This document is for Xerox internal use only

A Methodology for User Interface Design

by Charles Irby and Linda Bergsteinsson, ITG Systems Development Division
Thomas Moran, William Newman and Larry Tesler, Palo Alto Research Center

JANUARY 1977

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

INFORMATION TECHNOLOGY GROUP
SYSTEMS DEVELOPMENT DIVISION
3406 Hillview Avenue / Palo Alto / California 94304

This document is for Xerox internal use only

Preface

In April 1976, a working group was formed at the Xerox Palo Alto facility to study the problem of user interface design. The group was led by Charles Irby, and included Linda Bergsteinsson, Tom Moran, William Newman and Larry Tesler as its other full-time members. The group met on a weekly basis throughout the spring and summer of 1976. David Liddle and Butler Lampson attended some of the early meetings.

User interface design is an essential part of the design of interactive application programs. The user interface encompasses the set of commands by which the user operates the program, the display techniques used to present the state of the program to him, and the more abstract notions involved in learning how to operate the program effectively. Successful user interface design is crucial to the production of useful Office Information Systems.

The working group decided at one of its earliest meetings that it could not attempt to propose an actual design for an OIS user interface. The reasons for this lie in the complex nature of user interface design, and the difficulty of performing the design without a thorough analysis of the task. These problems are discussed further in the main report.

The group therefore concentrated its efforts on developing a straightforward methodology for user interface design. This methodology is presented in the report. Included with the report is a detailed description of a user interface developed along the lines suggested.

The working group will now turn its attention to interaction with the application designers and subsequent design of the user interface facilities for OIS.

Acknowledgement

We wish to thank the many people who provided useful comments and suggestions about this report, in particular D. DeSantis, A. Kay, R. Kimball, T. Mott, R. Sproull and R. Taylor.

TABLE OF CONTENTS

Introduction

One of the most troublesome and least understood aspects of interactive systems is the *user interface.* In the design of user interfaces, we are concerned with several issues: the provision of languages by which users can express their commands to the computer; the design of display representations that show the state of the system to the user; and other more abstract issues that affect the user's understanding of the system's behavior. Many of these issues are highly subjective, and are therefore often addressed in an *ad hoc* fashion. We believe, however, that more rigorous approaches to user interface design can be developed; we present such an approach in this report.

In preparing this document, our main concern has been to propose a *methodology* for user interface design. It is based primarily on the separation of user interface design into a small number of separate design phases, to be applied iteratively in order to reach a satisfactory design. For each of these phases we have developed *representations* in which to present the concepts involved; these representations help both in formalizing the design process, and in defining the user interface design for purposes of documentation and communication with others. We have included a number of *guidelines* that we feel may be useful to those designing user interfaces. To illustrate the use of our methodology, we have included a partial description of a user interface design for a simple interactive page layout system.

The main report is a fairly concise statement of our proposals, with only a few examples included to illustrate specific points. The user interface for page layout is described in the Appendix. Marginal notes are included with this description, to explain some of the issues that arose in the user interface design. The reader should note that this is not a complete user interface specification: a number of detailed design issues have been left unresolved, and some of the more repetitive passages have been curtailed. We hope that the reader will not be confused or misled by these omissions.

Task Analysis

In a subject as little understood as user interface design, it is not surprising to encounter many different methodologies. Some of these scarcely deserve to be called methodologies: an example is the very common practice of studying competitors'

products, making a list of what appear to be their most effective features, and then planning the new design from this list. Another approach is to proceed by inventing user functions one by one, implementing each function together with a command to invoke it. Both of these methods tend to produce inconsistent user interfaces. A more sensible approach, still unsatisfactory, is to implement a complete set of functions and then to try to design a consistent command language: the resulting language may indeed be consistent, but will probably involve unsuitable concepts and abstractions that the user cannot understand.

These design methodologies are all unsatisfactory for the same basic reason: they all omit an essential step that must precede the design of any successful user interface, namely *task analysis*. By this we mean the analysis of the task performed by the user, or users, prior to introducing the proposed computer system. Task analysis involves establishing who the users are, what their goals are in performing the task, what information they use in performing it, what information they generate, and what methods they employ. The descriptions of input and output information should include an analysis of the various *objects*, or individual types of information entities, employed by the user.

Analysis of a simple text-editing task, for example, will typically indicate the involvement of two people, an author and a typist. The goal of the author is to produce a set of corrections to the current draft of a document, while the typist's goal is to follow the instructions implied by these corrections, in order to generate a fresh draft. Thus the list of corrections (probably written on the current draft) is the author's output; this list, in conjunction with the current draft, forms the typist's input. Conversely, the output from the typist is the fresh draft, and this forms the input for the author's next editing task. In each case, the principal information objects are *pages* of text, arranged in *paragraphs*, each paragraph consisting of *lines* of text made up of *words*, each word consisting of text *characters*. The typist's input objects also include *corrections*, in the form of a set of marks relating to entities in the draft. The method employed by the author is to read the current draft, marking his corrections; the typist's method is to copy text from the current draft to the fresh one, performing each correction as encountered.

Task analysis thus leads to what we term the *current task description*. Its purpose is to define current procedures, informational objects and their inter-relationships, and terminology. It serves as a checkpoint against which further design steps can be measured. The Appendix includes the current task description for a somewhat more complex task than text editing, taken to considerably greater detail. In time, the form of the current task description may evolve to a yet more useful form.

The Three Components of the User Interface

The purpose of task analysis is to direct the remaining stages in user interface design. The current task description, with its breakdown of the information objects and methods presently employed, offers a starting point for the definition of a corresponding set of objects and methods to be provided by the computer system. The idea behind this phase of design is to build up a new *task environment* for the user, in which he can work to accomplish the same goals as before, surrounded now

by a different set of objects, and employing new methods. The new task environment can be described much as before, by means of a *new task description.* In the generation of this new task description, the three basic components of the user interface must be considered. These three components are:

1. The user's *conceptual model* (generally abbreviated to *user's model*) of the system with which he is interacting and how it relates to the tasks described in the current task description;
2. The *command language* that he uses in order to express commands to the system;
3. The *information display* used by the system to present the results of executing commands, and to show the state of the system in general.

We consider the user's model to be the most important single component of the user interface. This is the user's set of concepts, or abstractions, of the *objects* he is manipulating with the aid of the system, and of the processes, or *actions*, that he can apply to these data. This set of objects and actions is crucial in enabling the user to anticipate the behavior of the system. In general, the choice of abstractions for objects and actions to a large extent determines the actual data structures and procedures used in implementing the application program. It is important, however, to design these abstractions with the user's needs in mind; otherwise they may turn out to be inconsistent or incomprehensible.

The command language provides the user with a set of physical actions to use in performing the conceptual actions of the user's model. Command language design can be viewed as the problem of providing the means to apply each of these conceptual actions. In reality, however, command language design also involves a great deal of careful and detailed work on command syntax. It differs from user model design in that it is greatly influenced by the physical environment in which the system will be used. It also interacts strongly with the third component of the user interface, information display.

Information display, like the command language, is concerned with providing the user with a more concrete representation of the user's model; in this case, the objects of the user's model are furnished with visible representations. Information display interacts with the command language, in the sense that command language design involves a number of considerations of output effects. In particular, interactive command languages depend on *feedback* to reassure the user that his actions are being understood, and to prevent him wherever possible from taking a wrong step. Most display-oriented command languages also make use of the display of *control information:* cursor symbols, command menus, window boundaries, and so forth. The remaining class of information display, *data presentation*, is concerned purely with showing the user the state of the information he is manipulating.

In order to avoid a fragmented approach to information display, we distinguish between the *intrinsic objects* essential to the application, and *control objects*, such as menus and windows, used to control user actions. This concept allows us to treat both control information display and data presentation in a uniform way, as the display of objects.

The Design Process

In this section we describe in detail the steps involved in developing the user interface. This is an iterative process, in which the designer must return time and again to earlier phases of the process after encountering snags in the later phases. Thus the sequence of steps we describe here is best viewed as a framework for the actual steps followed in design.

