves a single node in the graph."
                                                  (SUBITEMS (|Move Single Node| 'MOVENODE "Moves a single node in
                                                                                          the graph.")
                                                            (|Move Node and Subtree| (EDITMOVESUBTREE GRAPHWINDOW)
                                                                   "Moves a subtree of nodes relative to the movement
                                                                   of their root.")
                                                            (Move% Region (EDITMOVEREGION GRAPHWINDOW)
                                                                   "Moves a group of nodes within a specified region
                                                                   to another region.")))
                                          ("Add Node" 'ADDNODE)
                                          ("Delete Node" 'DELETENODE)
                                          ("Add Link" 'ADDLINK)
                                          ("Delete Link" 'DELETELINK)
                                          ("Change label" 'CHANGELABEL)
                                          ("label smaller" 'SMALLER)
                                          ("label larger" 'LARGER)
                                          ("<-> Directed" 'DIRECTED)
                                          ("<-> Sides" 'SIDES)
                                          ("<-> Border" 'BORDER)
                                          ("<-> Shade" 'SHADE)
                                          STOP]
       (LOCALVARS . T)
       (RECORDS GRAPHNODE GRAPH)
```

```
        (DECLARE%: DONTCOPY (MACROS HALF))
        (COMS                                                    ; Grapher image objects
              (FNS GRAPHERIMAGEFNS)
              (FNS GRAPHERCOPYBUTTONEVENTFN GRAPHOBJ.FINDGRAPH)
              (FNS ALIGNMENTNODE GRAPHOBJ.CHECKALIGN)
              (FNS GRAPHEROBJ GRAPHOBJ.BUTTONEVENTINFN GRAPHOBJ.COPYBUTTONEVENTFN GRAPHOBJ.COPYFN
                   GRAPHOBJ.DISPLAYFN GRAPHOBJ.GETALIGN GRAPHOBJ.GETFN GRAPHOBJ.IMAGEBOXFN GRAPHOBJ.PUTALIGN
                   GRAPHOBJ.PUTFN)
              (FNS COPYGRAPH DUMPGRAPH READGRAPH)
              (INITVARS (GRAPHERIMAGEFNS))
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (GRAPHERIMAGEFNS)))
              (ALISTS (IMAGEOBJGETFNS GRAPHOBJ.GETFN])

;; Graph Editing

(DEFINEQ

(ADD/AND/DISPLAY/LINK
  [LAMBDA (FROMND TOND WIN G)                                    ; Edited 29-Apr-94 13:59 by sybalsky
                                                                 (* adds and displays a link.)
    (COND
       ((MEMBTONODES (fetch (GRAPHNODE NODEID) of TOND)
              (TOLINKS FROMND))
        (printout PROMPTWINDOW "Link already exists. " T)
        NIL)
       (T (GRAPHADDLINK FROMND TOND G WIN)
          (DISPLAYLINK FROMND TOND (CONSTANT (create POSITION
                                                     XCOORD _ 0
                                                     YCOORD _ 0))
                 WIN G)
          T])

(APPLYTOSELECTEDNODE
  [LAMBDA (WINDOW)                                               (* rmk%: "20-Nov-85 16:33")

          (* applys a function whenever the node is selected. Is used as BUTTONEVENTFN and gets called whenever cursor moves
          or button is down.)

    (GRAPHBUTTONEVENTFN WINDOW (WINDOWPROP WINDOW 'GRAPH)
           (WINDOWPROP WINDOW 'BROWSER/LEFTFN)
           (WINDOWPROP WINDOW 'BROWSER/MIDDLEFN)
           (WINDOWPROP WINDOW 'REGION])

(CALL.MOVENODEFN
  [LAMBDA (NODE NEWPOS GRAPH WINDOW OLDPOS)                      (* BBB "13-Sep-85 15:37")
                                                                 (* calls a graphs movenodefn.)
    (PROG ((MOVEFN (fetch (GRAPH GRAPH.MOVENODEFN) of GRAPH)))
          (AND MOVEFN (APPLY* MOVEFN NODE NEWPOS GRAPH WINDOW OLDPOS])

(CHANGE.NODEFONT.SIZE
  [LAMBDA (HOW NODE GRAPH WINDOW)                                ; Edited 22-Jul-87 16:32 by sye
                                                                 (* makes the label font of a node larger.)
    (PROG [(NEWFONT (NEXTSIZEFONT HOW (fetch (GRAPHNODE NODEFONT) of NODE]
          (COND
             (NEWFONT (DISPLAYNODE NODE (CONSTANT (create POSITION
                                                          XCOORD _ 0
                                                          YCOORD _ 0))
                                WINDOW GRAPH)
                      (PROG ((CHNGFN (fetch (GRAPH GRAPH.FONTCHANGEFN) of GRAPH)))
                            (AND CHNGFN (APPLY* CHNGFN HOW NODE GRAPH WINDOW)))
                      (replace (GRAPHNODE NODELABELBITMAP) of NODE with NIL)
                      (replace (GRAPHNODE NODEFONT) of NODE with NEWFONT)
                      (MEASUREGRAPHNODE NODE T)
                      (DISPLAYNODE NODE (CONSTANT (create POSITION
                                                          XCOORD _ 0
                                                          YCOORD _ 0))
                             WINDOW GRAPH])

(DEFAULT.ADDNODEFN
  [LAMBDA (GRAPH WINDOW BOXED)                                   ; Edited  9-Jan-89 15:57 by sye
                                                                 ; reads a node label name from the user and puts a node at the
                                                                 ; current cursor position.
    (PROG (NODELABEL NODENAME)
          (OR (SETQ NODELABEL (PROMPTINWINDOW "Node label?                    "))
              (RETURN))
      LP  (COND
             ((FASSOC (SETQ NODENAME (PACK* NODELABEL (GENSYM)))
                      (fetch (GRAPH GRAPHNODES) of GRAPH))
              (GO LP)))
          (RETURN (NODECREATE NODENAME NODELABEL (CURSORPOSITION NIL WINDOW)
                         NIL NIL (OR DEFAULT.GRAPH.NODEFONT DEFAULTFONT)
                         BOXED])
```

(DELETE/AND/DISPLAY/LINK

```
[LAMBDA (FROMND TOND WIN G)                                    ; Edited 29-Apr-94 13:59 by sybalsky
                                                               (* delete a link and updates the display.)
```

(* * rht 4/4/85%: Added temporary var LINKPARAMS to hold link parameters since they'll get tossed by GRAPHDELETELINK.)

```
   (COND
      ([NOT (OR (MEMBTONODES (fetch (GRAPHNODE NODEID) of TOND)
                             (TOLINKS FROMND))
                (AND (MEMBTONODES (fetch (GRAPHNODE NODEID) of FROMND)
                                  (TOLINKS TOND))
                     (NOT (fetch (GRAPH DIRECTEDFLG) of G))
                     (PROG ((TMP FROMND))                      (* editting graph, don't distinguish between links.)
                           (SETQ FROMND TOND)
                           (SETQ TOND TMP)
                           (RETURN T]
       (printout PROMPTWINDOW "Link does not exist. " T)
       NIL)
      (T (PROG ((LPARAMS (LINKPARAMETERS FROMND TOND)))
               (GRAPHDELETELINK FROMND TOND G WIN)
               (DISPLAYLINK FROMND TOND (CONSTANT (create POSITION
                                                          XCOORD _ 0
                                                          YCOORD _ 0))
                            WIN G NIL LPARAMS))
         T])
```

(DISPLAY/NAME

```
[LAMBDA (ND)                                                   ; Edited 29-Apr-94 13:59 by sybalsky
   (fetch (GRAPHNODE NODELABEL) of ND])
```

(DISPLAYGRAPH

```
[LAMBDA (GRAPH STREAM CLIP/REG TRANS)                          ; Edited 27-Jul-90 09:09 by tafel
```

;; Displays GRAPH with coordinates system translated to TRANS on STREAM. POS=NIL is interpreted as 0,0. Draws links first then labels so that ;; lattices don't have lines through the labels.

```
   (PROG (SCALE (LINEWIDTH 1)
                NNODES NODEHASHTABLE)
         [OR (type? POSITION TRANS)
             (SETQ TRANS (CONSTANT (create POSITION
                                           XCOORD _ 0
                                           YCOORD _ 0]
         (SETQ STREAM (\GETSTREAM STREAM 'OUTPUT))
         (COND
            ((DISPLAYSTREAMP STREAM)
```

;; This is because PRIN3 on displaystreams can sometimes cause CR's to be output. GRAPHER/CENTERPRINTINAREA doesn't ;; have the rightmargin kludge that the CENTERPRINTINAREA in MENU has.

```
             (DSPRIGHTMARGIN 65000 STREAM))
            (T (SETQ SCALE (DSPSCALE NIL STREAM))
               (SETQ GRAPH (SCALE/GRAPH GRAPH STREAM SCALE))
               [SETQ TRANS (create POSITION
                                   XCOORD _ (FIXR (FTIMES SCALE (fetch (POSITION XCOORD) of TRANS)))
                                   YCOORD _ (FIXR (FTIMES SCALE (fetch (POSITION YCOORD) of TRANS]
               (SETQ LINEWIDTH SCALE)))
```

;; nhb, 23-Feb-89: modified to create hashtable for nodeid to node lookup for cases where hash tables provide better performance than A-Lists.

```
         [COND
            ((IGREATERP (SETQ NNODES (LENGTH (fetch (GRAPH GRAPHNODES) of GRAPH)))
                        25)
             (SETQ NODEHASHTABLE (HASHARRAY NNODES))
             (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) do (PUTHASH (fetch (GRAPHNODE NODEID) of N)
                                                                       N NODEHASHTABLE]
         (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) do (DISPLAYNODELINKS N TRANS STREAM GRAPH T LINEWIDTH
                                                                            NODEHASHTABLE))
         (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) do (PRINTDISPLAYNODE N TRANS STREAM CLIP/REG])
```

(DISPLAYLINK

```
[LAMBDA (FRND TOND TRANS STREAM G LINEWIDTH PARAMS)            (* rht%: "13-Mar-85 13:58")
                                                               (* draws in a link from FRND TO TOND, translated by TRANS)
   (COND
      ((fetch (GRAPH SIDESFLG) of G)
       (COND
          ((OR (fetch (GRAPH DIRECTEDFLG) of G)
               (IGREATERP (GN/LEFT TOND)
                          (GN/RIGHT FRND)))                    (* in the horizontal case of LATTICE, always draw from right to left.)
           (DISPLAYLINK/RL TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
          ((IGREATERP (GN/LEFT FRND)
                      (GN/RIGHT TOND))
           (DISPLAYLINK/LR TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
          ((IGREATERP (GN/BOTTOM FRND)
                      (GN/TOP TOND))
```

```
            (DISPLAYLINK/BT TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
           ((IGREATERP (GN/BOTTOM TOND)
                  (GN/TOP FRND))
            (DISPLAYLINK/TB TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
           (T                                                    (* if on top of each other, don't draw.)
              NIL)))
       (T (COND
             ((OR (fetch (GRAPH DIRECTEDFLG) of G)
                  (IGREATERP (GN/BOTTOM FRND)
                         (GN/TOP TOND)))
```

(* if LATTICE, always draw from FROMNODE BOTTOM to TONODE TOP.
Otherwise find the one that looks best.)

```
              (DISPLAYLINK/BT TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
             ((IGREATERP (GN/BOTTOM TOND)
                    (GN/TOP FRND))
              (DISPLAYLINK/TB TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
             ((IGREATERP (GN/LEFT TOND)
                    (GN/RIGHT FRND))
              (DISPLAYLINK/RL TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
             ((IGREATERP (GN/LEFT FRND)
                    (GN/RIGHT TOND))
              (DISPLAYLINK/LR TRANS FRND TOND LINEWIDTH NIL STREAM PARAMS))
             (T                                                  (* if on top of each other, don't draw.)
                NIL])
```

(DISPLAYLINK/BT

```
[LAMBDA (TRANS GNB GNT WIDTH OPERATION STREAM PARAMS)          ; Edited 29-Apr-94 13:59 by sybalsky
```

(* draws a line from the bottom edge of GNB to the top edge of GNT translated by TRANS)

```
  (APPLY* (OR (LISTGET PARAMS 'DRAWLINKFN)
              'DRAWLINE)
          (IPLUS (fetch XCOORD of TRANS)
                 (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNB)))
          (IPLUS (fetch YCOORD of TRANS)
                 (SUB1 (GN/BOTTOM GNB)))
          (IPLUS (fetch XCOORD of TRANS)
                 (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNT)))
          (IPLUS (fetch YCOORD of TRANS)
                 (ADD1 (GN/TOP GNT)))
          (OR (LISTGET PARAMS 'LINEWIDTH)
              WIDTH 1)
          OPERATION STREAM (LISTGET PARAMS 'COLOR)
          (LISTGET PARAMS 'DASHING)
          PARAMS])
```

(DISPLAYLINK/LR

```
[LAMBDA (TRANS GNL GNR WIDTH OPERATION STREAM PARAMS)          ; Edited 29-Apr-94 13:59 by sybalsky
```

(* draws a line from the left edge of GNL to the right edge of GNR, translated by TRANS)

```
  (APPLY* (OR (LISTGET PARAMS 'DRAWLINKFN)
              'DRAWLINE)
          (IPLUS (fetch XCOORD of TRANS)
                 (SUB1 (GN/LEFT GNL)))
          (IPLUS (fetch YCOORD of TRANS)
                 (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNL)))
          (IPLUS (fetch XCOORD of TRANS)
                 (ADD1 (GN/RIGHT GNR)))
          (IPLUS (fetch YCOORD of TRANS)
                 (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNR)))
          (OR (LISTGET PARAMS 'LINEWIDTH)
              WIDTH 1)
          OPERATION STREAM (LISTGET PARAMS 'COLOR)
          (LISTGET PARAMS 'DASHING)
          PARAMS])
```

(DISPLAYLINK/RL

```
[LAMBDA (TRANS GNR GNL WIDTH OPERATION STREAM PARAMS)          ; Edited 29-Apr-94 13:59 by sybalsky
```

(* draws a line from the right edge of GNR, to the left edge of GNL translated by TRANS)

```
  (APPLY* (OR (LISTGET PARAMS 'DRAWLINKFN)
              'DRAWLINE)
          (IPLUS (fetch XCOORD of TRANS)
                 (ADD1 (GN/RIGHT GNR)))
          (IPLUS (fetch YCOORD of TRANS)
                 (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNR)))
          (IPLUS (fetch XCOORD of TRANS)
                 (SUB1 (GN/LEFT GNL)))
          (IPLUS (fetch YCOORD of TRANS)
                 (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNL)))
```

```
          (OR (LISTGET PARAMS 'LINEWIDTH)
              WIDTH 1)
          OPERATION STREAM (LISTGET PARAMS 'COLOR)
          (LISTGET PARAMS 'DASHING)
          PARAMS])
```

(DISPLAYLINK/TB

```
[LAMBDA (TRANS GNT GNB WIDTH OPERATION STREAM PARAMS)          ; Edited 29-Apr-94 13:59 by sybalsky
```

(* draws a line from the top edge of GNT to the bottom edge of GNR, translated by TRANS)

```
  (APPLY* (OR (LISTGET PARAMS 'DRAWLINKFN)
              'DRAWLINE)
          (IPLUS (fetch XCOORD of TRANS)
                 (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNT)))
          (IPLUS (fetch YCOORD of TRANS)
                 (ADD1 (GN/TOP GNT)))
          (IPLUS (fetch XCOORD of TRANS)
                 (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of GNB)))
          (IPLUS (fetch YCOORD of TRANS)
                 (SUB1 (GN/BOTTOM GNB)))
          (OR (LISTGET PARAMS 'LINEWIDTH)
              WIDTH 1)
          OPERATION STREAM (LISTGET PARAMS 'COLOR)
          (LISTGET PARAMS 'DASHING)
          PARAMS])
```

(DISPLAYNODE

```
[LAMBDA (NODE TRANS STREAM G TOSONLY)                          (* kvl "10-Aug-84 19:08")
```

(* displays a node and its links. IF TOSONLY IS NON-NIL, DRAWS ONLY THE TO LINKS.)

```
  (DISPLAYNODELINKS NODE TRANS STREAM G TOSONLY)
  (PRINTDISPLAYNODE NODE TRANS STREAM (DSPCLIPPINGREGION NIL STREAM])
```

(ERASE/GRAPHNODE

```
[LAMBDA (NODE STREAM TRANS)                                    ; Edited 29-Apr-94 13:59 by sybalsky
```

(* erases a node at its position translated by TRANS)

```
  (OR [NOT (OR (WINDOWP STREAM)
               (IMAGESTREAMTYPEP STREAM 'DISPLAY]
      (ZEROP (fetch (GRAPHNODE NODEHEIGHT) of NODE))
      (BITBLT NIL NIL NIL STREAM (COND
                                    (TRANS (IPLUS (fetch (POSITION XCOORD) of TRANS)
                                                  (GN/LEFT NODE)))
                                    (T (GN/LEFT NODE)))
            (COND
               (TRANS (IPLUS (fetch (POSITION YCOORD) of TRANS)
                             (GN/BOTTOM NODE)))
               (T (GN/BOTTOM NODE)))
            (fetch (GRAPHNODE NODEWIDTH) of NODE)
            (fetch (GRAPHNODE NODEHEIGHT) of NODE)
            'TEXTURE
            'REPLACE WHITESHADE])
```

(DISPLAYNODE

```
[LAMBDA (NODE TRANS STREAM G TOSONLY)                          (* kvl "10-Aug-84 19:08")
```

(* displays a node and its links. IF TOSONLY IS NON-NIL, DRAWS ONLY THE TO LINKS.)

```
  (DISPLAYNODELINKS NODE TRANS STREAM G TOSONLY)
  (PRINTDISPLAYNODE NODE TRANS STREAM (DSPCLIPPINGREGION NIL STREAM])
```

(DISPLAYNODELINKS

```
[LAMBDA (NODE TRANS STREAM G TOSONLY LINEWIDTH NODEHASHTABLE)  ; Edited 24-Feb-89 11:56 by Briggs
```

;; displays a node links. If TOSONLY is non-NIL, draws only the TO links.

;; nhb, 23-Feb-89: modified to accept a hash table of nodes by nodeid to assist GETNODEFROMID.

```
  (PROG ((NODELST (fetch (GRAPH GRAPHNODES) of G)))
        (for TONODEID TONODE in (TOLINKS NODE) do (DISPLAYLINK NODE (SETQ TONODE (GETNODEFROMID TONODEID
                                                                                   NODELST NODEHASHTABLE))
                                                                   TRANS STREAM G LINEWIDTH (LINKPARAMETERS NODE TONODE
                                                                                             )))
        (OR TOSONLY (for FROMNDID FROMND in (FROMLINKS NODE) do (DISPLAYLINK (SETQ FROMND
                                                                                  (GETNODEFROMID FROMNDID
                                                                                         NODELST NODEHASHTABLE))
                                                                             NODE TRANS STREAM G LINEWIDTH
                                                                             (LINKPARAMETERS FROMND NODE])
```

(DRAW/GRAPHNODE/BORDER

```
[LAMBDA (BORDER LEFT BOTTOM WIDTH HEIGHT STREAM)               (* lmm " 9-Jun-85 22:38")
```

(* interprets the node border. If the border is a shade, then bitblt twice in invert mode.
This will look ugly if a link runs underneath the node, but at least the label will be legible.)

```
(COND
   ((EQ BORDER NIL))
   ((EQ BORDER T)
    (DRAWAREABOX LEFT BOTTOM WIDTH HEIGHT 1 NIL STREAM))
   ((FIXP BORDER)
    (OR (ILEQ BORDER 0)
        (DRAWAREABOX LEFT BOTTOM WIDTH HEIGHT BORDER NIL STREAM)))
   ((LISTP BORDER)                                          (* Extract the PROG after Intermezzo is released)
    (DRAWAREABOX LEFT BOTTOM WIDTH HEIGHT (CAR BORDER)
           NIL STREAM (CADR BORDER)))
   (T (ERROR "Illegal border:" BORDER])
```

(DRAWAREABOX

```
[LAMBDA (BOXLEFT BOXBOTTOM BOXWIDTH BOXHEIGHT BORDER OP W TEXTURE)
                                                            (* lmm " 9-Jun-85 22:36")
   (OR TEXTURE (SETQ TEXTURE BLACKSHADE))                   (* lmm " 9-Jun-85 22:04")
                                                            (* draws lines inside the region.)
                                                            (* draw left edge)
   (BLTSHADE TEXTURE W BOXLEFT BOXBOTTOM BORDER BOXHEIGHT OP)   (* draw top)
   (BLTSHADE TEXTURE W (PLUS BOXLEFT BORDER)
          (DIFFERENCE (PLUS BOXBOTTOM BOXHEIGHT)
                 BORDER)
          (DIFFERENCE BOXWIDTH (PLUS BORDER BORDER))
          BORDER OP)                                        (* draw bottom)
   (BLTSHADE TEXTURE W (PLUS BOXLEFT BORDER)
          BOXBOTTOM
          (DIFFERENCE BOXWIDTH (PLUS BORDER BORDER))
          BORDER OP)                                        (* draw right edge)
   (BLTSHADE TEXTURE W (DIFFERENCE (PLUS BOXLEFT BOXWIDTH)
                              BORDER)
          BOXBOTTOM BORDER BOXHEIGHT OP])
```

(EDITADDLINK

```
[LAMBDA (W)                                                 (* kvl "20-APR-82 13:53")
                                                            (* reads and adds a link to the graph)
   (EDITAPPLYTOLINK (FUNCTION ADD/AND/DISPLAY/LINK)
          'added
          (WINDOWPROP W 'GRAPH)
          W])
```

(EDITADDNODE

```
[LAMBDA (W NewPosition MSGW NODELABELFN)                    ; Edited 29-Apr-94 13:59 by sybalsky
                                                            ; adds a node to the graph in the window W and displays it.
```

;; pmi 4/8/88: Added NewPosition argument so that the new position for a node may be specified programatically.

;; sye Jan/9/89: added MSGW & NODELABELFN args

```
   (DECLARE (GLOBALVARS PROMPTWINDOW))
   (PROG [NODE ORIGPOS NEWPOS NODELABEL (GRAPH (WINDOWPROP W 'GRAPH))
                (Stream (WINDOWPROP W 'DSP]
         (OR (SETQ NODE (GRAPHADDNODE GRAPH W))
             (RETURN))
         (MEASUREGRAPHNODE NODE)
         (if (POSITIONP NewPosition)
             then (SETQ ORIGPOS (create POSITION using (fetch (GRAPHNODE NODEPOSITION) of NODE)))
                  (MOVENODE NODE ORIGPOS NewPosition GRAPH Stream)
                  (FLIPNODE NODE Stream)
                  (EXTENDEXTENT (WFROMDS Stream)
                         (NODEREGION NODE))
                  (CALL.MOVENODEFN NODE NewPosition GRAPH (WFROMDS Stream)
                          ORIGPOS)
           else (printout (OR MSGW PROMPTWINDOW)
                        "Position node "
                        (OR (AND NODELABELFN (APPLY* NODELABELFN NODE))
                            (fetch (GRAPHNODE NODELABEL)
                                   NODE)))
                (PRINTDISPLAYNODE NODE (CONSTANT (create POSITION
                                                         XCOORD _ 0
                                                         YCOORD _ 0))
                        W
                        (DSPCLIPPINGREGION NIL W))
                (TRACKCURSOR NODE Stream GRAPH))
         (RETURN NODE])
```

(EDITAPPLYTOLINK

