THE INTERLEAF USER INTERFACE

by Robert A. Morris

Interleaf, Inc.[1]

The University of Massachusetts at Boston[2]

Abstract. The user interface of the Interleaf Publishing System is analyzed as a collection of different views of a document. Each view provides a context with which a user may model the system response to actions.

The Interleaf Publishing System (IPS) is a family of commercial electronic publishing systems which run on a variety of bit-mapped work stations with output to several different laser printers and typesetters. It is designed for a high degree of integration between text and graphics, either of which can be created interactively or imported from other systems. The system features in their state in late 1985 have been described elsewhere [3.] This paper expands on one of the themes presented there to cast the user interface of this software in the *document views* terminology of Meyrowitz and VanDam [2.]

The author is a member of the development staff of Interleaf, but did not design or implement any of the portion of the software described. The description is entirely the author's analysis, and does not represent views of Interleaf, nor of the architects of the user interface described here.

Among principal visible features of the user interface of IPS are

- object oriented document model
- rigorous mouse model
- progressive disclosure
- rapid response
- dynamic default determination
- limited undo facility
- uniform interface to graphics and text objects

IPS is an object-oriented system, with all interaction being of the form *select-act*. The mouse model of IPS rigorously uses the left button to select an object, the right button to extend or contract the selection, and the middle button to bring up a hierarchical popup menu of possible actions for the selected objects. If a given action has a sub-tree attached to it, the user sees a rightward pointing

---

This paper is in final form and no version of it will be submitted for publication elsewhere.

1. 10 Canal Park, Cambridge MA. 02141.

2. Department of Mathematics and Computer Science, Boston, MA. 02125.

arrow on that entry and may slide off the menu to bring up an additional menu (see Figure 3.). The user need expose only those choices whose default action is not known (indeed, a 150 ms. delay in displaying the popup permits the skilled user bypass the menu display altogether and accept the default by quickly releasing the middle mouse button.). Exactly what the default action is varies with the context in ways exhibited below. Most actions which do not have an explicit undo facility have a toggle or other simple way to return to the previous state. Text, graphics, and documents as a whole have a uniform user interface described by the select-act paradigm and conforming to the mouse model.

OFFICE VIEW. The first view presented to an IPS user is that of the desktop. Although designated as a desktop, this view might be deemed an *office view*, since it deals with the totality of the work flow associated to producing documents, including filing, printing and organizing them. In the present implementation, these icons are visible to the user either on the desktop or in containers on the desktop. These icons are illustrated in Figure 1. and described individually below.

IPS presently has a very limited notion of an object hierarchy, which begins with the desktop, the first piece of the system with which the user must interact. This follows a paradigm now familiar to users of bit mapped graphics systems. It is a window based system with a small number of icons representing different kinds of objects in the system. This model first became widely known with the Xerox Star system, described in [1.], which was a commercial implementation of research ideas at the Xerox Palo Alto Research Center.

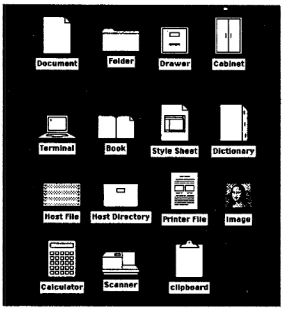

Figure 1. Portion of users desktop showing various icons.

Document – When such an icon is selected and opened the user will be editing the named document.

Containers – Container icons have the form of file folders, drawers, cabinets and books. When a container is opened, additional icons may be found of the same types as those on the desktop (with the exception of the clipboard). With the exception of books, described below, the semantics of a container is left entirely to the user. On a host with a tree structured file system they correspond to directories.

Terminals – When a terminal is opened, a native command processor runs in the resulting window. This virtual terminal facility is not available on all versions of the system.

