IS WHAT YOU SEE ENOUGH TO GET ?
A Description of the Interleaf Publishing System

Robert A. Morris
Interleaf, Inc.[1]
The University of Massachusetts at Boston[2]

Abstract. The author describes a commercial document preparation system which integrates text and graphics capabilities, composes documents interactively in real time, and maintains screen fidelity while producing high quality output on laser printers and typesetters.

I. Introduction

The Interleaf Publishing System (IPS) is a family of related document preparation systems which run on workstations manufactured by Apollo Corp., Cadmus Computer Systems, Digital Equipment Corp., PCS gmbH, and Sun Microsystems. Interleaf is a reseller of Apollo, Digital, and Sun equipment, and sells complete turnkey systems on those workstations. The system drives 12 dot/mm (300 dot/inch) laser printers and typesetters. Presently supported printers include the Imagen 8/300 and the Dataproducts LZR2630, and the Monotype Lasercomp typesetter. In addition, Interleaf has announced that it will support RIPrint, PostScript and Interpress printers, and the Compugraphic 8600G, Autologic APS-Micro5G, Allied Linotype Linotronic 300 and Information International, Inc. typesetters.

The IPS family ranges from the Workstation Publishing System (WPS), which runs on all the above-mentioned hosts and in some cases can be purchased from the workstation vendors, to the Technical Publishing System (TPS), which is available from Interleaf on the stations it resells. The differences are essentially a matter of features, configuration and pricing. There is no performance degradation with use of the full TPS system, provided that it is run with larger memory when using the image processing subsystem described below. Since the distinction is thus largely a commercial matter, this note will describe the entire high end system, referring to it as IPS.

[1] 1100 Massachusetts Ave., Cambridge, MA 02138
[2] Department of Mathematics and Computer Science, Boston, MA 02125
This paper is in final form and no version of it will be submitted for publication elsewhere.

The principal design goals of IPS are:

1. Maximum output quality possible on each supported device.
2. Screen fidelity (What-You-See-Is-What-You-Get or WYSIWYG).
3. Highly responsive editing and display in all subsystems.
4. A high degree of integration of text and graphics.
5. Document structuring facilities that can be manipulated by the user.
6. Ease of use.
7. Ease of portability to appropriate hardware.
8. Acceptance of text and graphics prepared on other systems.

II. The Document Model

IPS is an object oriented, property based system with a user interface heavily influenced by Smalltalk [4]. Consistency of the user interface has been rigorously sought. All action is of the form *noun-verb*, wherein an object is selected and then an action selected and applied. This is extremely easy to learn because the mouse model conforms naturally and rigidly to it: the left mouse button selects an object, the right button optionally extends or contracts that selection by adding or deleting the indicated object, and the middle button brings up a pop-up hierarchical menu of possible actions, one of which pends until the button is released (moving the mouse through the menu with the button held down changes the pending selection, which is highlighted). When object properties must be given numeric or textual properties, a property sheet appears and is manipulated with mouse and keyboard.

In the terminology of [10], an IPS document has four roughly orthogonal *document views* that can be manipulated by the user: (a) a collection of document structuring objects named *components*, whose properties are determined by the user, (b) the linear collection of text and graphics taken as a whole, (c) the entire document taken as a whole without regard to internal structure, and (d) the page structure.

The properties of one of these structures may impose defaults or constraints on those of another. For example, an attempt to create a graphic container wider than a page will provoke a complaint from the system and a requirement that the user confirm the intention.

(a) Components

The principal structure imposed on an IPS document is that of *components*, which are a collection of named objects into which text or graphics (in any mixture) are deposited. Components have properties whose values are entirely selected by the user (for example, the name is itself one such property). The system makes no assignments internally.

A user may enter text into a component merely by pointing at it and typing. Special containers, called *frames,* can be created in or anchored to a component to hold graphics objects. The size of a frame is a property independent of its contents, which will be clipped to the frame upon display or printing.

When a new document is created, it has, by default, only one component, whose name is *paragraph* and whose properties, such as default type face, margins, interline leading, etc., are predetermined. But this determination, including the name, is not internally defined. The creation of a document actually causes a copy to be made of a document from a conventional location in the file system. Each site, or even each host, can put an arbitrary document at that location, so that document creation can result in anything desired being the initial document. (IPS's present implementation does little in the way of per-user initialization, but it is also not difficult to modify the system so that document creation copies a document from a location defined on a changeable, per-user, basis). Since components may be duplicated and their properties changed, a user can cause the evolution of a document with arbitrary values of the properties. In practice, a user or a site will keep a variety of template documents and copying one of these is a common way to create a new document.

The component notions of IPS are a linearization of those of Etude [5], on which they are based. Although the *values* of properties are not internally defined, *what* the properties are is embedded in the system. Users can not create altogether new properties and must depend on the designers to anticipate the appropriate collection of properties.

Unlike those of Etude, components in IPS are not hierarchical. This contributes somewhat to its speed. A careful choice of properties by the IPS authors attempts to obviate the need for subcomponents. For example, in a hierarchical system, one purpose of having a paragraph be a subobject of chapter objects is to force certain constraints, such as that a chapter must begin on a new page and the first paragraph of that chapter be on the same page. The need for such constraints does not however imply an object hierarchy if the constraints themselves are properties of the objects. Such is the case for IPS components, among whose properties are *Begin New Page, Allow Break Within* and *Allow Break After.*

Components may be selected in a collection, either by the selection extension feature mentioned earlier, or by selection of the entire set of components with a given name. This collection can be copied or cut and pasted. Such manipulation is possible uniformly for all objects in the system. Because multiple documents can be open simultaneously, the manipulations can take place between documents as well as within documents (a paste operation simply applies to whatever document the mouse cursor is in when the operation is requested). Although the properties of only one component at a time can be manipulated, such manipulations are easily applied to all components of the same name in the document by a submenu choice *Global Apply.*

Figure 1 shows a screen image of two property sheets for components in version 2.5 of IPS, in beta testing as of this writing

