NoteCards Release1.2i Announcement

Xerox Corporation

Randy Trigg
Frank Halasz

[Location: {qv}<notecards>release1.2i>notecards.dcom]
[First written: 3/27/85 Randy Trigg]
[Last updated: 8/26/85 Randy Trigg]

This document updates the NoteCards Release1.1 User's Manual, describing changes and new features for Release1.2i. As usual, send bug reports to NoteCardsSupport.pa (or use the Lafite SendMail middle button menu) and matters of more global interest to Notecards↑.pa.

You must be in Intermezzo to run NoteCards Release1.2i. From now on, you can depend on the letter suffix following the release number to indicate the appropriate version of Interlisp.

Changes from 1.1 are mostly in the following areas: the NoteCards browser, notefiles interface, link icon display and user interface. In addition, there are various miscellaneous changes, a couple of new card types, and fixes of several outstanding 1.1 bugs.

1. Operating on a Notefile.

Checkpointing and aborting a session:

A fundamental change was made to the way Notecards updates its working notefile that allows 1.2i users to checkpoint their work, abort a session (losing work since the last checkpoint), and recover more gracefully from crashes. First, a word about the way Notecards notefiles are structured.

A notefile consists of two parts, an index area and a data area. The index includes for each notecard, several pointers into the data area. There are separate pointers for the notecard's substance, title, prop list, and links. When, say, a notecard's title is changed, the new title is written at the end of the data area (in fact the end of the file) and the index pointer is changed. In Release1.1 (and earlier), the index modifications happened out on the file as they occured. Now, in Release1.2i they happen in an in-core array and are not written to the file till checkpoint (or close) time. In addition, there is a checkpoint pointer that points to the end of file at the time of the last checkpoint or close. New data (such as a new title) is still written to the file, but always at the end of the file. Thus if a crash occurs and later the notefile is reopened, Notecards can notice the extra data beyond the checkpoint pointer and truncate the file at that point (if you confirm).

More concretely, there are now two new NoteFile Ops menu entries: "Checkpoint Session" and "Abort Session." Checkpointing causes any active cards to have their contents saved to the notefile (but not closed), the index array to be written back out to the file, and the checkpoint pointer to be reset to the end of the file. (Note that closing a notefile automatically does a CheckpointSession.) Aborting a session causes Notecards to close down, discarding all work since the last checkpoint or close.

When a notefile is opened, the checkpoint pointer is compared with the end of file pointer. If they don't agree, then you're asked whether the file should be truncated. You're also given the option of saving the extra work since the last checkpoint to a file. If valuable cards were created (or modifications made) since the last checkpoint, then you should answer yes and provide the name of a file in which to store the truncated information.

Next, you should open the truncated notefile and bring up a separate TEdit window on the file containing the truncated information. Though TEdit formatting information is lost, you can recover a card's text by browsing this file. (Note that scrolling from back to front will retrieve the most recent version of each card.)

[Note that closing (or saving without closing) a card writes it out to the file, but does not force the index to be updated. Thus, if crashes are anticipated, do CheckpointSession often.]

Compacting a Notefile:

Because Notecards never actually overwrites any information in the data area of a notefile, it is necessary to periodically compact the notefile. This facility has been improved in Release1.2i in two ways. It is now possible to specify a target file name for the compaction (rather than always going to the same name), and it is now possible to compact a notefile in place. These two choices form a submenu of the CompactNotefile entry in the Notefile Ops menu.

Copying, restoring, and backing up notefiles:

The menu entries for RestoreFromFloppy and BackupToFloppy have been removed from the NotefileOps menu. In their place is a general CopyNotefile option. It prompts you for source and target file names for the copy.

There is a new facility for checking in and out notefiles using locks for multiple users sharing a notefile. Still in the experimental stages, it must be called via the programmer's interface. See the programmer's interface documentation.

Inspecting and healing broken notefiles:

The old Repair option on the Notefile Ops menu is now called Inspect&Repair and has been improved considerably. Before rebuilding the links of your notefile, it reads the entire data area looking for good card parts (including outdated and deleted versions). It then allows you to delete and/or back up card parts to previous versions. All this is done interactively through a menu driven interface. Only when the notefile is deemed healthy are you allowed to perform the link rebuilding. For details on the operation of Inspect&Repair, see the document titled NotefileInspector.ted.