Books – A book is a special container used for the support of large documents. All the sub–books and documents of a book are maintained as though they were a single document. In particular, pagination, automatic numbering, automatic cross–references, indexing and table of contents apply to all the objects in a book, but it is necessary only to open any sub–object for editing. All automatic numerical entities (e.g. page numbers, auto–numbers, etc.) are maintained in the geometric collation order of the sub-objects. That is, numbering proceeds from object to object, left to right, top to bottom, recursively through any sub–books. Thus, selecting and moving a sub–object will automatically change these numbers.

Style Sheets – Editable like a document, except that when found in a book, they is used to define the formatting style of the objects in a book and their components.

Dictionaries – When a dictionary is opened, the user is editing a plain-text file consisting of private exceptions to the spelling checker. In the current implementation, only a system-wide dictionary and one dictionary on the desktop are examined.

Host files – Files not recognizable as having been created by IPS, but typically found as ASCII plain-text files imported from other software are designated by this icon. Upon opening them, IPS attempts to make a document with the same text in it and operate on that instead of the original.

Host directories – Directories in the host file system behave like containers but appear to IPS to have been created by external software.

Printer files – It is possible to prepare files for printing without actually queuing them to a printer. In this case, the resulting file can not be opened, but it has a special appearance for the convenience of the user's maintenance.

Image – When files are created by digital scanners they are available for inclusion in documents and for subsequent editing. Image files are not edited directly, but rather are first pasted into a document.

Devices – Some installations have special devices such as digital scanners which can be controlled from the desktop to create documents directly. These have special icons and individually defined interactions on their menus. A graphic RPN calculator is also available. When opened, it has the appearance of a hand calculator, with keys pressed via mouse button clicks.

Clipboard – All objects cut or copied in any context reside in the clipboard until deleted either by (implicitly or explicitly) selecting and pasting them, or by opening the clipboard and purging all selected objects. In any context, cutting or copying an object or collection of objects leaves those objects selected on the clipboard even if not open. The subsequent paste operation will refer to them. No emacs-like "kill ring" or stack is kept, but the last cut from each document is kept and cut-and-paste between documents is transparent to the user. Pasted objects are deleted from the clipboard, so, for example, to paste multiple copies of something it is necessary to do a sequence

of *copy-paste* operations. However, after *paste*, the pasted object is selected, and the default action is *copy*, and after *copy* the default action is *paste*, so that very few mouse clicks are needed for repeated copying. This is an example of the dynamic defaulting mentioned above.

Once a document is opened, IPS presents the user with five views of it

- printed view
- collection of pages
- input view
- linear view
- structured view
- display view

PRINTED VIEW. The printed view comprises *printer properties* of a document which are manipulated from a property sheet invoked through an anchored popup menu at the top of the open document. It permits the choice of the target printer, whether revision bars, strike through and underlining should be suppressed on printing, and the control of the printing process itself. The latter presently includes such things as whether to invoke duplex printing, whether to insert a header page, and whether operator intervention is required at the printer for special forms.

COLLECTION OF PAGES. When viewed as a collection of pages, the document has properties relating to its pagination and layout such as page margins, page number and header/footer style, columnation, and orientation.

The property sheet menus for the printed view and the collection of pages view are property sheets invoked from anchored popup menus at the top of the menu. Examples are shown in Figure 2.

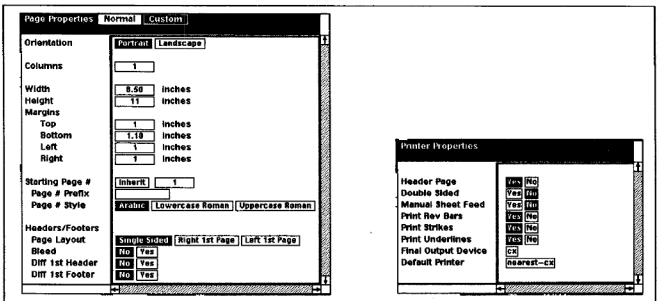

Figure 2. Page view and printed view property sheets

INPUT VIEW. Selecting a point in text (i.e., clicking the left button while the mouse cursor points at text) will cause the input cursor to move to that point. Subsequent input (whether from the keyboard or by a *paste* action) will take place at that point. Viewed in this way, the user has control over the document both from the keyboard and the mouse. From the keyboard, the system supports a collection of text attribute modifications with function and control keys. These include font weight, and underlining. In addition, editing features normally needed from the keyboard are also supported in this way. These include deleting characters, words or regions, creating new instances of components, and moving back and forth through the text.