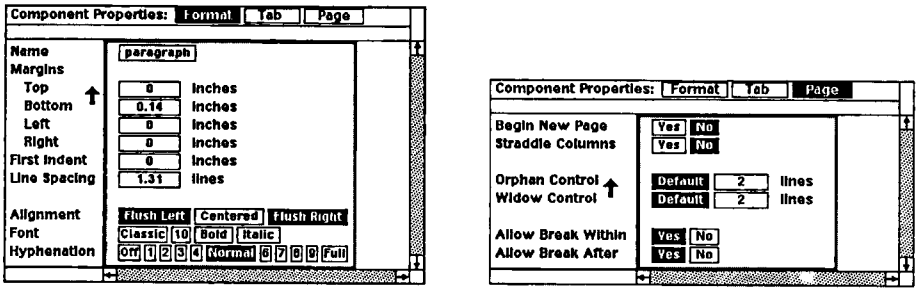


Figure 1. Component property sheets

(b) Linear Structure

A substantially less structured organization of a document consists of all the text and graphics objects of the document. Any contiguous subset of this structure may be selected by delineating it with mouse actions and a small number of useful actions applied to it (e.g. cut and paste, font changes, duplication). When the selection comprises a graphics object, editing its properties entails invoking one of the graphics subsystem editors described below, or changing the total shape of the frame in which the graphics object is displayed. Frames can be set off or inline, anchored to positions in text (i.e., floating on the page), fixed on the page, or fixed at a page location bound to the text (e.g., footnotes). Again, pasting between documents happens transparently, merely by moving to the window containing the target document.

In practice, manipulation of these unstructured regions is local in nature. Selections tend to be made for the purpose of minor rearrangement or correction of text, font changes, or the invocation of a graphics editing subsystem on a frame. Major rearrangement of the document, including moving paragraphs, is usually by cutting and pasting of components (indeed, deleting the *contents* of a component does not delete the component, nor change its properties).

Selection of such a region takes place in one of several ways. The user may select it manually by selecting a point in the document and then extending that selection. Consistent with the mouse model, this is done by pointing at a place and pressing the left button, then pointing at a second place and pressing the right button. The selected region appears in reverse video to indicate it is selected. Throughout the system, drag selection is also possible, so that holding the right button down causes the selection to

track the mouse. A standard search and replace facility also causes a region to be selected if it meets the specification of a requested search. Finally, the embedded spelling checker, when invited to do so, will select the next misspelled word after the current location and await the user's action to correct it.

(c) Page Structure

IPS documents are divided into pages with constraints derived in some cases from the components (namely those whose property lists include a forced page break or whose widow/orphan properties imply a page break), but mostly from the property list of the document as a whole. These properties can be manipulated in the same fashion as those for components. They include the page dimensions and margins (components have left and right margins relative to page margins, and top and bottom margins relative to adjacent components), the number of columns and the gutter width in the case of multi-column setting. (Prior to version 2.5 of IPS, headers and footers were properties of pages. Now they are merely a suitably constrained frame.)

Both formatting (i.e., line breaking) and page makeup thus are influenced by component as well as page properties (and, of course, text properties including font, tabs and required linebreaks and spaces).

(d) Global Document Structure

A small number of properties, largely of an advisory nature, pertain to the document as a whole.

The most important of these global properties is the *final output device*. All composition is done with the setwidths of the final output device, in order to optimize the appearance of the final document. (This imposes severe requirements on the quality of screen fonts, whose widths must remain close to those of the final device while at the same time retain characters artistically faithful to the given face.) In essence, all computerized composition systems behave in this way, but in non-WYSIWYG systems, it is perhaps not so immediately apparent to a user that a change in the printer will change the composition. A second printer, the *default target printer*, tells to which printer the document should actually be queued. It is the responsibility of the print spooling system to reconcile these (the output file carries enough information to make this possible, provided that the actual output device has sufficient capability). Of course if the target printer and the final device are identical or sufficiently similar, there is little issue. Interleaf supplies highly tuned fonts for 12 dot/mm printers so that they can be used as near typeset quality printers.

The output file contains fonts by name and the printing system must deal appropriately with the names. In addition, the setwidths are carried in the output file for each character used, so that a printing system can make compromises with knowledge of what the

composition system assumed. It is also possible at print request time to request a specific target or just to produce an output file on the file system. In each of the distributed versions of the system, the host's native print spooler is invoked but little other interaction with spoolers is provided from within IPS. It is not necessary to open a document to print it.

The only other global properties in the present version are whether the printing should carry a cover page (in an office environment a large number of documents are a single page and a cover is wasteful), and the number of consecutive hyphenated lines permitted. Only the latter is a composition constraint.

In the terminology of the Office Document Architecture standard of the European Computer Manufacturers' Association (ECMA) [6], one may say that IPS fixes *property rules,* that the user may influence the *relation rules* and also such of the few *content rules* which IPS admits. Note, however, that most such standards do not appear to deal adequately with graphics represented as high level geometric objects more complex than vectors and bitmaps. As will be clearer when graphics are discussed below, it is difficult to fit IPS precisely into any model which omits high level graphics.

III. User Interface

As mentioned earlier, the Interleaf Publishing System is an object oriented system in which the user selects an object and then an action to apply to that object. Recall that a uniform mouse model in all subsystems reserves the left mouse button for selection, the right for extending or contracting the selection, and the middle for bringing up a menu of possible actions. This section details the IPS user interface.

Default actions have been chosen with some care and and an attempt has been made to avoid presenting users with actions which do not apply in the current context. A 150 millisecond delay occurs before the popup menus are actually presented, so that a user experienced with a particular action can press and release the middle button immediately, invoking the default action before the menu appears. These defaults, like the menus themselves, are dynamic and context dependent. For example, after cut operations, the default becomes paste, so that a user can very quickly move objects within or between documents.

A series of popup menus anchored to the top of document window is invoked in the same way. They affect the document as a whole and are comprised of choices for acting on document and page properties and for printing.

Documents appear in windows which are themselves system objects and have the same scheme for dealing with them. By use of popups anchored to their borders, windows can be sized, positioned, or brought to the front or back of the current collection of

windows on the screen. Other objects can appear in these windows also. For example, property sheets appear in windows and provide for the assignment to objects of property values best suited to keyboard entry, such as numeric or alphabetic properties. Property lists are not always textual but may also be iconic. Thus, fill patterns are selected not by naming them, but by selecting a box of the desired pattern. Similarly, in the data driven chart subsystem the style of a chart is edited by selecting icons which represent the kind of chart (pie, bar, line, etc.) as well as certain other limited choice style parameters (such as the thickness of the border, the nature of hash marks, etc.). In the latter case, a sophisticated user has the capability to manipulate numerically some of the parameters with a more conventional property sheet.

To move an object from one open document to another, the user selects and cuts it in the first document and then moves the cursor to the second document and selects the point at which to paste it. When that selection is made, the paste operation will put the object in the new document. This is in all respects identical to moving an object *within* a document, except that in the latter case the mouse stays in the same window throughout.