```
[LAMBDA (FN MSG GRAPH DS MSGW NODELABELFN)                  ; Edited 9-Jan-89 09:10 by sye
   (SETQ MSGW (OR MSGW PROMPTWINDOW))
   (CLEARW MSGW)
   (CLRPROMPT)
   (COND
      [(fetch (GRAPH GRAPHNODES) of GRAPH)
```

```
        (PROG (FROM TO (ABORTMSG "No selection was made ... operation aborted."))
              (printout MSGW "Specify the link by selecting the FROM node, then the TO node." T "FROM?" T)
                                                                  (* "if no FROM node was selected, abort the operation")
              (OR (SETQ FROM (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                        DS))
                  (RETURN (printout PROMPTWINDOW ABORTMSG T)))
              (FLIPNODE FROM DS)
              (printout MSGW "TO?" T)
              (COND
                 [(ERSETQ (SETQ TO (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                              DS]
                 (T (FLIPNODE FROM DS)
                    (ERROR!)))
              (FLIPNODE FROM DS)                                  (* "if no TO node was selected, abort the operation")
              (OR TO (RETURN (printout PROMPTWINDOW ABORTMSG T)))
              (COND
                 ((APPLY* FN FROM TO DS GRAPH)                    (* return non-nil if changed anything.)
                  (printout PROMPTWINDOW "Link from " (OR (AND NODELABELFN (APPLY* NODELABELFN FROM))
                                                         (DISPLAY/NAME FROM))
                            " to "
                            (OR (AND NODELABELFN (APPLY* NODELABELFN TO))
                                (DISPLAY/NAME TO))
                            %, MSG T)
                  (RETURN T]
       (T (printout PROMPTWINDOW "There are no nodes.  You can create nodes with the Add Node command." T])
```

(EDITCHANGEFONT

```
  [LAMBDA (HOW W)                                                 ; Edited  7-Jan-89 13:14 by sye
                                                                  (* prompts the user for a node and deletes it)
    (PROG ((GRAPH (WINDOWPROP W 'GRAPH))
           (DS (WINDOWPROP W 'DSP))
           NODE)
          (COND
             ((NOT (fetch (GRAPH GRAPHNODES) of GRAPH))
              (PROMPTPRINT " No nodes in graph yet. ")
              (RETURN)))
          (CLRPROMPT)
          (printout PROMPTWINDOW "Select node to be made " (COND
                                                              ((EQ HOW 'SMALLER)
                                                               "smaller.")
                                                              (T "larger.")))
          (OR (SETQ NODE (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                    DS))
              (RETURN (printout PROMPTWINDOW T "No selection was made ... operation aborted." T)))
          (CHANGE.NODEFONT.SIZE HOW NODE GRAPH W)
          (RETURN NODE])
```

(EDITCHANGELABEL

```
  [LAMBDA (W MSGW)                                                ; Edited  7-Jan-89 13:31 by sye
                                                                  (* prompts the user for a node and deletes it)
    (PROG ((GRAPH (WINDOWPROP W 'GRAPH))
           (DS (GETSTREAM W))
           (TRANS (CONSTANT (create POSITION
                                    XCOORD _ 0
                                    YCOORD _ 0)))
           NODE NEWLABEL)
          (COND
             ((NOT (fetch (GRAPH GRAPHNODES) of GRAPH))
              (PROMPTPRINT "No nodes in graph yet. ")
              (RETURN)))
          (CLRPROMPT)
          (SETQ MSGW (OR MSGW PROMPTWINDOW))
          (CLEARW MSGW)
          (printout MSGW "Select node to have label changed.")
          (OR (SETQ NODE (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                    DS))
              (RETURN (printout PROMPTWINDOW T "No selection was made ... operation aborted." T)))
          (if (NULL (SETQ NEWLABEL (GRAPHCHANGELABEL GRAPH W NODE)))
              then (RETURN))
          (DISPLAYNODE NODE TRANS W GRAPH)
          (ERASE/GRAPHNODE NODE DS TRANS)
          (replace (GRAPHNODE NODELABEL) of NODE with NEWLABEL)
          (replace (GRAPHNODE NODELABELBITMAP) of NODE with NIL)
          (MEASUREGRAPHNODE NODE T)
          (DISPLAYNODE NODE TRANS W GRAPH)
          (RETURN NODE])
```

(EDITDELETELINK

```
  [LAMBDA (W)                                                     (* kvl "20-APR-82 13:54")
                                                                  (* reads and adds a link to the graph)
    (EDITAPPLYTOLINK (FUNCTION DELETE/AND/DISPLAY/LINK)
           'deleted
           (WINDOWPROP W 'GRAPH)
           W])
```

(EDITDELETENODE

```
  [LAMBDA (W)                                                    ; Edited  9-Jan-89 09:14 by sye
                                                                 (* prompts the user for a node and deletes it)

    (RESETFORM (TTYDISPLAYSTREAM PROMPTWINDOW)
           (CLRPROMPT)
           (PROG ((GRAPH (WINDOWPROP W 'GRAPH))
                  (DS (WINDOWPROP W 'DSP))
                  NODE NODELABEL)
                 (COND
                    ((NOT (fetch (GRAPH GRAPHNODES) of GRAPH))
                     (PROMPTPRINT " No nodes to delete. ")
                     (RETURN)))
                 (PROMPTPRINT "Select node to be deleted. ")
                 (OR (SETQ NODE (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                           DS))
                     (RETURN (printout T T "No selection was made ... operation aborted." T)))
                 (TERPRI T)
                 (FLIPNODE NODE DS)
                 (COND
                    ((EQ [ASKUSER NIL NIL (LIST "delete node " (SETQ NODELABEL (DISPLAY/NAME NODE]
                         'Y)
                     (FLIPNODE NODE DS)
                     (DISPLAYNODE NODE (CONSTANT (create POSITION
                                                         XCOORD _ 0
                                                         YCOORD _ 0))
                           DS GRAPH)
                     (for TOND in (APPEND (TOLINKS NODE)) do (GRAPHDELETELINK NODE (GETNODEFROMID
                                                                                    TOND
                                                                                    (fetch (GRAPH GRAPHNODES)
                                                                                       of GRAPH))
                                                                                GRAPH W))
                     (for FROMND in (APPEND (FROMLINKS NODE)) do (GRAPHDELETELINK (GETNODEFROMID
                                                                                   FROMND
                                                                                   (fetch (GRAPH GRAPHNODES)
                                                                                      of GRAPH))
                                                                               NODE GRAPH W))
                     (GRAPHDELETENODE NODE GRAPH W)
                     (printout T "Node " NODELABEL " deleted." T)
                     (RETURN NODE))
                    (T (FLIPNODE NODE DS)
                       (printout T "nothing deleted." T)
                       (RETURN NIL])
```

(EDITGRAPH

```
  [LAMBDA (G W)
    (SHOWGRAPH G W NIL NIL T T])
```

(EDITGRAPH1

```
  [LAMBDA (GRAPH WINDOW)                                         ; Edited 19-Aug-88 08:30 by sye
```

;; top level function for editing a graph. If there is no graph, create one empty. IF there is no window, create on the right size for the graph. After ;; getting the arguments right, put the right button functions on, display it and enter the main loop.

```
    (OR GRAPH (SETQ GRAPH (create GRAPH)))
    (SETQ WINDOW (SIZE/GRAPH/WINDOW GRAPH WINDOW))
    (WINDOWPROP WINDOW 'GRAPH GRAPH)
    (WINDOWPROP WINDOW 'REPAINTFN (FUNCTION REDISPLAYGRAPH))
    (WINDOWPROP WINDOW 'SCROLLFN (FUNCTION SCROLLBYREPAINTFN))
    (DSPOPERATION 'INVERT WINDOW)
    (REDISPLAYGRAPH WINDOW)
    (EDITGRAPH2 WINDOW)
   GRAPH])
```

(EDITGRAPH2

```
  [LAMBDA (W)                                                    (* rrb " 7-NOV-83 14:51")
```

(* Can also be called from top level if the given window W has a graph on its GRAPH windowprop and the graph has been displayed by SHOWGRAPH or its equivalent. It waits for mouse hits, does the comand, then waits for mouse clear. Each edit command function takes only the window so that they can be hung separately on button event functions. However, the window must have INVERT as its display operation mode.)

```
    (PROG (VAL)
          (CLRPROMPT)
          (printout PROMPTWINDOW "Use the left button to move nodes." T "Use the middle button to get a menu of
                 edit commands." T "During an edit command, the middle button can be used to abort.")
      LP  (until (MOUSESTATE (OR LEFT MIDDLE)) do)
          (COND
             [(LASTMOUSESTATE MIDDLE)
              (SETQ VAL (ERSETQ (GRAPHEDITCOMMANDFN W)))
              (COND
                 ((NULL VAL)                                     (* aborted)
                  (printout PROMPTWINDOW T T "command aborted." T))
                 ((EQ (CAR VAL)
```

```
                    'STOP)
                  (RETURN (CLRPROMPT]
            ((fetch (GRAPH GRAPHNODES) of (WINDOWPROP W 'GRAPH)) (* track the nearest node.)
             (TRACKNODE W))
            (T (printout PROMPTWINDOW T "There are no nodes to move yet." T "Press the middle button and select
                    the 'Add a node' command.")))
          (until (MOUSESTATE UP) do)
          (GO LP])
```

(EDITMOVENODE

```
  [LAMBDA (WINDOW)                                          ; Edited  7-Jan-89 13:22 by sye
                                                            (* hilite nodes until the cursor goes down then move it)
    (PROG ((DS (WINDOWPROP WINDOW 'DSP))
           (REG (WINDOWPROP WINDOW 'REGION))
           (GRAPH (WINDOWPROP WINDOW 'GRAPH))
           OLDPOS NOW NEAR NODELST)
          (COND
             (GRAPH (SETQ NODELST (fetch (GRAPH GRAPHNODES) of GRAPH)))
             (T (RETURN)))
          (CLRPROMPT)
          (printout PROMPTWINDOW "Move the cursor to the node " "you want to move " "and press any button.")
          [SETQ NEAR (NODELST/AS/MENU NODELST (SETQ OLDPOS (CURSORPOSITION NIL DS]
      FLIP
          (AND NOW (FLIPNODE NOW DS))
          (AND NEAR (FLIPNODE NEAR DS))
          (SETQ NOW NEAR)
      LP  (GETMOUSESTATE)
          (COND
             ((LASTMOUSESTATE (NOT UP))                     (* button up, process it.)
              (AND NOW (FLIPNODE NOW DS))                   (* NOW node has been selected.)
              )
             ([EQ NOW (SETQ NEAR (NODELST/AS/MENU NODELST (CURSORPOSITION NIL DS OLDPOS]
              (GO LP))
             (T (GO FLIP)))
          (printout PROMPTWINDOW T "Holding the button down, " "move the node to its new position" "and release
                the button.")
          (TRACKCURSOR NOW DS GRAPH)
          (printout PROMPTWINDOW T "Done."])
```

(EDITTOGGLEBORDER

```
  [LAMBDA (W)                                               ; Edited  7-Jan-89 13:38 by sye
                                                            ; prompts the user for a node and inverts its border
    (RESETFORM (TTYDISPLAYSTREAM PROMPTWINDOW)
           (CLRPROMPT)
           (PROG ((GRAPH (WINDOWPROP W 'GRAPH))
                  (DS (WINDOWPROP W 'DSP))
                  NODE)
                 (COND
                    ((NOT (fetch (GRAPH GRAPHNODES) of GRAPH))
                     (PROMPTPRINT "No nodes to invert. ")
                     (RETURN)))
                 (PROMPTPRINT "Select node to have border inverted. ")
                 (OR (SETQ NODE (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                           DS))
                     (RETURN (printout T T "No selection was made ... operation aborted." T)))
                 (TERPRI T)
                 (RESET/NODE/BORDER NODE 'INVERT W GRAPH)
                 (AND (fetch (GRAPH GRAPH.INVERTBORDERFN) of GRAPH)
                      (APPLY* (fetch (GRAPH GRAPH.INVERTBORDERFN) of GRAPH)
                              NODE GRAPH W))
                 (RETURN NODE])
```

(EDITTOGGLELABEL

```
  [LAMBDA (W)                                               ; Edited  7-Jan-89 13:17 by sye
                                                            (* prompts the user for a node and inverts its lable)
    (RESETFORM (TTYDISPLAYSTREAM PROMPTWINDOW)
           (CLRPROMPT)
           (PROG ((GRAPH (WINDOWPROP W 'GRAPH))
                  (DS (WINDOWPROP W 'DSP))
                  NODE)
                 (COND
                    ((NOT (fetch (GRAPH GRAPHNODES) of GRAPH))
                     (PROMPTPRINT " No nodes to invert.")
                     (RETURN)))
                 (PROMPTPRINT "Select node to have label inverted. ")
                 (OR (SETQ NODE (READ/NODE (fetch (GRAPH GRAPHNODES) of GRAPH)
                                           DS))
                     (RETURN (printout T T "No selection was made ... operation aborted." T)))
                 (TERPRI T)
                 (RESET/NODE/LABELSHADE NODE 'INVERT W)
                 (AND (fetch (GRAPH GRAPH.INVERTLABELFN) of GRAPH)
                      (APPLY* (fetch (GRAPH GRAPH.INVERTLABELFN) of GRAPH)
                              NODE GRAPH W))
                 (RETURN NODE])
```

(FILL/GRAPHNODE/LABEL

```
[LAMBDA (SHADE LEFT BOTTOM WIDTH HEIGHT NBW STREAM)          (* kvl "10-Sep-84 14:41")
                                                             (* NBW is the border, which must be subtracted from the node's
  region)
  (PROG ((NS SHADE))
        (OR (WINDOWP STREAM)
            (DISPLAYSTREAMP STREAM)
            (RETURN))
        (COND
           ((EQ SHADE T)
            (SETQ NS BLACKSHADE))
           ((NULL SHADE)
            (SETQ NS WHITESHADE)))
        (BITBLT NIL NIL NIL STREAM (IPLUS LEFT NBW)
               (IPLUS BOTTOM NBW)
               (IDIFFERENCE WIDTH (IPLUS NBW NBW))
               (IDIFFERENCE HEIGHT (IPLUS NBW NBW))
               'TEXTURE
               'INVERT NS])
```

(FIX/SCALE

```
[LAMBDA (PARAMVALUE SCALE)                                   (* dgb%: "28-Jan-85 10:01")

        (* * fixes PARAMVALUE by SCALE If PARAMVALUE is a list, then fixes the elements of the list)

  (COND
     ((LISTP PARAMVALUE)
      (for V in PARAMVALUE collect (FIX/SCALE V SCALE)))
     (T                                                      (* Note that some parameters may go to zero)
        (FIXR (FTIMES SCALE PARAMVALUE])
```

(FLIPNODE

```
[LAMBDA (NODE DS)                                            ; Edited 29-Apr-94 14:00 by sybalsky
                                                             (* flips the region around a node.)
  (BITBLT NIL NIL NIL DS (IDIFFERENCE (GN/LEFT NODE)
                                      1)
         (IDIFFERENCE (GN/BOTTOM NODE)
                1)
         (IPLUS (fetch (GRAPHNODE NODEWIDTH) of NODE)
                2)
         (IPLUS (fetch (GRAPHNODE NODEHEIGHT) of NODE)
                2)
         'TEXTURE
         'INVERT BLACKSHADE])
```

(FONTNAMELIST

```
[LAMBDA (FONTDESC)                                           (* rrb " 2-NOV-83 21:00")
  (LIST (FONTPROP FONTDESC 'FAMILY)
        (FONTPROP FONTDESC 'SIZE)
        (FONTPROP FONTDESC 'FACE])
```

(FROMLINKS

```
[LAMBDA (NODE)                                               ; Edited 29-Apr-94 14:00 by sybalsky
  (fetch (GRAPHNODE FROMNODES) of NODE])
```

(GETNODEFROMID

```
[LAMBDA (ID NODELST NODEHASHTABLE)                           ; Edited 24-Feb-89 11:55 by Briggs
```

;; Allow Link parameters to be passed as a property list of the node description.

;; nhb, 23-Feb-89: modified -- If the (optional) NODEHASHTABLE is passed then we will use this rather than assoc'ing in the node list to find the
;; node. Also switched order of listp check and bare FASSOC

```
  (COND
     (NODEHASHTABLE (OR (AND (LISTP ID)
                             (EQ 'Link% Parameters (CAR ID))
                             (GETHASH (CADR ID)
                                      NODEHASHTABLE))
                        (GETHASH ID NODEHASHTABLE)
                        (ERROR "No graphnode for nodeid:" ID)))
     (T (OR (AND (LISTP ID)
                 (EQ 'Link% Parameters (CAR ID))
                 (FASSOC (CADR ID)
                         NODELST))
            (FASSOC ID NODELST)
            (ERROR "No graphnode for nodeid:" ID])
```

(GN/BOTTOM

```
[LAMBDA (NODE)                                               ; Edited 29-Apr-94 14:00 by sybalsky
  (IDIFFERENCE (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE))
         (HALF (fetch (GRAPHNODE NODEHEIGHT) of NODE])
```

(GN/LEFT

```
  [LAMBDA (NODE)                                                    ; Edited 29-Apr-94 14:00 by sybalsky
    (IDIFFERENCE (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE))
           (HALF (fetch (GRAPHNODE NODEWIDTH) of NODE])
```

(GN/RIGHT

```
  [LAMBDA (NODE)                                                    ; Edited 29-Apr-94 14:00 by sybalsky

          (* Assumes that the big-half of width is to the left of the center, for even width)

    (IPLUS (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE))
           (SUB1 (HALF (ADD1 (fetch (GRAPHNODE NODEWIDTH) of NODE])
```

(GN/TOP

```
  [LAMBDA (NODE)                                                    ; Edited 29-Apr-94 14:00 by sybalsky

          (* Assumes that big-half of height is under the center, for even height.
          Result is -1 for height=0, which is correct.)

    (IPLUS (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE))
           (SUB1 (HALF (ADD1 (fetch (GRAPHNODE NODEHEIGHT) of NODE])
```

(GRAPHADDLINK

```
  [LAMBDA (FROM TO GRAPH WINDOW)                                    ; Edited 29-Apr-94 14:00 by sybalsky
                                                                    (* links two nodes)
    (PROG ((ADDFN (fetch (GRAPH GRAPH.ADDLINKFN) of GRAPH)))
          (AND ADDFN (APPLY* ADDFN FROM TO GRAPH WINDOW)))
    (push (fetch (GRAPHNODE FROMNODES) of TO)
          (fetch (GRAPHNODE NODEID) of FROM))
    (push (fetch (GRAPHNODE TONODES) of FROM)
          (fetch (GRAPHNODE NODEID) of TO])
```

(GRAPHADDNODE

```
  [LAMBDA (GRAPH W)                                                 (* rrb " 2-NOV-83 20:29")
                                                                    (* adds a node to the graph GRAPH)
    (PROG (ADDFN NODE)
          (OR [SETQ NODE (COND
                             ((SETQ ADDFN (fetch (GRAPH GRAPH.ADDNODEFN) of GRAPH))
                              (APPLY* ADDFN GRAPH W))
                             (T (DEFAULT.ADDNODEFN GRAPH W T]
              (RETURN))
          (replace (GRAPH GRAPHNODES) of GRAPH with (NCONC1 (fetch (GRAPH GRAPHNODES) of GRAPH)
                                                            NODE))
          (RETURN NODE])
```

(GRAPHBUTTONEVENTFN

```
  [LAMBDA (WINDOW GRAPH LEFTFNOFNODE MIDDLEFNOFNODE REG)            (* rmk%: "20-Nov-85 16:33")

          (* applys a function whenever the node is selected. Is used as BUTTONEVENTFN and gets called whenever cursor moves
          or button is down.)

    (TOTOPW WINDOW)
    (PROG ((NODELST (fetch (GRAPH GRAPHNODES) of GRAPH))
           (DS (GETSTREAM WINDOW))
           BUTTON OLDPOS REG NOW NEAR)                              (* note which button is down.)
          (COND
             ((LASTMOUSESTATE LEFT)
              (OR LEFTFNOFNODE (RETURN))
              (SETQ BUTTON 'LEFT))
             ((LASTMOUSESTATE MIDDLE)
              (OR MIDDLEFNOFNODE (RETURN))
              (SETQ BUTTON 'MIDDLE))
             (T                                                     (* no button down, not interested.)
                (RETURN)))                                          (* get the region of this window.)
          [SETQ NEAR (NODELST/AS/MENU NODELST (SETQ OLDPOS (CURSORPOSITION NIL DS]
      FLIP
          (AND NOW (FLIPNODE NOW DS))
          (AND NEAR (FLIPNODE NEAR DS))
          (SETQ NOW NEAR)
      LP                                                            (* wait for a button up or move out of region)
          (GETMOUSESTATE)
          (COND
             ((NOT (LASTMOUSESTATE (OR LEFT MIDDLE)))               (* button up, process it.)
              (AND NOW (FLIPNODE NOW DS))                           (* NOW node has been selected.)
              (RETURN (APPLY* (SELECTQ BUTTON
                                       (LEFT LEFTFNOFNODE)
                                       (MIDDLE MIDDLEFNOFNODE)
                                       (SHOULDNT))
                              NOW WINDOW)))
             ((NOT (INSIDE? (WINDOWPROP WINDOW 'REGION)
```

```
                        LASTMOUSEX LASTMOUSEY))                    (* outside of region, return)
          (AND NOW (FLIPNODE NOW DS))
          (RETURN))
         ([EQ NOW (SETQ NEAR (NODELST/AS/MENU NODELST (CURSORPOSITION NIL DS OLDPOS]
          (GO LP))
         (T (GO FLIP])
```

(GRAPHCHANGELABEL

```
  [LAMBDA (GRAPH W NODE)                                           (* rmk%: "19-Sep-85 10:50")
                                                                   (* Returns a new label for NODE)
    (LET (CHANGEFN)
         (COND
            ((SETQ CHANGEFN (fetch (GRAPH GRAPH.CHANGELABELFN) of GRAPH))
             (APPLY* CHANGEFN GRAPH W NODE))
            (T (PROMPTINWINDOW "Node label?                  "])
```

(GRAPHDELETELINK

```
  [LAMBDA (FROM TO GRAPH WINDOW)                                   ; Edited 29-Apr-94 14:00 by sybalsky
                                                                   (* deletes a link from a graph)

          (* * rht 4/4/85%: Changed to call REMOVETONODES to remove either nodeID or paramlist thingie for nodeID.)