2. Changes to the Notecards user interface.

Stylesheets:

Several places in Notecards now use Tayloe Stansbury's stylesheet package for user interaction, in particular, changing a link's display mode, a browser's specs, or the default text and link icon fonts from the global parameters menu. Stylesheets allow packaging of several menus together with "buttons" governing individual menus and the stylesheet as a whole. Menus within a stylesheet can optionally allow multiple selections. All stylesheets have three global buttons "Done," "Reset," and "Abort." "Done" causes the new values to be accepted. "Reset" causes the

original values (when the stylesheet was entered) to be recovered. "Abort" causes the stylesheet to be exited without changing any values. Menus allowing multiple selections also have the buttons "All" and "Clear" attached. "All" causes all values in the menu to be selected while "Clear" unselects the entire menu. Toggling of menu entries is accomplished by left clicking the entry.

New global parameters:

The top level global parameters menu has several new additions. These (as well as some old 1.1 ones) are described below. To change the value of a global parameter, click on the variable name. The value will toggle between "Yes" and "No" if binary, and allow selection from an appropriate menu otherwise.

ForceSources, ForceFiling, ForceTitles: These dictate whether to bother you at card closing time about incomplete information for the card. If ForceFiling is set, for example, then you are asked to designate parent fileboxes of the card before closing. Similarly, for sources and titles. If ForceFiling is off (value is "No"), then cards without parents will be filed automatically in the ToBeFiled filebox at closing time. If ForceTitles is off, then an untitled card will be left with the title "Untitled." ForceTitles and ForceFiling default to "Yes," while ForceSources defaults to "No."

CloseCardsOffScreen: If "Yes," then when a card is closed, it is first dragged off screen so that the close happens invisibly.

MarkersInFileBoxes: If "Yes," then new fileboxes will contain the markers "FILE BOXES" and "NOTE CARDS." New child boxes are inserted under the FILE BOXES marker and new child cards under the NOTE CARDS marker. If "No," then new fileboxes come up without markers and new children are inserted at the current cursor position. Note that regardless of the MarkersInFileBoxes setting, if a filebox has no markers (because you've deleted them) then new children are inserted at the cursor position.

AlphabetizedFileBoxChildren: If "Yes," then new fileboxes will have the property OrderingFn set to NC.IDAlphOrder. This will cause any new cards put into such a filebox to be inserted in alphabetical order. For further details on OrderingFn's for fileboxes see Section 4.

DefaultLinkIconAttachBitmap, DefaultLinkIconShowTitle, DefaultLinkIconShowLinkType: These dictate the manner in which link icons are displayed if not currently specified in the icon. There are three fields of a link's display mode that can be set, unset, or floated independently. If a field is floated, then the global parameter for that field is consulted. For example, if a link icon's display mode has value FLOAT for the ShowTitle field, then whether the title gets shown inside the link icon depends on the value of DefaultLinkIconShowTitle. See below for a further description of a link's display mode.

LinkDashingInBrowsers: If "Yes," then browser links are drawn with dashed lines with the dashing style corresponding to the link's type. See Section 3 for further details on browser changes. Defaults to "No."

ArrowHeadsInBrowsers: This dictates whether arrow heads are drawn on browser links. The variable can be set to either "AtMidpoint," "AtEndpoint," or "None." See Section 3 for details. Defaults to "None."

EnableBravoToTEditConversion: If "Yes" then TEdit checks when getting a file whether that file is in Bravo format and if so, converts. This defaults to "No" for efficiency.

DefaultFont: This dictates the font that new text cards default to.

LinkIconFont: This dictates the font for text appearing in link icons.

Link icon display mode:

The display mode of a link icon can be changed by middle buttoning in the icon and selecting from the three menus in the resulting stylesheet. These are: AttachBitmap, ShowTitle, and ShowLinkType. AttachBitmap, if "Yes," causes link icons to be shown with a bitmap representing the type of the destination card attached at the left. ShowTitle and ShowLinkType, if "Yes," cause the link icon to contain the title of the destination card and/or the link type. Any of the three fields can have the value FLOAT, in which case the appropriate global parameter will be consulted. (See description of global parameters above.) If all fields are set to "No" (or the floating ones inherit No from the global parameters), then a small, uninformative icon is used to display the link.

"Pushing" and "Pulling" link icons:

There are now two ways to move or copy a link icon between cards or within a card. "Pulling" works like TEdit shift-select. That is, to move an icon, put the cursor where you want to move to and hold down the shift key (or shift and ctrl keys) while left clicking in the left or right quarter of the icon. The new style is called "pushing" and is done by holding down the shift key while left clicking in the middle part of the icon. Then move the cross-hairs cursor to the icon's new home and left-click. To abort a "push," just left click in the background. Note that "pushing" currently only works for copying, not moving.