There is a message system comprised of two parts. Advisory messages are posted in a small box at the top of each document. Each new message replaces the last. When the system requires input from the user, a window appears on the screen requiring the user to make a choice or enter keyboard input. When this is done, the window is closed and the action requested by the input is applied.

One important feature of the menu system (in addition to the fact that menus appear near where they are used, minimizing eye and mouse travel) is that submenus of popups are well hidden until the user desires their appearance. To implement progressive disclosure, any menu item with a submenu has a small arrow pointing to the right. If the user slides off the menu near this arrow, keeping the middle mouse button depressed (Figure 2a, below), a submenu appears adjacent to its parent (Figure 2b). Throughout the menu tree, the principle holds that the default choice need not be displayed. For example, the default choice for *Fonts Size* in the subtree below is *Fonts Size Larger*. The user could release the middle mouse button when the popup is as in Figure 2b and get the same effect as releasing it in Figure 2c, namely, the type size of the selected object would increase to the next available size.

Finally, it should be mentioned that some actions are available also from the keyboard. These are actions which are naturally required when entering text, such as deleting a word or character, deleting a region selected by a search command, or toggling bold or italic. These actions are obtained by the use of designated function keys, alternate shift keys such as the Control key, or keyboard escape sequences.

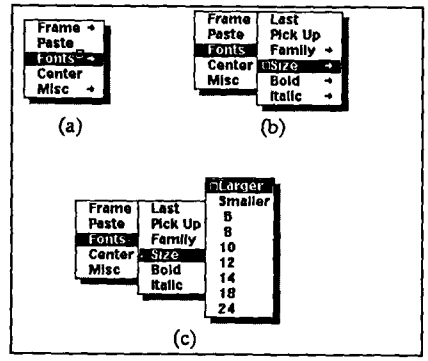


Figure 2. Portion of a popup menu tree

All versions of IPS follow the desktop paradigm apparently introduced in the Bravo editor developed at Xerox Palo Alto Research Center in the mid 1970's (Bravo and the Star, Xerox's product dervied from it, are described in [3].) The screen contains icons which denote documents or the containers of documents, and a few others described below. The containers are icons of cabinets, drawers and folders, but the hierarchical organization of containers is entirely the user's choice. Like other objects in the system, documents and containers can be selected and (a small number of) actions can be applied to them. These are moving their display location, cutting them (for deletion or later pasting elsewhere in the tree), opening them, and printing them (which results in a recursive descent through the container, printing all documents in the subtree). When multiple objects are selected, the chosen action applies to all of them. Objects remember their display location and size, so that when opened they always have the same appearance as when they were last closed.

Several other icons can appear in containers or on the desktop, which is essentially the root of the container tree and need not be distinguished from other containers, except that closing it means closing the IPS system and returning control to its invoker. One such icon is a computer terminal, which represents a virtual terminal. When opened, the host's command processor is invoked in the resulting window. A small amount of exchange of text is possible between documents and these windows. This permits text being manipulated by tools in the (possibly remote) host to be added directly to open documents. This feature requires some operating system support not available on all the platforms on which the system runs. (In some implementations IPS runs under the native window system which may provide a similar functionality. This is true of the Apollo but not the Sun implementation, where the window system presently entails too much overhead and insufficient information to support IPS running under it.)

A few special purpose icons representing foreign objects (see Section IV, below), including host files and directories, can appear and be manipulated. Typically these are placed in the desktop hierarchy by some external software. For example, the author keeps on his desktop a file representing his home directory and another representing the root directory of his node on the network. Opening these permits perusing these directories, requiring no return to the operating system to do so. Other icons represent plain text files, perhaps deposited in a container from another computer altogether.

IV. Foreign Interfaces

In a continuing effort, Interleaf releases support for the inclusion in IPS documents of objects produced by other software. This support consists of filters to produce internal or document exchange formats from standard ones, such as those produced by popular word processors, spread sheets (including the ability to produce charts or simple tables in a document), and the output from CAD/CAM systems, including CALCOMP 960 and 925 format files and IGES files (IGES and other graphics standards are described in [12] and [1].)

The filters range in sophistication. They attempt to leave the user with little or no work within IPS to usefully incorporate the output of the foreign system into a document. Little attention has been given to returning IPS objects to foreign creators, although, somewhat to Interleaf's surprise, many users rely heavily on the filters to prepare text and even sophisticated graphics on low cost stations such as IBM P.C.'s. Such users are essentially using IPS principally as a composition system, not for editing, and would benefit by greater information flow back to their principal work station. In addition to the filters, Interleaf provides some communications programs for direct connection to popular word processors, personal computers, and mainframes.

A form of the internal markup and graphics languages, which consists only of the printable ASCII character set, is published (and supported by the IO choices in the menus). This permits IPS documents to be exchanged readily between IPS systems on disparate hardware. It also permits sophisticated users to write their own filters or even to prepare "fully composed" documents with a standard text editor. When such documents are read by an IPS system they will appear exactly as if they had been composed on the reading system.

V. Word Processing Features

IPS systems provide a relatively standard collection of text editing and word processing features chosen to support the production of documents.

A search feature permits strings to be selected when indicated in full. The full range of actions described earlier can then be applied to the selected text. (This method of

selecting does not produce different results from marking with the mouse.) In addition, the indicated text can be replaced globally or with an interactive query—each repetition of the search query will move the cursor to the corresponding text and select it.

Moves through the text on a per character, word or line basis can be made with keyboard commands.

Dictionary based spelling checking is done continuously. Upon request, the system will flag the next misspelled word and invite the user to correct it, ignore it or add it to a local dictionary. The system is presently delivered with the Houghton-Mifflin dictionary of 85,000 words.

Hyphenation is also dictionary based, but words not in the dictionary are hyphenated algorithmically with the original method of TeX [8]). The user can control the effect of hyphenation by globally limiting the number of successive lines to be hyphenated or by controlling the extent to which hyphenation will tighten lines. As with the internal flagging of misspelling, hyphenation points are continuously added as typing takes place (or when the document is first composed, if it is read from the file system as a foreign document). The points are stored with the document when it is written in the default storage form which is designed for fast document loading.

The formatting of text is determined by constraints implied by the properties of its containing component and page, as well as the fonts of the characters.

VI. Layout

IPS distinguishes *formatting* from *pagination*. The former is concerned with line breaks, the latter with page layout. Both are accomplished continuously on each keystroke and in real time.