    (PROG ((DELFN (fetch (GRAPH GRAPH.DELETELINKFN) of GRAPH)))
          (AND DELFN (APPLY* DELFN FROM TO GRAPH WINDOW)))
    (replace (GRAPHNODE TONODES) of FROM with (REMOVETONODES (fetch (GRAPHNODE NODEID) of TO)
                                                             (fetch (GRAPHNODE TONODES) of FROM)))
    (replace (GRAPHNODE FROMNODES) of TO with (REMOVE (fetch (GRAPHNODE NODEID) of FROM)
                                                      (fetch (GRAPHNODE FROMNODES) of TO])
```

(GRAPHDELETENODE

```
  [LAMBDA (NODE GRAPH WINDOW)                                      (* kvl " 5-Sep-84 19:03")
    (PROG ((DELFN (fetch (GRAPH GRAPH.DELETENODEFN) of GRAPH)))
          (AND DELFN (APPLY* DELFN NODE GRAPH WINDOW))
          (replace (GRAPH GRAPHNODES) of GRAPH with (DREMOVE NODE (fetch (GRAPH GRAPHNODES) of GRAPH])
```

(GRAPHEDITCOMMANDFN

```
  [LAMBDA (GRAPHWINDOW)                                            (* rmk%: "19-Sep-85 11:12")
    (DECLARE (SPECVARS GRAPHWINDOW))                               (* So that window is available to functions called from menu
                                                                   items)
    (SELECTQ [MENU (COND
                       ((type? MENU EDITGRAPHMENU)
                        EDITGRAPHMENU)
                       (T (SETQ EDITGRAPHMENU (create MENU
                                                      ITEMS _ EDITGRAPHMENUCOMMANDS
                                                      CENTERFLG _ T
                                                      CHANGEOFFSETFLG _ T]
        (STOP 'STOP)
        (MOVENODE (EDITMOVENODE GRAPHWINDOW))
        (ADDNODE (EDITADDNODE GRAPHWINDOW))
        (DELETENODE (EDITDELETENODE GRAPHWINDOW))
        (ADDLINK (EDITADDLINK GRAPHWINDOW))
        (SMALLER (EDITCHANGEFONT 'SMALLER GRAPHWINDOW))
        (LARGER (EDITCHANGEFONT 'LARGER GRAPHWINDOW))
        (DELETELINK (EDITDELETELINK GRAPHWINDOW))
        (CHANGELABEL (EDITCHANGELABEL GRAPHWINDOW))
        (DIRECTED (TOGGLE/DIRECTEDFLG GRAPHWINDOW))
        (SIDES (TOGGLE/SIDESFLG GRAPHWINDOW))
        (BORDER (EDITTOGGLEBORDER GRAPHWINDOW))
        (SHADE (EDITTOGGLELABEL GRAPHWINDOW))
        NIL])
```

(GRAPHEDITEVENTFN

```
  [LAMBDA (GRWINDOW)                                               (* rmk%: "16-Feb-85 10:15")
                                                                   (* implements a graph editor on the right button transition of a
                                                                   window.)
    (COND
       ((NOT (INSIDE? (DSPCLIPPINGREGION NIL GRWINDOW)
                      (LASTMOUSEX GRWINDOW)
                      (LASTMOUSEY GRWINDOW)))
        (DOWINDOWCOM GRWINDOW))
       ((SHIFTDOWNP 'CTRL)
        (TRACKNODE GRWINDOW))
       ((EQ (GRAPHEDITCOMMANDFN GRWINDOW)
            'STOP)                                                 (* do menu)
        (CLOSEW GRWINDOW])
```

(GRAPHER/CENTERPRINTINAREA

```
  [LAMBDA (EXP X Y WIDTH HEIGHT STREAM)                            (* kvl "15-Aug-84 11:01")
    ;; prints an expression in a box.  The system CENTERPRINTINAREA on MENU worried about overflowing the right margin, which we ignore here.
    (SETQ STREAM (\GETSTREAM STREAM 'OUTPUT))
```

```
    (PROG (XPOS (STRWIDTH (STRINGWIDTH EXP STREAM)))
          (MOVETO (SETQ XPOS (IPLUS X (IQUOTIENT (ADD1 (IDIFFERENCE WIDTH STRWIDTH))
                                                 2)))
                  (IPLUS Y (IQUOTIENT (IPLUS (IDIFFERENCE HEIGHT (FONTPROP STREAM 'ASCENT))
                                             (FONTPROP STREAM 'DESCENT))
                                      2))
                  STREAM)
          (PRIN3 EXP STREAM])
```

(GRAPHERPROP

```
  [LAMBDA (GRAPH PROP NEWVALUE)                                    ; Edited 19-Aug-88 14:09 by sye
    (LET (PROPLIST)
         (SETPROPLIST PROPLIST (fetch (GRAPH GRAPH.PROPS) of GRAPH))
         (if NEWVALUE
             then (PROG1 (PUTPROP PROPLIST PROP NEWVALUE)
                         (replace (GRAPH GRAPH.PROPS) of GRAPH with (GETPROPLIST PROPLIST)))
          else (GETPROP PROPLIST PROP])
```

(GRAPHNODE/BORDER/WIDTH

```
  [LAMBDA (BORDER)                                                 (* kvl " 5-Sep-84 16:19")
                                                                   (* returns a non-negative interger)
    (COND
       ((NULL BORDER)
        0)
       ((EQ BORDER T)
        1)
       ((FIXP BORDER)
        (ABS BORDER))
       ((AND (LISTP BORDER)
             (FIXP (CAR BORDER))
             (IGEQ (CAR BORDER)
                   0))
        (CAR BORDER))
       (T (ERROR "Illegal border:" BORDER])
```

(GRAPHREGION

```
  [LAMBDA (GRAPH)                                                  ; Edited 29-Apr-94 14:01 by sybalsky
                                                                   (* Returns the minimum region containing the graph.)
    (PROG (LEFTOFFSET BOTTOMOFFSET (NODELST (fetch (GRAPH GRAPHNODES) of GRAPH)))
          (RETURN (COND
                     [NODELST                                      (* Determine the dimensions of the node labels)
                            (for N in NODELST do (MEASUREGRAPHNODE N))
                            (CREATEREGION (SETQ LEFTOFFSET (MIN/LEFT NODELST))
                                   (SETQ BOTTOMOFFSET (MIN/BOTTOM NODELST))
                                   (ADD1 (IDIFFERENCE (MAX/RIGHT NODELST)
                                                      LEFTOFFSET))
                                   (ADD1 (IDIFFERENCE (MAX/TOP NODELST)
                                                      BOTTOMOFFSET]
                     (T (CREATEREGION 0 0 0 0])
```

(HARDCOPYGRAPH

```
  [LAMBDA (GRAPH/WINDOW FILE IMAGETYPE TRANS)                      ; Edited 23-Apr-92 16:51 by jds
    (LET* ((LANDSCAPE-FLAG (EQ (LISTGET GRAPH/HARDCOPY/FORMAT 'MODE)
                               'LANDSCAPE))
           [PSTREAM (OR (AND FILE (OPENP FILE 'OUTPUT)
                             (GETSTREAM FILE))
                        (OPENIMAGESTREAM FILE IMAGETYPE (APPEND '(CLIP.INCLUSIVE T)
                                                                (AND LANDSCAPE-FLAG '(LANDSCAPE T]
           (PSCALE (DSPSCALE NIL PSTREAM))
           (ORIGINAL-CLIPREGION (DSPCLIPPINGREGION NIL PSTREAM))
           (GRAPH (COND
                     ((WINDOWP GRAPH/WINDOW)
                      (WINDOWPROP GRAPH/WINDOW 'GRAPH))
                     (T GRAPH/WINDOW)))
           (GRAPH-REGION (GRAPHREGION GRAPH))
           (GRAPH-LEFT (fetch (REGION LEFT) of GRAPH-REGION))
           (GRAPH-BOTTOM (fetch (REGION BOTTOM) of GRAPH-REGION))
           (GRAPH-WIDTH (fetch (REGION WIDTH) of GRAPH-REGION))
           (GRAPH-HEIGHT (fetch (REGION HEIGHT) of GRAPH-REGION))
           (SCREENPOINTS-PER-INCH 72)
           (PAGENUMBERS-FLAG (LISTGET GRAPH/HARDCOPY/FORMAT 'PAGENUMBERS))
           [RIGHT-MARGIN (FIXR (TIMES 0 SCREENPOINTS-PER-INCH (OR (LISTGET GRAPH/HARDCOPY/FORMAT 'RIGHTMARGIN)
                                                                  0.5]
           [UPPER-MARGIN (FIXR (TIMES 0 SCREENPOINTS-PER-INCH (OR (LISTGET GRAPH/HARDCOPY/FORMAT 'UPPERMARGIN)
                                                                  0.4]
           (PAGE-WIDTH (- (FIXR (QUOTIENT (fetch (REGION WIDTH) of ORIGINAL-CLIPREGION)
                                          PSCALE))
                          RIGHT-MARGIN))
           (PAGE-HEIGHT (- (FIXR (QUOTIENT (fetch (REGION HEIGHT) of ORIGINAL-CLIPREGION)
                                           PSCALE))
                           UPPER-MARGIN))
           (NUMBER-OF-X-PAGES (CL:CEILING GRAPH-WIDTH PAGE-WIDTH))
           (NUMBER-OF-Y-PAGES (CL:CEILING GRAPH-HEIGHT PAGE-HEIGHT))
```

```
          [X-POSITION (FIXR (TIMES PSCALE (PLUS PAGE-WIDTH (TIMES 0.2 RIGHT-MARGIN]
          [Y-POSITION (FIXR (TIMES PSCALE (PLUS PAGE-HEIGHT (TIMES 0.5 UPPER-MARGIN]
          (BOTTOM-CENTERING-OFFSET NIL)
          [LEFT-CENTERING-OFFSET (LET (TRAN)
                                      (COND
                                         ((type? POSITION TRANS)
                                          (SETQ BOTTOM-CENTERING-OFFSET (fetch YCOORD of TRANS))
                                          (fetch XCOORD of TRANS))
                                         ([type? POSITION (SETQ TRAN (LISTGET GRAPH/HARDCOPY/FORMAT 'TRANS]
                                          (SETQ BOTTOM-CENTERING-OFFSET (fetch YCOORD of TRAN))
                                          (fetch XCOORD of TRAN))
                                         (T (SETQ BOTTOM-CENTERING-OFFSET (QUOTIENT (PLUS UPPER-MARGIN
                                                                                          (DIFFERENCE
                                                                                           PAGE-HEIGHT
                                                                                           (REMAINDER
                                                                                                  GRAPH-HEIGHT
                                                                                                  PAGE-HEIGHT))
                                                                                          )
                                                                                    2))
                                            (QUOTIENT (PLUS RIGHT-MARGIN (DIFFERENCE PAGE-WIDTH
                                                                                     (REMAINDER GRAPH-WIDTH
                                                                                            PAGE-WIDTH)))
                                                      2]
          [CLIPREGION (CREATEREGION 0 0 (FIXR (TIMES PSCALE PAGE-WIDTH))
                                        (FIXR (TIMES PSCALE PAGE-HEIGHT]
          (SCALED-GRAPH (SCALE/GRAPH GRAPH PSTREAM)))
        ;;
        ;; set up  margins and clip/region for the print stream
        ;;
        (DSPLEFTMARGIN 0 PSTREAM)
        (DSPBOTTOMMARGIN 0 PSTREAM)
        (DSPTOPMARGIN (fetch (REGION HEIGHT) of ORIGINAL-CLIPREGION)
               PSTREAM)
        (DSPRIGHTMARGIN (TIMES 2 (fetch (REGION WIDTH) of ORIGINAL-CLIPREGION))
               PSTREAM)
        (DSPCLIPPINGREGION CLIPREGION PSTREAM)
        ;;
        ;; print  graph
        ;;
        [for Y-PAGE-NUMBER from 1 to NUMBER-OF-Y-PAGES
           do (for X-PAGE-NUMBER from 1 to NUMBER-OF-X-PAGES
                 do (LET [(PTRANS (create POSITION
                                          XCOORD _ [FIXR (FTIMES PSCALE (PLUS LEFT-CENTERING-OFFSET
                                                                              (MINUS GRAPH-LEFT)
                                                                              (MINUS (TIMES (SUB1 X-PAGE-NUMBER
                                                                                            )
                                                                                     PAGE-WIDTH]
                                          YCOORD _ (FIXR (FTIMES PSCALE (PLUS BOTTOM-CENTERING-OFFSET
                                                                              (MINUS GRAPH-BOTTOM)
                                                                              (MINUS (TIMES (SUB1 Y-PAGE-NUMBER
                                                                                            )
                                                                                     PAGE-HEIGHT]
                          ;;
                          ;; write a page-full of graph to the print stream
                          ;;
                          (for N in (fetch (GRAPH GRAPHNODES) of SCALED-GRAPH)
                             do (DISPLAYNODELINKS N PTRANS PSTREAM SCALED-GRAPH T PSCALE))
                          (for N in (fetch (GRAPH GRAPHNODES) of SCALED-GRAPH)
                             do (PRINTDISPLAYNODE N PTRANS PSTREAM CLIPREGION))
                          ;;
                          ;; print the page number & start a new page
                          ;;
                          (CL:UNLESS (AND (= X-PAGE-NUMBER NUMBER-OF-X-PAGES)
                                          (= Y-PAGE-NUMBER NUMBER-OF-Y-PAGES))
                              (COND
                                 (PAGENUMBERS-FLAG (DSPCLIPPINGREGION ORIGINAL-CLIPREGION PSTREAM)
                                        (MOVETO X-POSITION Y-POSITION PSTREAM)
                                        (printout PSTREAM Y-PAGE-NUMBER "-" X-PAGE-NUMBER)
                                        (DSPCLIPPINGREGION CLIPREGION PSTREAM)))
                              (DSPNEWPAGE PSTREAM))]
        (CLOSEF PSTREAM])
```

(**INTERSECT/REGIONP/LBWH**

```
  [LAMBDA (L B W H REG HOW NODE)                    ; Edited 11-Jun-90 16:15 by mitani
                                                    ; like intersect regions, but without requiring the consing
                                                    (*
                                                    |how = partial :check if the nodelabel was partially intersect with
REG|)                                               (*
                                                    |otherwise     :check if the whole nodelabel was contained in RE
```

```
G|)
    (SELECTQ HOW
         (PARTIAL (NOT (OR (IGREATERP (fetch (REGION BOTTOM) of REG)
                                      (IPLUS B H))
                           (ILESSP (fetch (REGION PRIGHT) of REG)
                                   L)
                           (IGREATERP (fetch (REGION LEFT) of REG)
                                      (IPLUS L W))
                           (ILESSP (fetch (REGION PTOP) of REG)
                                   B))))
         (EQUAL (INTERSECTREGIONS REG (LIST L B W H))
                (LIST L B W H])
```

(INVERTED/GRAPHNODE/BORDER

```
  [LAMBDA (BORDER)                                          (* kvl " 5-Sep-84 18:49")
                                                            (* returns the right thing to invert a graphnode's border)
    (COND
       ((EQ BORDER T)
        NIL)
       ((NULL BORDER)
        T)
       ((FIXP BORDER)
        (IMINUS BORDER))
       ((AND (LISTP BORDER)
             (FIXP (CAR BORDER)))
        (LIST (CAR BORDER)
              (INVERTED/SHADE/FOR/GRAPHER (CADR BORDER])
```

(INVERTED/SHADE/FOR/GRAPHER

```
  [LAMBDA (SHADE)                                           (* rmk%: "20-Sep-85 09:31")
                                                            (* funny name because hopefully will become system function)
    (COND
       ((EQ SHADE T)
        NIL)
       ((NULL SHADE)
        T)
       ((FIXP SHADE)
        (LOGNOT SHADE))
       ((BITMAPP SHADE)
        (PROG ((NB (BITMAPCOPY SHADE)))
              (BLTSHADE BLACKSHADE NB NIL NIL NIL NIL 'INVERT)
              (RETURN NB)))
       (T (ERROR "Illegal shade:" SHADE])
```

(LAYOUT/POSITION

```
  [LAMBDA (NODE)                                            ; Edited 29-Apr-94 14:00 by sybalsky
    (fetch (GRAPHNODE NODEPOSITION) of NODE])
```

(LINKPARAMETERS

```
  [LAMBDA (FROMND TOND)                                     ; Edited 29-Apr-94 14:00 by sybalsky
    (PROG (TOPARAMS)
          (RETURN (AND (SETQ TOPARAMS (MEMBTONODES (fetch (GRAPHNODE NODEID) of TOND)
                                                   (TOLINKS FROMND)))
                       (LISTP TOPARAMS)
                       (EQ 'Link% Parameters (CAR TOPARAMS))
                       (CDDR TOPARAMS])
```

(MAX/RIGHT

```
  [LAMBDA (NODES)                                           (* rmk%: "20-Dec-84 09:33")
    (for NODE in NODES largest (GN/RIGHT NODE) finally (RETURN $$EXTREME])
```

(MAX/TOP

```
  [LAMBDA (NODES)                                           (* rmk%: "20-Dec-84 09:34")
    (for NODE in NODES largest (GN/TOP NODE) finally (RETURN $$EXTREME])
```

(MEASUREGRAPHNODE

```
  [LAMBDA (NODE RESETFLG)                                   ; Edited 29-Apr-94 14:00 by sybalsky
                                                            (* Measure the nodelabel image)
    (SET/LABEL/SIZE NODE RESETFLG)
    (SET/LAYOUT/POSITION NODE (OR (fetch (GRAPHNODE NODEPOSITION) of NODE)
                                  (ERROR "This graphnode has not been given a position:" NODE])
```

(MEMBTONODES

```
  [LAMBDA (TOND TONODES)                                    (* dgb%: "24-Jan-85 08:05")
    (for Z in TONODES do (COND
                           ([OR (EQ TOND Z)
                                (AND (LISTP Z)
                                     (EQ (CAR Z)
                                         'Link% Parameters)
```

```
                                    (EQ TOND (CADR Z]
                            (RETURN Z])
```

(MIN/BOTTOM

```
(MIN/BOTTOM
  [LAMBDA (NODES)                                                    (* rmk%: "20-Dec-84 09:34")
                                                                     (* returns the bottommost point of the graph.)
    (for NODE in NODES smallest (GN/BOTTOM NODE) finally (RETURN $$EXTREME])
```

(MIN/LEFT

```
(MIN/LEFT
  [LAMBDA (NODES)                                                    (* rmk%: "20-Dec-84 09:34")
                                                                     (* returns the leftmost point of the graph.)
    (for NODE in NODES smallest (GN/LEFT NODE) finally (RETURN $$EXTREME])
```

(MOVENODE

```
(MOVENODE
  [LAMBDA (NODE OLDPOS POS GRAPH STREAM)                             (* rmk%: "10-Apr-84 12:31")
                                                                     (* moves a node from its current position to POS)
    (COND
       ((EQUAL OLDPOS POS)                                           (* don't move if position hasn't changed)
        NIL)
       (T                                                            (* node is flipped, flip it back.)
          (FLIPNODE NODE STREAM)                                     (* erase current position)
          (DISPLAYNODE NODE (CONSTANT (create POSITION
                                              XCOORD _ 0
                                              YCOORD _ 0))
                STREAM GRAPH)                                        (* put it in new one.)
          (SET/LAYOUT/POSITION NODE POS)
          (DISPLAYNODE NODE (CONSTANT (create POSITION
                                              XCOORD _ 0
                                              YCOORD _ 0))
                STREAM GRAPH)
          (FLIPNODE NODE STREAM])
```

(NODECREATE

```
(NODECREATE
  [LAMBDA (ID LABEL POS TONODEIDS FROMNODEIDS FONT BORDER LABELSHADE)
                                                                     (* Randy.Gobbel "13-May-87 12:04")
                                                                     (* creates a node for a grapher.)
    (create GRAPHNODE
            NODEID _ ID
            NODEPOSITION _ POS
            NODELABEL _ LABEL
            NODEFONT _ (COND
                          (FONT)
                          ((IMAGEOBJP LABEL)
                           NIL)
                          (DEFAULT.GRAPH.NODEFONT)
                          (T (FONTNAMELIST DEFAULTFONT)))
            TONODES _ TONODEIDS
            FROMNODES _ FROMNODEIDS
            NODEBORDER _ BORDER
            NODELABELSHADE _ LABELSHADE])
```

(NODELST/AS/MENU

```
(NODELST/AS/MENU
  [LAMBDA (NODELST POS)                                              ; Edited 29-Apr-94 14:00 by sybalsky
                                                                     (* finds the node that is closest to POS)
    (for N in NODELST bind (X _ (fetch XCOORD of POS))
                           (Y _ (fetch YCOORD of POS))
                           T1 T2
       thereis (AND (ILESSP [IDIFFERENCE (SETQ T1 (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of N)))
                                         (SETQ T2 (HALF (fetch (GRAPHNODE NODEHEIGHT) of N]
                                Y)
                    (ILESSP Y (IPLUS T1 T2))
                    (ILESSP [IDIFFERENCE (SETQ T1 (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of N)))
                                         (SETQ T2 (HALF (fetch (GRAPHNODE NODEWIDTH) of N]
                                X)
                    (ILESSP X (IPLUS T1 T2])
```

(NODEREGION

```
(NODEREGION
  [LAMBDA (NODE)                                                     (* kvl "10-Aug-84 17:25")
                                                                     (* returns the region taken up by NODE)
    (CREATEREGION (GN/LEFT NODE)
           (GN/BOTTOM NODE)
           (fetch (GRAPHNODE NODEWIDTH) of NODE)
           (fetch (GRAPHNODE NODEHEIGHT) of NODE])
```

(PRINTDISPLAYNODE

```
(PRINTDISPLAYNODE
  [LAMBDA (NODE TRANS STREAM CLIP/REG)                               ; Edited 29-Apr-94 14:00 by sybalsky
                                                                     ; Edited 12-Aug-88 12:58 by sye
    ;; prints a node at its position translated by TRANS.  Takes the operation from the stream so that when editor has set the operation to invert, this
    ;; may erase as well as draw;  but when the operation is paint, then nodes obliterate any link lines that they are drawn over.
```

```
    (OR (ZEROP (fetch (GRAPHNODE NODEHEIGHT) of NODE))
        (PROG* [(LABEL (fetch (GRAPHNODE NODELABEL) of NODE))
                (LEFT (IPLUS (fetch (POSITION XCOORD) of TRANS)
                             (GN/LEFT NODE)))
                (BOTTOM (IPLUS (fetch (POSITION YCOORD) of TRANS)
                               (GN/BOTTOM NODE)))
                (WIDTH (fetch (GRAPHNODE NODEWIDTH) of NODE))
                (HEIGHT (fetch (GRAPHNODE NODEHEIGHT) of NODE))
                (FONT (fetch (GRAPHNODE NODEFONT) of NODE))
                (NBW (GRAPHNODE/BORDER/WIDTH (fetch (GRAPHNODE NODEBORDER) of NODE]
               [AND (WINDOWP STREAM)
                    (SETQ STREAM (WINDOWPROP STREAM 'DSP]
               (COND
                  ([AND CLIP/REG (NOT (INTERSECT/REGIONP/LBWH LEFT BOTTOM WIDTH HEIGHT CLIP/REG 'PARTIAL]
                   (RETURN NODE))
                  ((BITMAPP (fetch (GRAPHNODE NODELABELBITMAP) of NODE))
                   (BITBLT (fetch (GRAPHNODE NODELABELBITMAP) of NODE)
                          0 0 STREAM LEFT BOTTOM WIDTH HEIGHT 'INPUT))
                  [(BITMAPP LABEL)
                   (COND
                      ((NEQ 0 NBW)
                       (DRAW/GRAPHNODE/BORDER (fetch (GRAPHNODE NODEBORDER) of NODE)
                              LEFT BOTTOM WIDTH HEIGHT STREAM)
                       (BITBLT LABEL 0 0 STREAM (IPLUS LEFT NBW)
                              (IPLUS BOTTOM NBW)
                              (BITMAPWIDTH LABEL)
                              (BITMAPHEIGHT LABEL)
                              'INPUT))
                      (T (BITBLT LABEL 0 0 STREAM LEFT BOTTOM WIDTH HEIGHT 'INPUT]
                  ((IMAGEOBJP LABEL)
                   (OR (ZEROP NBW)
                       (DRAW/GRAPHNODE/BORDER (fetch (GRAPHNODE NODEBORDER) of NODE)
                              LEFT BOTTOM WIDTH HEIGHT STREAM))
```

(* RMK--In order to place image objects properly, must take into account their XKERN and YDESC)

