File created: 2-Nov-2020 16:42:38 **{DSK}<Users>arunwelch>SkyDrive>Documents>unix>lisp>lde>notecards>SYSTEM>GRAPHERPATCH.;2**

changes to: (FNS EDITMOVEREGION EDITMOVESUBTREE NOT.TRACKCURSOR RECURSIVE.COLLECTDESCENDENTS MOVEDESCENDENTS COLLECT.CHILD.NODES CREATE.NEW.NODEPOSITION GETBOXPOSITION.FROMINITIALREGION INIT/NODES/FOR/LAYOUT NODECREATE COLLECTDESCENDENTS DISPLAYGRAPH DISPLAYNODELINKS GETNODEFROMID EDITADDNODE EDITCHANGELABEL EDITAPPLYTOLINK EDITCHANGEFONT PRINTDISPLAYNODE GRAPHOBJ.GETFN ERASE/GRAPHNODE DEFAULT.ADDNODEFN EDITTOGGLELABEL EDITMOVENODE EDITTOGGLEBORDER EDITDELETENODE DELETE/AND/DISPLAY/LINK ADD/AND/DISPLAY/LINK)

previous date: 25-Nov-93 16:26:15 {DSK}<Users>arunwelch>SkyDrive>Documents>unix>lisp>lde>notecards>SYSTEM>GRAPHERPATCH.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ GRAPHERPATCHCOMS
       (
```

;;; RG 1/28/87 From Dave Newman's DVNPATCH003 for NoteCards. Adds MoveRegion and MoveSubtree facilities.

;;; pmi 4/8/88: Added change to EDITADDNODE which allows a position to be specified for the node being added.

```
        [DECLARE%: DONTEVAL@LOAD FIRST (P (NC.LoadFileFromDirectories 'GRAPHER 'LISPUSERSDIRECTORIES]
        (FNS EDITMOVEREGION EDITMOVESUBTREE NOT.TRACKCURSOR RECURSIVE.COLLECTDESCENDENTS MOVEDESCENDENTS
             COLLECT.CHILD.NODES CREATE.NEW.NODEPOSITION GETBOXPOSITION.FROMINITIALREGION INIT/NODES/FOR/LAYOUT
             NODECREATE COLLECTDESCENDENTS)
```

;;; Nick Briggs performanct fixes

```
        (FNS DISPLAYGRAPH DISPLAYNODELINKS GETNODEFROMID)
        (FNS EDITADDNODE EDITCHANGELABEL EDITAPPLYTOLINK EDITCHANGEFONT)
        ;; When the new release of Grapher is available, the following fns can be removed
        (FNS PRINTDISPLAYNODE GRAPHOBJ.GETFN ERASE/GRAPHNODE)
        (FNS DEFAULT.ADDNODEFN EDITCHANGEFONT EDITTOGGLELABEL EDITMOVENODE EDITTOGGLEBORDER EDITDELETENODE
             DELETE/AND/DISPLAY/LINK ADD/AND/DISPLAY/LINK)
        (DECLARE%: DONTEVAL@LOAD (P (DREMOVE (SASSOC "Move Node" EDITGRAPHMENUCOMMANDS)
                                             EDITGRAPHMENUCOMMANDS)
                                    (ADDTOVAR 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.")))
)
                                    (SETQ EDITGRAPHMENU NIL)))
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              GRAPHERPATCH)))
```

;;; RG 1/28/87 From Dave Newman's DVNPATCH003 for NoteCards. Adds MoveRegion and MoveSubtree facilities.

;;; pmi 4/8/88: Added change to EDITADDNODE which allows a position to be specified for the node being added.

```
(DECLARE%: DONTEVAL@LOAD FIRST

(NC.LoadFileFromDirectories 'GRAPHER 'LISPUSERSDIRECTORIES)
)

(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  3-Aug-88 14:51 by pmi
```

;; 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 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 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."])
```

(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])
```

(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])
```

(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)))])
```

```
)
```

;;; Nick Briggs performanct fixes

```
(DEFINEQ
```

(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])
```

(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])
```

(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])
```

```
)

(DEFINEQ
```

(EDITADDNODE

```
[LAMBDA (W NewPosition MSGW NODELABELFN)                    ; Edited 9-Jan-89 14:23 by sye
                                                            ; 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 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])
```

(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])
```

(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])

)

;; When the new release of Grapher is available, the following fns can be removed

(DEFINEQ
```

(PRINTDISPLAYNODE

```
  [LAMBDA (NODE TRANS STREAM CLIP/REG)                           ; Edited 27-Jul-90 09:10 by tafel
                                                                 ; 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 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 NODEWIDTH of NODE))
                (HEIGHT (fetch NODEHEIGHT of NODE))
                (FONT (fetch (GRAPHNODE NODEFONT) of NODE))
                (NBW (GRAPHNODE/BORDER/WIDTH (fetch 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 NODELABELBITMAP of NODE))
                   (BITBLT (fetch NODELABELBITMAP of NODE)
                          0 0 STREAM LEFT BOTTOM WIDTH HEIGHT 'INPUT))
                  [(BITMAPP LABEL)
                   (COND
                      ((NEQ 0 NBW)
                       (DRAW/GRAPHNODE/BORDER (fetch 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 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 NODEBORDER of NODE)
                                 LEFT BOTTOM WIDTH HEIGHT STREAM))
                    (DSPFONT FONT STREAM)
                    (GRAPHER/CENTERPRINTINAREA LABEL LEFT BOTTOM WIDTH HEIGHT STREAM)
                    (AND (fetch NODELABELSHADE of NODE)
                         (FILL/GRAPHNODE/LABEL (fetch 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 NODELABELBITMAP of NODE with (BITMAPCREATE WIDTH HEIGHT))
                        (BITBLT STREAM LEFT BOTTOM (fetch NODELABELBITMAP of NODE)
                               0 0 WIDTH HEIGHT 'INPUT])
```

(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])
```

(ERASE/GRAPHNODE

```
  [LAMBDA (NODE STREAM TRANS)                                     ; Edited 27-Jul-90 09:10 by tafel
                                                                  (* 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 NODEWIDTH of NODE)
                (fetch NODEHEIGHT of NODE)
                'TEXTURE
                'REPLACE WHITESHADE])

)

(DEFINEQ
```

(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])
```

(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])
```

(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])
```

(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])
```

(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])
```

(DELETE/AND/DISPLAY/LINK

```
  [LAMBDA (FROMND TOND WIN G)                                    ; Edited 11-Jan-89 14:24 by sye
                                                                 (* 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 NODEID of TOND)
                              (TOLINKS FROMND))
                 (AND (MEMBTONODES (fetch 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])
```

(ADD/AND/DISPLAY/LINK

```
  [LAMBDA (FROMND TOND WIN G)                                   ; Edited 11-Jan-89 17:18 by sye
                                                                (* adds and displays a link.)
    (COND
       ((MEMBTONODES (fetch 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])

)

(DECLARE%: DONTEVAL@LOAD

(DREMOVE (SASSOC "Move Node" EDITGRAPHMENUCOMMANDS)
       EDITGRAPHMENUCOMMANDS)

(ADDTOVAR 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."))))

(SETQ EDITGRAPHMENU NIL)
)

(PUTPROPS GRAPHERPATCH FILETYPE :TCOMPL)

(PUTPROPS GRAPHERPATCH MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(PUTPROPS GRAPHERPATCH COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1989 1990 1993 2020))
```

FUNCTION INDEX

PROPERTY INDEX

VARIABLE INDEX