1. Task Analysis

Task analysis is the process in which a task description is developed. This analysis attempts to describe the *current task* as performed prior to introduction of the computer system:

1. Definitions are given, itemizing each of the important components involved in the task, and the terms used to describe them;
2. First-level tasks are identified. Typically each of these represents a portion of the entire task domain performed by a single person; in some cases, it may be useful to divide up the duties of each person into several of these tasks.
3. Under each major task heading, descriptions are drawn up of the different entities involved, as follows:

   Agents: i.e. the person, persons or machines performing the task;
   Goals: the rationale behind the task, and the criteria that determine its completion;
   Inputs: the items of information required in order to perform the task;
   Outputs: the items of information generated by performing the task;
   Methods: the procedures employed to get the task done; each method identifies a series of *subtasks* that are then defined by recursively applying step 3, until a sufficiently fine level of detail is reached.

2. The User's Model

The user's model is based on a set of conceptual objects, and on a corresponding set of actions that may be performed on these objects. From the user's point of view, the objects are abstract entities that behave in a consistent and predictable way when the actions are performed on them. The designer, on the other hand, may find it useful to consider objects to be items of data structure in the user's domain: corresponding to each object seen by the user, there will generally be an item in the data structure maintained by the system. Design of the user's model must steer a careful path between the often conflicting requirements of the user and the system designer.

We suggest the following guidelines for user's model design:

Completeness: if a concept, or set of concepts, is introduced to the user's model, it should be supported thoroughly and convincingly.

Consistency: the behavior of the system should be consistent, whatever the context

of use. Objects should not exhibit different behavior at different times for no apparent reason, and actions should have consistent effects.

Use of concrete objects: it is sometimes helpful to base the choice of objects in the user's model on concrete examples. It would be appropriate, for example, to allow the user of a page layout system to manipulate the same words, text lines and paragraphs that he would lay out manually. While this can be a useful guideline in choosing a user's model, we feel it can also be limiting.

3. The New Task Environment

A description of the user's model forms the starting point for a description of the new, computer-based task. Many features of this description bear close resemblance to corresponding parts of the manual task description completed earlier. The purpose of this description is different, however: it is the foundation upon which the detailed command language design is based. Descriptions of objects and actions are therefore taken to a level of detail that will support direct development of the command language. The actual task description is preceded by a section describing each object and action in some detail.

The representation used for the current task description can be used almost unchanged for the new task. The main differences are (a) a complete list of objects and actions should be provided, with control objects included; and (b) tasks are broken down into subtasks until they reach the level of actions, so as to provide a basis for command language design.

4. Information Display

Information display is on the whole a poorly documented area. As we have mentioned earlier, it encompasses three topics: data presentation, feedback and control information display. The last two topics relate closely to command language design.

Data presentation is the means by which the system presents to the user a view of the objects he is manipulating. The purpose of data presentation is twofold: it conveys to the user the effects of his previous actions; and it enables him to plan his next move. The design of data presentation techniques should take both of these into account. To some extent, this raises conflicting requirements: in order to show the effects of past actions, data presentation techniques should show the state of the user's model in an economical and consistent fashion; to help in planning the next action, they should highlight the information on which this action is likely to operate. We suggest that any such conflict be resolved in favor of overall consistency.

The design of data presentation methods is considerably easier when the user's model is based on concrete objects. In text-editing and formatting applications, for example, it is obvious that text should generally be displayed in the form of character symbols -- anything else would require the user to make a mental transformation from the displayed data to the text itself. In applications such as

information retrieval, however, there may be no obvious method of displaying the data, and the designer must choose among several alternatives. When it is not possible to portray all aspects of an object accurately, the designer must decide which aspects are the most important.

Guidelines for data presentation include the following:

Faithfulness: the display should show as faithfully as possible the true state of the stored data. Often the data cannot be shown absolutely faithfully because of poor display resolution; this creates a problem well-known in text editor design -- "what you get" when printing the text is not "what you see" on the screen. A related problem is that of "cleaning up" the screen after one or more deletions: in cases where this is not done immediately, the user should be made fully aware that the screen shows anomalies.

Economy: in applications such as information retrieval, it is very valuable to perform some analysis of the data to be displayed, and to try to isolate the different attributes that are to be shown. This will help in the design of an economical and effective display. Bertin [1] has suggested some further guidelines to the use of symbol shapes, line qualities, etc.

Control information display is concerned with the *control objects* used in operating the program. Many of the same techniques used in data presentation design are useful here. Guidelines include:

Physical convenience: the control objects should lie close at hand. Command menus, if used, should not be placed at the extreme edge of the screen where they can be reached only with excessive hand movement.

Selections should be visible: if commands operate on a *selected object*, then the user should always be able to find that object easily.

Feedback is distinguished from other forms of information display in that it relates much more closely to actions than it does to objects, and in particular is generally very closely coupled to the actual commands of the command language. Feedback is used wherever it can help the user to give his commands more quickly and accurately. Guidelines include:

Economy: feedback should show only the minimum information to permit the command to be given. Feedback should not be used where it is not needed: one should be sparing in the use of dynamic effects like dragging, rubber-band lines and zooming.

Smooth feedback: where the action requiring feedback is a smooth one, the feedback should be smooth; this is particularly true of pointer movement. The user should be able to easily follow the transformations that are taking place on the display.

5. The Command Language

The final section of the user interface specification is a description of the command language. We suggest the use of a relatively formal command language description, organized in the same order as the lowest-level subtasks. A suggested syntax for this formal description is given in the Appendix. It is also suggested that the designer compile cross references of user input and output: these list the effects of each input, and the meaning of each style of output, and help to expose ambiguities and sources of confusion. It should be obvious that command language descriptions are aided by the use of illustrations, figures and examples.

Command language design involves decisions at two levels: the global level, at which the choices are made between alternatives such as menus and function buttons, prefix or postfix operators, and so forth; and at the level of individual commands, where detailed design can be extremely important. Guidelines for command language design include:

Consistency: function keys should retain their meaning throughout the system if possible, and if not should have analogous meanings (e.g. an "insert" key that adds text or dispenses ink).

Minimum effort: the most frequently used commands should be easier to apply, and should not require great dexterity.

Safeguards: dangerous commands should require confirmation from the user, and should use keys or menu symbols that are not easily confused with others.

Modes: command languages should avoid the use of several different command modes, in which different meanings are assigned to the same set of input actions. The mode of a command language is one additional concept that the user must understand; and even when understood, it is a frequent cause of erroneous user action.

Conclusion

The methodology proposed in this report involves the use of three levels of representations for ideas involved in the user interface: the Task Description, the User's Model, and the Command Language. We believe the use of such a three-stage approach to user interface design offers a number of advantages over other approaches. Some particular benefits are worthy of mention:

1. The designer is encouraged to focus his attention on the *user's view* of the task and of the system. An attempt is made to treat the user's view as a precise and specific concept in these representations, not just a metaphor.

2. The representations define the *state of the design*, both before and after completion. Thus they serve both as working documents and as design

specifications for program implementation. As working documents, they can provide several general benefits to the designer: they force completeness of the design for the given task, and they tend to encourage consistency and simplicity. As specifications, they communicate in a precise manner the intended design to the user manual writer, the system programmers, etc. They can also be helpful in coordinating the efforts of several designers.

3. The representations help to rationalize the design, i.e. to show the reasons for designing the user interface in a particular way.

4. The representations can serve as a framework for the development of design principles and for conducting user studies.

5. The representations potentially provide the basis for useful design analysis, involving such aspects as learning time, interference between commands, performance for various types of task, etc.

We have suggested a relatively formal syntax for user interface descriptions. We consider this appropriate where communication amongst designers is important. We do not recommend the use of such a syntax in the user's manual.

The methodology we propose implies different design activities at the different stages of design. We believe that user model design is the most "creative" aspect of the entire process; it must be preceded by careful task analysis, and followed by equally careful command language design. This in turn suggests the possibility that different kinds of professionals, each a specialist of sorts, be involved in the design process, working either as a team, or as consultants to a "systems architect" who has overall responsibility for the design.