When a point in text is selected, the *create* menu will have these sub-popups:

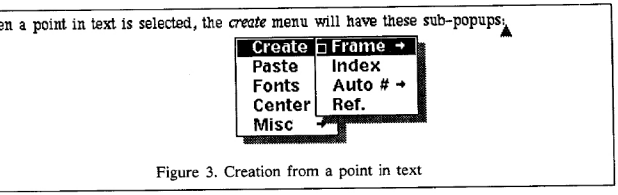

Figure 3. Creation from a point in text

The selection of *frame* on the sub-popup causes the creation of an object called a *frame*. Frames are rectangular portions of a document whose position and size is constrained in a number of ways controllable by the user. The most common constraint is to anchor the frame to the point of its creation (which is how the frame containing Figure 3. was created, anchored to the point of the line above the frame), but footnotes, for example, are frames anchored to the bottom of the page on which the anchor point falls. Once created, the anchor point is selected whenever the frame is, and cut-and-paste operations operate on both.

A frame can be as small as desired, but does not extend onto pages other than that on which its constraints force it. If a frame is bigger than the page on which it falls, the result should not interfere with printing the visible part. Frames have properties, which are edited with a property sheet similar to that for the printed and page views of a document. Editing these properties does not edit the *contents* of the frame, which can be done only by selecting those contents and acting on them.

Various objects can be created in frames, and these objects are clipped to the frame boundaries. It is curious that early printer software was not powerful enough to support this clipping properly and at least one large installation made innovative use of this "misfeature" to provide a capability not otherwise then available under IPS: the user placed a small frame anchored to the top of the page and the full width of the text. This frame was filled with a light gray box which was as long as the total text depth of the page. Since this box printed first, in its unclipped entirety, the result was the overprinting of formatted text on a textured background. The current release (3.0) of IPS now supports the notion of *micro-documents*, described below, which in large measure could provide this capability (although in the current release, text can not flow across page boundaries in a micro-document).

In the early commercial releases of IPS, graphics and unformatted text could be created inside a frame. The graphics objects were fully editable and could be combined in powerful ways to make

complex objects. Again, for each collection or individual object, interaction consists of selection followed by action. The range of possible actions has been steadily enhanced. At this release graphics objects can be scaled, rotated, filled given a border of various thickness, aligned to other objects or to a rectangular or isometric grid, and grouped with other objects in the same frame.

The graphics objects now include arcs of conic sections, continuous tone images displayable on grayscale monitors and printed on high resolution typesetters, as well as the polygonal and elliptical objects, data driven charts, and scanned line art that have been in the system previously. The user has a wide variety of methods for entering this data, including mouse, stylus, and puck interfaces, scanners, and filters from externally produced data such as plotter files from CAD-CAM systems. Once entered, it is fully editable. Most transformations apply to all objects except for charts, which do not rotate, and body text, which do not presently rotate or scale (the size of type is of course a property of text, so alterable). Details may be found in [3.], along with details of the image editing. Since that writing, image editing has been enhanced to support a variety of airbrushing features on black and white and on grayscale displays.