Specifying notefile names and card titles:

A different editor has been incorporated into Notecards for obtaining card titles, file names, etc. This editor is the same one used in the top level lisp exec window (TTYIN). Thus you can change the title (or file name) given as prompt via mouse edits.

3. Changes to the Notecards browser.

Multiple roots:

Browsers can now contain multiple roots, in which case the graph will be laid out as a forest.

Dashed links:

Dashed browser links was a rarely used option in Release1.1, largely because of speed considerations. The speed of drawing dashed links has improved in Release1.2i by taking advantage of improvements in Grapher. There are currently nine different dashing styles possible. If a browser contains instances of more than nine different link types, then the last dashing style will be used repeatedly for each link type beyond the ninth. As before, link dashing is a user-settable option in the GlobalParameters menu (see Section 2).

Arrowheads:

Arrowheads can now be drawn on browser links. These show the direction of the notecards link being represented in the browser. This is a user-settable parameter in the GlobalParameters

menu with possible values AtMidpoint, AtEndpoint, or None. If AtMidpoint or AtEndpoint is specified, then arrowheads will be drawn at link midpoints or endpoints, respectively. However, in either case, if two browser nodes are connected by more than one link, then any arrowheads for those links will appear at the midpoints (so as not to overlap).

Browser specs:

Whereas in Release1.1 only the link types to traverse could be specified, in Release1.2i, link types is one of a number of browser specs. Also included are browser depth, format, and orientation. These are accessible through a BrowserSpecs stylesheet, a collection of 5 menus. For general details on the stylesheet interface see Section 2. In this case, the forward and backward link types menus are multi-selectable, that is, more than one entry can be chosen. The other three menus are used to make single selections.

Forward and backward link types function as in Release1.1. That is, the browser will contain only nodes for cards reachable from the root cards by following forward links in "line of direction" or backward links in "reverse line of direction."

Browser depth is chosen from a menu containing entries for the integers 0 through 9 and INF (or infinite depth). The default is INF, meaning that the browser will not be cut off until there are no more links to follow from leaf nodes. Choosing depth 0 means that only the root nodes will appear (and no links).

Browser format is one of *GRAPH*, LATTICE, COMPACT, or FAST. The latter three are provided by the grapher package and correspond to lattice, compact forest and fast forest, respectively. COMPACT and FAST generate virtual nodes (in double boxes) whenever two or more links would be drawn to the same node. LATTICE only generates virtual nodes when a cycle exists in the graph. *GRAPH* is a new format that never generates virtual nodes. The drawback to using *GRAPH* is that a cycle can cause lines to be drawn that cross boxes or overlap other lines. Thus you may have to move nodes around for legibility after computing the browser. The default is LATTICE.

Browser orientation is one of Horizontal, Vertical, Reverse/Horizontal, or Reverse/Vertical. These specify whether the graph is layed out left-to-right, top-to-bottom, right-to-left, or bottom-to-top, respectively. The default is Horizontal.

New middle button title bar menu options:

Several new entries have been added to the middle button menu invoked from a browser’s title bar. The options are now RecomputeBrowser, RelayoutGraph, ReconnectNodes, UnconnectNodes, ExpandBrowserNode, GraphEditMenu, and ChangeBrowserSpecs.

RecomputeBrowser causes the current contents of the browser to be thrown away and recomputed as in Release1.1. However, in Release1.2i, you can optionally specify a new set of root nodes.

RelayoutGraph does not rebuild the graph, but rather causes the nodes and links of the graph to be repositioned on the screen (using Grapher’s LAYOUTGRAPH). This will destroy any work you have done moving nodes within the graph.

ReconnectNodes first causes any link edges in the graph to be erased. (Note, however, that non-link edges, those created by "AddEdge" as described below, are ignored.) Then, each node in the

graph is connected to every other node in the graph for which there is a link between them having one of the currently selected link types. This can be useful for several reasons:

1. when the linking structure between cards has changed, but the current browser layout needs to be preserved.

2. when some browser nodes need to be moved, but dragging the connected links is too slow. In this case, do UnconnectNodes followed by ReconnectNodes (after you've moved the nodes around).