References

1. Bertin, J., *Semiologie Graphique*, Paris: Mouton, 1967.

2. Newman, W. M., and Sproull, R. F., *Principles of Interactive Computer Graphics*, McGraw-Hill, 1973, Chapter 11.

APPENDIX:

A User Interface for Page Layout

Note to the reader

This Appendix contains a partial description of a user interface design for a simple interactive page layout system. It has been included to illustrate the use of the methodology described in the main report, and should not be viewed as a complete, polished user interface design. We would like to emphasize the importance of the *form* of this description, rather than its *content.*

Readers should note in particular that the Task Description sections have been abbreviated, since the Working Group could not afford the time to undertake a proper study of page layout. Also only the first part of the Command Language section is illustrated with the aid of figures; the last part of the description is printed in small type to indicate this distinction.

The marginal notes have been included to assist the reader in following the user interface description. They would not normally be included in such a document.

CURRENT TASK DESCRIPTION

THE LAYOUT TASK:

Definitions

The production of a book is under the control of two people: the *editor*, who is in charge of the contents of the text, and the *graphics designer*, in charge of all graphics work and spatial text layout. All decisions regarding the book's layout must be discussed and agreed upon by these two people. The *layout artist* does the page-by-page layout of the text and graphics; this may on occasion be done by the graphics designer himself, although it is usually done by someone working for the graphics designer.

A *galley* consists of a "scroll" of text lines, each *text line* being in its final form, i.e. in the correct type font, justified and hyphenated to the correct *width* between margins, and with the minimum *spacing* between lines. Each line in a galley has a sequential *line number* located to the left of the line margin. Text galleys also contain *instruction lines* which indicate such things as figure positions. Instruction lines do not include line numbers. For any one book there are several galleys: a galley for the main running text, including section headers, indented quotations, equations, etc.; a galley for footnotes; a galley for chapter titles; a galley for figure captions. Other galleys may exist, for such things as running text to be meshed with the main text in the layout.

A graphics *portfolio* contains a complete collection of the graphic illustrations (photographs, diagrams, drawings) for the book. Each *illustration* has an *identification number* and a *size specification* (the physical size of the illustration, a marked size, or some constraints on the final size).

A *pattern page* is a "blank" book page for layout. It contains rectangular outlines, called *layout blocks*, and *guidelines*, showing the layout page structure (e.g. how many columns, where the margins are, where to put page headings and page numbers). There may be several different kinds of pattern page for a book, usually not more than two or three.

The *layout instructions* are an informal set of documents that bring together all of the parts of the layout, and embody the basic layout decisions agreed upon at the outset by the editor and the graphics designer. The instructions spell out the *correspondences* between the galley texts and graphics (e.g. where to put the illustrations from the

This section describes the existing techniques and methodologies in use within the task domain. The principal tasks are defined, with subtasks derived from the methods used in the task. This section could be reviewed by customers and experts in the field to assure its accuracy.

A key part of this section is clarification of terms and concepts found in the current manual environment. These can be used later in the new system to reduce learning time for new users who are already familiar with the manual techniques and terminology.

Note that the new system chose to use Illustration Galley instead of Graphics Portfolio, since the illustrations do not appear on pages in the new system, but rather are scrolled.

portfolio in relation to the text in the galley, how to relate the text pieces in two or more galleys). These correspondences are given on the galleys when possible, but sometimes a separate document is needed. The instructions refer by line numbers to the galleys and by id numbers to the graphics. The instructions also provide some global layout conventions (e.g. how much white space to include per page, where figures should usually go), in a document called the *style sheet.* Usually, a few *sample pages* prepared by the graphics designer are also included as part of the instructions.

After a pattern page has been pasted up with text and graphics, it is called a *dummy page.* There are two kinds of dummy page: A precisely laid-out, clean, camera-ready dummy page is called a *mechanical,* while less exact dummy page is called a *rough.*

Agent: The layout artist.

Goals:

a) To assign to all text and graphic pieces that go into a book specific locations on pages in the book.
b) To compose each page of the book esthetically and according to given layout instructions.

Inputs:

a) A set of galleys.
b) A graphics portfolio.
c) A set of pattern pages.
d) The layout instructions.

The inputs indicate to the designer the set of objects that must be on hand in order for the user to perform his task.

Outputs:

a) A final set of mechanicals.
b) The galleys are all consumed in the process, as are the illustrations in the portfolio.

The outputs of the task are often the inputs to other tasks. This should be pointed out whenever possible.

Method:

a) Plan the layout of the whole book.
b) Paste-up a set of rough layout pages.
c) Submit the roughs for approval (or corrections or changes) to the editor and graphic designer.
d) Revise the rough layout, making all the above corrections and changes. If necessary, return to (c).
e) Make the mechanicals from the roughs.

This is the procedure followed by the agent in converting the inputs into outputs. The steps described here often form the basis of subtasks that expand the process further.

THE PLANNING SUBTASK:

Definitions:

A *layout plan* consists of an informal specification (markings on the galleys and illustrations) of the pages on which to put the illustrations and the different segments of the galley text. The layout plan sometimes includes a set of *thumbnail sketches* of the book's pages, showing the approximate layout of each page.

Goals:

a) To determine how many the book contains, and what goes on each page.

b) To anticipate any difficulties (violations of the layout instructions) in getting all the material onto pages.

Inputs: Same as in the layout task.

Output: A layout plan.

Method 1 (For simple books):

This method is used only for simple books with little or no graphics. Using the galley line numbers, simply scan down the galley, marking the page breaks.

It is sometimes necessary to describe several alternative methods. Care should be taken to explain when each is used.

Method 2 (The most common method):

a) Completely compile all the layout instructions by marking on each galley the points of correspondence to other galleys and to illustrations.

b) Proceeding sequentially through the pages, estimate by rough measurements the amount of galley text, and which illustrations to put on each page, and mark the cut lines on the galleys and the page numbers on the illustrations.

Method 3 (Usually employed only with graphically complex books):

a) Same as in Method 2.

b) Consider the pairs of facing pages sequentially, first draw a thumbnail sketch of the composition of the pages, then estimate the amount of galley text needed, and finally mark the galley cuts.

THE PASTE-UP SUBTASK:

Goals:

a) To compose each page, making sure that all the assigned pieces fit, and can be arranged esthetically.

b) To produce a set of roughs for approval.

Inputs:

a) The layout instructions.
b) The pattern pages.
c) The marked galleys and illustrations.
d) Possibly the thumbnail sketches.

Output: The roughs, plus possibly some suggestions to the editor for changes.

Method: For each page:

a) Cut out the segments from the galleys and the illustrations that go on this page and place them approximately on a pattern page.

b) Slide them around on the pattern page until all the instructions regarding the pieces are satisfied. If this cannot be done, and some pieces must be shuffled between pages, then determine just which pieces must be moved to make this page work, and go back to the planning subtask to readjust the layout plan.

c) Align all pieces with the appropriate blocks and guidelines, and with each other. Slide the pieces around until the arrangement is neat.

d) Paste all the pieces to the pattern page.

Note that the methods involved in these tasks may change in the new environment. Comparison of this section with the later New Task Description should show this clearly.

More details of the alignment problem should have been provided here. In addition, more quantitative measures of the number of cutouts being manipulated simultaneously would have been helpful.

Timing data for the steps involved would also help the designer understand what aspects of the task are most time-consuming. This could be expanded into an explicit statement of what the problems are with the current procedures -- why the computer system is desirable.

THE APPROVAL SUBTASK:

Agents: The graphics designer and the editor.

Input: The outputs from the paste-up subtask.

Outputs:

a) Possibly, the rough layout marked-up with corrections, or a list of changes to be made, keyed to the rough layout.

b) Possibly, an altered set of layout instructions.

c) Possibly, a new version of the galleys, galley segments or illustrations to be incorporated into the revised layout.