```
                   (LET ((IMAGEBOX (APPLY* (IMAGEOBJPROP LABEL 'IMAGEBOXFN)
                                           LABEL STREAM 0 WIDTH)))
                                                          (* Formerly just LEFT and BOTTOM)
                        (MOVETO (IPLUS NBW LEFT (fetch XKERN of IMAGEBOX))
                               (IPLUS NBW BOTTOM (fetch YDESC of IMAGEBOX))
                               STREAM))
```

(* * End of modifications. RMK)

```
                   (APPLY* (IMAGEOBJPROP LABEL 'DISPLAYFN)
                          LABEL STREAM))
                  ((EQ FONT 'SHADE)                          (* so small just use texture)
                   (LET [(2SCALE (ITIMES 2 (DSPSCALE NIL STREAM]
                        (BLTSHADE BLACKSHADE STREAM LEFT BOTTOM 2SCALE 2SCALE)))
                  ((NULL FONT))
                  (T (OR (FONTP FONT)
                         (SETQ FONT (FONTCREATE FONT NIL NIL NIL STREAM)))
                     (AND (NEQ NBW 0)
                          (DRAW/GRAPHNODE/BORDER (fetch (GRAPHNODE NODEBORDER) of NODE)
                                 LEFT BOTTOM WIDTH HEIGHT STREAM))
                     (DSPFONT FONT STREAM)
                     (GRAPHER/CENTERPRINTINAREA LABEL LEFT BOTTOM WIDTH HEIGHT STREAM)
                     (AND (fetch (GRAPHNODE NODELABELSHADE) of NODE)
                          (FILL/GRAPHNODE/LABEL (fetch (GRAPHNODE NODELABELSHADE) of NODE)
                                 LEFT BOTTOM WIDTH HEIGHT NBW STREAM))
                     (COND
                        ((AND CACHE/NODE/LABEL/BITMAPS (DISPLAYSTREAMP STREAM)
                              CLIP/REG
                              (INTERSECT/REGIONP/LBWH LEFT BOTTOM WIDTH HEIGHT CLIP/REG 'WHOLE))
                         (replace (GRAPHNODE NODELABELBITMAP) of NODE with (BITMAPCREATE WIDTH HEIGHT))
                         (BITBLT STREAM LEFT BOTTOM (fetch (GRAPHNODE NODELABELBITMAP) of NODE)
                                0 0 WIDTH HEIGHT 'INPUT])
```

(PROMPTINWINDOW

```
  [LAMBDA (PROMPTSTR POSITION WHICHCORNER BORDERSIZE)          (* jds "18-Mar-86 17:49")
                                                               (* opens a small window for prompting at a position and
                                                               PROMPTFORWORD's a word.)
```

(* POSITION is the location in screen coordinate of the window.
Default is the cursor position.)

(* WHICHCORNER can be a list of up to two of the atoms LEFT RIGHT TOP BOTTOM which specify which corner position is intended to be. Default is lower left.)

(* BORDERSIZE is the border size of the prompt window. Default is 6.0)

```
    (PROG ((PROMPTWBORDER (OR (NUMBERP BORDERSIZE)
                              6))
           (X (COND
                 (POSITION (fetch (POSITION XCOORD) of POSITION))
                 (T LASTMOUSEX)))
```

```
           (Y (COND
                  (POSITION (fetch (POSITION YCOORD) of POSITION))
                  (T LASTMOUSEY)))
           HGHT WDTH READSTR PREVTTY)
          (SETQ HGHT (HEIGHTIFWINDOW (ITIMES (FONTPROP (DEFAULTFONT 'DISPLAY)
                                                       'HEIGHT)
                                             2)
                                     T PROMPTWBORDER))
          (SETQ WDTH (WIDTHIFWINDOW (IMAX (STRINGWIDTH PROMPTSTR WindowTitleDisplayStream)
                                          60)
                                    PROMPTWBORDER))
          (SETQ PREVTTY (CREATEW (CREATEREGION (COND
                                               ((MEMB 'RIGHT WHICHCORNER)
                                                (DIFFERENCE X WDTH))
                                               (T X))
                                            (COND
                                               ((MEMB 'TOP WHICHCORNER)
                                                (DIFFERENCE Y HGHT))
                                               (T Y))
                                            WDTH HGHT)
                                 PROMPTSTR PROMPTWBORDER))
          (DSPLEFTMARGIN (IMAX 0 (fetch (CURSOR CUHOTSPOTX) of (CARET)))
                 PREVTTY)
          (MOVETOUPPERLEFT PREVTTY)
          [SETQ READSTR (ERSETQ (PROMPTFORWORD NIL NIL NIL PREVTTY NIL NIL (LIST (CHARCODE EOL]
          (CLOSEW PREVTTY)
          (RETURN (COND
                     (READSTR (CAR READSTR))
                     (T                                          (* pass back the error.)
                        (ERROR!])
```

(READ/NODE

```
[LAMBDA (NODES DS)                                               ; Edited 23-Jul-87 18:20 by sye
```

(* * rht 8/20/85%: Modified "until" statement so it waits till user clicks inside of window.)

```
    [bind (CR _ (DSPCLIPPINGREGION NIL DS)) until (AND (MOUSESTATE (OR LEFT MIDDLE RIGHT))
                                                       (INSIDEP CR (CURSORPOSITION NIL DS]
    (PROG (NEAR NOW OLDPOS)
          [SETQ NEAR (NODELST/AS/MENU NODES (SETQ OLDPOS (CURSORPOSITION NIL DS]
      FLIP
                                                                 (* turn off old flip (if one) and turn on new flip.)
          (AND NOW (FLIPNODE NOW DS))
          (AND NEAR (FLIPNODE NEAR DS))
          (SETQ NOW NEAR)
      LP  (COND
             ((MOUSESTATE UP)
              (AND NOW (FLIPNODE NOW DS))
              (RETURN NOW))
             ([EQ NOW (SETQ NEAR (NODELST/AS/MENU NODES (CURSORPOSITION NIL DS OLDPOS]
              (GO LP))
             (T (GO FLIP])
```

(REDISPLAYGRAPH

```
[LAMBDA (WINDOW REGION)                                          (* kvl "10-Aug-84 19:52")
```

(* displays the graph that is in a window. REGION if given is the clipping region.
Later this could be used to make things run faster.)

```
    (DSPFILL NIL NIL 'REPLACE WINDOW)
    (DISPLAYGRAPH (WINDOWPROP WINDOW 'GRAPH)
           WINDOW
           (OR REGION (DSPCLIPPINGREGION NIL WINDOW])
```

(REMOVETONODES

```
[LAMBDA (TOND TONODES)                                           (* rht%: " 4-Apr-85 19:32")
```

(* * Removes either TOND or a paramlist thingie for TOND.)

```
    (for Z in TONODES unless [OR (EQ Z TOND)
                                 (AND (LISTP Z)
                                      (EQ (CAR Z)
                                          'Link% Parameters)
                                      (EQ TOND (CADR Z]
       collect Z])
```

(RESET/NODE/BORDER

```
[LAMBDA (NODE BORDER STREAM GRAPH TRANS)                         ; Edited 29-Apr-94 14:00 by sybalsky
```

;; gives the node a new border, and displays it if there is a stream. Might not be a stream if being called just to finagle a graph datastructure.

```
    (PROG [(ONBW (GRAPHNODE/BORDER/WIDTH (fetch (GRAPHNODE NODEBORDER) of NODE]
          [OR TRANS (SETQ TRANS (CONSTANT (create POSITION
                                                  XCOORD _ 0
```

```
                                                YCOORD _ 0]
        (COND
           (STREAM (ERASE/GRAPHNODE NODE STREAM TRANS)
                   [OR GRAPH (AND (WINDOWP STREAM)
                                  (SETQ GRAPH (WINDOWPROP STREAM 'GRAPH]
                   (DISPLAYNODELINKS NODE TRANS STREAM GRAPH)))
        (replace (GRAPHNODE NODEBORDER) of NODE with (COND
                                                        ((EQ BORDER 'INVERT)
                                                         (INVERTED/GRAPHNODE/BORDER (fetch (GRAPHNODE
                                                                                             NODEBORDER)
                                                                                           of NODE)))
                                                        (T BORDER)))
        (replace (GRAPHNODE NODELABELBITMAP) of NODE with NIL)
        (OR (IEQP ONBW (GRAPHNODE/BORDER/WIDTH (fetch (GRAPHNODE NODEBORDER) of NODE)))
            (SET/LABEL/SIZE NODE T))
        (AND STREAM (DISPLAYNODE NODE TRANS STREAM GRAPH))
        (RETURN NODE])
```

(RESET/NODE/LABELSHADE

```
  [LAMBDA (NODE SHADE STREAM TRANS)                          ; Edited 29-Apr-94 14:00 by sybalsky
                                                             (* gives the node a new SHADE and displays it if there is a
                                                             stream)
    (AND STREAM (ERASE/GRAPHNODE NODE STREAM TRANS))
    (replace (GRAPHNODE NODELABELSHADE) of NODE with (COND
                                                       ((EQ SHADE 'INVERT)
                                                        (INVERTED/SHADE/FOR/GRAPHER (fetch (GRAPHNODE
                                                                                             NODELABELSHADE)
                                                                                           of NODE)))
                                                       (T SHADE)))
    (replace (GRAPHNODE NODELABELBITMAP) of NODE with NIL)
    (AND STREAM (PRINTDISPLAYNODE NODE (OR TRANS (CONSTANT (create POSITION
                                                                  XCOORD _ 0
                                                                  YCOORD _ 0)))
                        STREAM
                        (DSPCLIPPINGREGION NIL STREAM)))
    NODE])
```

(SCALE/GRAPH

```
  [LAMBDA (GRAPH STREAM)                                     ; Edited 29-Apr-94 14:01 by sybalsky
```

;; Scale the graph GRAPH so it'll look right when rendered on the image stream STREAM. This involves both scaling all the coordinates, and fixing ;; node positions (because we keep track of teh CENTER of each node, even though we really want the lower, left corner to be in the right place).

```
    (LET ((SCALE (DSPSCALE NIL STREAM))
          [LAYOUT-IS-VERTICAL (EQMEMB 'VERTICAL (LISTGET (fetch (GRAPH GRAPH.PROPS) of GRAPH)
                                                         'FORMAT]
          HEIGHT WIDTH)
         (create GRAPH
            using GRAPH GRAPHNODES _
                  (for N in (fetch (GRAPH GRAPHNODES) of GRAPH)
                     collect
```

;; Move each node to its new position.

;; Start by finding the node's lower, left corner, then scaling that.

```
                     (SETQ WIDTH (fetch (GRAPHNODE NODEWIDTH) of N))
                     (SETQ HEIGHT (fetch (GRAPHNODE NODEHEIGHT) of N))
                     (SETQ N (create GRAPHNODE
                                using N NODEPOSITION _
                                      [COND
                                         [LAYOUT-IS-VERTICAL
```

;; Layout is vertical, so make the center correct.

```
                                          (create POSITION
                                                  XCOORD _ [FIXR (FTIMES SCALE (fetch XCOORD
                                                                                  of (fetch (GRAPHNODE
                                                                                              NODEPOSITION
                                                                                              )
                                                                                            of N]
                                                  YCOORD _ (FIXR (FTIMES SCALE (fetch YCOORD
                                                                                  of (fetch (GRAPHNODE
                                                                                              NODEPOSITION
                                                                                              )
                                                                                            of N]
                                         (T
```

;; Horizontal layout, so make the left bottom correct.

```
                                          (create POSITION
                                                  XCOORD _ [FIXR (FTIMES SCALE
                                                                         (IDIFFERENCE
                                                                          (fetch XCOORD
                                                                             of (fetch (GRAPHNODE
                                                                                         NODEPOSITION)
                                                                                       of N))
                                                                          (LRSH WIDTH 1]
                                                  YCOORD _ (FIXR (FTIMES SCALE
                                                                         (IDIFFERENCE
                                                                          (fetch YCOORD
```

```
                                                                    of (fetch (GRAPHNODE
                                                                                NODEPOSITION)
                                                                       of N))
                                                                 (LRSH HEIGHT 1]
                                NODEWIDTH _ NIL NODEHEIGHT _ NIL NODEFONT _
                                (FONTCREATE (fetch (GRAPHNODE NODEFONT)
                                                   N)
                                       NIL NIL NIL STREAM)
                                TONODES _ (SCALE/TONODES N SCALE)
                                NODEBORDER _ (SCALE/GRAPHNODE/BORDER (fetch (GRAPHNODE NODEBORDER)
                                                                        of N)
                                                                     SCALE)))
                    ;; Now figure out the new width & height of the node:
                    (SET/LABEL/SIZE N NIL STREAM)
                    ;; Now find the new center point, so the node prints in the right place:
                    [COND
                       ((NOT LAYOUT-IS-VERTICAL)
                        ;; Only do this if the layout is horizontal.
                        (add (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of N))
                             (LRSH (fetch (GRAPHNODE NODEHEIGHT) of N)
                                   1))
                        (add (fetch XCOORD of (fetch (GRAPHNODE NODEPOSITION) of N))
                             (LRSH (fetch (GRAPHNODE NODEWIDTH) of N)
                                   1]
                    N])
```

(SCALE/GRAPHNODE/BORDER

```
[LAMBDA (BORDER SCALE)                                  (* kvl " 5-Sep-84 18:06")
                                                        (* returns a new setting for the border appropriate for the given
                                                        SCALE)

   (COND
      ((NULL BORDER)
       0)
      ((EQ BORDER T)
       (FIXR (FTIMES SCALE NODEBORDERWIDTH)))
      ((FIXP BORDER)
       (FIXR (FTIMES SCALE BORDER)))
      ((AND (LISTP BORDER)
            (FIXP (CAR BORDER)))
       (CONS (FIXR (FTIMES SCALE (CAR BORDER)))
             (CDR BORDER])
```

(SCALE/TONODES

```
[LAMBDA (NODE SCALE)                                    ; Edited 29-Apr-94 14:00 by sybalsky
   (for NODEID in (fetch (GRAPHNODE TONODES) of NODE)
      collect                                           (* copy the property list so that the scaling doesn't change the
                                                        original.)
            (COND
               [(AND (LISTP NODEID)
                     (EQ 'Link% Parameters (CAR NODEID))
                     (SETQ NODEID (APPEND NODEID))
                     (for prop val in ScalableLinkParameters do (AND (SETQ val (LISTGET NODEID prop))
                                                                     (LISTPUT NODEID prop (FIX/SCALE val SCALE]
               (T NODEID])
```

(SET/LABEL/SIZE

```
[LAMBDA (NODE RESET/FLG STREAM)                         ; Edited 29-Apr-94 14:00 by sybalsky
                                                        (* the SHADE and null font stuff is for ZOOMGRAPH)

   (OR (AND (NOT RESET/FLG)
            (FIXP (fetch (GRAPHNODE NODEHEIGHT) of NODE))
            (FIXP (fetch (GRAPHNODE NODEWIDTH) of NODE)))
       (PROG ((SCALE (DSPSCALE NIL STREAM))
              (FONT (fetch (GRAPHNODE NODEFONT) of NODE))
              (LAB (fetch (GRAPHNODE NODELABEL) of NODE))
              (NBW (GRAPHNODE/BORDER/WIDTH (fetch (GRAPHNODE NODEBORDER) of NODE)))
             WIDTH HEIGHT)
             [COND
                [(BITMAPP LAB)                          (* (* ; "set up appropriate width & height by checking the scale
                                                        of stream"))
                 (SETQ WIDTH (OR [AND (NEQ SCALE 1)
                                      (FIXR (FTIMES SCALE (BITMAPWIDTH LAB]
                                 (BITMAPWIDTH LAB)))
                 (SETQ HEIGHT (OR [AND (NEQ SCALE 1)
                                       (FIXR (FTIMES SCALE (BITMAPHEIGHT LAB]
                                  (BITMAPHEIGHT LAB]
                ((IMAGEOBJP LAB)
                 (SETQ WIDTH (APPLY* (IMAGEOBJPROP LAB 'IMAGEBOXFN)
                                     LAB STREAM))
                 (SETQ HEIGHT (fetch (IMAGEBOX YSIZE) of WIDTH))
                 (SETQ WIDTH (fetch (IMAGEBOX XSIZE) of WIDTH)))
                ((EQ FONT 'SHADE)                       (* node image is very small)
```

```
                (SETQ WIDTH (SETQ HEIGHT 2)))
               [(NULL FONT)                                    (* FONT of NIL means that the node is smaller than displays)
                (SETQ NBW (SETQ WIDTH (SETQ HEIGHT 0]
               (T (OR (FONTP FONT)
                      (SETQ FONT (FONTCREATE FONT NIL NIL NIL STREAM)))
                  [SETQ WIDTH (IPLUS (STRINGWIDTH (fetch (GRAPHNODE NODELABEL) of NODE)
                                                  FONT)
                                     (FONTPROP FONT 'DESCENT]
                  (SETQ HEIGHT (IPLUS (FONTPROP FONT 'HEIGHT)
                                      (FONTPROP FONT 'DESCENT]
            (OR (AND (NOT RESET/FLG)
                     (FIXP (fetch (GRAPHNODE NODEWIDTH) of NODE)))
                (replace (GRAPHNODE NODEWIDTH) of NODE with (IPLUS WIDTH NBW NBW)))
            (OR (AND (NOT RESET/FLG)
                     (FIXP (fetch (GRAPHNODE NODEHEIGHT) of NODE)))
                (replace (GRAPHNODE NODEHEIGHT) of NODE with (IPLUS HEIGHT NBW NBW)))
            (RETURN NODE])
```

(SET/LAYOUT/POSITION

```
[LAMBDA (NODE POS)                                                  ; Edited 29-Apr-94 14:00 by sybalsky
                                                                    (* sets a nodes position)
  (replace XCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE) with (fetch XCOORD of POS))
  (replace YCOORD of (fetch (GRAPHNODE NODEPOSITION) of NODE) with (fetch YCOORD of POS))
  NODE])
```

(SHOWGRAPH

```
[LAMBDA (GRAPH WINDOW LEFTBUTTONFN MIDDLEBUTTONFN TOPJUSTIFYFLG ALLOWEDITFLG COPYBUTTONEVENTFN CENTERFLG)
                                                                    ; Edited 28-Sep-93 17:20 by jds
  ;; puts a graph in the given window, creating one if a window is not given.
  (SETQ WINDOW (SIZE/GRAPH/WINDOW (COND
                                     ((NULL GRAPH)
                                      (SETQ GRAPH (create GRAPH)))
                                     (T GRAPH))
                   (COND
                      (WINDOW)
                      (ALLOWEDITFLG                             ; put on a title so there will be a place to get window commands.
                           "Graph Editor Window"))
                   TOPJUSTIFYFLG CENTERFLG))
  (WINDOWPROP WINDOW 'GRAPH GRAPH)
  (WINDOWPROP WINDOW 'REPAINTFN (FUNCTION REDISPLAYGRAPH))
  (WINDOWPROP WINDOW 'SCROLLFN (FUNCTION SCROLLBYREPAINTFN))
  (WINDOWPROP WINDOW 'HARDCOPYFN (FUNCTION HARDCOPYGRAPH))
  (COND
     (ALLOWEDITFLG                                                  ; change the mode to invert so lines can be erased by being
                                                                    ; redrawn.
          (DSPOPERATION 'INVERT WINDOW)
          (WINDOWPROP WINDOW 'RIGHTBUTTONFN (FUNCTION GRAPHEDITEVENTFN)))
     (T (WINDOWPROP WINDOW 'RIGHTBUTTONFN NIL)))
  (WINDOWPROP WINDOW 'COPYBUTTONEVENTFN (OR COPYBUTTONEVENTFN (FUNCTION GRAPHERCOPYBUTTONEVENTFN)))
  (WINDOWPROP WINDOW 'BUTTONEVENTFN (FUNCTION APPLYTOSELECTEDNODE))
  (WINDOWPROP WINDOW 'BROWSER/LEFTFN LEFTBUTTONFN)
  (WINDOWPROP WINDOW 'BROWSER/MIDDLEFN MIDDLEBUTTONFN)
  (REDISPLAYGRAPH WINDOW)
  WINDOW])
```

(SIZE/GRAPH/WINDOW

```
[LAMBDA (GRAPH WINDOW/TITLE TOPJUSTIFYFLG CENTERFLG)                ; Edited 28-Sep-93 17:21 by jds
  ;; returns a window sized to fit the given graph.  WINDOW/TITLE can be either a window to be printed in or a title of a window to be created.  If
  ;; TOPJUSTIFYFLG is true, scrolls so top of graph is at top of window, else puts bottom of graph at bottom of window.
  (PROG ((GRAPHREG (GRAPHREGION GRAPH))
         TITLE WINDOW)
        (COND
           ((WINDOWP WINDOW/TITLE)
            (SETQ WINDOW WINDOW/TITLE))
           (T (SETQ TITLE WINDOW/TITLE)))
   ;; if there is not already a window, ask the user for one to fit.
        (COND
           ((NULL WINDOW)
            (SETQ WINDOW (CREATEW (GETBOXREGION (WIDTHIFWINDOW (IMIN (IMAX (fetch (REGION WIDTH) of GRAPHREG)
                                                                          100)
                                                                     (FIXR (CAR DEFAULT.GRAPH.WINDOWSIZE))
                                                                     SCREENWIDTH))
                                                (HEIGHTIFWINDOW (IMIN (IMAX (fetch (REGION HEIGHT) of GRAPHREG)
                                                                            60)
                                                                      (FIXR (CADR DEFAULT.GRAPH.WINDOWSIZE))
                                                                      SCREENHEIGHT)
                                                                TITLE))
                                  TITLE)))
           (T (CLEARW WINDOW)))
        (WINDOWPROP WINDOW 'EXTENT GRAPHREG)
        (WXOFFSET [COND
```

```
                    [CENTERFLG (IDIFFERENCE (WXOFFSET NIL WINDOW)
                                      (IDIFFERENCE (IPLUS (fetch (REGION LEFT) of GRAPHREG)
                                                          (LRSH (fetch (REGION WIDTH) of GRAPHREG)
                                                                1))
                                                   (LRSH (WINDOWPROP WINDOW 'WIDTH)
                                                         1]
                    (T                                                  ; Put it at the left edge.
                       (IDIFFERENCE (WXOFFSET NIL WINDOW)
                              (fetch (REGION LEFT) of GRAPHREG]
             WINDOW)
          (WYOFFSET [IDIFFERENCE (WYOFFSET NIL WINDOW)
                           (COND
                              [TOPJUSTIFYFLG (IDIFFERENCE (fetch (REGION PTOP) of GRAPHREG)
                                                          (WINDOWPROP WINDOW 'HEIGHT]
                              (T (fetch (REGION BOTTOM) of GRAPHREG]
             WINDOW)
          (RETURN WINDOW])
```

(TOGGLE/DIRECTEDFLG

```
(LAMBDA (WIN)                                                   (* kvl "20-APR-82 13:38")
                                                                (* flips the value of the flag that indicates whether the graph is a
  lattice.)
  [replace (GRAPH DIRECTEDFLG) of (WINDOWPROP WIN 'GRAPH) with (NOT (fetch (GRAPH DIRECTEDFLG)
                                                                          of (WINDOWPROP WIN 'GRAPH]
  (DSPFILL NIL (DSPTEXTURE NIL WIN)
         'REPLACE WIN)
  (REDISPLAYGRAPH WIN])
```

(TOGGLE/SIDESFLG

```
(LAMBDA (WIN)                                                   (* kvl "20-APR-82 13:15")

        (* flips the value of the flag that indicates whether the graph is to be layed out vertically or horizontally.)