Internally, composition follows the boxes and glue paradigm of Knuth [8,9] but, unlike TeX, the formatter does not look backwards. Only the current line is reformatted, together with only those succeeding lines whose formatting must change as a consequence. In addition, any implied changes to the page layout are carried out immediately. For most real documents, it is impossible to type faster than the recomposition. (In the distributed system there is an exception to this real time composition ability: the automatic backup facility from time to time will sieze control and write the current copy of the document to the file system as protection against crashes or user errors. IO time is heavily determined by the graphical complexity of the document, but a typical document can be saved at the rate of about 2-3 pages/second. Typeahead buffers the intervening keystrokes and the real-time composition will continue after the backup).

The number of columns, the gutter width, and whether to justify vertically are properties of the page objects in IPS, but they are global to the document. It is not presently

possible to have some components set in N columns and some in M columns. However, whether a component straddles *all* columns is a component property. More sophisticated page layout does not violate IPS user or internal models, and design is under way to accomplish it without sacrificing performance, as was achieved when passing from single to multi-column setting.

VII. Graphics Features

The graphics subsystems of IPS comprise three relatively independent pieces. These are the the diagramming system, the image system and the data driven chart system.

Recall that all graphics objects are contained in a rectangular object called a *frame*. Frame properties are edited as all others. Such editing does not affect the display of the graphics within, except that they are clipped to the frame boundaries. (The display of clipped objects presents interesting questions about exactly where the clipping should take place upon output—at the output device or in the display file. These questions are not relevant to the present discussion, except to express the opinion shared by the other IPS developers that the output device is the appropriate clipper, and that graphics objects should carry a representation which specifies both the object and its display bounds. This is discussed further in describing incremental redisplay below).

Frame properties are largely numeric (for example, size) or positional. Frames may be constrained to follow an associated anchor, to follow the *text* on the page with the anchor, or to appear at a specific location on the page containing the anchor. In the latter two cases the frame can also be constrained to the left, right, center or with a numeric margin with respect to the column or page containing the anchor. Frame properties are edited by calling up a property sheet which stays on the screen until closed. Footnotes are simply implemented as frames constrained to the bottom of the page of the anchor. Headers and footers are implemented as (internally) named frames constrained to top and bottom of the page and containing text which is constant or derived from the values of internal variables such as the page counter and the document name (in release 2.5 these are the only such variables, but future releases will contain more generality and control of the variables by the user).

Frames, such as this one  , may occur anchored to text or anchored to a page position. The contents of a frame are edited by first selecting and opening the frame. Within a frame, objects of each graphics subsystem may be freely mixed. All graphics objects in the current implementation appear as though in successive planes, with display determined by the painter's algorithm: successors are painted on top of their predecessors. High speed display software controls incremental redisplay so that as little as possible is repainted upon the motion or removal of an object (but when windows and menus appear, the system simply stores the bit array they are overwriting for restoration when they are removed).

Here again display clipping issues arise, since repainting after removal of a partially overlaying object requires redisplaying only that portion of the object underneath revealed by the removal. In this case it becomes somewhat clearer why the display system should clip. Consider the use of some variant of the Cohen-Sutherland clipping algorithm [2]. Suppose there is a black line underneath an opaque box which is then removed. The boundaries of the box may be regarded as the clipping window for the Cohen-Sutherland algorithm. Such an algorithm attempts to find the intersection of the line with the box and request a new line from one intersection to another. But in the discrete pixel plane it is not possible to guarantee that this intersection actually hits precisely the same pixels at the border as would be turned on by the lower level line painter if it redisplayed the entire line. It is therefore appropriate to use "restartable" drawing primitives in which the primitive object, in this example a line, is given by its complete specification but computed only inside the clipping window. A restartable Bresenham algorithm for line drawing seems well known and was first described publicly in [11]. "Naive clipping," in which the entire object is calculated but only the visible pixels displayed, is possible but time consuming on a screen. On a printer it is more feasible, and one can envision drawing objects in memory in their entirety and imaging by writing to the imaging hardware only a region of memory corresponding to the clipping boundaries.

The creation and editing of each of the three kinds of objects is described below.

(a) The Diagramming System

The diagramming system comprises a system akin to the MacDraw package of the MacIntosh computer. These systems have their antecedents in CAD-CAM software and in the Draw package of the Xerox Bravo editor mentioned earlier.

Under mouse control, the user can create a rather small number of primitive objects but combine them in powerful ways to make sophisticated graphics in a short time. The primitive objects are lines, polylines (piecewise linear paths which cannot be decomposed), ellipses, boxes, and certain kinds of splines. These objects may be grouped in any combination, and grouping can be recursive; groups can be made of other groups.

When a collection is grouped, selecting any piece of it (by pointing at it and clicking the left mouse button) causes the *group* to be selected. Editing the properties of the group causes each of its members to be given the chosen properties, but a *subedit* facility permits the pieces of a group to be edited without first decomposing the group.

Properties of graphics objects include their texture and thickness, if any (when filled zero width objects appear borderless), the font of any text they contain and a set of constraints called *locks* described later.

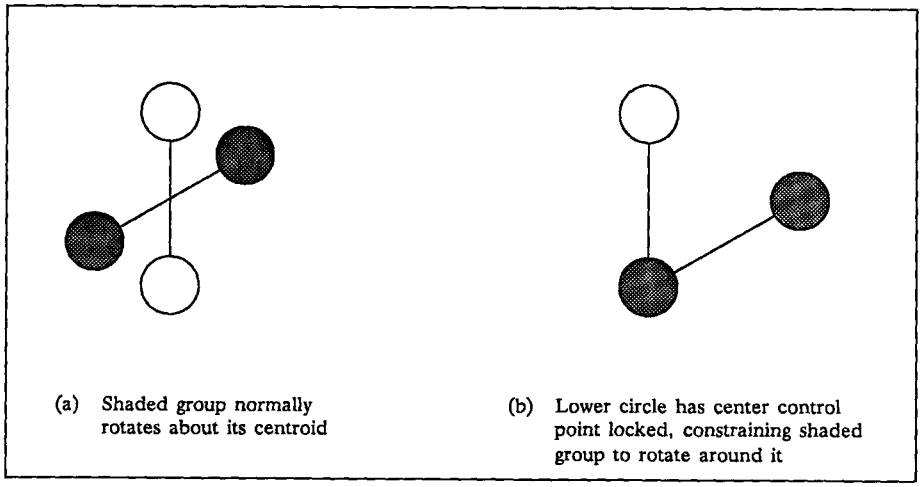
Objects can be transformed in a number of ways. They can be scaled (anamorphically or uniformly), rotated and translated. A user interface is provided to do the scaling interactively, numerically or to fit the the objects to the frame containing them. The latter provides one way to force a very complex graphic to have a specific total size.