When it is known that some task areas are not to be automated, the description may be more sketchy. It is important to bring out the interdependencies between tasks, but details may often be omitted.

THE REVISION SUBTASK:

Goals:

To revise the rough layout according to the changes from the approval subtask.

Inputs:

The outputs from the approval subtask.

Method:

Similar to Paste-up except that new galley material replaces old pasted-down material on the pages. New pages may be added and old pages deleted as a result of the revision.

THE MECHANICAL LAYOUT SUBTASK:

Agent: A contract draftsman.

Inputs:

a) The pattern pages.
b) A clean set of galleys.
c) A clean graphics portfolio.
d) The roughs.

Output: The mechanicals.

USER'S CONCEPTUAL MODEL

SCOPE OF THIS USER'S MODEL:

A more precise statement of the scope is to be found in the New Task Description below. This scope section acts as an introduction to the model and attempts to provide the user with a perspective for reading the subsequent sections.

This user's model is intended to address the layout artist's task of creating a laid-out book from galleys, pattern pages, style sheets, and so forth (see Current Task Description section above). Little attention is paid here to the planning subtask. Furthermore, as was the case in the Current Task Description, the creation of author manuscripts, galleys and illustrations is completely outside the scope of this user's model. We also make the simplifying assumption that the task of alignment is so discretionary that only very basic facilities should be offered. We are omitting recovery mechanisms for catastrophic user errors, although we understand that any real system of this type would have to address them. Instead, we assume that the user has some way of obtaining new copies of galleys and can manually correct whatever has gone wrong.

The user's model does address the process of extracting blocks of material from galleys and illustrations and placing them onto pattern pages for the final book. However, no facilities are provided for rotating or scaling the material extracted from the galley, or for placing material in such a way that its textual or graphical contents extend beyond the boundaries of the pattern page or overlap visible, previously-placed material.

DEFINITION OF TERMS:

The following objects and actions form the foundation of the user's model. Care should be taken to make sure that they are complete and consistent.

Intrinsic Online Objects:

Intrinsic objects are analogous to the objects found in the current task environment. They are stored within the computer and are represented on the display screen in some fashion (see Data Presentation section). The presentation of these objects to the user often involves the use of control objects (see below).

Line of text - A contiguous string of characters that usually begins at the begining of a word and ends at the end of a word, possibly followed by some punctuation characters. The width of such a line, its contents, fonts, faces, and spacing are all determined in the creation of the line and are considered unchangeable in the layout process itself.

Book - A book is the finished product of the layout operation. It consists of some number of laid-out pages, and front and back *covers*. A newly created book consists only of the covers. The front cover contains information supplied by the user when the book was created (title, descriptive prose, date of creation, etc.); this descriptive information is used as the book's *identification* in a list of books and galleys (see Create Window below). The back cover contains a list of the pattern pages associated with this book, copied from the style sheet specified to Create Book (see below). Only the front cover may be modified by the user.

Object names are usually nouns. They are defined here only to the depth of detail needed to support the user's model. Additional detail is provided in later sections. Special care should be taken to avoid commitment to display representations in this section. Only aspects of object display that materially effect the user's model should be defined here (see Display Window below).

Page - A page is constructed by the layout artist during the layout process. In its final form, it is

part of the finished product. While in an incomplete form, it is called a dummy page (see below).

Illustration - This is a graphic representation of something that is meaningful when used in conjunction with the textual material of the book. For the purposes of our user's model, such an illustration has an identity of some sort (which indicates to the layout artist where it is to go within the book) and a bounding rectangle which is used in the alignment process.

Galley - A typeset version of the text of a book, adjusted for the correct width but with no page breaks; it may include notes indicating where illustrations are to be placed. A galley consists of lines of text, often organized into paragraphs. Each such line of text is numbered. Line numbers appear to the left of the lines of text. Several galleys may be used in the process of laying out a single page.

Illustration Galley - the illustrations for a book. Like the lines of text with a galley, the illustrations within the illustration galley are already of the proper size and orientation. If several copies of the same illustration are to appear in the book, then the appropriate number of copies will be supplied in the illustration galley.

Cutout - a rectangular portion of galley or illustration galley with the line numbers removed. Many cutouts may exist at one time. Cutouts may overlap each other. The boundaries of a cutout are clearly visible and the white space within a cutout is opaque relative to the display of other material which it overlays on the display screen. A cutout may be slid almost off the screen. However, some clearly visible portion of it must remain on the screen.

Pattern Page - a blank book page (created prior to the page layout task) with guidelines denoting the portions of the page to use for various purposes (e.g. the columnar boundaries for the body text, the location of the page number). A grid is also associated with the pattern page. Pasting and selections are constrained to the grid points. The horizontal resolution of the grid can be different from the vertical resolution. Neither the grid nor the guidelines shows on the finished printed page, but both are useful in the aligning material.

Dummy Page - pattern pages on which cutouts have been pasted. Dummies vary in accuracy from rough planning mockups to the finished page. Material that has been placed on a dummy page may subsequently be rearranged.

Style Sheet - A list of pattern pages to be used

Cross references are often helpful at this level. However, care should be taken that the material is still readable and comprehensible. The covers were invented to act as natural boundaries for the Turn Page actions and to allow a way of naming a book and associating a set of pattern pages with it. It might also be wise to point out in this description that the book is online, although there is a print command to create the more traditional paper form.

The line numbers are carried over from the Current Task Description as an aid to planning, which will generally involve marking up hardcopy of the galleys (which also have the line numbers). Recall that the line numbers help the user count lines in his planning subtask, which is not automated.

These display characteristics are a necessary part of the user's model and are therefore discussed here. More detailed information display discussion may be found in the Data Presentation and Scenario section and in the Terminal Description.

The Data Representation section points out that the grid does not display to the user but the guidelines do. It would be inappropriate to make such a statement here.

within a book. New books are defined in terms of the style sheet they will follow. When a new book is created from a style sheet, the contents of the style sheet are copied to the new book's back cover.

Control Objects:

Control objects are introduced to facilitate the manipulation of other objects via a display screen.

Display Window - A rectangular area of the user's display screen. Such areas are used to display textual and graphical information to the user from a book or a galley. Windows may overlap each other and behave as would overlapping pieces of paper on a table top, i.e. if two windows overlap, one is said to be *on top* of the other, and within the region of overlap only the contents of the window on top are visible.

The user may create any number of layout or galley windows. Multiple layout windows are useful for laying out facing pages and for propagating material from page to page, as seen in the New Task Description.

Selection - A portion of a galley or page that is being viewed in a window and is highlighted in some way as a result of a *Select Rectangular Area of Page or galley* action (see below). The *Cut Selection* action operates on the *current* selection (there is only one selection current at one time).

Selected Window - A layout or galley window that is to be used in one of several of the actions listed below. There is only one *current* layout or galley window. The current window always obscures the display of other windows it overlaps.

Actions:

Each action listed below should have a corresponding command in the command language section. The actions act on the objects defined above, and are grouped according to the type of object. There should probably be statements of response time requirements along with these actions.

Action definitions are in the form of imperatives. The definition should briefly describe the action's effect on the objects involved. Again, it should not specify what is seen on the display unless there is some visual property (e.g. that windows overlap) that strongly effects the user's model.

Note that these definitions should not include the keystrokes or menu items used to invoke the actions. The Command Language section serves this purpose.

Select Rectangular Area of Page or Galley - The specification of one or more lines of text, or one illustration, to be cut from a galley or illustration galley respectively, or from a Dummy Page. Each new selection replaces the old one; thus there is only one *current* selection.

Cut Selection - The act of cutting the current selection, removing it from its current location and making it a *cutout.* This might be thought of as *un-anchoring* that portion of text or illustration. If material is cut from a galley or illustration galley, the subsequent material within the galley will move up to fill the hole automatically.

Select Cutout - The act of selecting a cutout for sliding or pasting. While a cutout is selected for sliding, it will overlay the display of other material on the display screen.