  [replace (GRAPH SIDESFLG) of (WINDOWPROP WIN 'GRAPH) with (NOT (fetch (GRAPH SIDESFLG)
                                                                       of (WINDOWPROP WIN 'GRAPH]
  (DSPFILL NIL (DSPTEXTURE NIL WIN)
         'REPLACE WIN)
  (REDISPLAYGRAPH WIN])
```

(TOLINKS

```
(LAMBDA (NODE)                                                  ; Edited 29-Apr-94 14:00 by sybalsky
  (fetch (GRAPHNODE TONODES) of NODE])
```

(TRACKCURSOR

```
(LAMBDA (ND DS GRAPH)                                           ; Edited 29-Apr-94 14:00 by sybalsky
                                                                (* causes ND to follow cursor.)
  (PROG (OLDPOS ORIGPOS DOWNFLG)                                (* maybe there aren't any nodes)
        (OR ND (RETURN))
        (SETQ ORIGPOS (create POSITION using (fetch (GRAPHNODE NODEPOSITION) of ND)))
        (SETQ OLDPOS (CURSORPOSITION (fetch (GRAPHNODE NODEPOSITION) of ND)
                           DS))
        (FLIPNODE ND DS)
        (until (COND
                  (DOWNFLG (MOUSESTATE UP))
                  ((SETQ DOWNFLG (MOUSESTATE (NOT UP)))
                   NIL))
           do (MOVENODE ND (fetch (GRAPHNODE NODEPOSITION) of ND)
                        (CURSORPOSITION NIL DS OLDPOS)
                        GRAPH DS))
        (FLIPNODE ND DS)
        (COND
           ([NOT (EQUAL ORIGPOS (SETQ OLDPOS (fetch (GRAPHNODE NODEPOSITION) of ND]
            (EXTENDEXTENT (WFROMDS DS)
                   (NODEREGION ND))
            (CALL.MOVENODEFN ND OLDPOS GRAPH (WFROMDS DS)
                   ORIGPOS])
```

(TRACKNODE

```
(LAMBDA (W)                                                     ; Edited 17-Jul-87 15:26 by sye

        (* grabs the nearest nodes and hauls it around with the cursor, leaving it where it is when the button goes up.)

  (TRACKCURSOR (NODELST/AS/MENU (fetch (GRAPH GRAPHNODES) of (WINDOWPROP W 'GRAPH))
                       (CURSORPOSITION NIL W))
         (WINDOWPROP W 'DSP)
         (WINDOWPROP W 'GRAPH])
```

(TRANSGRAPH

```
(LAMBDA (GRAPH X Y)                                             ; Edited 29-Apr-94 14:01 by sybalsky
  (create GRAPH reusing GRAPH GRAPHNODES _ (for N in (fetch (GRAPH GRAPHNODES) of GRAPH)
                                                   collect (create GRAPHNODE
```

```
                                                    reusing N NODEPOSITION _
                                                          (create POSITION
                                                                XCOORD _ (PLUS X (fetch XCOORD
                                                                                    of (fetch (GRAPHNODE
                                                                                               NODEPOSITION)
                                                                                       of N)))
                                                                YCOORD _ (PLUS Y (fetch YCOORD
                                                                                    of (fetch (GRAPHNODE
                                                                                               NODEPOSITION)
                                                                                       of N]))

)

          (* * Was MODERNIZE loaded before?)

(CL:WHEN (GETD 'MODERNWINDOW.SETUP)
    (MODERNWINDOW.SETUP 'APPLYTOSELECTEDNODE))

;; Support for EDITSUBGRAPH and EDITREGION

(DEFINEQ

(EDITMOVEREGION
  [LAMBDA (Window)                                          (* Newman "27-Jan-87 11:08")

          (* * This function moves all the nodes within a selected region to another region of similar shape and size.)

    (if (NOT (WINDOWP Window))
        then (ERROR Window " not a window.")
      else (PROMPTPRINT " Select the region containing the nodes you wish to move.")
           (PROG* ((DisplayStream (WINDOWPROP Window 'DSP))
                   (Region (GETWREGION Window))
                   (Graph (WINDOWPROP Window 'GRAPH))
                   (NodeList (for Node in (fetch (GRAPH GRAPHNODES) of Graph)
                                when (OR (INTERSECTREGIONS Region (NODEREGION Node))
                                         (SUBREGIONP Region (NODEREGION Node)))
                                collect Node)))
                  (if (NULL Graph)
                      then (ERROR Window " not a graph window.")
                    elseif (NULL NodeList)
                      then (PROMPTPRINT "No nodes in the region selected."))
                  (for Node in NodeList do (FLIPNODE Node DisplayStream))
                  (bind OldPos (NewRegionPosition _ (GETBOXPOSITION.FROMINITIALREGION Window Region
                                                           DisplayStream))
                    for SelectedNode in NodeList eachtime (SETQ OldPos (fetch (GRAPHNODE NODEPOSITION) of
                                                                                                SelectedNode
                                                                                                ))
                    do (MOVENODE SelectedNode OldPos (CREATE.NEW.NODEPOSITION SelectedNode
                                                          (DIFFERENCE (fetch (POSITION XCOORD) of
                                                                                                NewRegionPosition
                                                                                                )
                                                                      (fetch (REGION LEFT) of Region))
                                                          (DIFFERENCE (fetch (POSITION YCOORD) of
                                                                                                NewRegionPosition
                                                                                                )
                                                                      (fetch (REGION BOTTOM) of Region)))
                                                    Graph DisplayStream)
                       (EXTENDEXTENT (WFROMDS DisplayStream)
                              (NODEREGION SelectedNode))        (* extent the graph extent because the node may be outside the
                       old extent.)
                       (FLIPNODE SelectedNode DisplayStream])

(EDITMOVESUBTREE
  [LAMBDA (WINDOW)                                          (* Newman "27-Jan-87 11:10")

          (* * Code derived from EDITMOVENODE by Richard Burton. Changes to prompt strings, and changes the to
          TRACKCURSOR to a call to NOT.TRACKCURSOR)
                                                            (* hilite nodes until the cursor goes down then move it)
    (PROG ((DS (WINDOWPROP WINDOW 'DSP))
           (REG (WINDOWPROP WINDOW 'REGION))
           (GRAPH (WINDOWPROP WINDOW 'GRAPH))
           OLDPOS NOW NEAR NODELST)
          (COND
             (GRAPH (SETQ NODELST (fetch (GRAPH GRAPHNODES) of GRAPH)))
             (T (RETURN)))
          (printout PROMPTWINDOW T "Move the cursor to the node " "that is the common root of " "the subtree you
                want to move " "and press any button.")
          [SETQ NEAR (NODELST/AS/MENU NODELST (SETQ OLDPOS (CURSORPOSITION NIL DS]
      FLIP
          (AND NOW (FLIPNODE NOW DS))
          (AND NEAR (FLIPNODE NEAR DS))
          (SETQ NOW NEAR)
      LP  (GETMOUSESTATE)
          (COND
             ((LASTMOUSESTATE (NOT UP))                     (* button up, process it.)
```

```
            (AND NOW (FLIPNODE NOW DS))                    (* NOW node has been selected.)
           )
          ([EQ NOW (SETQ NEAR (NODELST/AS/MENU NODELST (CURSORPOSITION NIL DS OLDPOS]
           (GO LP))
          (T (GO FLIP)))
        (printout PROMPTWINDOW T "Holding the button down, " "move the node to its new position" "and release
              the button.")
        (NOT.TRACKCURSOR NOW DS GRAPH)
        (printout PROMPTWINDOW T "Done."])
```

(NOT.TRACKCURSOR

```
[LAMBDA (Node DisplayStream Graph)                         ; Edited 3-Aug-88 14:50 by pmi
```

;; Gets an old, and a new region from the user, and uses these to calculate all the new positions for all the children of Node.

;; rht 4/28/87: Changed from APPLY of UNIONREGIONS to for loop doing successive UNIONREGIONS calls.

;; pmi 8/3/88: Changed to call COLLECTDESCENDENTS instead of RECURSIVE.COLLECTDESCENDENTS.

```
(if (NULL Node)
    then (PROMPTPRINT "No node selected.")
  else (PROG* ((Children (COLLECTDESCENDENTS Node Graph))
               (OldRegion (for EachNode in (CONS Node Children) bind (TotalRegion _ (NODEREGION Node))
                             do (FLIPNODE EachNode DisplayStream)
                                (SETQ TotalRegion (UNIONREGIONS TotalRegion (NODEREGION EachNode)))
                             finally (RETURN TotalRegion)))
               (NewRegionPosition (GETBOXPOSITION.FROMINITIALREGION (WFROMDS DisplayStream)
                                                                    OldRegion DisplayStream))
               (deltaX (DIFFERENCE (fetch (POSITION XCOORD) of NewRegionPosition)
                                   (fetch (REGION LEFT) of OldRegion)))
               (deltaY (DIFFERENCE (fetch (POSITION YCOORD) of NewRegionPosition)
                                   (fetch (REGION BOTTOM) of OldRegion)))
               (OldPos (fetch (GRAPHNODE NODEPOSITION) of Node))
               (NewPos (CREATE.NEW.NODEPOSITION Node deltaX deltaY)))
              [if (NOT (EQUAL OldPos NewPos))
                  then (MOVENODE Node OldPos NewPos Graph DisplayStream)
                       (EXTENDEXTENT (WFROMDS DisplayStream)
                                     (NODEREGION Node))
                       (CALL.MOVENODEFN Node OldPos Graph (WFROMDS DisplayStream)
                                        NewPos)
                       (if Children
                           then (PROG [(MovedNodes (LIST (fetch (GRAPHNODE NODEID) of Node]
                                                         (MOVEDESCENDENTS Graph Node DisplayStream deltaX deltaY]
              (for EachNode in (CONS Node Children) do (FLIPNODE EachNode DisplayStream])
```

(RECURSIVE.COLLECTDESCENDENTS

```
[LAMBDA (Node Graph)                                       ; Edited 5-Aug-88 16:06 by pmi
```

;; Collect all descendents of Node in Graph.

;; pmi 8/2/88: Changed to break infinite recursion on circular graphs. Now marks nodes as visited.

;; pmi 8/5/88: FIxes bug introduced by previous fix.

```
(LET (NodeId)
```

;; Node's NODEID may be a list if it is a virtual node.

```
     (if (LISTP (SETQ NodeId (fetch (GRAPHNODE NODEID) of Node)))
         then (SETQ NodeId (CAR NodeId)))
     (NC.GraphNodeIDPutProp NodeId 'Visited T)
     (for ChildNode in (COLLECT.CHILD.NODES Node Graph) bind ChildNodeID
        when [PROGN (SETQ ChildNodeID (fetch (GRAPHNODE NODEID) of ChildNode))
```

;; This node has not been visited, and it is not a virtual node.

```
                    (NOT (NC.GraphNodeIDGetProp (if (LISTP ChildNodeID)
                                                    then (CAR ChildNodeID)
                                                  else ChildNodeID)
                                                'Visited]
        join (CONS ChildNode (RECURSIVE.COLLECTDESCENDENTS ChildNode Graph])
```

(MOVEDESCENDENTS

```
[LAMBDA (Graph Node DisplayStream deltaX deltaY)           ; Edited 29-Apr-94 14:00 by sybalsky
```

;; Moves Node and all Children of Node by deltaX and deltaY.

;; first, finds all descendents of Node. For each of these, create a new position based on the old and the deltas. Then, if the child has not been
;; moved yet, we add it to the list of moved nodes, move the node, and call the MOVENODEFN,

;; pmi 8/3/88: Changed to call COLLECTDESCENDENTS instead of RECURSIVE.COLLECTDESCENDENTS.

```
(bind (MovedNodes _ (LIST Node))
      NewPos for Child in (COLLECTDESCENDENTS Node Graph) eachtime (SETQ NewPos (CREATE.NEW.NODEPOSITION
                                                                                 Child deltaX deltaY))
   unless (MEMBER (fetch (GRAPHNODE NODEID) of Child)
                  MovedNodes)
   do (SETQ MovedNodes (CONS (fetch (GRAPHNODE NODEID) of Child)
                             MovedNodes))
      (MOVENODE Child (fetch (GRAPHNODE NODEPOSITION) of Child)
            NewPos Graph DisplayStream)
      (EXTENDEXTENT (WFROMDS DisplayStream)
```

```
            (NODEREGION Child))
      ;; we must call EXTENDEXTENT to extend the graph extent in case we have moved a node outside the previous extent.
      (CALL.MOVENODEFN Child NewPos Graph (WFROMDS DisplayStream)
            (fetch (GRAPHNODE NODEPOSITION) of Child])
```

(COLLECT.CHILD.NODES

```
  [LAMBDA (Node Graph)                                                    (* Newman "27-Jan-87 11:16")

          (* * collect all immediate children (only one generation) of Node in Graph.)

    (bind (GraphNodes _ (fetch (GRAPH GRAPHNODES) of Graph)) for NodeID in (fetch (GRAPHNODE TONODES) of Node)
       collect                                                            (* ??? (ASSOC (if (AND (LISTP NodeID)
                                                                          (EQUAL (CAR NodeID) (QUOTE Link% Parameters))) then
                                                                          (* Special case where the second item in the list is the NodeID)
                                                                          (CADR NodeID) else NodeID) GraphNodes))
          (GETNODEFROMID NodeID GraphNodes])
```

(CREATE.NEW.NODEPOSITION

```
  [LAMBDA (Node deltaX deltaY)                                            (* Newman "27-Jan-87 11:06")

          (* * Creates a new position for Node by adding deltaX and deltaY to the appropriate coordinates.)

    (PROG ((OldPos (fetch (GRAPHNODE NODEPOSITION) of Node)))
          (RETURN (create POSITION
                          XCOORD _ (PLUS deltaX (fetch (POSITION XCOORD) of OldPos))
                          YCOORD _ (PLUS deltaY (fetch (POSITION YCOORD) of OldPos])
```

(GETBOXPOSITION.FROMINITIALREGION

```
  [LAMBDA (Window Region DisplayStream)                                   (* Newman "26-Jan-87 11:38")

          (* * This function obtains a new region from the user, and it prompts the user using the region passed in as Region.
          DisplayStream is the displaystream of Window, and Region is considered to be a region within Window.
          This function was written to be called from EDITMOVEREGION.)

          (* All of the garbage below to calculate the third and fourth arguments to GETBOXPOSITION exists to put the ghost box
          prompting the user in exactly the same place as the region passed in.)

    (GETBOXPOSITION (fetch (REGION WIDTH) of Region)
           (fetch (REGION HEIGHT) of Region)
           (DIFFERENCE (PLUS (fetch (REGION LEFT) of Region)
                             (fetch (REGION LEFT) of (WINDOWPROP Window 'REGION))
                             (WINDOWPROP Window 'BORDER))
                  (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL DisplayStream)))
           (DIFFERENCE (PLUS (fetch (REGION BOTTOM) of Region)
                             (fetch (REGION BOTTOM) of (WINDOWPROP Window 'REGION))
                             (WINDOWPROP Window 'BORDER))
                  (fetch (REGION BOTTOM) of (DSPCLIPPINGREGION NIL DisplayStream)))
           Window "Select new region for nodes."])
```

(COLLECTDESCENDENTS

```
  [LAMBDA (Node Graph)                                                    ; Edited  5-Aug-88 15:40 by pmi
    ;; pmi 8/3/88: Created to wrap RESETLST around call to RECURSIVE.COLLECTDESCENDENTS.  Prevents infinite looping on circular graph
    ;; structures by marking where we have been.

    ;; Clean up the Visited markers placed on the nodes traversed.

    ;; pmi 8/5/88: Now also cleans up Visited marker on Node.

    (LET (NodeID Descendents)
         (RESETLST
             [RESETSAVE NIL '(PROGN (for VisitedNode in (CONS Node Descendents) bind VisitedNodeID
                                       do (NC.GraphNodeIDPutProp (if (LISTP (SETQ VisitedNodeID
                                                                                  (fetch (GRAPHNODE NODEID)
                                                                                     of VisitedNode)))
                                                                     then (CAR VisitedNodeID)
                                                                   else VisitedNodeID)
                                                                 'Visited NIL]
             (SETQ Descendents (RECURSIVE.COLLECTDESCENDENTS Node Graph)))])

)

;; functions for finding larger and smaller fonts

(DEFINEQ

(NEXTSIZEFONT
  [LAMBDA (WHICHDIR NOWFONT)                                              (* rmk%: "15-Sep-84 00:14")

          (* returns the next sized font either SMALLER or LARGER that on of size FONT.
          (NEXTSIZEFONT (QUOTE LARGER) DEFAULTFONT))

    (PROG [(NOWSIZE (FONTPROP NOWFONT 'HEIGHT]
          (RETURN (COND
```

```
                    [(EQ WHICHDIR 'LARGER)
                     (COND
                        ((IGEQ NOWSIZE (FONTPROP (CAR DECREASING.FONT.LIST)
                                                 'HEIGHT))          (* nothing larger)
                         NIL)
                        (T (for FONTTAIL on DECREASING.FONT.LIST when [AND (CDR FONTTAIL)
                                                                           (IGEQ NOWSIZE (FONTPROP (CADR
                                                                                                    FONTTAIL
                                                                                                    )
                                                                                                 'HEIGHT]
                              do (RETURN (FONTNAMELIST (CAR FONTTAIL]
                    (T (for FONT in DECREASING.FONT.LIST when (LESSP (FONTPROP FONT 'HEIGHT)
                                                                     NOWSIZE)
                          do (RETURN (FONTNAMELIST FONT])
```

(DECREASING.FONT.LIST

```
  [LAMBDA NIL                                                       (* rrb "16-Dec-83 12:28")
```

(* returns a list of the font descriptors for the fonts sketch windows are willing to print in.)

```
    (for SIZE in '(18 14 12 10 8 5) collect (FONTCREATE 'HELVETICA SIZE])
```

(SCALE.FONT

```
  [LAMBDA (WID STR)                                                 (* rrb " 7-NOV-83 11:35")
```

(* returns the font that text should be printed in to have the text STR fit into a region WID points wide)

```
    (COND
       ((GREATERP WID (TIMES (STRINGWIDTH STR (CAR DECREASING.FONT.LIST))
                             1.5))                                  (* scale it too large for even the largest font.)
        NIL)
       (T (for FONT in DECREASING.FONT.LIST when (NOT (GREATERP (STRINGWIDTH STR FONT)
                                                                WID))
             do (RETURN FONT) finally (RETURN 'SHADE])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQ DECREASING.FONT.LIST (DECREASING.FONT.LIST))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DECREASING.FONT.LIST)
)

;; functions for LAYOUTGRAPH And LAYOUTLATTICE

(DEFINEQ
```

(BRH/LAYOUT

```
  [LAMBDA (N X Y MOMLST GN)                                         (* kvl "26-DEC-83 16:44")
```

(* X and Y are the lower left corner of the box that will surround the tree headed by the browsenode N.
MOMLST is the mother node inside a cons cell. GN is the graphnode for the nodeid N.
It is crucial that the NODEPOSITION be set before recursion because this marks that the node has been
(is being) laid out already. BRH/OFFSET is used to raise the daughters in those rare cases where the label is bigger than
the daughters.)

```
    (DECLARE (USEDFREE MOTHERD PERSONALD NODELST))
    (PROG ((DS (fetch (GRAPHNODE TONODES) of GN))
           (W (fetch (GRAPHNODE NODEWIDTH) of GN))
           (YHEIGHT (IPLUS PERSONALD (fetch (GRAPHNODE NODEHEIGHT) of GN)))
           DHEIGHT)
          (replace (GRAPHNODE FROMNODES) of GN with MOMLST)
          [replace (GRAPHNODE NODEPOSITION) of GN with (create POSITION
                                                               XCOORD _ (IPLUS X (HALF W]
          (COND
             ((NULL DS))
             [[IGREATERP YHEIGHT (SETQ DHEIGHT (BRH/LAYOUT/DAUGHTERS DS (IPLUS X W MOTHERD)
                                                                      Y
                                                                      (LIST N]
              (BRH/OFFSET DS (HALF (IDIFFERENCE YHEIGHT DHEIGHT]
             (T (SETQ YHEIGHT DHEIGHT)))
          (replace YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GN) with (IPLUS Y (HALF YHEIGHT)))
          (RETURN YHEIGHT])
```

(BRH/LAYOUT/DAUGHTERS

```
  [LAMBDA (DS X Y MOMLST)                                           (* rmk%: " 5-Feb-84 15:01")
```

(* DS are the daughters of (CAR MOMLST)%. X is where the left edge of their labels will be, and Y is the bottom of the mother's box. Returns the height of the mother's box. Tests to see if a node has been layout out already If so, it replaces the daughter with one that has no descendents, and splices into the mother's daughter list, side-effecting the graphnode

structure.)

```
(DECLARE (USEDFREE NODELST))
(for D (FLOOR _ Y) in DS do [SETQ FLOOR (IPLUS FLOOR (BRH/LAYOUT D X FLOOR MOMLST (GETNODEFROMID D NODELST
                                                                                                    ]
   finally (RETURN (IDIFFERENCE FLOOR Y])
```

(BRH/OFFSET

```
[LAMBDA (NODEIDS YINC)
  (DECLARE (USEDFREE NODELST))                                    (* kvl "11-Dec-84 14:35")
  (for N in NODEIDS do (SETQ N (GETNODEFROMID N NODELST))
                       (add (fetch YCOORD of (fetch (GRAPHNODE NODEPOSITION) of N))
                            YINC)
                       (BRH/OFFSET (fetch (GRAPHNODE TONODES) of N)
                              YINC])
```

(BRHC/INTERTREE/SPACE

```
[LAMBDA (TTC BTC)                                                 (* kvl "21-DEC-83 10:23")
```

(* Given the top transition chain of the old daughter and the bottom transition chain of the new daughter, where BTC is sitting on the bottom of the box, calculate how much the bottom must be raised so that it just clears the TTC. OP is the top left corner of some label. NP is the bottom left corner.)

```
  (PROG ((RAISE -1000)
         NP DIST OP)
        (SETQ OP (pop TTC))
        (SETQ NP (pop BTC))
    L   (SETQ DIST (IDIFFERENCE (fetch YCOORD of OP)
                          (fetch YCOORD of NP)))
        (AND (IGREATERP DIST RAISE)
             (SETQ RAISE DIST))
        [COND
           ((NULL BTC)
            (RETURN RAISE))
           ((NULL TTC)
            (RETURN RAISE))
           ((IEQP (fetch XCOORD of (CAR BTC))
                  (fetch XCOORD of (CAR TTC)))
            (SETQ NP (pop BTC))
            (SETQ OP (pop TTC)))
           ((ILESSP (fetch XCOORD of (CAR BTC))
                    (fetch XCOORD of (CAR TTC)))
            (SETQ NP (pop BTC)))
           (T (SETQ OP (pop TTC]
        (GO L])
```

(BRHC/LAYOUT

```
[LAMBDA (N X MOMLST GN)                                           (* rmk%: " 5-Feb-84 14:47")
```

(* See comment on BRH/LAYOUT. Instead of keeping only the graphnode in layed out node's position field, keep the offset as well. The offset is how much this nodes box must be raised relative to the inclosing box.
Uses two free variables to return transition chains. RETURNTTC is the top left corners of all the labels.
RETURNBTC is the bottom left corners.)