Outline fonts from the Bitstream library permit the use of high quality display faces which can be manipulated geometrically:

Figure 4. Outline fonts manipulated with the graphics subsystem

One of the most consequential additions to the latest version is the *micro-document*. A micro-document behaves like an automatically sizing document inside a frame. It behaves like an entire document in all respects but two: It is limited to the frame (hence to a single page) and frames may not be created within it. These restrictions appears only in the current implementation. Nothing in the design inherently prohibits text from crossing page boundaries or even flowing into connected micro-documents, although this behavior is not presently implemented. The micro-document feature allows a limited implementation of complex interactive user-defined layout, as these micro-documents can be selected and re-sized with the mouse just as an ordinary graphic. In this case, the micro-document is continuously kept composed according to the constraints implied by its current size.

Aside from frames, the user can also create at the input point a reference to another point in the document, an index token or an auto-number object. These comprise user named objects whose ordinal is maintained by the system.

Finally, at the input point one can change the current font, split the current component or create another instance of it, invoke the spelling checker[1] from that point forward, search for various objects from that point forward (text strings, references, etc.), and indicate that hyphenation is permitted at the current point.

The current input point can be changed either by selection with the mouse or with keyboard action. A number of the actions invoked at the current point are available from the keyboard also.

LINEAR VIEW. In the linear view, a document is an unstructured collection of text and frames. Any contiguous portion of the document can be selected by easy mouse action. Clicking the right mouse button will select all of the document from the input point to the point of the mouse cursor. Drag selection, wherein the right mouse button is held down while the mouse is moved, is also possible. The search actions mentioned above, when invoked from the input point, will also leave a portion of the document selected if they succeed.

When a region of the document is selected, keyboard and mouse provide some of the obvious actions, such as cut and copy (the region or a copy of it are moved to the clipboard), or change the font attributes of the region (face, weight, size, underline or strike through)

In the linear view of a document, the user can select, in various ways, a contiguous stream in the document, including text and the frames which contain graphics or micro-documents, and apply actions to the selected region. These actions include changing the font, examining alternative spellings, and cutting or copying the selected region to the clipboard. Here again, attempts are made to have intelligent dynamic defaults. For example, if the user delineates a region and selects the *italic* choice on the popup menu, then the *next* region selected will have *italic* as the default choice. Since popup menu display is suppressed for about 150 ms. after pressing the middle mouse button, the user who has internalized this behavior can very rapidly go through text italicizing selected portions without being presented menus.

STRUCTURED VIEW. The structured view of the document permits the selection of one or more named *components*. The properties of these components are local to the component, although a facility is provided for the global application of property changes to all components of the same name.

Unlike new pages, which are created by the system to conform to the typographic constraints implied by the properties of the document and its sub-objects, new components are created by explicit action of the user. The simplest way to do this is with the line-feed key. This creates a new instance of the same kind of component as the one in which typing currently is happening. A selected component can be changed to one of another name, in which case its other properties change to that of the other named one also. It is also possible to create from a popup menu, a component of an arbitrary name from among the existing component names. To create a totally new kind of component, the user creates one of existing type, and edits its properties including changing the name. The new name is now added to the list of existing component names, and relevant popup menus will display it. When the last instance of a component type is deleted, the user must explicitly delete the "master" description of this component if desired.

1. Actually one invokes the interactive spelling *corrector*. Misspellings are continuously and silently flagged during input.

Components are selected by pointing at the component name in a special column, the component bar, at the left of the document window and pushing the left mouse button. As throughout IPS, this selection can be extended or contracted by pointing at another component label and pushing the right mouse button. Although it is not possible to simultaneously edit the properties of multiple components, multiple selection of components is possible and useful, for example, for cutting and pasting to remove or reposition large pieces of the document.

In addition to the selection by direct mouse action, the component bar may in a sense itself be regarded as an object in the document (although it exists only when a document is displayed), with which the user controls the application of the selection and editing mechanism itself. Clicking the left button in the component window space between two components selects a point in the document between two components (the name of the first will be underlined, whereas when the component itself is selected, it is highlighted). Under these circumstances, the middle mouse button will evoke any of several component-related actions. These actions presently include creating new components, pasting components, searching for or selecting components by name, and joining the current component to its successor.