Optional gravity between objects permits the guaranteed meeting of objects on all output devices. Internally, all diagramming objects are stored at a resolution smaller than that of any conceivable output device. All internal computations are done at this resolution, and if two objects are brought near one another on the screen with gravity turned on they will be made to touch in the high resolution plane, insuring that they do so even on a device of greater resolution than the screen.

Other user-setable methods of controlling the size and placement of objects include alignment to a (usually invisible) grid and pairwise alignment of objects on their tops, bottoms, left or right edges and centers (left-right, top-bottom, or both). As with gravity, this alignment is calculated in the high resolution plane. Thus, for example, circles made concentric by aligning their centers will be concentric on all output devices.

Perhaps one of the most novel ideas of the diagramming system is a set of monadic constraints called *locks*. These simple constraints modify, for the selected object, the way in which a given transformation is applied or property changed. In the simplest case, a lock can *prohibit* the change from being applied. Many of the most useful applications of locks involve locking an object before it is grouped with another (the locks do not become constraints on the group as a whole). For example, if one wishes to make arrows of varying length and orientation, it is unlikely that as the arrow is transformed it is desireable to change the size of the arrow head, but rather only the shaft. The IPS figure which forms the arrowhead can have its size locked before grouping it with the shaft. Subsequent transformations of the arrow as a whole will then have the desired effect. Template objects can be kept in a document for replication, but locked against printing so that the template does not appear in the final document (however, in practice, cut-and-paste across multiple open documents tends to encourage one to keep graphics templates in a separate document, to be opened when working). A more subtle and powerful lock constrains an object's control point to be dominant in a group. For example, in Figure 3a below shows the normal default rotation of a simple group around its centroid. In Figure 3b, the lower shaded circle has its center control point locked, which requires it to become the center of any rotation command

applied to the entire group.



(a) Shaded group normally rotates about its centroid

(b) Lower circle has center control point locked, constraining shaded group to rotate around it

Figure 3. Control locks

The diagramming system is implemented internally with a Smalltalk style message passing system [4]. This makes it relatively straightforward to add either new objects or new capabilities to the diagramming system, even if some objects cannot respond to all messages. For example, data driven chart objects (described below) do not respond to rotation messages. They do however respond to requests to scale or translate themselves. A response to a request to edit their properties is answered by invoking the chart editing subsystem, described in detail later. Similar behavior is exhibited by the image objects, which, however, respond to more messages than the charts. Although invisible to the user, this implementation meshes well with the user interface. In response to the user's selection of an action to apply to a selected object, the diagramming system constructs and sends a message to that object designed to insure the intended effect.

Such a message passing system insulates the user interface of the high level graphics system from details of the subsystem implementations, including IO, which can then be maintained and enhanced without further development requirements being made on the higher level software. For example, each subsystem is responsible for being able to deal with objects in documents which were created with older versions of the subsystem.

Figure 4 shows composite diagramming objects. Each was created by interactively.

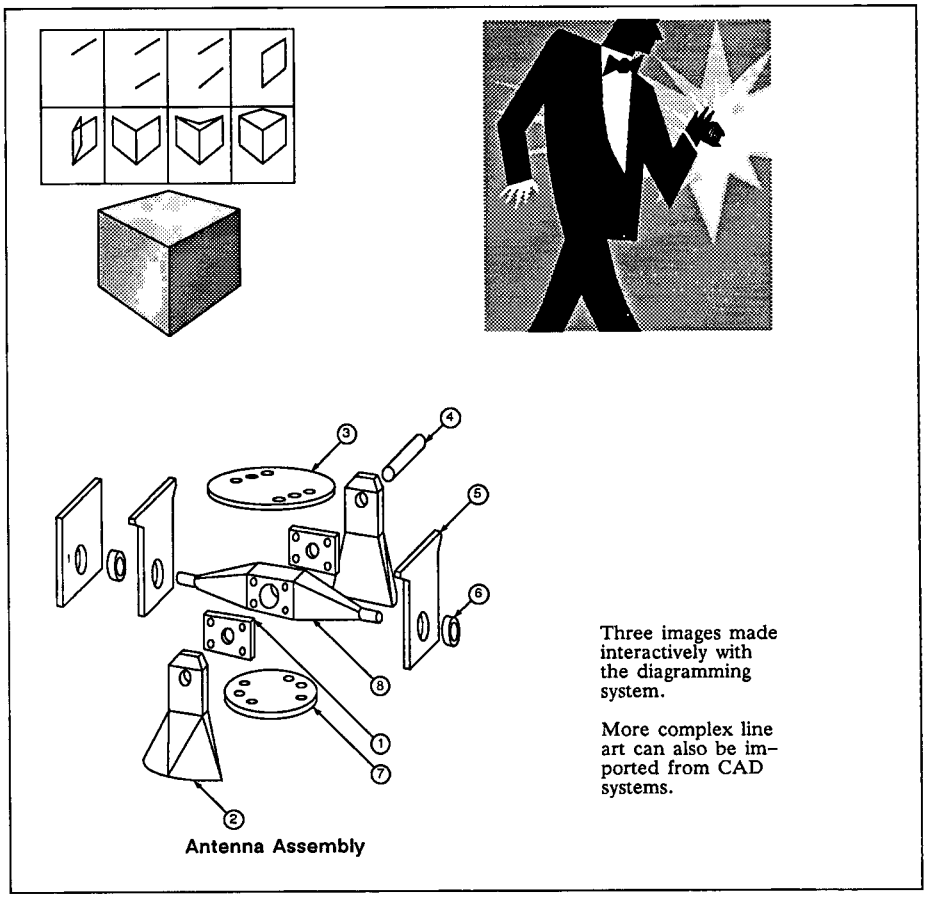


Figure 4. Diagramming images

(b) The Image Subsystem

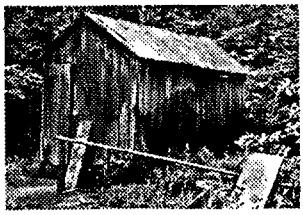
IPS supports several different kinds of bit mapped images, usually derived from external sources, such as a digital scanner. The present systems reflect the needs of document producers to input images from photographs or line art produced by conventional means.