```
  (DECLARE (USEDFREE PERSONALD RETURNTTC RETURNBTC))
  (PROG ((DS (fetch (GRAPHNODE TONODES) of GN))
         (W (fetch (GRAPHNODE NODEWIDTH) of GN))
         (H (fetch (GRAPHNODE NODEHEIGHT) of GN))
         YCENTER X/SW H/2)
        (SETQ H/2 (HALF H))
        (SETQ X/SW (IPLUS X W))
        (replace (GRAPHNODE FROMNODES) of GN with MOMLST)
        (replace (GRAPHNODE NODEPOSITION) of GN with (LIST 0))
        [SETQ YCENTER (COND
                         (DS (BRHC/LAYOUT/DAUGHTERS DS X/SW (LIST N)))
                         (T (BRHC/LAYOUT/TERMINAL GN X/SW]
        (RPLACD (fetch (GRAPHNODE NODEPOSITION) of GN)
               (create POSITION
                       XCOORD _ (IPLUS X (HALF W))
                       YCOORD _ YCENTER))
        [push RETURNTTC (create POSITION
                                XCOORD _ X
                                YCOORD _ (IPLUS PERSONALD (IPLUS (IDIFFERENCE YCENTER H/2)
                                                                 H]
        (push RETURNBTC (create POSITION
                                XCOORD _ X
                                YCOORD _ (IDIFFERENCE YCENTER H/2)))
        (RETURN YCENTER])
```

(BRHC/LAYOUT/DAUGHTERS

```
[LAMBDA (DS X/SW MOMLST)
  (DECLARE (USEDFREE MOTHERD FAMILYD NODELST RETURNTTC RETURNBTC))
                                                                  (* rmk%: " 5-Feb-84 14:52")
```

(* see comment on BRH/LAYOUT/DAUGHTERS. First daughter is always laid out on the bottom of the box.
Subsequent daughters have the amount that they are to be raised calculated by comparing the top edge of the old daughter (in TTC) with the bottom edge of the new daughter (in RETURNBTC)%.
TTC is update by adding the new daughter's transition chain to the front, because the new daughter's front is guaranteed to be higher than the old daughter's front. Conversely, BTC is updated by adding the new daughter's transition chain to the back, because the old daughter's front is guaranteed to be lower.)

```
(for D in DS bind GN BTC TTC 1ST/DCENTER LST/DCENTER (OFFSET _ 0)
                  (X _ (IPLUS X/SW MOTHERD))
   do (SETQ GN (GETNODEFROMID D NODELST))
      (SETQ LST/DCENTER (BRHC/LAYOUT D X MOMLST GN))
      [COND
         ((NULL TTC)                                          (* first daughter)
          (SETQ 1ST/DCENTER LST/DCENTER)
          (SETQ TTC RETURNTTC)
          (SETQ BTC RETURNBTC))
         (T (SETQ OFFSET (BRHC/INTERTREE/SPACE TTC RETURNBTC))
            (RPLACA (fetch (GRAPHNODE NODEPOSITION) of GN)
                   OFFSET)
            (SETQ TTC (EXTEND/TRANSITION/CHAIN (RAISE/TRANSITION/CHAIN RETURNTTC OFFSET)
                              TTC))
            (SETQ BTC (EXTEND/TRANSITION/CHAIN BTC (RAISE/TRANSITION/CHAIN RETURNBTC OFFSET]
   finally
```

(* add a mythical top left corner at the height of the highest daughter because diagnonal links are getting clobbered. Move lowest daughter's bottom left corner to the left for the same reason.)

```
   (SETQ RETURNTTC (CONS (create POSITION
                                 XCOORD _ X/SW
                                 YCOORD _ (fetch YCOORD of (CAR TTC)))
                         TTC))
   (replace XCOORD of (CAR BTC) with X/SW)
   (add (fetch YCOORD of (CAR TTC))
        FAMILYD)
   (SETQ RETURNBTC BTC)
```

(* center of mother is halfway between first and last daughter's label centers using fact that offset of first daughter is zero and last daughter's offset is OFFSET)

```
   (RETURN (HALF (IPLUS 1ST/DCENTER OFFSET LST/DCENTER])
```

(BRHC/LAYOUT/TERMINAL

```
[LAMBDA (GN X/SW)                                             (* rmk%: " 3-Feb-84 09:55")
```

(* initiallizes the transition chains to the right edge of the node label, and returns the label's center.)

```
(DECLARE (USEDFREE RETURNTTC RETURN/TBC))
(SETQ RETURNTTC (LIST (create POSITION
                              XCOORD _ X/SW
                              YCOORD _ 0)))
[SETQ RETURNBTC (LIST (create POSITION
                              XCOORD _ X/SW
                              YCOORD _ (fetch (GRAPHNODE NODEHEIGHT) of GN]
(HALF (fetch (GRAPHNODE NODEHEIGHT) of GN])
```

(BRHC/OFFSET

```
[LAMBDA (N ABSY)                                              (* dgb%: "22-Jan-85 07:17")
                                                              (* Adds in all the offsets. See comment on
                                                              BRHC/LAYOUT/DAUGHTERS.)
(DECLARE (USEDFREE NODELST))
(PROG ((GN (GETNODEFROMID N NODELST)))
      [SETQ ABSY (IPLUS ABSY (pop (fetch (GRAPHNODE NODEPOSITION) of GN]
      [replace YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GN) with (IPLUS ABSY (fetch YCOORD
                                                                                   of (fetch (GRAPHNODE
                                                                                              NODEPOSITION
                                                                                              )
                                                                                      of GN]
      (for D in (fetch (GRAPHNODE TONODES) of GN) do (BRHC/OFFSET D ABSY])
```

(BRHL/LAYOUT

```
[LAMBDA (N X Y MOMLST GN)                                     (* kvl "26-DEC-83 16:36")
```

(* X and Y are the lower left corner of the box that will surround the tree headed by the browsenode N.
MOMLST is the mother node inside a cons cell. GN is the graphnode for the nodeid N.
It is crucial that the NODEPOSITION be set before recursion because this marks that the node has been laid out already.
If in addition, the YCOORD is NIL, then the node is still in the process of being laid out.
BRHL/LAYOUT/DAUGHTERS uses this fact to break loops by inserting boxed nodes.)

```
(DECLARE (USEDFREE MOTHERD PERSONALD NODELST))
(COND
   ((fetch (GRAPHNODE NODEPOSITION) of GN)
```

(* This case only occurs if this node has been put in the roots list, and has already been visited by recursion.
Value won't be used)

```
        0)
       (T (PROG [(DS (fetch (GRAPHNODE TONODES) of GN))
                 (W (fetch (GRAPHNODE NODEWIDTH) of GN))
                 (YHEIGHT (IPLUS PERSONALD (fetch (GRAPHNODE NODEHEIGHT) of GN]
                (replace (GRAPHNODE FROMNODES) of GN with MOMLST)  (* This is first time for layout, so set FROMNODES)
                [replace (GRAPHNODE NODEPOSITION) of GN with (create POSITION
                                                                     XCOORD _ (IPLUS X (HALF W]
                (AND DS (SETQ YHEIGHT (IMAX (BRHL/LAYOUT/DAUGHTERS DS (IPLUS X W MOTHERD)
                                                                    Y
                                                                    (LIST N))
                                                    YHEIGHT)))
                (replace YCOORD of (fetch (GRAPHNODE NODEPOSITION) of GN) with (IPLUS Y (HALF YHEIGHT)))
                (RETURN YHEIGHT])
```

(BRHL/LAYOUT/DAUGHTERS

```
[LAMBDA (DS X Y MOMLST)                                          ; Edited 29-Apr-94 14:00 by sybalsky
```

(* DS are the daughters of (CAR MOMLST)%. X is where their the left edge of their labels will be, and Y is the bottom of the mother's box. Returns the height of the mother's box. Tests to see if a node has been laid out out already If so, it sees if the node is far enought to the right; if not it moves the node and its daughters.)

```
    (DECLARE (USEDFREE NODELST YHEIGHT))
    (for DTAIL on DS bind D GN NP DELTA (FLOOR _ Y) finally (RETURN (IDIFFERENCE FLOOR Y))
       do (SETQ GN (GETNODEFROMID (SETQ D (CAR DTAIL))
                                  NODELST))
          (COND
             ((SETQ NP (fetch (GRAPHNODE NODEPOSITION) of GN))
              [COND
                 [(NULL (fetch YCOORD of NP))
                  (SETQ GN (NEW/INSTANCE/OF/GRAPHNODE GN))
                  (RPLACA DTAIL (fetch (GRAPHNODE NODEID) of GN))
                  (SETQ FLOOR (IPLUS FLOOR (BRHL/LAYOUT (fetch (GRAPHNODE NODEID) of GN)
                                                        X FLOOR MOMLST GN]
                 (T (BRHL/MOVE/RIGHT GN X NIL)
                    (push (fetch (GRAPHNODE FROMNODES) of GN)
                          (CAR MOMLST]                           (* Add this mother to the fromLinks)
              )
             (T (SETQ FLOOR (IPLUS FLOOR (BRHL/LAYOUT D X FLOOR MOMLST GN])
```

(BRHL/MOVE/RIGHT

```
[LAMBDA (GN X STACK)                                             ; Edited 29-Apr-94 14:00 by sybalsky
                                                                 (* Move this node and its children right)
    (DECLARE (USEDFREE NODELST))
    (PROG ((W (fetch (GRAPHNODE NODEWIDTH) of GN))
           (NP (fetch (GRAPHNODE NODEPOSITION) of GN)))
          (AND (FMEMB GN STACK)
               (ERROR "Loop caught in BRHL/MOVE/RIGHT at" (fetch (GRAPHNODE NODELABEL) of GN)))
          (COND
             ((ILESSP X (IDIFFERENCE (fetch XCOORD of NP)
                                     (HALF W)))
              (RETURN)))
          (for D in (TOLINKS GN) bind (NEWX _ (IPLUS X W MOTHERD))
                                      (NSTACK _ (CONS GN STACK))
             do (BRHL/MOVE/RIGHT (GETNODEFROMID D NODELST)
                                 NEWX NSTACK))
          (replace XCOORD of NP with (IPLUS X (HALF W])
```

(BROWSE/LAYOUT/HORIZ

```
[LAMBDA (ROOTIDS)                                                ; Edited 19-Aug-88 08:32 by sye
```

(* each subtree is given a box centered vertically on its label. Sister boxes abut but do not intrude as they do in the compacting version.)

```
    (DECLARE (USEDFREE NODELST))
    [for N in ROOTIDS bind (Y _ 0) do (SETQ Y (IPLUS Y (BRH/LAYOUT N 0 Y NIL (GETNODEFROMID N NODELST]
    (create GRAPH
            GRAPHNODES _ NODELST
            SIDESFLG _ T
            DIRECTEDFLG _ NIL])
```

(BROWSE/LAYOUT/HORIZ/COMPACTLY

```
[LAMBDA (ROOTS)
    (DECLARE (USEDFREE NODELST MOTHERD))                         ; Edited 19-Aug-88 08:33 by sye
```

(* See comments on BRH/LAYOUT and BRH/LAYOUT/DAUGHTERS first.
This differs in that it keeps (on the stack) a representation of the shape of the tree that fills the node's box.
The representation is a list of POSITIONs. If one starts drawing a line from left to right starting at the CAR, each point is a step in the line, and the point begins the new plateau (or valley)%.
The last point is where the line would turn around and head back to the left.)
                                                                 (* builds dummy top node for ROOTS if necessary, and adjusts the horizontal distance accordingly.)

```
    [PROG (RETURNTTC RETURNBTC)
```

```
        (DECLARE (SPECVARS RETURNTTC RETURNBTC))
        (COND
           ((NLISTP ROOTS)
            (BRHC/LAYOUT ROOTS 0 NIL (GETNODEFROMID ROOTS NODELST))
            (BRHC/OFFSET ROOTS 0))
           ((NULL (CDR ROOTS))
            (BRHC/LAYOUT (CAR ROOTS)
                   0 NIL (GETNODEFROMID (CAR ROOTS)
                                NODELST))
            (BRHC/OFFSET (CAR ROOTS)
                   0))
           (T (PROG ((GN (create GRAPHNODE
                                 NODELABEL _ (PACK)
                                 NODEID _ (CONS)
                                 TONODES _ ROOTS
                                 NODEWIDTH _ 0
                                 NODEHEIGHT _ 0))
                     TOPNODE)
                    (push NODELST GN)
                    (SETQ TOPNODE (fetch (GRAPHNODE NODEID) of GN))
                    (BRHC/LAYOUT TOPNODE (IMINUS MOTHERD)
                          NIL GN)
                    (BRHC/OFFSET TOPNODE 0)
                    [for N GN in ROOTS do (replace (GRAPHNODE FROMNODES) of (SETQ GN (FASSOC N NODELST))
                                                  with (DREMOVE TOPNODE (fetch (GRAPHNODE FROMNODES) of GN]
                    (SETQ NODELST (DREMOVE GN NODELST]
    (create GRAPH
            GRAPHNODES _ NODELST
            SIDESFLG _ T
            DIRECTEDFLG _ NIL])
```

(BROWSE/LAYOUT/LATTICE

```
  [LAMBDA (NS)                                                    ; Edited 19-Aug-88 08:33 by sye
```

(* almost the same as BROWSE/LAYOUT/HORIZ, except that it doesn't box nodes unless there are cycles. Instead, a single node is placed at the rightmost of the positions that would be laid out by for all of its (boxed) occurrences by BROWSE/LAYOUT/HORIZ.)

```
    (DECLARE (USEDFREE NODELST))
    [for N in NS bind (Y _ 0) do (SETQ Y (IPLUS Y (BRHL/LAYOUT N 0 Y NIL (GETNODEFROMID N NODELST]
    (create GRAPH
            GRAPHNODES _ NODELST
            SIDESFLG _ T
            DIRECTEDFLG _ NIL])
```

(BRV/OFFSET

```
  [LAMBDA (N ABSX)                                                (* dgb%: "22-Jan-85 07:25")
```

(* Adds in offset which are kept in car of NODEPOSITION. TERMY is Y of lowest node. Adding it in raises tree so lowest node is at zero.)

```
    (DECLARE (USEDFREE NODELST TERMY))
    (PROG (P (GN (GETNODEFROMID N NODELST)))
          [SETQ ABSX (IPLUS ABSX (pop (fetch (GRAPHNODE NODEPOSITION) of GN]
          (replace XCOORD of (SETQ P (fetch (GRAPHNODE NODEPOSITION) of GN)) with (IPLUS ABSX (fetch XCOORD
                                                                                                  of P)))
          (replace YCOORD of P with (IDIFFERENCE (fetch YCOORD of P)
                                                 TERMY))
          (for D in (fetch (GRAPHNODE TONODES) of GN) do (BRV/OFFSET D ABSX])
```

(EXTEND/TRANSITION/CHAIN

```
  [LAMBDA (LTC RTC)                                               (* kvl "21-DEC-83 11:00")
```

(* Extends the left transition chain by appending the part of the right transition chain that is to the right of the end of the left transition chain. End point of left transition chain is changed to intersect right transition chain)

```
    (PROG ((LTAIL LTC)
           (RTAIL RTC)
           LX RX)
      L    [COND
              ((NULL (CDR RTAIL))
               (replace YCOORD of (CAR (FLAST LTAIL)) with (fetch YCOORD of (CAR RTAIL)))
               (RETURN LTC))
              ((NULL (CDR LTAIL))
               (RPLACD LTAIL (CDR RTAIL))
               (replace YCOORD of (CAR LTAIL) with (fetch YCOORD of (CAR RTAIL)))
               (RETURN LTC))
              ([IEQP (SETQ LX (fetch XCOORD of (CADR LTAIL)))
                     (SETQ RX (fetch XCOORD of (CADR RTAIL]
               (SETQ LTAIL (CDR LTAIL))
               (SETQ RTAIL (CDR RTAIL)))
              ((ILESSP LX RX)
               (SETQ LTAIL (CDR LTAIL)))
              (T (SETQ RTAIL (CDR RTAIL]
```

```
          (GO L])
```

(FOREST/BREAK/CYCLES

```
  [LAMBDA (NODE)                                                    (* kvl "14-Aug-84 09:19")
                                                                    (* Breaks any cycles by inserting new nodes and boxing)
    (DECLARE (USEDFREE NODELST))
    (replace (GRAPHNODE NODEPOSITION) of NODE with T)
    (for DTAIL DN on (fetch (GRAPHNODE TONODES) of NODE) do (SETQ DN (GETNODEFROMID (CAR DTAIL)
                                                                                   NODELST))
                                                            (COND
                                                               ((fetch (GRAPHNODE NODEPOSITION) of DN)
                                                                   (* We've seen this before)
                                                                (SETQ DN (NEW/INSTANCE/OF/GRAPHNODE DN))
                                                                (RPLACA DTAIL (fetch (GRAPHNODE NODEID)
                                                                                     of DN)))
                                                               (T (FOREST/BREAK/CYCLES DN])
```

(INIT/NODES/FOR/LAYOUT

```
  [LAMBDA (NS FORMAT ROOTIDS FONT)                                  (* Randy.Gobbel " 8-May-87 16:22")
    (for GN in NS do [replace (GRAPHNODE NODEPOSITION) of GN with (NOT (NOT (FMEMB (fetch (GRAPHNODE NODEID)
                                                                                     of GN)
                                                                                   ROOTIDS]
                                                                    (* T Used to indicate prior visitation.
                                                                    Roots are already visited)
                     (OR (IMAGEOBJP (fetch (GRAPHNODE NODELABEL) of GN))
                         (fetch (GRAPHNODE NODEFONT) of GN)
                         (replace (GRAPHNODE NODEFONT) of GN with FONT)))
    [for R in ROOTIDS do (COND
                            ((EQMEMB 'LATTICE FORMAT)
                             (LATTICE/BREAK/CYCLES (GETNODEFROMID R NODELST)
                                   NIL))
                            (T (FOREST/BREAK/CYCLES (GETNODEFROMID R NODELST]
    (for GN in NODELST do (replace (GRAPHNODE NODEPOSITION) of GN with NIL)
                          (SET/LABEL/SIZE GN])
```

(INTERPRET/MARK/FORMAT

```
  [LAMBDA (FORMAT)                                                  (* rmk%: "20-Sep-85 08:59")
                                                                    (* sets specvars for NEW/INSTANCE/OF/GRAPHNODE and
                                                                    MARK/GRAPH/NODE)
    (DECLARE (USEDFREE BOX.BOTH.FLG BOX.LEAVES.FLG BORDER.FOR.MARKING LABELSHADE.FOR.MARKING))
    (PROG (PL)
          (AND (EQMEMB 'COPIES/ONLY FORMAT)
               (SETQ BOX.BOTH.FLG NIL))
          (AND (EQMEMB 'NOT/LEAVES FORMAT)
               (SETQ BOX.LEAVES.FLG NIL))
          (COND
             ((NLISTP FORMAT)
              (RETURN))
             ((EQ (CAR FORMAT)
                  'MARK)
              (SETQ PL (CDR FORMAT)))
             ((SETQ PL (FASSOC 'MARK FORMAT))
              (SETQ PL (CDR PL)))
             (T (RETURN)))
          [COND
             [(FMEMB 'BORDER PL)
              (SETQ BORDER.FOR.MARKING (LISTGET PL 'BORDER]
             (T (SETQ BORDER.FOR.MARKING 'DON'T]
          (COND
             [(FMEMB 'LABELSHADE PL)
              (SETQ LABELSHADE.FOR.MARKING (LISTGET PL 'LABELSHADE]
             (T (SETQ LABELSHADE.FOR.MARKING 'DON'T])
```

(LATTICE/BREAK/CYCLES

```
  [LAMBDA (NODE STACK)                                              ; Edited 29-Apr-94 14:01 by sybalsky
    (replace (GRAPHNODE NODEPOSITION) of NODE with T)
    (for DTAIL on (fetch (GRAPHNODE TONODES) of NODE) bind D GN do (SETQ GN (GETNODEFROMID (SETQ D (CAR DTAIL))
                                                                                           NODELST))
                                                                   (COND
                                                                      ((FMEMB D STACK)
                                                                       (SETQ GN (NEW/INSTANCE/OF/GRAPHNODE GN))
                                                                       (RPLACA DTAIL (fetch (GRAPHNODE NODEID)
                                                                                            of GN)))
                                                                      ((NULL (fetch (GRAPHNODE NODEPOSITION)
                                                                                    of GN))
                                                                       (LATTICE/BREAK/CYCLES GN (CONS D STACK])
```

(LAYOUTFOREST

```
  [LAMBDA (NODELST ROOTIDS FORMAT BOXING FONT MOTHERD PERSONALD FAMILYD)
                                                                    ; Edited 16-Apr-90 19:05 by gadener
                                                                    (* This is an older version of LayoutGraph, kept around
                                                                    temporarily but de-documented)
```

```
(LAYOUTGRAPH NODELST ROOTIDS (CL:IF (LISTP FORMAT)
                                    (APPEND FORMAT BOXING)
                                    (CONS FORMAT BOXING))
       FONT MOTHERD PERSONALD])
```

(LAYOUTGRAPH

```
[LAMBDA (NODELST ROOTIDS FORMAT FONT MOTHERD PERSONALD FAMILYD)
                                                                    ; Edited 29-Apr-94 14:01 by sybalsky
    ;; takes a list of GRAPHNODE records and a list node ids for the top level nodes, where the graphnodes have only the NODEID, NODELABEL and
    ;; TONODES fields filled in.  It fills in the other fields appropriately according the format switch and the boxing switch so that the graph becomes a
    ;; forest.  If there are loops in the graph, they are snapped and the NODELST is extended with Push This function returns a GRAPH record with the
    ;; display slots filled in appropriately.
    (DECLARE (SPECVARS NODELST MOTHERD PERSONALD FAMILYD))
    (PROG ((BOX.BOTH.FLG T)
           (BOX.LEAVES.FLG T)
           (BORDER.FOR.MARKING T)
           (LABELSHADE.FOR.MARKING 'DON'T)
           G)
          (DECLARE (SPECVARS BOX.BOTH.FLG BOX.LEAVES.FLG BORDER.FOR.MARKING LABELSHADE.FOR.MARKING))
          (OR (LISTP ROOTIDS)
              (ERROR "LAYOUTGRAPH needs a LIST of root node ids"))
          (for R in ROOTIDS unless (FASSOC R NODELST) do (ERROR R "is in ROOTIDS but no GRAPHNODE for it in
                                                                   NODELST."))
          (OR FONT (SETQ FONT (OR DEFAULT.GRAPH.NODEFONT DEFAULTFONT)))
          (OR MOTHERD (SETQ MOTHERD (STRINGWIDTH "AAAAAA" FONT)))
          [OR PERSONALD (SETQ PERSONALD (COND
                                            ((EQMEMB 'VERTICAL FORMAT)
                                             (STRINGWIDTH "AA" FONT))
                                            (T 0]
          [OR FAMILYD (SETQ FAMILYD (HALF (FONTPROP FONT 'ASCENT]
          (INTERPRET/MARK/FORMAT FORMAT)
          (INIT/NODES/FOR/LAYOUT NODELST FORMAT ROOTIDS FONT)
          (AND (EQMEMB 'VERTICAL FORMAT)
               (SWITCH/NODE/HEIGHT/WIDTH NODELST))
          [SETQ G (COND
                     ((EQMEMB 'LATTICE FORMAT)
                      (BROWSE/LAYOUT/LATTICE ROOTIDS))
                     ((EQMEMB 'FAST FORMAT)
                      (BROWSE/LAYOUT/HORIZ ROOTIDS))
                     (T (BROWSE/LAYOUT/HORIZ/COMPACTLY ROOTIDS]
          (replace (GRAPH GRAPH.PROPS) of G with (LIST 'FORMAT FORMAT))
          [for N in NODELST do (OR (type? POSITION (fetch (GRAPHNODE NODEPOSITION) of N))
                                   (ERROR "Disconnected graph.  Root(s) didn't connect to:" (fetch (GRAPHNODE
                                                                                                    NODELABEL)
                                                                                              of N]
          [COND
             ((EQMEMB 'VERTICAL FORMAT)
              (SWITCH/NODE/HEIGHT/WIDTH NODELST)
              (REFLECT/GRAPH/DIAGONALLY G)
              (OR (EQMEMB 'REVERSE FORMAT)
                  (REFLECT/GRAPH/VERTICALLY G))
              (AND (EQMEMB 'REVERSE/DAUGHTERS FORMAT)
                   (REFLECT/GRAPH/HORIZONTALLY G)))
             (T (AND (EQMEMB 'REVERSE FORMAT)
                     (REFLECT/GRAPH/HORIZONTALLY G))
                (AND (EQMEMB 'REVERSE/DAUGHTERS FORMAT)
                     (REFLECT/GRAPH/VERTICALLY G]
          (RETURN G])
```

(LAYOUTLATTICE

```
[LAMBDA (NODELST ROOTIDS FORMAT FONT MOTHERD PERSONALD FAMILYD)
                                                                    (* rmk%: " 6-Dec-85 12:19")

    (* takes a list of GRAPHNODE records and a list node ids for the top level nodes, where the graphnodes have only the
    NODEID, NODELABEL and TONODES fields filled in. It fills in the other fields appropriately according the format switch If
    there are loops in the graph, they are detected in BRHL/MOVE/RIGHT and an error occurs.
    This function returns a GRAPH record with the display slots filled in appropriately.)