3. when special browser layouts are desired. For example, suppose you like the layout that Grapher gives you when certain links are left out or when you limit the depth. Then calling ReconnectNodes will fill in the missing links without affecting the graph's layout.

UnconnectNodes simply erases all edges in the browser. This is useful for positioning a browser's nodes before invoking ReconnectNodes.

ExpandBrowserNode allows you to enlarge the graph under a given node. After selecting a node, you're asked for a depth (defaults to 1). The graph is then expanded under the selected node to the given depth, following any currently selected links. Note that ExpandBrowserNode calls LAYOUTGRAPH so any existing special node arrangements will be lost.

GraphEditMenu brings up the graph editing menu. See the description below.

ChangeBrowserSpecs brings up the BrowserSpecs stylesheet to allow you to change any of the browser specs. These changes will be noticed at the next RecomputeBrowser, ExpandBrowserNode, etc.

Editing the browser manually and "structure editing":

The browser can be edited through the use of the GraphEditMenu. This menu can be obtained either by right-buttoning in the browser window or by choosing GraphEditMenu from the title bar middle button menu. The GraphEditMenu includes options for "structure editing"; that is, changing underlying NoteCards structure by editing the browser. The old options for editing without changing structure are also present. Given below are the menu items in GraphEditMenu and the actions they engender.

CreateCard&Node causes a new card to be created in the current Notefile and a corresponding node for it to be included in the browser. You're asked for the type of the new card, its title, and where to position the node representing it.

CreateLink&Edge causes a new link to be created between two existing cards and a corresponding edge to be drawn in the browser. (We call such an edge representing a Notecards link, a "link edge." See AddEdge below for creating non-link edges.) You're asked for the "From" and "To" nodes in the browser corresponding to the cards to be linked as well as a link type. The link icon for the new link is positioned at the cursor point in the From card if the card has text substance and an open window. Text cards with closed windows have links inserted at the start of the text stream. Otherwise, the new link is a global link. You can have multiple link edges between pairs of cards. In this case the edges are displayed in a spline or "flower" arrangement.

DeleteCard&Node causes a card to be deleted and its corresponding node in the browser to be removed. You are asked first to choose the node representing the card to be deleted and then to confirm the removal of the node (type "y" to confirm) and the deletion of the card. If the selected node is one of a set of virtual nodes (double boxed), then all nodes in the set (i.e. representing the given card) are removed.

DeleteLink&Edge causes a link in the Notefile to be deleted and the corresponding edge in the browser to be removed. You first pick the "From" and "To" nodes corresponding to the source and destination ends of the link respectively. Then, if there is only one link between those two cards, the link is deleted after user confirms. If there are multiple links between the two cards, then the user chooses from a menu of link types.

AddLabel puts a "label node" into the browser that does not represent a Notecard. You are prompted for a string forming the node's label and then must position the label node. This node is not boxed. (But note that "virtual" label nodes can be boxed and thus can be confused with non-virtual regular nodes.)

AddNode adds a node into the browser corresponding to some existing card. You are asked to point to a card (title bar or link icon) on the screen that this node is to represent and then to position the node.

AddEdge draws a line between two nodes in the browser. This edge does not correspond to a real link in the Notefile. To avoid confusion, it is best to have the arrowheads option on (see Section ??) in this case, since edges formed by AddEdge do not have arrowheads (or dashing). Only one such edge is allowed between any two nodes and none if there are already link edges between the nodes. Thus doing CreateLink&Edge will remove any existing non-link edge.

RemoveNode removes a node from the browser. It does not delete the card (if any) that the node represents. Edges into and out of the node are also removed. If the selected node is one of a set of virtual nodes representing the same card, then you will be told how many nodes will be removed with this one and will be asked to confirm. The only way to remove only one node of a set of virtual nodes, is to first manually remove edges into and out of it using RemoveEdge. Then RemoveNode can be used to remove only the one virtual node.

RemoveEdge removes an edge from the browser. It does not delete the link (if any) that the edge represents. The user is asked to select the "From" and "To" nodes of the edge.

MoveNode allows you to change the position of any node, rubber banding any edges pointing to it. You're asked to point to the node by left-buttoning, and holding down the left button, drag the node to its new position.

LabelSmaller is used to decrease the font size of label nodes. Note that it does not work for regular non-label nodes.

LabelLarger is used to increase the font size of label nodes.

<->Shade toggles the shade of a node between black-on-white and white-on-black. This can only be performed on label nodes (not on nodes representing Notecards).