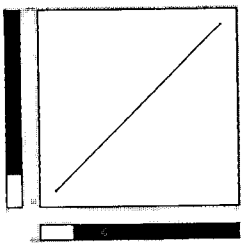
Two types of images are presently dealt with: line art and continuous tone images (contones). All images can be scaled, rotated and translated without entering the image editing subsystem. Images are internally stored at their input resolution and extremely high performance routines apply the transform to the (usually higher than display reso-

lution) input resolution image at print time and to the screen resolution image at display time. As with the diagramming system, this scheme insures that no information is lost before passing to the higher resolution output device. High performance image manipulation software is applied to do the transforms: a 300x300 pixel image can be rotated on the screen in about 5 seconds.

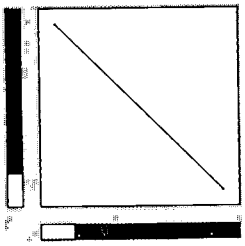
Selecting an image object and choosing the Edit operation causes one to enter the image editor. A blowup of the image is displayed at input resolution and the user can paint on the image with a small collection of black or white brushes. In addition, the viewport through which the image is viewed can be translated or scaled on the image (but it always remains a rectangle parallel to the page edges). Thus the image editor modifies the properties of an image at the pixel level, but also some of its display attributes.

For contones, more sophisticated control is offered. On black and white terminals and printers, gray scale is simulated by a screening mechanism, and the transfer function can be controlled by the user in a simple fashion: a graph of the transfer function is manipulated exactly as if it were a diagramming object. Some examples are shown in Figure 5 with a screen dump of the editing window state which gave rise to them.

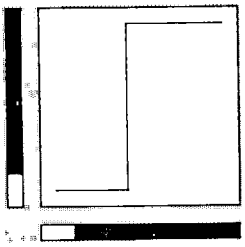




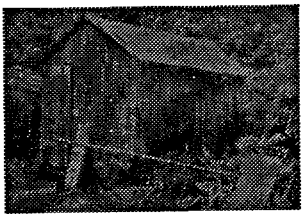
The original image

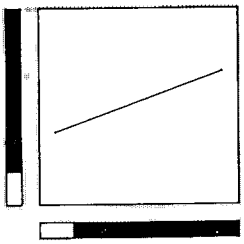


Negative



Two tone posterization





Reduced contrast

Figure 5. Effect on the screening process of manipulating transfer function

Image objects were introduced into IPS only in release 2.0, about 1 year after the first commercial release and are a prime example of how sophisticated graphic subsystems can be added to IPS without perturbing the general graphics interface.

(c) Charts

A chart is a graphics object driven by data which the user can enter at the keyboard or obtain through a number of filters which accept data from other programs such as spreadsheets.

As with image objects, charts appear in frames and respond to requests to modify their properties. At this level, only requests to scale, translate, cut or duplicate a chart are honored. When a request is made to edit a chart, the system invokes a property sheet based chart editing subsystem, which permits modification to the style, data or display attributes of a chart. Figure 6 shows one such property sheet.

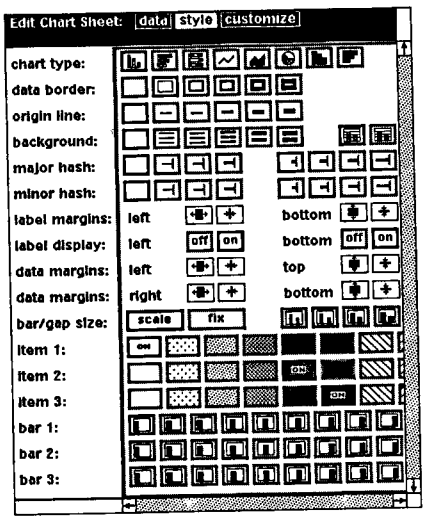


Figure 6. Chart style property sheet

The chart style property sheet is largely iconic, with choices representing the chart type (bar, line, etc.) and the presentation (e.g., the nature of hashmarks, borders, background lines). As with other property sheets in IPS, indicating selections causes these choices to pend until an *Apply* command is given (since this is the default action in a property sheet, pressing and releasing the middle mouse button within 150 ms. will cause it to happen with no further popup menu display). The entire chart display is recomputed with the new style or data (data are manipulated with a second property

sheet invoked by selecting the data box at the top of the property sheet). Generally this process, including the redisplay, is complete within 1-2 seconds.

The principal design considerations of the chart subsystem were responsiveness, flexibility and rational defaults. For example, by default a chart will rescale its axes when data or style changes imply it should do so, but a chart user can control this and force a specific range to be displayed.

As with the diagramming primitives, the chart system has a very small number of primitive chart types, but control of their imaging gives the user a wealth of complex charts. For eaxample, the charts shown in Figure 7 are internally identical, and differ only in which data are displayed, whether the lines are displayed and whether marks are placed at the data points.

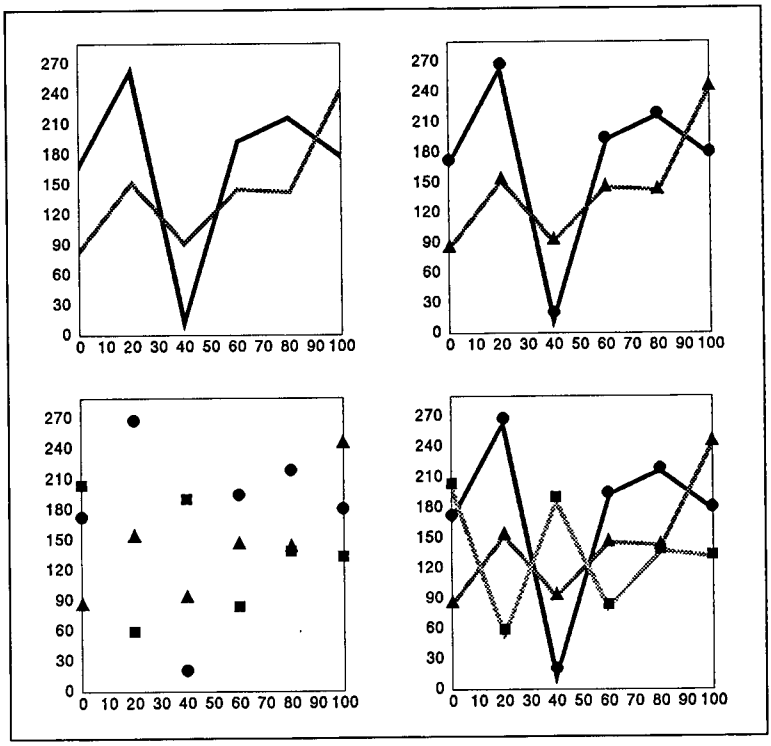


Figure 7. Charts with identical internal structure but different display attributes

Similarly, a bar chart and a histogram are not fundamentally different charts—they differ only in their display attributes.