    (DECLARE (SPECVARS NODELST MOTHERD PERSONALD FAMILYD))
    (for R in ROOTIDS unless (FASSOC R NODELST) do (ERROR R "is in ROOTIDS but no GRAPHNODE for it in NODELST."))
    (SETQ FONT (OR FONT DEFAULTFONT))
    (INIT/NODES/FOR/LAYOUT NODELST FORMAT ROOTIDS FONT)
    [OR FAMILYD (SETQ FAMILYD (HALF (FONTPROP FONT 'ASCENT]
    (OR MOTHERD (SETQ MOTHERD (STRINGWIDTH "AAAAAA" FONT)))
    [OR PERSONALD (SETQ PERSONALD (COND
                                      ((EQ FORMAT 'VERTICAL)
                                       (STRINGWIDTH "AA" FONT))
                                      (T 0]
    (BROWSE/LAYOUT/LATTICE ROOTIDS])
```

(LAYOUTSEXPR

```
[LAMBDA (TREE FORMAT BOXING FONT MOTHERD PERSONALD FAMILYD)    ; Edited  1-Sep-92 17:26 by jds
    ;; assumes CAR of tree is node label, CDR is daughter trees.
```

```
    (COND
       [TREE (PROG (RESULT)
                   (DECLARE (SPECVARS RESULT))
                   (LAYOUTSEXPR1 TREE)
             ;; Boxing arg will only be taken into account if they are valid Format arguments
                                                                  ; otherwise, it is ignored
             (AND (OR (NLISTP BOXING)
                      (EQ (CAR BOXING)
                          'MARK))
                  (SETQ BOXING (CONS BOXING)))
             (RETURN (LAYOUTGRAPH RESULT (LIST TREE)
                          (APPEND (MKLIST FORMAT)
                                  BOXING)
                          FONT MOTHERD PERSONALD FAMILYD]
       (T (ERROR "Cannot layout NIL as S-EXPRESSION"])
```

(LAYOUTSEXPR1

```
  [LAMBDA (TREE)                                                  (* dgb%: "22-Jan-85 07:07")
    (DECLARE (SPECVARS RESULT))
    (COND
       [(for R in RESULT thereis (EQ TREE (fetch (GRAPHNODE NODEID) of R]
       ((NLISTP TREE)
        (push RESULT (create GRAPHNODE
                             NODEID _ TREE
                             NODELABEL _ TREE)))
       (T [push RESULT (create GRAPHNODE
                               NODEID _ TREE
                               NODELABEL _ (CAR TREE)
                               TONODES _ (APPEND (CDR TREE]
          (for D in (CDR TREE) do (LAYOUTSEXPR1 D])
```

(MARK/GRAPH/NODE

```
  [LAMBDA (NODE)                                                  ; Edited 29-Apr-94 14:01 by sybalsky
                                                                  (* changes appearance of graph node to indicate that a link has
    been snapped.)
    (DECLARE (USEDFREE BORDER.FOR.MARKING LABELSHADE.FOR.MARKING))
    (OR (EQ BORDER.FOR.MARKING 'DON'T)
        (replace (GRAPHNODE NODEBORDER) of NODE with BORDER.FOR.MARKING))
    (OR (EQ LABELSHADE.FOR.MARKING 'DON'T)
        (replace (GRAPHNODE NODELABELSHADE) of NODE with LABELSHADE.FOR.MARKING])
```

(NEW/INSTANCE/OF/GRAPHNODE

```
  [LAMBDA (GN)
    (DECLARE (USEDFREE NODELST BOX.LEAVES.FLG BOX.BOTH.FLG))      ; Edited 29-Apr-94 14:01 by sybalsky
                                                                  (* returns a second instance of the node, boxing it appropriately.
                                                                  No daughters.)
    (PROG [(NEW (create GRAPHNODE
                        NODEID _ (LIST (fetch (GRAPHNODE NODEID) of GN))
                        NODELABEL _ (fetch (GRAPHNODE NODELABEL) of GN)
                        NODEFONT _ (fetch (GRAPHNODE NODEFONT) of GN)
                        NODEWIDTH _ (fetch (GRAPHNODE NODEWIDTH) of GN)
                        NODEHEIGHT _ (fetch (GRAPHNODE NODEHEIGHT) of GN)
                        NODEBORDER _ (COPY (fetch (GRAPHNODE NODEBORDER) of GN))
                        NODELABELSHADE _ (fetch (GRAPHNODE NODELABELSHADE) of GN]
          (push NODELST NEW)
          [COND
             ((OR BOX.LEAVES.FLG (fetch (GRAPHNODE TONODES) of GN))
              (MARK/GRAPH/NODE NEW)
              (COND
                 (BOX.BOTH.FLG (MARK/GRAPH/NODE GN]
          (RETURN NEW])
```

(RAISE/TRANSITION/CHAIN

```
  [LAMBDA (TC RAISE)                                              (* kvl "21-DEC-83 10:25")

          (* raises a daughters transition chain by adding in the offset of the daughter's box relative to the mother's box.)

    (for P in TC do (add (fetch YCOORD of P)
                         RAISE)
       finally (RETURN TC])
```

(REFLECT/GRAPH/DIAGONALLY

```
  [LAMBDA (GRAPH)                                                 (* kvl "26-DEC-83 10:58")
    (replace (GRAPH SIDESFLG) of GRAPH with (NOT (fetch (GRAPH SIDESFLG) of GRAPH)))
    [for N in (fetch (GRAPH GRAPHNODES) of GRAPH) do (SETQ N (fetch (GRAPHNODE NODEPOSITION) of N))
                                                     (replace XCOORD of N with (PROG1 (fetch YCOORD of N)
                                                                                      (replace YCOORD of N
                                                                                         with (fetch XCOORD of N))))]
    GRAPH])
```

```
(REFLECT/GRAPH/HORIZONTALLY
  [LAMBDA (GRAPH)                                                    (* kvl "10-Aug-84 17:23")
    (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) bind [W _ (IPLUS (MAX/RIGHT (fetch (GRAPH GRAPHNODES) of GRAPH))
                                                                   (MIN/LEFT (fetch (GRAPH GRAPHNODES) of GRAPH]
       do (SETQ N (fetch (GRAPHNODE NODEPOSITION) of N))
          (replace XCOORD of N with (IDIFFERENCE W (fetch XCOORD of N])

(REFLECT/GRAPH/VERTICALLY
  [LAMBDA (GRAPH)                                                    (* kvl "10-Aug-84 16:48")
    (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) bind [H _ (IPLUS (MAX/TOP (fetch (GRAPH GRAPHNODES) of GRAPH))
                                                                   (MIN/BOTTOM (fetch (GRAPH GRAPHNODES)
                                                                                      of GRAPH]
       do (SETQ N (fetch (GRAPHNODE NODEPOSITION) of N))
          (replace YCOORD of N with (IDIFFERENCE H (fetch YCOORD of N])

(SWITCH/NODE/HEIGHT/WIDTH
  [LAMBDA (NL)                                                       (* rmk%: " 2-Feb-84 22:19")
    (for N in NL do (swap (fetch (GRAPHNODE NODEWIDTH) of N)
                          (fetch (GRAPHNODE NODEHEIGHT) of N])

)

(DECLARE%: EVAL@COMPILE

(RPAQQ LINKPARAMS Link% Parameters)

(CONSTANTS (LINKPARAMS 'Link% Parameters))
)

(RPAQQ DEFAULT.GRAPH.NODEBORDER NIL)

(RPAQQ DEFAULT.GRAPH.NODEFONT NIL)

(RPAQQ DEFAULT.GRAPH.NODELABELSHADE NIL)

(RPAQQ ScalableLinkParameters (LINEWIDTH))

(RPAQQ CACHE/NODE/LABEL/BITMAPS NIL)

(RPAQQ NODEBORDERWIDTH 1)

(RPAQQ GRAPH/HARDCOPY/FORMAT (MODE PORTRAIT PAGENUMBERS T TRANS NIL))

(RPAQ? DEFAULT.GRAPH.WINDOWSIZE (LIST (TIMES SCREENWIDTH 0.7)
                                      (TIMES SCREENHEIGHT 0.4)))

(RPAQ? EDITGRAPHMENUCOMMANDS
       '((Move% Node 'MOVENODE "Moves a single node in the graph." (SUBITEMS (|Move Single Node| 'MOVENODE
                                                                                 "Moves a single node in the
                                                                                 graph.")
                                                                              (|Move Node and Subtree| (
                                                                                                        EDITMOVESUBTREE
                                                                                                          GRAPHWINDOW)
                                                                                 "Moves a subtree of nodes
                                                                                 relative to the movement of
                                                                                 their root.")
                                                                              (Move% Region (EDITMOVEREGION
                                                                                                  GRAPHWINDOW)
                                                                                 "Moves a group of nodes within
                                                                                 a specified region to another
                                                                                 region.")))
         ("Add Node" 'ADDNODE)
         ("Delete Node" 'DELETENODE)
         ("Add Link" 'ADDLINK)
         ("Delete Link" 'DELETELINK)
         ("Change label" 'CHANGELABEL)
         ("label smaller" 'SMALLER)
         ("label larger" 'LARGER)
         ("<-> Directed" 'DIRECTED)
         ("<-> Sides" 'SIDES)
         ("<-> Border" 'BORDER)
         ("<-> Shade" 'SHADE)
         STOP))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)

(DECLARE%: EVAL@COMPILE

(RECORD GRAPHNODE (NODEID NODEPOSITION NODELABELBITMAP NIL NODELABELSHADE NODEWIDTH NODEHEIGHT TONODES FROMNODES
                          NODEFONT NODELABEL NODEBORDER)
       NODEBORDER _ DEFAULT.GRAPH.NODEBORDER NODELABELSHADE _ DEFAULT.GRAPH.NODELABELSHADE NODEFONT _
       DEFAULT.GRAPH.NODEFONT)
```

```
(RECORD GRAPH (GRAPHNODES SIDESFLG DIRECTEDFLG GRAPH.MOVENODEFN GRAPH.ADDNODEFN GRAPH.DELETENODEFN
                    GRAPH.ADDLINKFN GRAPH.DELETELINKFN GRAPH.FONTCHANGEFN GRAPH.INVERTBORDERFN
                    GRAPH.INVERTLABELFN GRAPH.CHANGELABELFN . GRAPH.PROPS))
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS HALF MACRO ((X)
                      (LRSH X 1)))
)
)
```

;; Grapher image objects

```
(DEFINEQ
```

(GRAPHERIMAGEFNS

```
  [LAMBDA NIL                                                    ; Edited 11-Apr-2018 09:02 by rmk:
                                                                 ; Edited 11-Apr-2018 09:01 by rmk:
    (DECLARE (USEDFREE GRAPHERIMAGEFNS))
    (OR GRAPHERIMAGEFNS (SETQ GRAPHERIMAGEFNS (IMAGEFNSCREATE (FUNCTION GRAPHOBJ.DISPLAYFN)
                                                              (FUNCTION GRAPHOBJ.IMAGEBOXFN)
                                                              (FUNCTION GRAPHOBJ.PUTFN)
                                                              (FUNCTION GRAPHOBJ.GETFN)
                                                              (FUNCTION GRAPHOBJ.COPYFN)
                                                              (FUNCTION GRAPHOBJ.BUTTONEVENTINFN)
                                                              (FUNCTION GRAPHOBJ.COPYBUTTONEVENTFN)
                                                              (FUNCTION NILL)
                                                              (FUNCTION NILL)
                                                              (FUNCTION NILL)
                                                              (FUNCTION NILL)
                                                              (FUNCTION NILL)
                                                             NIL
                                                             'GRAPHER])

)

(DEFINEQ
```

(GRAPHERCOPYBUTTONEVENTFN

```
  [LAMBDA (WINDOW)                                               ; Edited  1-Aug-87 14:54 by sye
    ;; Called on down transition in WINDOW.  If GRAPHOBJ.FINDGRAPH locates a graph in window, it is copy inserted.  Another callers of
    ;; GRAPHOBJ.FINDGRAPH might also specify alignments to GRAPHEROBJ.
    (PROG* [(GRAPH (OR (GRAPHOBJ.FINDGRAPH WINDOW)
                       (RETURN)))
            (REG (GRAPHREGION GRAPH))
            (LEFT (MINUS (fetch (REGION LEFT) of REG)))
            (BOTTOM (MINUS (fetch (REGION BOTTOM) of REG)))
            (LEFTBUTTONFN (WINDOWPROP WINDOW 'BROWSER/LEFTFN))
            (MIDDLEBUTTONFN (WINDOWPROP WINDOW 'BROWSER/MIDDLEFN]
           (if (NOT (AND (ZEROP LEFT)
                         (ZEROP BOTTOM)))
               then (SETQ GRAPH (TRANSGRAPH GRAPH LEFT BOTTOM)))
           (COPYINSERT (GRAPHEROBJ GRAPH NIL NIL LEFTBUTTONFN MIDDLEBUTTONFN])
```

(GRAPHOBJ.FINDGRAPH

```
  [LAMBDA (WINDOW)                                               (* rmk%: "22-Dec-84 11:29")

          (* Get control on down transition, track until key goes up or mouse leaves the window)

    (bind (DS _ (GETSTREAM WINDOW))
          (REG _ (WINDOWPROP WINDOW 'REGION)) first (DSPFILL NIL BLACKSHADE 'INVERT DS)
       do (GETMOUSESTATE)
          (COND
             ((NOT (INSIDE? REG LASTMOUSEX LASTMOUSEY))
              (DSPFILL NIL BLACKSHADE 'INVERT DS)
              (RETURN))
             ((NOT (LASTMOUSESTATE (OR LEFT MIDDLE RIGHT)))
              (DSPFILL NIL BLACKSHADE 'INVERT DS)
              (RETURN (COPYGRAPH (WINDOWPROP WINDOW 'GRAPH])

)

(DEFINEQ
```

(ALIGNMENTNODE

```
  [LAMBDA (NODESPEC GRAPH)                                       ; Edited 29-Apr-94 14:01 by sybalsky
                                                                 (* Returns the alignment node specified by NODESPEC)

          (* Early implementation had *TOP, but documentation says *TOP*.
          Remove earlier ones (*TOP) at some point)
```

```
    (SELECTQ NODESPEC
        ((*TOP* *TOP)
             (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) largest (GN/TOP N)))
        ((*BOTTOM* *BOTTOM)
             (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) smallest (GN/BOTTOM N)))
        ((*RIGHT* *RIGHT)
             (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) largest (GN/RIGHT N)))
        ((*LEFT* *LEFT)
             (for N in (fetch (GRAPH GRAPHNODES) of GRAPH) smallest (GN/LEFT N)))
        (GETNODEFROMID NODESPEC (fetch (GRAPH GRAPHNODES) of GRAPH])
```

(GRAPHOBJ.CHECKALIGN

```
  [LAMBDA (GRAPH ALIGNSPEC)                                  ; Edited 29-Apr-94 14:02 by sybalsky
                                                             (* Makes sure that the ALIGNMENTSPEC is valid, putting it into
    standard form if necessary)
    (OR (AND (NULL ALIGNSPEC)
             (SETQ ALIGNSPEC 0))
        (NUMBERP ALIGNSPEC)
        [AND (LISTP ALIGNSPEC)
             (SELECTQ (CAR ALIGNSPEC)
                  ((*TOP* *BOTTOM* *LEFT* *RIGHT* *TOP *BOTTOM *LEFT *RIGHT)
                    T)
                  (GETNODEFROMID (CAR ALIGNSPEC)
                        (fetch (GRAPH GRAPHNODES) of GRAPH)))
             (LISTP (CDR ALIGNSPEC))
             (OR (NUMBERP (CADR ALIGNSPEC))
                 (EQ (CADR ALIGNSPEC)
                     'BASELINE)
                 (AND (NULL (CADR ALIGNSPEC))
                      (SETQ ALIGNSPEC (LIST (CAR ALIGNSPEC)
                                            0]
        (ERROR "ILLEGAL GRAPH ALIGNMENT SPECIFICATION" ALIGNSPEC))
    ALIGNSPEC])

)

(DEFINEQ
```

(GRAPHEROBJ

```
  [LAMBDA (GRAPH HALIGN VALIGN LEFTBUTTONFN MIDDLEBUTTONFN COPYBUTTONEVENTFN)
                                                             ; Edited 10-Apr-2018 11:01 by rmk:
                                                             (* rmk%: " 6-Dec-85 11:35")
                                                             (* Constructs a Grapher image object.)

          (* HALIGN and VALIGN specify the horizontal or vertical alignment.
          Each can be a floating point number between 0 and 1, specifying that the alignment point is located at that portion of the
          width/height of the graphregion, or a list of the form (nodespec align)%, where nodespec is a node ID or one of the atoms
          LEFT, RIGHT, BOTTOM, TOP, and align is either a floating point number bewtween 0 and 1, or the atom BASELINE)

    (LET ((REG (GRAPHREGION GRAPH))
          (OBJ (IMAGEOBJCREATE (LIST GRAPH (GRAPHOBJ.CHECKALIGN GRAPH HALIGN)
                                     (GRAPHOBJ.CHECKALIGN GRAPH VALIGN))
                               GRAPHERIMAGEFNS)))
         [IMAGEOBJPROP OBJ 'OBJECTORIGIN (CREATEPOSITION (MINUS (fetch (REGION LEFT) of REG))
                                                         (MINUS (fetch (REGION BOTTOM) of REG]
         (AND LEFTBUTTONFN (IMAGEOBJPROP OBJ 'LEFTBUTTONFN LEFTBUTTONFN))
         (AND MIDDLEBUTTONFN (IMAGEOBJPROP OBJ 'MIDDLEBUTTONFN MIDDLEBUTTONFN))
         (AND COPYBUTTONEVENTFN (IMAGEOBJPROP OBJ 'COPYBUTTONEVENTFN COPYBUTTONEVENTFN))
         OBJ])
```

(GRAPHOBJ.BUTTONEVENTINFN

```
  [LAMBDA (GROBJ WINDOW)                                     ; Edited  1-Aug-87 16:16 by sye
                                                             (* the user has pressed a button inside the grapher object
                                                             IMAGEOBJ.)
    (LET [(LEFT (IMAGEOBJPROP GROBJ 'LEFTBUTTONFN))
          (MIDDLE (IMAGEOBJPROP GROBJ 'MIDDLEBUTTONFN]
         (if (OR LEFT MIDDLE)
             then (GRAPHBUTTONEVENTFN WINDOW (CAR (IMAGEOBJPROP GROBJ 'OBJECTDATUM))
                         LEFT MIDDLE)
           elseif [MENU (create MENU
                                ITEMS _ '((Edit% graph T " Opens a window to edit this graph"]
             then (PROG [W (DATUM (IMAGEOBJPROP GROBJ 'OBJECTDATUM]
                        (SETQ W (SIZE/GRAPH/WINDOW (CAR DATUM)
                                     NIL T))
                        (IMAGEOBJPROP GROBJ 'OBJECTDATUM (LIST (EDITGRAPH1 (COPYGRAPH (CAR DATUM))
                                                                           W)
                                                               (CADR DATUM)
                                                               (CADDR DATUM)))
                        (CLOSEW W))
                  'CHANGED])
```

(GRAPHOBJ.COPYBUTTONEVENTFN

```
  [LAMBDA (GROBJ WINDOW)                                     (* rmk%: " 6-Dec-85 11:42")
```

(* the user has pressed a button inside the grapher object IMAGEOBJ while a copy key was down)

```
(LET [(CBEFN (IMAGEOBJPROP GROBJ 'COPYBUTTONEVENTFN]
     (if CBEFN
         then (APPLY* CBEFN GROBJ WINDOW)
      else (COPYINSERT (GRAPHOBJ.COPYFN GROBJ])
```

(GRAPHOBJ.COPYFN

```
[LAMBDA (GROBJ)                                          (* rmk%: " 6-Dec-85 12:07")
                                                         (* makes a copy of a grapher image object.)
  (LET* [(DATUM (IMAGEOBJPROP GROBJ 'OBJECTDATUM))
         (NEW (GRAPHEROBJ (COPYGRAPH (CAR DATUM))
                          (CADR DATUM)
                          (CADDR DATUM]
        [IMAGEOBJPROP NEW 'OBJECTORIGIN (create POSITION using (IMAGEOBJPROP GROBJ 'OBJECTORIGIN]
        (IMAGEOBJPROP NEW 'LEFTBUTTONFN (IMAGEOBJPROP GROBJ 'LEFTBUTTONFN))
        (IMAGEOBJPROP NEW 'MIDDLEBUTTONFN (IMAGEOBJPROP GROBJ 'MIDDLEBUTTONFN))
        (IMAGEOBJPROP NEW 'COPYBUTTONEVENTFN (IMAGEOBJPROP GROBJ 'COPYBUTTONEVENTFN))
        NEW])
```

(GRAPHOBJ.DISPLAYFN