Slide Cutout - the act of moving a cutout to a new position. Discontinuing the slide operation does not imply that the cutout is anchored to the pattern

page over which it may currently lie. It is, however, anchored to its current position on the display screen until the user slides it again or pastes it. As discussed above, a cutout may be slid partly off the display screen.

Paste Cutout - the act of anchoring a cutout to the location that it currently overlies within a pattern page. This operation terminates the existence of the cutout.

Split Cutout - The specified cutout is subdivided into two cutouts along a horizontal line specified by the user. If this split line lies within the bounding rectangle of an illustration, the split will occur above or below the illustration, whichever is closer. Once split, the cutouts may be manipulated separately and can be joined back together by first pasting them into the book and then cutting the whole region to form a new cutout.

Delete Cutout - The specified cutout is eliminated and no longer appears on the display. The contents are lost. This action is used primarily when revising a portion of a book with new galley material.

Create Window - A new display window is created, of the dimensions and at the screen location indicated by the user. The user may associate a book with the window, in which case it becomes a layout window (positioned at the front cover) or he may associate a galley with it, making it a galley window (positioned at the begining of the galley).

Note that in the Command Language section the designer chose to combine part of this action with the Create Book action.

Change Window Size - The size of the window is adjusted as indicated by the user.

Delete Window - The indicated window is removed from view and ceases to exist.

Move Window - The indicated window is moved to a new position on the display screen. Note the similarity of this operation to *slide* above. The moved window overlays other displayed material that it overlaps. Windows may be moved partially off the display screen.

Note that in the Command Language section the designer chose to combine this action with the Change Window Size action.

Select Layout-window - A display window is specified for subsequent layout actions. The specified window will overlay any other material that it overlaps on the screen. There is always one *current* layout window.

Note that this action is very similar to Select Galley Window. The two actions should probably have been combined.

Turn Page Forward - This action turns to the next

page in the book represented in the current layout window. Prior to this operation, the view in the current layout window is of a particular page within a particular book. As a result of this operation, the view in the window is changed to portray the material in the next page of the same book. If no such page exists, the back cover of the book is shown. The back cover is the end of the book; there are no pages beyond it.

Turn Page Backward - Same as above except that the previous page (or the front cover) is retrieved.

Create Page - A new page is inserted into the book following the current page in the current layout window, using the specified pattern page. This causes the current layout window to be paged forward, thus causing the blank pattern page to appear in the current layout window.

Delete Page - The current page in the current layout window is eliminated and its contents are lost. The current layout window is paged forward.

Print Page - Print (on paper) the page shown in the current layout window or the portion of the galley shown in the current galley window.

Select galley-*window* - Same as above, except that the action applies to galley windows.

Scroll Forward - The continuous movement of the view of lines of text within a galley in a display window in a forward (start of galley to end) direction. The top line will disappear and a new line will appear at the bottom. Scrolling will stop when the user so indicates or when the end of the galley is reached.

Scroll Backward - Same as above except in the opposite direction.

Create Book - Used by the layout artist when begining work on a new book. The title and pattern pages to be used with the book are specified at its creation.

Print Book, Galley, or Cutout - The print operation is provided so that the artist may see the current state of a book, galley, or cutout in paper form. In the case of a book, only material that has been pasted into the book will be printed. Any material left in galleys or cutouts will not appear in the printed form of the book.

NEW TASK DESCRIPTION:

The same structure established by the Current Task Description should be followed in this section. The reader should be able to understand what segments of the task are being automated and what are to remain manual after reading this section.

THE LAYOUT TASK:

Agent: The layout Artist (same).

Goals: Same.

Inputs:

a) A set of online galleys;
b) A set of online illustration galleys;
c) A set of pattern pages (a style sheet);
d) The offline layout instructions.

Outputs:

a) A finished online and printed book;
b) Empty galleys and illustration galleys.

Method:
a) Plan the layout of the whole book as before.
b) Create a new blank book, using Create Book action.
c) Paste up the pages of the book using the online facilities described above.
d) Submit an printed version of the book (or sections thereof) for approval, corrections or changes to the editor and graphic designer.
e) Revise the layout, making all the corrections and changes generated in the approval process. If necessary, return to (d).

Steps (a), (c), (d), and (e) are the same as (a) through (d) of Current Task Description except that (c) and (e) involve online activity. Current Task Description step (e) is eliminated: there is no longer a distinction between *roughs* and *mechanicals*. The new step (b) here is introduced for the automated system.

THE PLANNING SUBTASK:

Same as in Current Task Description.

THE PASTE-UP SUBTASK:

Goals: Same..

Inputs:

a) A set of online galleys and illustration galleys;
b) Online pattern pages (style sheet);
c) The offline layout instructions (marked up hardcopy of the galleys and illustration galleys and possibly thumbnail sketches);
d) Author's offline Manuscript.

Outputs:

Online and printed form of the book to be approved. Note: unused online input is also retained by the system.

Methods: For each page:

Layout and galley windows are generally present on the display screen. The status of the paste-up task at any point will consist of:

. Cutouts;
. The current state of the book;
. The current state of the galleys;
. One or more layout windows displaying the new book or a cover;
. One or more galley windows displaying a galley, positioned at the material following the last cut.

The steps below describe a normal sequence for the most typical paste-up actions. The user is free to vary the order as he desires.

a) *Select Rectangular Area of Page or Galley* to be cut.
b) *Cut Selection* creating a *cutout.*
c) *Slide Cutout* to the desired position on a page in a layout window (or anywhere else on the screen)
d) *Split Cutout* as needed.
e) When satisfied with alignment, *Paste Cutout* to a page.
f) *Create Page, Select Cutout, Select Layout Window, Select Galley Window, Print Page, Scroll, Turn Pages,* and *Create Window* as needed.

THE APPROVAL SUBTASK:

Same as Current Task Description, except that new galleys are online.

THE REVISION SUBTASK:

Goal: Same.

Inputs: Same

Outputs: New state of the book.

Method:

Same as Paste-up with following additional steps:

a) Replace a portion of a page, such as a revised paragraph, with new material in a new galley (or delete unwanted material). The old, unwanted material in the page is selected and cut as described above. *Delete Cutout* is then applied to the unwanted cutout.

b) Add or replace material on a page, causing overflow to the following page, which propagates forward to one or more following pages. The user may use more than one layout window to facilitate this process.

c) Delete or replace material on a page; this shortens the page, causing material to be "brought back" to the modified page from the next page. Propagation of text may occur. Again, the user may employ multiple layout windows.

d) Remove all material from a page, causing an empty page in the middle of the book, and use *Delete Page.*

e) Add material to the middle of the book, requiring the use of *Create Page.*

APPENDIX:

There are no error messages in the interface proposed here. Illegal operations are null operations.

This section would include details of selection algorithms, line break algorithms, error handling details, and so forth.

DATA PRESENTATION

The following descriptions define how objects are to be displayed:

Intrinsic Online Objects:

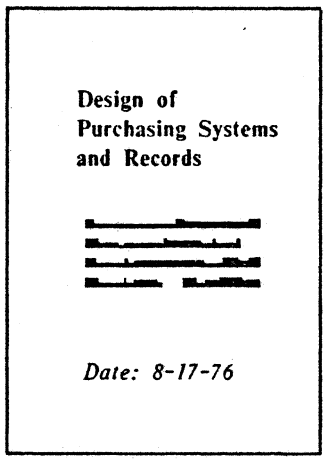
Line of text - to be shown as a string of character symbols similar in style to the font in use, scaled to the appropriate size in proportion to the page or galley containing the line of text.

Text may not be legible at the scale used for display. This is desirable to the use of legible but out-of-scale text, which will apear differently on display and hard-copy.

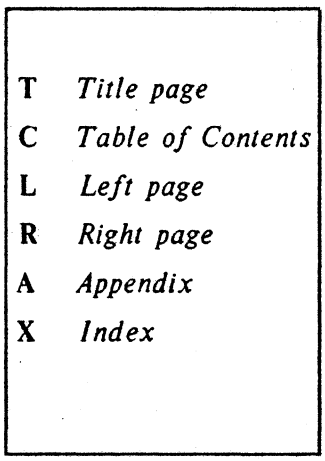
A cure for management-by-crisis and panic purchasing which is easier to achieve than either of these, is to devise better systems and procedures. Efficient forms and really useful records can substantially improve the service given to other departments.