As with the diagramming system, a large number of template charts are provided in documents distributed with the system.

VII. Integration

IPS aims for a high level of integration of text and graphics objects. From the view of the composition software, these are totally integrated by virtue of the frame system because the line breaker and paginator are ignorant of the contents of a frame. The consistentcy and power of the user interface across the entire system leaves subsystem implementors free to focus on details unique to their application.

Frames may be in text or may form an entire component. Text may occur in frames, as may each kind of graphics object. However, the three types of graphics objects are not presently tightly coupled but are evolving towards greater cooperation. Thus, it is possible to overlay a chart or image with a diagramming object in order to add graphics not generated by those subsystems. However, it is not presently possible to decompose a chart object into primitive diagramming objects, which would then be able to be manipulated without regard to the data from which they arose. Nor can diagramming objects be decomposed into a collection of pixels for editing with the image system.* Such transformations, in which the semantic content of graphics objects is progressively reduced, could give a graphic artist finer control than is presently available without sacrificing the speed and ease with which sophisticated graphics can presently be produced in IPS.

The most consequential lack of cooperation between subsystems of IPS is the absence of the text composition and editing software within a frame. Only the simplest formatting and editing is supported. There is no justification of text within a graphics frame, and text strings are nearly atomic. One major addition, scheduled for the next commercial release of IPS, due in 1986, is to make text in frames exactly the same kind of object as text in a document. It would then be subject to all operations now or ultimately embedded in IPS, including rapid editing, hyphenation and justification, and the cross-reference and index features also planned for the next release.

The central success of IPS's integration has been the presentation of a single user interface and the fact that a user need not invoke various pieces of software to produce a document with both text and graphics. Any changes to either are immediately reflected in the entire document.

*However, a screen capture facility is provided for Interleaf documentors. Using this, IPS can be interrupted and a section of the screen designated for capture. The resulting bitmap is automatically placed on the system clipboard, so that the next paste operation will add it to the document. The menu and property sheet examples in this paper were produced in this way, whereas all other graphics are at full output resolution on any output device.

VIII. Speed

It is now commonly accepted that immediate feedback, together with a usable *undo* facility, is a distinct aid in the learning of a complex system. Undoubtedly there is debate about the extent to which quality and sophistication can be compromised in the search for speed. As with the choice of programming languages, editors, spread sheets or any tools for productivity enhancement, these arguments turn on personal preference and the problem at hand, but IPS main usefulness comes from its continuous composition: a user always knows exactly what the document will look like when printed. IPS users report that this alone makes their document turnaround time substantially faster than with batch pagination systems.

IPS reflects an evolutionary approach to the details of these issues, largely market driven. For example, the current release has little "document control" or semantic control. Thus, there is not presently any automatic cross-referencing or indexing facility nor provision for subdocuments by reference. Some of the omitted facilities are inherently slow (these three are not, but line breaking optimized over the entire paragraph, discussed below, may be), but none seem necessarily contrary to the design choices of the current implementation of IPS or to the philosophy of incremental computation and display which IPS attempts internally. The IPS implementors believe that no other existing system is as highly integrated or as fast on the present generation of workstations.

Some speeds attained by IPS are susbstantially faster than existing batch composition systems, whose time is often measured in seconds per page. Under optimal conditions (which generally occur during typing), IPS composes as fast as 10-15 *pages per second.* Even in situations where a drastic amount of data must be examined, such as a global font change, speeds are still high. In the current document, changing all paragraphs from 12 to 10 point type required about 8 seconds before control was returned to the user.

IX. Implementation History

At this writing about 25 programmer-years have been invested in the development of IPS and it consists of about 210,000 lines of code, almost all in C.

The order in which features have been implemented is largely determined by market considerations. Originally, IPS was conceived principally as an office publication tool, but in the 12 months preceding this writing a substantial number of large installations have been made in technical publications facilities. This has accelerated typesetter development, more sophisticated composition (such as multi-column setting) and table and equation makeup.

For the first 6-8 months of IPS development, fewer than 6 people worked on the software and some of the scheduling was influenced by the search for venture capital. Features which were comprehensible to investors, such as the business chart and word processing capabilities, were developed first. The diagramming system, although in design earlier, was only implemented when the company was well financed for its initial development. One consequence of this is that the chart and the diagramming system evolved independently and were merged after their completion. This accounted for their loose coupling, and also meant that there was replication of some display handling functionality.

Since the commercial release of the IPS system, the development effort continues to strive for greater integration. Also, independent efforts are in place to transport the code to other systems (see *Portability* below) and when importable code is found in these efforts it is either added to the small list of functions required to be ported, or "repaired" in the main code.

X. Portability

Portability is a central consideration in coding. The system runs on all Sun and Apollo workstations and on the VaxStation I and II of Digital Equipment Corp., as well as on workstations made by Cadmus Computer Corp. and PCS GmBH. Several unannounced implementations are also in progress.

In order to promote portability, a small collection of primitive functions is used by each application in the system to insulate the software from machine dependencies. These are divided into four groups: (a) graphics display primitives, (b) operating system abstractions, (c) high precision arithmetic and, (d) avoidance of some miscellaneous but common C compiler deficiencies.

The graphics primitives are few and of an obvious nature. They draw and fill lines, arcs, polygons, etc. It is Interleaf's experience that none of the more general vendor-provided graphics packages give sufficient speed. It is necessary to touch the hardware directly. For example, the programs of [7] provide a benchmark for vector drawing speed using various of Sun Microsystems's standard graphics packages. The benchmark accounts for its own overhead, and in essence measures that of the vector drawing routines used, together with their raw pixel painting speed. For large vectors (1024 pixels long when horizontal), there is little difference between the benchmark run with Sun's window graphics package (pw_vector()). The Sun package averaged 104 vectors/second on the author's Sun 2/120 and the corresponding Interleaf software averaged 107 (the benchmark draws vectors at 20 different angles, in five degree increments). But the Interleaf software has a nearly linear speedup in vector size: vectors of 1/20th this length (51 pixels long when horizontal) are painted at 1623 per/second with the Interleaf routines called by the benchmark, whereas with the Sun routines these vectors

are painted at 297 vectors/second, only 3 times faster than the vectors which are 20 times longer. From this one can conclude that the startup overhead of the Sun vector painters is consequential. Indeed, a prototype implementation of the Interleaf graphics library using the Sun pixrect software proved intolerably slow. Other vendors doubtless do as poorly. There is simply too much generality in these packages.