```
[LAMBDA (GROBJ STREAM)                                   (* rmk%: " 2-Apr-85 10:56")
                                                         (* display function for a grapher image object)
```

(* Scale the streams position back to display coordinates, since DISPLAYGRAPH translates the translation. Might be simplest to define DISPLAYGRAPH without a translation, as locating the graph coordinate system at the current X,Y position)

```
  (PROG [REG (BOX (IMAGEOBJPROP GROBJ 'BOUNDBOX))
             (SCALE (DSPSCALE NIL STREAM))
             (GRAPH (CAR (IMAGEOBJPROP GROBJ 'OBJECTDATUM]
        (OR BOX (SETQ BOX (APPLY* (IMAGEOBJPROP GROBJ 'IMAGEBOXFN)
                                  GROBJ STREAM)))
        [SETQ REG (GRAPHREGION (COND
                                  ((EQP SCALE 1)
                                   GRAPH)
                                  (T (SCALE/GRAPH GRAPH STREAM SCALE]
```

(* Kludgy%: we have to scale the graph to get the real region, but then DISPLAYGRAPH will do it again, cause it assumes screen points.)

(* Other kludge is that the translation is also in screen points)

```
        (DISPLAYGRAPH GRAPH STREAM NIL (CREATEPOSITION (QUOTIENT (DIFFERENCE (DIFFERENCE (DSPXPOSITION NIL
                                                                                          STREAM)
                                                                                 (fetch XKERN of BOX))
                                                                          (fetch (REGION LEFT) of REG))
                                                               SCALE)
                                                      (QUOTIENT (DIFFERENCE (DIFFERENCE (DSPYPOSITION NIL STREAM)
                                                                                        (fetch YDESC of BOX))
                                                                            (fetch (REGION BOTTOM) of REG))
                                                                SCALE])
```

(GRAPHOBJ.GETALIGN

```
[LAMBDA (STREAM GRAPH)                                   ; Edited 29-Apr-94 14:02 by sybalsky
  (PROG ((ALIGN (READ STREAM FILERDTBL)))
        [if [AND (LISTP ALIGN)
                 (NOT (MEMB (CAR ALIGN)
                            '(*TOP* *BOTTOM* *LEFT* *RIGHT* *TOP *BOTTOM *LEFT *RIGHT]
            then (SETQ ALIGN (CONS [fetch (GRAPHNODE NODEID) of (CAR (NTH (CAR ALIGN)
                                                                          (fetch (GRAPH GRAPHNODES) of GRAPH]
                                   (CDR ALIGN]
        (RETURN ALIGN])
```

(GRAPHOBJ.GETFN

```
[LAMBDA (STREAM)                                         ; Edited  7-Dec-88 18:38 by sye
                                                         ; reads a grapher image object from a file.
  (OR (EQ (SKIPSEPRCODES STREAM FILERDTBL)
          (CHARCODE %()))
      (ERROR "ILLEGAL GRAPHOBJECT FORMAT"))
  (READCCODE STREAM)                                     (* Read the paren)
  (PROG ((GRAPH (READGRAPH STREAM))
         IMAGEOBJ)
        (SETQ IMAGEOBJ (GRAPHEROBJ GRAPH (GRAPHOBJ.GETALIGN STREAM GRAPH)
                                   (GRAPHOBJ.GETALIGN STREAM GRAPH)))
```

;; read leftbuttonfn & middlebuttonfn & copybuttoneventfn

```
        [COND
           ((NEQ (SKIPSEPRCODES STREAM FILERDTBL)
                 (CHARCODE %)))                          ; ) means extra props don't exist
            (IMAGEOBJPROP IMAGEOBJ 'LEFTBUTTONFN (HREAD STREAM))
            (IMAGEOBJPROP IMAGEOBJ 'MIDDLEBUTTONFN (HREAD STREAM))
            (IMAGEOBJPROP IMAGEOBJ 'COPYBUTTONEVENTFN (HREAD STREAM))
```

;; read imageobject origin

```
            (IMAGEOBJPROP IMAGEOBJ 'OBJECTORIGIN (CREATEPOSITION (READ STREAM)
                                                                 (READ STREAM]
          (RATOM STREAM FILERDTBL)                                    ; Skip the closing paren
          (RETURN IMAGEOBJ])
```

(GRAPHOBJ.IMAGEBOXFN

```
  [LAMBDA (GROBJ STREAM)                                              ; Edited 29-Apr-94 14:01 by sybalsky
                                                                      (* size function for a tedit bitmap object.)
    (PROG (REGION GRAPH HALIGN VALIGN ALNODE (DATUM (IMAGEOBJPROP GROBJ 'OBJECTDATUM))
                (SCALE (DSPSCALE NIL STREAM))
                BMW BMH)
          (SETQ GRAPH (CAR DATUM))
          (SETQ HALIGN (CADR DATUM))
          (SETQ VALIGN (CADDR DATUM))
          (OR (EQ 1 SCALE)
              (SETQ GRAPH (SCALE/GRAPH GRAPH STREAM SCALE)))
          (SETQ REGION (GRAPHREGION GRAPH))
          (RETURN (create IMAGEBOX
                          XSIZE _ (fetch (REGION WIDTH) of REGION)
                          YSIZE _ (fetch (REGION HEIGHT) of REGION)
                          YDESC _ [COND
                                      ((NUMBERP VALIGN)
                                       (TIMES VALIGN (fetch (REGION HEIGHT) of REGION)))
                                      (T                              (* Must be a list, cause of checks in GRAPHEROBJ)
                                         (SETQ ALNODE (ALIGNMENTNODE (CAR VALIGN)
                                                                     GRAPH))
                                         (PLUS (GN/BOTTOM ALNODE)
                                               (COND
                                                  ((EQ (CADR VALIGN)
                                                       'BASELINE)
                                                   (IQUOTIENT (IPLUS (IDIFFERENCE (fetch (GRAPHNODE NODEHEIGHT)
                                                                                     of ALNODE)
                                                                                  (FONTPROP (fetch (GRAPHNODE NODEFONT)
                                                                                               of ALNODE)
                                                                                            'ASCENT))
                                                                                (FONTPROP (fetch (GRAPHNODE NODEFONT)
                                                                                             of ALNODE)
                                                                                          'DESCENT))
                                                                     2))
                                                  (T (TIMES (CADR VALIGN)
                                                            (fetch (GRAPHNODE NODEHEIGHT) of ALNODE]
                          XKERN _ (COND
                                      ((NUMBERP HALIGN)
                                       (TIMES HALIGN (fetch (REGION WIDTH) of REGION)))
                                      (T                              (* Must be a list, cause of checks in GRAPHEROBJ)
                                         (SETQ ALNODE (ALIGNMENTNODE (CAR HALIGN)
                                                                     GRAPH))
                                         (PLUS (GN/LEFT ALNODE)
                                               (TIMES (COND
                                                          ((EQ (CADR HALIGN)
                                                               'BASELINE)
                                                           0)
                                                          (T (CADR HALIGN)))
                                                      (fetch (GRAPHNODE NODEWIDTH) of ALNODE])
```

(GRAPHOBJ.PUTALIGN

```
  [LAMBDA (STREAM GRAPH ALIGN)                                        ; Edited 29-Apr-94 14:02 by sybalsky
    (PRIN2 [COND
              ([OR (NLISTP ALIGN)
                   (MEMB (CAR ALIGN)
                         '(*TOP* *BOTTOM* *LEFT* *RIGHT* *TOP *BOTTOM *LEFT *RIGHT]
               ALIGN)
              (T                                                      (* Convert node ID to node index)
                 (CONS (for I from 1 as N in (fetch (GRAPH GRAPHNODES) of GRAPH)
                          when (EQ (CAR ALIGN)
                                   (fetch (GRAPHNODE NODEID) of N))
                          do (RETURN I))
                       (CDR ALIGN]
           STREAM FILERDTBL])
```

(GRAPHOBJ.PUTFN

```
  [LAMBDA (GROBJ STREAM)                                              (* rmk%: "31-Dec-84 12:25")
                                                                      (* Put a description of a grapher object into the file.)
    (PROG [ALIGN GRAPH (DATUM (IMAGEOBJPROP GROBJ 'OBJECTDATUM))
                 (OBJORIGIN (IMAGEOBJPROP GROBJ 'OBJECTORIGIN]
          (PRIN1 "(" STREAM)
```

;; dump graph

```
          (SETQ GRAPH (CAR DATUM))
          (DUMPGRAPH GRAPH STREAM)
          (TERPRI STREAM)
```

;; dump halign and valign

```
          (GRAPHOBJ.PUTALIGN STREAM GRAPH (CADR DATUM))
```

```
          (SPACES 1 STREAM)
          (GRAPHOBJ.PUTALIGN STREAM GRAPH (CADDR DATUM))
          (TERPRI STREAM)
    ;; dump leftbuttonfn & middlebuttonfn & copybuttoneventfn
          (HPRINT (IMAGEOBJPROP GROBJ 'LEFTBUTTONFN)
                 STREAM)
          (HPRINT (IMAGEOBJPROP GROBJ 'MIDDLEBUTTONFN)
                 STREAM)
          (HPRINT (IMAGEOBJPROP GROBJ 'COPYBUTTONEVENTFN)
                 STREAM)
    ;; dump objectorigin
          (PRIN1 (fetch XCOORD of OBJORIGIN)
                 STREAM)
          (SPACES 1 STREAM)
          (PRIN1 (fetch YCOORD of OBJORIGIN)
                 STREAM)
          (printout STREAM ")" T])

)

(DEFINEQ

(COPYGRAPH
  [LAMBDA (GRAPH)                                                    ; Edited 29-Apr-94 14:02 by sybalsky
    (create GRAPH using GRAPH GRAPHNODES _ (for N L in (fetch (GRAPH GRAPHNODES) of GRAPH)
                                                  collect (create GRAPHNODE
                                                                  using N NODEPOSITION _ (create POSITION
                                                                                                 using (fetch (GRAPHNODE
                                                                                                                 NODEPOSITION)
                                                                                                        of N))
                                                                  NODELABEL _ (CL:TYPECASE (SETQ L (fetch (GRAPHNODE
                                                                                                                  NODELABEL)
                                                                                                          of N))
                                                                                           (BITMAP (BITMAPCOPY L))
                                                                                           (IMAGEOBJ (APPLY* (IMAGEOBJPROP
                                                                                                                 L
                                                                                                                 'COPYFN)
                                                                                                             L))
                                                                                           (T L))])))

(DUMPGRAPH
  [LAMBDA (GRAPH STREAM)                                             ; Edited 29-Apr-94 14:02 by sybalsky
                                                                     (* Put a description of a graph into a file.)
    (RESETLST
        (RESETSAVE (SETREADTABLE FILERDTBL))
        (PROG (BORDERS FONTS IDS SHADES (%#BORDERS 0)
                      (%#FONTS 0)
                      (%#SHADES 0)
                      (%#IDS 0))
              (printout STREAM "(" T "FIELDS (")
              (if (fetch (GRAPH SIDESFLG) of GRAPH)
                  then (printout STREAM 2 "SIDESFLG " .P2 (fetch (GRAPH SIDESFLG) of GRAPH)))
              (if (fetch (GRAPH DIRECTEDFLG) of GRAPH)
                  then (printout STREAM 2 "DIRECTEDFLG " .P2 (fetch (GRAPH DIRECTEDFLG) of GRAPH)))
              (if (fetch (GRAPH GRAPH.MOVENODEFN) of GRAPH)
                  then (printout STREAM 2 "MOVENODEFN " .P2 (fetch (GRAPH GRAPH.MOVENODEFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.ADDNODEFN) of GRAPH)
                  then (printout STREAM 2 "ADDNODEFN " .P2 (fetch (GRAPH GRAPH.ADDNODEFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.DELETENODEFN) of GRAPH)
                  then (printout STREAM 2 "DELETENODEFN " .P2 (fetch (GRAPH GRAPH.DELETENODEFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.ADDLINKFN) of GRAPH)
                  then (printout STREAM 2 "ADDLINKFN " .P2 (fetch (GRAPH GRAPH.ADDLINKFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.DELETELINKFN) of GRAPH)
                  then (printout STREAM 2 "DELETELINKFN " .P2 (fetch (GRAPH GRAPH.DELETELINKFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.FONTCHANGEFN) of GRAPH)
                  then (printout STREAM 2 "FONTCHANGEFN " .P2 (fetch (GRAPH GRAPH.FONTCHANGEFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.INVERTBORDERFN) of GRAPH)
                  then (printout STREAM 2 "INVERTBORDERFN " .P2 (fetch (GRAPH GRAPH.INVERTBORDERFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.INVERTLABELFN) of GRAPH)
                  then (printout STREAM 2 "INVERTLABELFN " .P2 (fetch (GRAPH GRAPH.INVERTLABELFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.CHANGELABELFN) of GRAPH)
                  then (printout STREAM 2 "CHANGELABELFN " .P2 (fetch (GRAPH GRAPH.CHANGELABELFN) of GRAPH)))
              (if (fetch (GRAPH GRAPH.PROPS) of GRAPH)
                  then (printout STREAM 2 "PROPS ")
                       (HPRINT (fetch (GRAPH GRAPH.PROPS) of GRAPH)
                              STREAM))
              (PRIN1 ")" STREAM)
              [for N TEMP in (fetch (GRAPH GRAPHNODES) of GRAPH)
                 do [OR (ASSOC (fetch (GRAPHNODE NODEID) of N)
                               IDS)
                        (push IDS (CONS (fetch (GRAPHNODE NODEID) of N)
                                        (add %#IDS 1]
                    [AND (SETQ TEMP (fetch (GRAPHNODE NODELABELSHADE) of N))
                         (OR (ASSOC TEMP SHADES)
```

```
                        (push SHADES (CONS TEMP (add %#SHADES 1]
                  [OR (ASSOC (fetch (GRAPHNODE NODEFONT) of N)
                             FONTS)
                      (push FONTS (CONS (fetch (GRAPHNODE NODEFONT) of N)
                                        (add %#FONTS 1]
                  (SELECTQ (SETQ TEMP (fetch (GRAPHNODE NODEBORDER) of N))
                       ((T NIL))
                       (OR (ASSOC TEMP BORDERS)
                           (push BORDERS (CONS TEMP (add %#BORDERS 1]
            (printout STREAM T "IDS " %#IDS %,)
            (for X in (SETQ IDS (DREVERSE IDS)) do (PRIN2 (CAR X)
                                                          STREAM)
                                                   (SPACES 1 STREAM))
            (printout STREAM T "FONTS " %#FONTS %,)
            (for X in (SETQ FONTS (DREVERSE FONTS)) do (SETQ X (CAR X))
                                                       (PRIN2 (if (LISTP X)
                                                                  elseif (type? FONTDESCRIPTOR X)
                                                                    then (FONTUNPARSE X)
                                                                  elseif (FONTP X)
                                                                    then
                                                                  (* Mark it as a class)
                                                                         (CONS 'CLASS (FONTCLASSUNPARSE X)))
                                                              STREAM)
                                                       (SPACES 1 STREAM))
            [COND
               (BORDERS (printout STREAM T "BORDERS " %#BORDERS %,)
                        (for X (POS _ (POSITION STREAM)) in (SETQ BORDERS (DREVERSE BORDERS))
                           do (TAB POS 1 STREAM)
                              (HPRINT (CAR X)
                                      STREAM]
            [COND
               (SHADES (printout STREAM T "SHADES " %#SHADES %,)
                        (for X (POS _ (POSITION STREAM)) in (SETQ SHADES (DREVERSE SHADES))
                           do (TAB POS 1 STREAM)
                              (HPRINT (CAR X)
                                      STREAM]
            (printout STREAM T "NODES (")
            (for N POS in (fetch (GRAPH GRAPHNODES) of GRAPH)
               do (printout STREAM 2 "(" .P2 (CDR (ASSOC (fetch (GRAPHNODE NODEID) of N)
                                                         IDS))
                             %,)
                  (SETQ POS (POSITION STREAM))
                  (HPRINT (fetch (GRAPHNODE NODELABEL) of N)
                          STREAM)
                  (printout STREAM %, .TAB POS .P2 (fetch (GRAPHNODE NODEPOSITION) of N)
                          %, .P2 (CDR (ASSOC (fetch (GRAPHNODE NODEFONT) of N)
                                             FONTS))
                          %, .P2 (SELECTQ (fetch (GRAPHNODE NODEBORDER) of N)
                                       ((NIL T)
                                           (fetch (GRAPHNODE NODEBORDER) of N))
                                       (CDR (ASSOC (fetch (GRAPHNODE NODEBORDER) of N)
                                                   BORDERS)))
                          %, .P2 (AND (fetch (GRAPHNODE NODELABELSHADE) of N)
                                      (CDR (ASSOC (fetch (GRAPHNODE NODELABELSHADE) of N)
                                                  SHADES)))
                          %,)
                  (if (fetch (GRAPHNODE TONODES) of N)
                      then (PRIN1 "(" STREAM)
                           (for X in (fetch (GRAPHNODE TONODES) of N)
                              do (printout STREAM .P2 [COND
                                                         [(EQ (CAR (LISTP X))
                                                              'Link% Parameters)
                                                          (CONS (CAR X)
                                                                (CONS (CDR (ASSOC (CADR X)
                                                                                  IDS))
                                                                      (CDDR X]
                                                         (T (CDR (ASSOC X IDS]
                                      %,))
                           (PRIN1 ") " STREAM)
                    else (PRIN1 "NIL " STREAM))
                  (if (fetch (GRAPHNODE FROMNODES) of N)
                      then (PRIN1 "(" STREAM)
                           (for X in (fetch (GRAPHNODE FROMNODES) of N)
                              do (printout STREAM .P2 (CDR (ASSOC X IDS))
                                      %,))
                           (PRIN1 ")" STREAM)
                    else (PRIN1 NIL STREAM))
                  (printout STREAM ")" T))
            (PRIN1 "))" STREAM)))])
```

(**READGRAPH**

```
  [LAMBDA (STREAM)                                    ; Edited 29-Apr-94 14:02 by sybalsky
                                                      (* reads a graph from a file.)
    (OR (EQ (SKIPSEPRS STREAM FILERDTBL)
            '%()
        (ERROR "ILLEGAL GRAPH FORMAT"))
```

```
    (READC STREAM)                                          (* Read the paren)
    (bind NUM TEMP FONTS BORDERS SHADES IDS (GRAPH _ (create GRAPH))
       do (SELECTQ (SETQ TEMP (RATOM STREAM FILERDTBL))
                (FIELDS [for F on (READ STREAM FILERDTBL) by (CDDR F)
                           do (SELECTQ (CAR F)
                                    (SIDESFLG (replace (GRAPH SIDESFLG) of GRAPH with (CADR F)))
                                    (DIRECTEDFLG (replace (GRAPH DIRECTEDFLG) of GRAPH with (CADR F)))
                                    (MOVENODEFN (replace (GRAPH GRAPH.MOVENODEFN) of GRAPH with (CADR F)))
                                    (ADDNODEFN (replace (GRAPH GRAPH.ADDNODEFN) of GRAPH with (CADR F)))
                                    (DELETENODEFN (replace (GRAPH GRAPH.DELETENODEFN) of GRAPH with (CADR F)))
                                    (ADDLINKFN (replace (GRAPH GRAPH.ADDLINKFN) of GRAPH with (CADR F)))
                                    (DELETELINKFN (replace (GRAPH GRAPH.DELETELINKFN) of GRAPH with (CADR F)))
                                    (FONTCHANGEFN (replace (GRAPH GRAPH.FONTCHANGEFN) of GRAPH with (CADR F)))
                                    (INVERTBORDERFN
                                         (replace (GRAPH GRAPH.INVERTBORDERFN) of GRAPH with (CADR F)))
                                    (INVERTLABELFN
                                         (replace (GRAPH GRAPH.INVERTLABELFN) of GRAPH with (CADR F)))
                                    (CHANGELABELFN
                                         (replace (GRAPH GRAPH.CHANGELABELFN) of GRAPH with (CADR F)))
                                    (PROPS (replace (GRAPH GRAPH.PROPS) of GRAPH with (CADR F)))
                                    (ERROR "UNRECOGNIZED GRAPH FIELD" (CAR F])
                (IDS (SETQ NUM (RATOM STREAM FILERDTBL))
                     (SETQ IDS (ARRAY NUM))
                     (for I to NUM do (SETA IDS I (READ STREAM FILERDTBL))))
                (BORDERS (SETQ NUM (RATOM STREAM FILERDTBL))
                         (SETQ BORDERS (ARRAY NUM))
                         (for I to NUM do (SETA BORDERS I (HREAD STREAM))))
                (FONTS (SETQ NUM (RATOM STREAM FILERDTBL))
                       (SETQ FONTS (ARRAY NUM))
                       [for I to NUM do (SETA FONTS I (COND
                                                ((EQ (SETQ TEMP (READ STREAM FILERDTBL))
                                                     'C)   (* A font class)
                                                 (SETQ TEMP (READ STREAM FILERDTBL))
                                                 (FONTCLASS (CAR TEMP)
                                                        (CDR TEMP)))
                                                ((EQ (CAR (LISTP TEMP))
                                                     'CLASS)
                                                 (FONTCLASS (CADR TEMP)
                                                        (CDDR TEMP)))
                                                (T TEMP])
                (NODES (RATOM STREAM)                       (* Skip paren)
                       [replace (GRAPH GRAPHNODES) of GRAPH
                          with (while (EQ (SKIPSEPRS STREAM FILERDTBL)
                                          '%()
                                  collect (READC STREAM)
                                          (PROG1 (create GRAPHNODE
                                                         NODEID _ (ELT IDS (RATOM STREAM FILERDTBL))
                                                         NODELABEL _ (HREAD STREAM)
                                                         NODEPOSITION _ (READ STREAM FILERDTBL)
                                                         NODEFONT _ (ELT FONTS (RATOM STREAM FILERDTBL))
                                                         NODEBORDER _ (SELECTQ (SETQ TEMP (RATOM STREAM FILERDTBL))
                                                                              ((NIL T)
                                                                                   TEMP)
                                                                              (ELT BORDERS TEMP))
                                                         NODELABELSHADE _ (AND (SETQ TEMP (RATOM STREAM FILERDTBL))
                                                                               (ELT SHADES TEMP))
                                                         TONODES _ [for X in (READ STREAM FILERDTBL)
                                                                      collect (COND
                                                                                [(EQ (CAR (LISTP X))
                                                                                     'Link% Parameters)
                                                                                 (CONS (CAR X)
                                                                                       (CONS (ELT IDS (CADR X))
                                                                                             (CDDR X]
                                                                                (T (ELT IDS X]
                                                         FROMNODES _ (for X in (READ STREAM FILERDTBL)
                                                                        collect (ELT IDS X)))
                                                                                    (* Skip the closing paren)
                                                (RATOM STREAM FILERDTBL))]
                                                                                    (* Skip the closing paren)
                       (RATOM STREAM FILERDTBL))
                (SHADES (SETQ NUM (RATOM STREAM FILERDTBL))
                        (SETQ SHADES (ARRAY NUM))
                        (for I to NUM do (SETA SHADES I (HREAD STREAM))))
                (%)                                         (* The closing paren)
                    (RETURN GRAPH))
                (ERROR "INVALID GRAPHER IMAGE OBJECT" STREAM])

)

(RPAQ? GRAPHERIMAGEFNS )

(DECLARE%: DONTEVAL@LOAD DOCOPY

(GRAPHERIMAGEFNS)
)

(ADDTOVAR IMAGEOBJGETFNS (GRAPHOBJ.GETFN))
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX

MACRO INDEX

CONSTANT INDEX