DISPLAY VIEW. This view consists of the location in its container of the icon for the closed object, the size and location of the window displaying the open version of the object, and the clipped placement of the object within that window in the event it does not fit entirely. Upon selection of a closed object *move* is one of the actions applicable to it. If selected, the mouse motion will move it around in its container until the *deselect* action is taken. For an open object of any sort, scroll bars at the bottom and right of the object permit the scrolling, re-sizing, and moving of the display window (Figure 5.) Clicking the left button *within* the window constitutes making a selection of the object pointed at. If this object is text, then the user has selected the text insertion point(for subsequent typing or the paste operation) and control is via the input view. If the object is the component bar, the result is as described above for the component view.

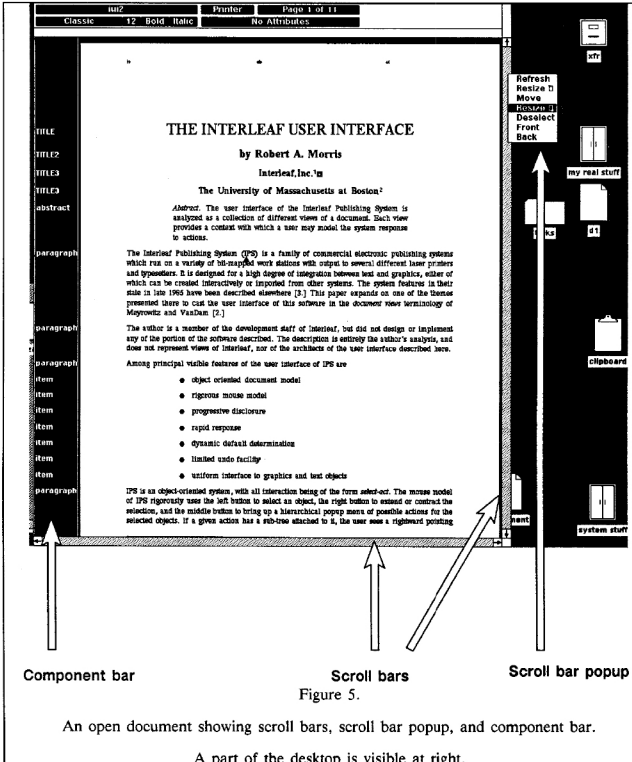

Figure 5.

An open document showing scroll bars, scroll bar popup, and component bar.

A part of the desktop is visible at right.

The display view of an object, whether a document or container, is most relevant to open objects, although the system remembers it when the object is closed. Although it does not automatically accompany the document if it is transmitted to the desktop of another user or to another host, the internal architecture of IPS would permit this, since the display view is recorded in a hidden file which does not normally accompany the document in transmission. In this sense, the display view of an object is really a view induced by the desktop, not the object itself.

SUMMARY. It has been shown how the user may model the behavior of the Interleaf Publishing System with any of five *views* of a document and one of the desktop. Each such view represents a paradigm for the system's reaction to user action.

References

1. Richard Furuta, Jeffrey Scofield, and Alan Shaw, "Document Formatting Systems: Survey, Concepts, and Issues," *Computing Surveys*, Vol 14, No. 3, 1982

2. Norman Meyrowitz and Andries van Dam, "Interactive Editing Systems, *Computing Surveys*, Vol. 14, No. 3, 1982.

3. Robert A. Morris, "Is What You See Enough To Get? A Description of the Interleaf Publishing System", in *Protext II, Proceedings of the Second International Conference on Text Processing Systems*, J.J.H Miller, ed., The Boole Press, Dublin, 1985.

This document was prepared on an Interleaf system. It is set in 12 point Interleaf Classic type and photographically reduced about 15% for final publication. Original printing was on an Interleaf LP308 300 dot/inch (12 dot/mm) laser printer based on the Canon CX marking engine. The outlines in Figure 4. are Copyright by Bitstream, Inc.