Line of text -> Efficient forms and really useful records can

Book - only individual pages of the book are displayed at any one time. The cover is shown as a normal page of text; the back cover is displayed as a list of pattern page names.



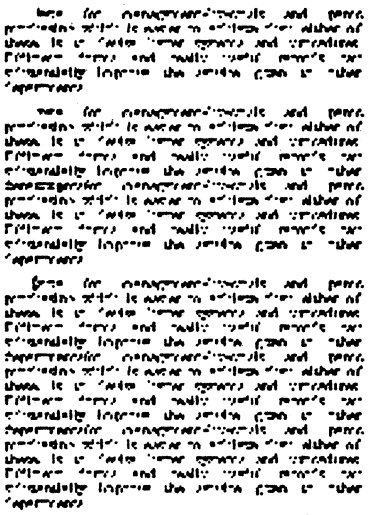
Front cover of Book



Back cover of Book

Page - to be shown as lines of text, with inter-line spacing scaled to match the required leading.

For the same reason as before, line spacing must be accurately scaled.



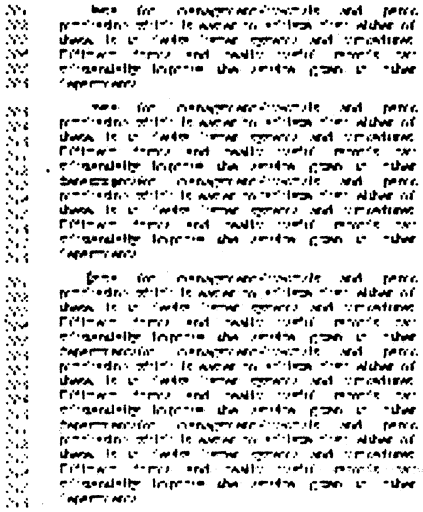
Page

Illustration - to be shown as a black-and-white image, scaled appropriately.

Illustrations need not be displayed with full resolution and gray scale. For many layout purposes, only a rough representation of the image, or even just a block of gray tone, will suffice.

Illustration

Galley - shown as lines of text, with appropriate leading, and with line numbers displayed to the left of each line.



Galley

Illustration Galley - to be shown as a sequence of illustrations, arranged one below the other in a continuous scroll.

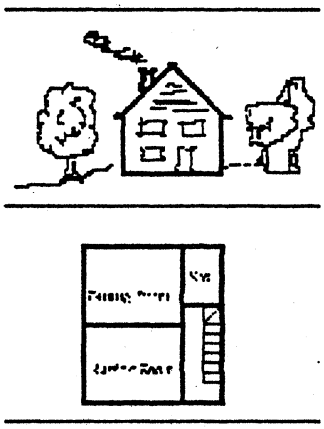
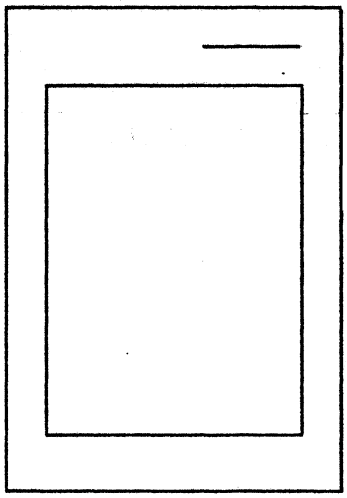


Illustration Galley

Pattern Page - to be shown as a page-sized rectangular area, with thin lines drawn on it marking the position of margins and folios.



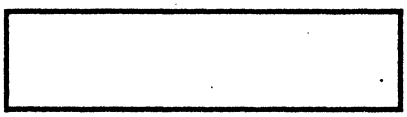
Pattern page

Dummy Page - to be shown in the same manner as a standard page.

Style Sheet - each of the individual pattern pages in the style sheet to be shown as described above.

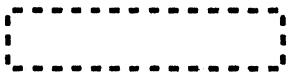
Control Objects:

Display Window - to be shown as a rectangle surrounded by a solid outline.



Display Window

Cutout - to be shown as a rectangle surrounded by a dashed outline.



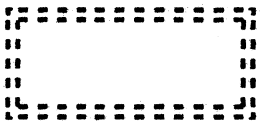
Cutout

Selection - to be shown by inverting the selected area of the screen, i.e. from black to white and vice versa.

Selection

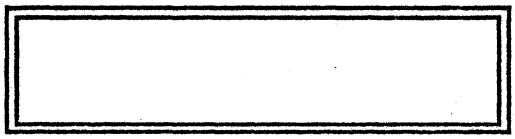
Selected cutout - to be shown as a rectangle surrounded by a double dashed outline.

The use of double outlines for all selected objects adds to the consistency of data presentation.



Selected Cutout

Selected window - to be shown as a rectangle surrounded by a double solid outline.



Selected Window

The display also shows a single cursor which follows the motions of the mouse.

Multiple cursor symbols can aid feedback, but were considered unnecessary here.

Cursor

As illustrated in the Scenario section that follows, the

display can show partial images of books and galleys in *windows.* It can also show cutouts from pages, and show them or windows sliding from one place to another around the screen. Both cutouts and windows can be moved almost all the way off the screen. A list of books, galleys, and style sheets can also be shown in a window, so that the user can choose among them using the mouse.

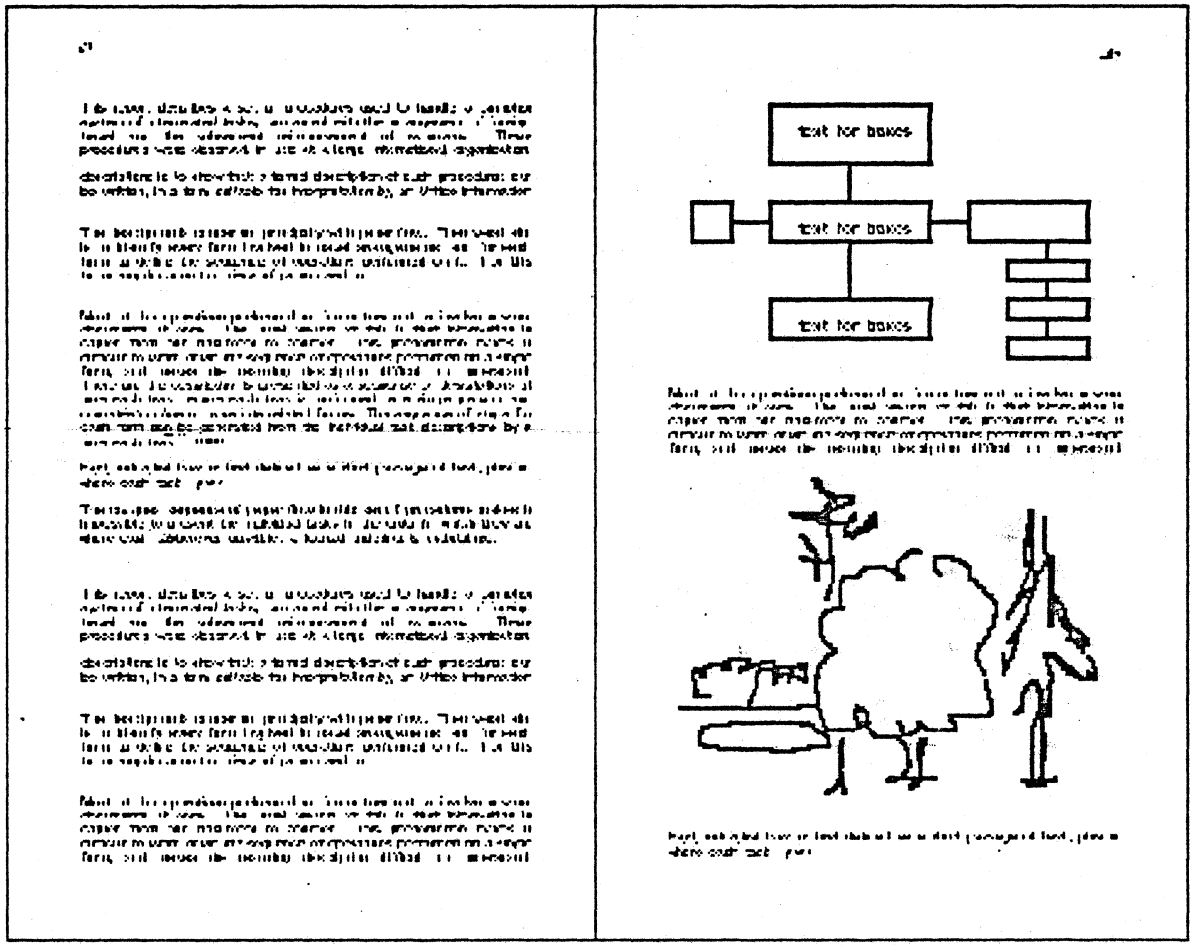
In addition, portions of a window are complemented (white characters on a black background) during selections. Selections also have a rectangular border around them during the actual specification of the selection. The complimented material inside this border indicates what is actually included in the selection. Text lines or illustrations that extend beyond the border are not included and are thus not complemented. When the specification of the selection is complete (the mouse button is released), the border disappears.