FIX MENU causes the GraphEditMenu to be affixed to the lower right edge of the browser window. Note that this does not prevent you from obtaining the menu via right button inside the window.

[Note that the above editing commands do not work on old 1.1 browsers. Such browsers should either be recomputed (via RecomputeBrowser) or unconnected and reconnected.]

4. Miscellaneous changes.

Links ordering within text cards:

The internal list of outgoing links in a text card is now kept in the same order that the links appear in the card's text. This means, for example, that the daughters of a browser node for a filebox will appear in the correct order.

Link insertion:

The title bar menu entry for "InsertLinks" now has an attached submenu containing entries for adding single links, multiple links, and global links. When inserting multiple links (or adding multiple global links) you're only asked for one link type which is used to label all the new links and all are inserted at the same place in the text.

Show links:

This is now a normal entry in the left button title bar menu of a card (rather than a subentry under Edit Properties). The format of the ShowLinks display has been changed slightly. The prefix is now either TO, FROM, or Global TO. The link type is shown in the icon. Also, for text cards, the TO links should appear in the correct order.

Sketch changes and fixes:

Notecards now uses the latest version of sketch. See the sketch documentation for details on changes. Several long-standing bugs having to do with link icons in sketch cards have been fixed.

Sketches and graphs in text cards:

It is possible to shift-copy the contents of sketch and graph/browser cards into text cards. In addition, the Document card is now able to include the contents of sketch and graph cards if encountering them during card gathering. (It is still not possible for Document to include the contents of cards having user-defined substance types such as NCFile cards.)

Data saved at card closing:

When a card is closed, only those parts that are dirty are written out to the notefile. A message indicating which parts are being saved is now printed to the card's prompt window during closing. Furthermore, certain card types (in particular, browsers) were saving their substance even if no changes were made. This source of space inefficiency has been fixed in Release1.2i.

Ordering cards in a filebox:

It is now possible to dictate the relative placement of new cards in a filebox. If the OrderingFn property of a card has a value, it should be a lisp function that takes two card ID arguments and returns T if the first should appear before the second and NIL otherwise. You can make such a function appear automatically on new boxes for the case of alphabetizing by using the global parameter AlphabetizeFileBoxChildren. See section 2.

Programmer's interface:

The Programmer's interface has been updated. Thus users with existing programmer's interface code should read the revised PI documentation. The changes are not all forward compatible.

Notecards system date:

You can find the date of your Notecards system in the variable NC.SystemDate. The 'NewestFile property on the NC.SystemDate atom contains the name of the last modified Notecards file.

The Notecards library packages:

The old Release1.1 library packages have been converted to 1.2i and documented and several new ones have been added. These can be found on {qv}<notecards>release1.2>library> and include NCScreen, NCCluster, NCChain, NCFileCard, NCKeys, NCHacks, and ARIDemo. Documentation can be found in <filename>.ted.

NCScreen defines several handy functions for arranging cards on the screen callable from the programmer's interface. NCCluster defines several new card types, most notably CaseCluster, a cluster of cards for use in the sample domain of legal case analysis. NCChain defines the Chain card type, useful for breaking up a large text card into a linked chain of cards. NCFileCard defines the new File card type and FILE substance allowing a notefile to link to external files via standard Notecards links. NCKeys provides a shorthand language for invoking various handy programmer's interface functions. NCHacks contains several handy functions written using the programmer's interface. Two of these allow global text searches and replaces throughout a notefile. In addition there is a function that searches by last card modification date and one that links cards to form chains. Finally, ARIDemo is an example of how the programmer's interface can be used to construct notefiles that demo themselves.

Loading NoteCards from different directories:

NoteCards now uses the values of four directories variables to decide from whence to load the code. These are NOTECARDSDIRECTORIES, NOTECARDSMAPDIRECTORIES, QUADTREEDIRECTORY, and MAPFILEDIRECTORY. They default to ({QV}<NOTECARDS>RELEASE1.2I>), ({QV}<NOTECARDS>MAPS>NEW>), {QV}<NOTECARDS>MAPS>, and {QV}<NOTECARDS>MAPS> respectively.

5. Known bugs and plans for future improvements:

o The compactor should check first for available space.

o There are major speed problems in redrawing large browsers. Changing link display mode could also use some streamlining.

o Integrate the document compiler and the types mechanism so that instances of new card types can be sucked into TEdit documents.

o Make links into full-fledged objects having properties and type hierarchies.