Operating system abstractions are mostly those of file systems. The object containers mentioned earlier must be mapped into the host file system and provision made for creating, copying, renaming and deleting them. To the user, the desktop must not appear or behave differently from one platform to another.

In addition to the file system abstractions, a simple hook is provided to the host print spooling system. IPS invokes this hook, passing to the spooler a file to be printed on the laser printer or typesetter, together with a small amount of ancillary information to assist the host spooler in identifying the target printer. Each implementation must construct any additional information required to induce the host to print the resulting file on the desired printer.

Finally, a low level mouse and keyboard handler must be re-implemented on each system. Since keyboard and mouse must be asynchronously serviced, the minimum requirement of the operating system is that it either be able to provide non-blocking input, in which case the implementation polls the devices, or software interrupts. In either case, the system performs idle time tasks between servicing events. For the most part, these presently comprise updating the visible portion of the desktop to reflect any changes in the file system from outside IPS, such as the arrival of a document from another host. They could, of course, easily be expanded to service other such asynchronous events, such as mail arrival or the display of a time of day clock on the screen. Indeed, some such facilities run experimentally at Interleaf.

The third class of non-portable code consists of multi-precision arithmetic (this usually hinges on byte and word order questions) and some miscellaneous routines such as high speed memory-to-memory block transfers.

A few small routines overcome compiler deficiencies where they occur. For example, the formal definition of arithmetic in C permits the multiplication of 16 bit short integers to be carried out not merely *as if* by, but *in fact* by first converting to 32 bit integers and reconverting after the arithmetic. This is grossly inefficient on Motorola 68010 based hardware which has no 32 bit multiply, but does have a 16x16 multiply which, however, is not necessarily used by all compilers. Code writers dealing with such multiplication use a preprocessor macro *Muls(a,b)* which is expanded to a*b on systems where reasonable arithmetic is done, but otherwise calls a small assembly language program (or inserts inline assembly language if the compiler accepts it) which is host dependent.

XI. Enhancements

As mentioned earlier, some omissions from IPS reflect marketing decisions which, in turn, influence the allocation of engineering resources.

Among those enhancements already in design or under construction are: improved composition; more document maintainence capability: indexing, auto-referencing and auto-numbering; full text processing and editing in graphics; better integration of the graphics subsystems; table and mathematics makeup. Among the additional features being added to the image processing subsystem are: zooming, geometrical objects, local contrast control, airbrushing, user definable halftone screens, support of gray scale terminals, and color (including the production of color separations).

Some problems require more careful consideration to determine their desirability. For example, the effects of backward rippling line breaking, such as would occur with schemes like Knuth's, could be disconcerting to users. Indeed, even forward rippling proves annoying to some users, but most typing is done at the end of a paragraph and so such rippling is rarely experienced. In addition, while the average running time is quite reasonable for the dynamic program algorithms of Knuth, the worst case is not. Some argue that *consistency* of response is as least as important for users as speed, and it is appropriate to investigate whether variation in behavior, say, between large and small paragraphs, is unacceptable. Many interactive systems solve this problem by use of a "paginate key," whereby text is added to the page, lengthening it like a printer's galley, until the user requests explicit repagination. Of course, this gives up the valuable cpu time available between keystrokes.

Ability to run on cheaper platforms is desired by some. It might be appropriate to consider whether a subset of the capabilties described here, e.g., text only, might be suitable.

Document inclusion by reference is a useful feature in an environment where large numbers of large documents are being maintained.

Of perhaps greater difficulty is more sophisticated page makeup, such as runarounds and other non-linear makeup frequently encountered in advertising graphics.

Finally, there is a small class of problems whose solution may be slow for display reasons. Most interesting among these is to be able to display rotated text in real time at any angle. Since this can require scan conversion on demand of fonts from outlines, one faces issues both of quality (screen resolutions are not high enough to display small type faces faithfully at arbitrary angles) and speed. On the other hand, many of these applications are for large characters, again such as in advertising layout, where character presentation time may be dominated by filling, not by finding the edges of

the character outline. It might be reasonable to provide such a facility for large type and a few specific common rotations for small type.

Acknowledgements

RIPrint is a trademark of Interleaf Inc. Interpress is a trademark of Xerox Corp. PostScript is a trademark of Adobe Systems, Inc. VaxStation is a trademark of Digital Equipment Corp. The images of Figure 5 are derived from a standard test photograph of the Graphics Arts Technical Foundation which is used with their permission.

This document was prepared on an Interleaf Technical Publishing System and printed on a RIPrint 12 dot/mm (300 dpi) laser printer. The final copy was photographically reduced about 15%.

Bibliography

[1] - Peter R. Bono, "A Survey of Graphics Standards and Their Role in Information Interchange," *IEEE Computer*, Vol. 18, No. 10., 1985, pp. 63-75.

[2] - J.D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA., 1982.

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

[4] - Goldberg, Adele and Robson, David, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, Reading, MA 1983.

[5] - M. Hammer, R. Ilson, T. Anderson, E.J. Gilbert, B. Niamir, L. Rosenstein, and S. Schoichet, "The Implementation of Etude, An Integrated and Interactive Document Production System", in *Proc. ACM SIGPLAN/SIGOA Conference on Text Manipulation*, Portland, Oregon, June 1981, pp. 137-146.

[6] - Wolfgang Horak, "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standarization", *IEEE Computer*, Vol. 18, No. 10., 1985, pp. 50-59. This article describes Standard 101 of the European Computer Manufacturers Association, available as ECMA/TC29/85/16.

[7] - Robert L. Judd, "Benchmarking Workstations", short course notes, PROCEEDINGS, ACM SIGGRAPH '85.

[8] - Donald Knuth, *TeX and MetaFont*, Digital Press and the American Mathematical Society, 1979. (Note that for TeX this book is obsoleted by Knuth's *The TeX Book*, Addison-Wesley, Reading MA, 1984).

[9] - Donald Knuth and Michael Plass, *Breaking paragraphs into lines*, Software Practice and Experience, Nov. 1981, pp. 1119-1184.

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

[11] - Rob Pike, "Graphics in Overlapping Bitmap Layers", *ACM Transactions on Graphics*, Vol. 2, No. 2, 1985, pp. 135-160.

[12] - Edmund Van Dusen, *Graphics Standards Handbook*, CC Exchange, P.O. Box 1251, Laguna Beach CA., 1985