When the user is sliding a cutout or moving a window, only the border moves with the cursor. When the operation is complete, the contents of the cutout or window are erased from their old location and regenerated at the new location. (While this seems undesirable from an alignment point of view, the hardwhere is not capable of smoothly moving the whole image at once.)

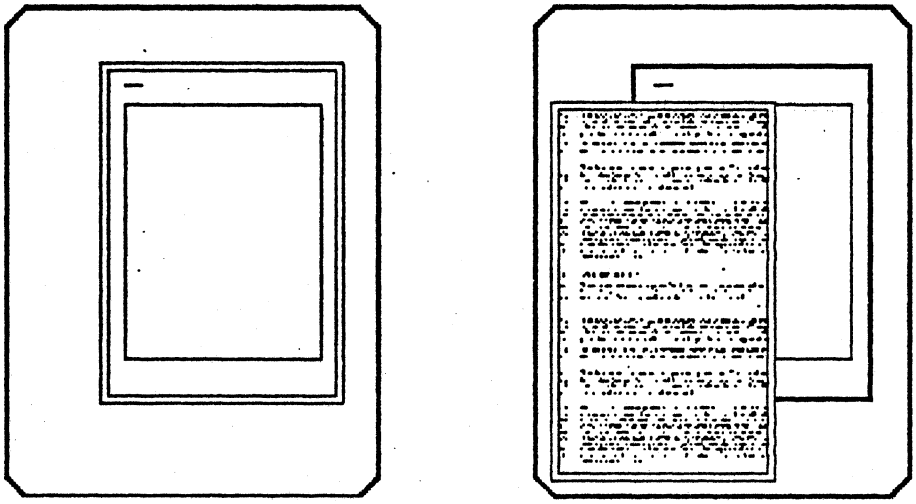
SCENARIO

This section presents a scenario of typical action sequences, with illustrations showing the general appearance of the display at each step. Only a few actions are illustrated out of the full command repertoire.

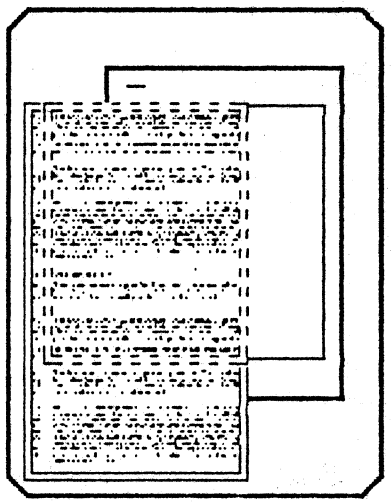
In our first example, we follow the steps involved in laying out a pair of facing pages. The aim is to produce an effect such as this:



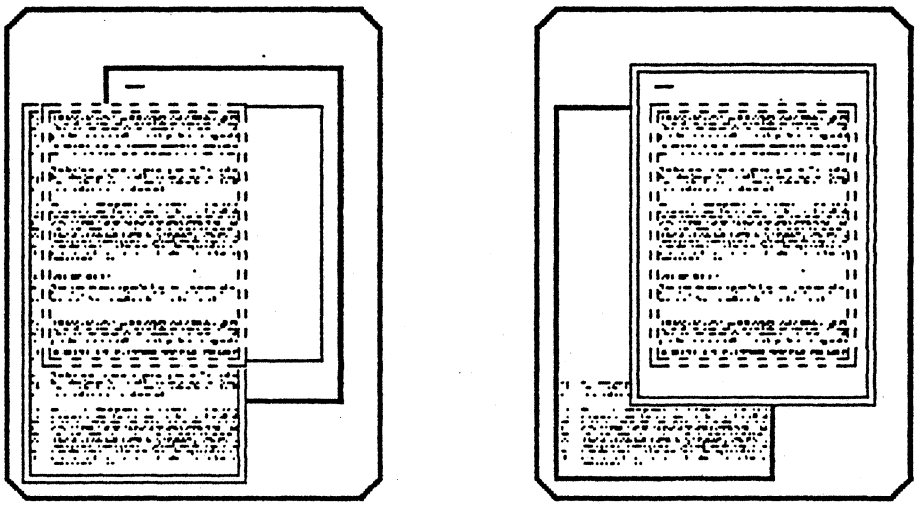
The layout artist starts by selecting a blank pattern page and placing it in the layout window. He then creates a second window and selects the desired text galley, which is displayed in that window.



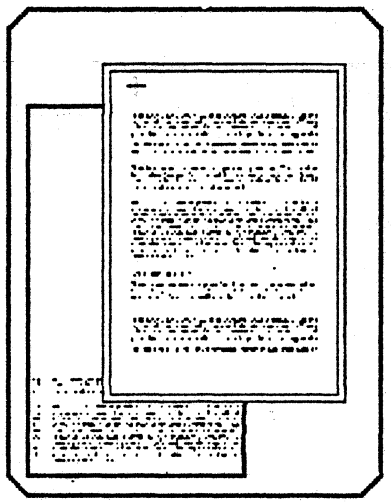
The artist scrolls through the galley until the desired text comes into view. If he is working in the normal manner, removing material from the galley in sequence, the next material will automatically be positioned at the top of the galley window after galley selection. When the selection has been made, a cut is done. This creates a cutout, containing the selected galley material.



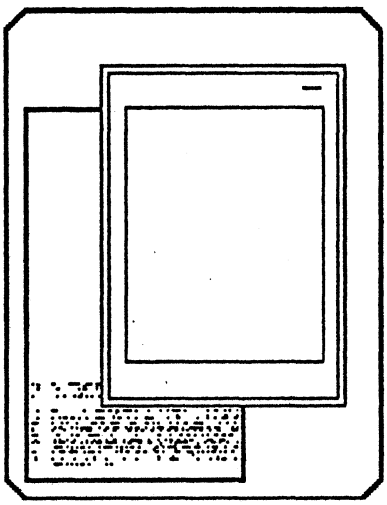
The artist now selects the cutout and moves it over the destination layout page, adjusting the position until it is correct.



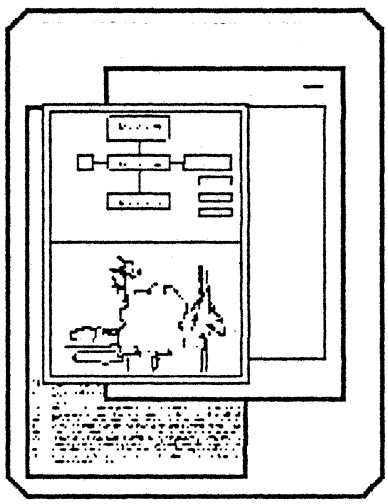
The artist pastes the cutout down. The material is now on the pattern page. The cutout no longer exists, and its boundaries disappear.

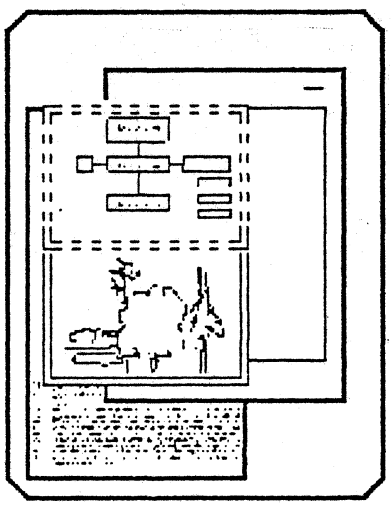


The artist now must open up a fresh page. A *select pattern page* command causes a new page to be created in the book, using the blank pattern page selected.

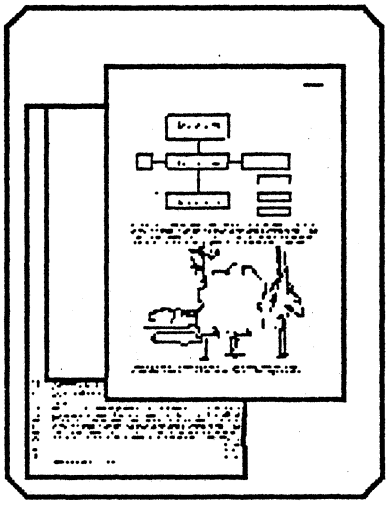


The artist now selects the illustration galley, bringing up the next sequential illustration up into a fresh window. He selects the required illustration, and cuts it into a cutout.



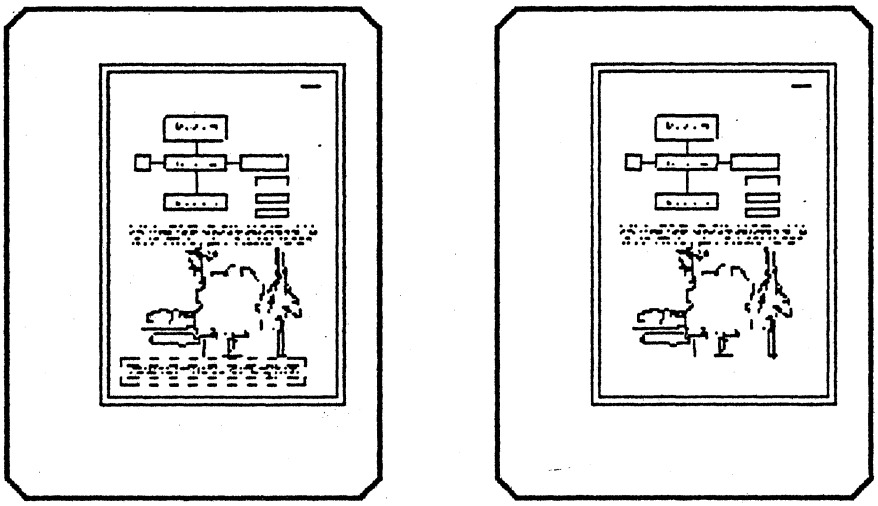


He does a number of cut and paste operations is succession, until the dummy page is completed.

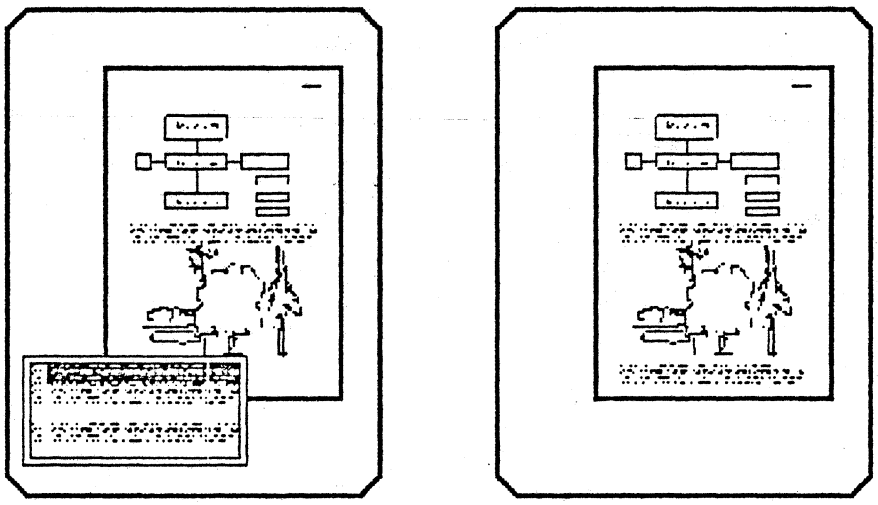


In our second sequence, we show the steps that would be involved in modifying a page. After a set of dummy pages have been laid out, it may be necessary to replace some material on a dummy page with other material from a galley. Regardless of the source of the other material, a cutout must be created containing the new material.

The dummy page to be revised is brought into view in the layout window. The material to be replaced is selected and cut from the dummy page. The cutout with this material is discarded.



The new material is positioned over the dummy page, and pasted down.



If the revision moves the material following the material replaced, it will also be cut from the dummy page, repositioned and pasted. If the material would be covered up by the revised cutout, this cut would have to precede the pasting of the revision. If it would not be covered up, the material could be moved after the revision.

TERMINAL DESCRIPTION

The work station is that set of physical devices with which a person interacts when using the system.

Special features of the devices used by the user should be spelled out in this section. This is also a convenient place to introduce terminology that will facilitate subsequent discussion.

Keyboard
A 60-key typewriter-style keyboard.

The keyboard contains a few function keys, whose names are represented herein by small-capitalized words or phrases.

The LOOK key is used as a shift key in conjunction with other alphabetic keys. Some other function keys are used to invoke operations as described in the Action Productions section below. They are:

DEL
SPLIT
CUT/PASTE
PRINT
PRINT PAGE

Mouse
A pointing and drawing device.

As the mouse is rolled across a surface a cursor on the display follows its movements. All three buttons are used interchangeably. The following operations can be performed with the mouse:

position means: roll the mouse until the cursor is at the desired spot.

hold means: **position** the mouse, depress the button, and hold it down.

point means: **hold** the mouse at one spot and release the button.

pick means: **point** the mouse at a menu-item in a list.

draw means: **hold**, roll the mouse, and release the button.

Keyset
A five-key chorded keyboard.

These function keys operate as long as they are held down.

Key Name	**Chord** (x means key depressed, o means released)
BACKWARD	(xoooo)
MOVE	(oxooo)

NEW	(ooxoo)
SIZE	(oooxo)
FORWARD	(oooox)

Disk Drawer
This is located on the top of the computer.

A user disk containing the documents being worked on is inserted here and the switch is set to run before starting the system.

On/Off Switch
This switch is found on the back of the keyboard.

Switching to the *on* position starts the system. The initial display and program state is the same as it was when the system was switch *off* using the same user disk.

Display
A high resolution, black-on-white, 8-1/2" by 11", CRT.

The discussion of the display should include descriptions of the feedback and control information display, since the user perceives this as part of the display device. Recall that the display representation of the user data was covered earlier in the Data Presentation section.

When the SPLIT key is depressed a bold horizontal line appears in the designated cutout at the location of the split. If the cursor is moved before the SPLIT key is released, the bold line will move to show the new split location. When the key is released, the split will occur at the indicated location. Note that a split will not occur in the bounding rectangle of an illustration or in the midst of a line of text. In such cases, the bold horizontal line will appear below or above the illustration/text, which ever is closer to the cursor.

While the DEL key is depressed, the cutout or page currently under the cursor will turn gray. When the key is released, the current gray cutout or page will be deleted. If nothing is gray (the cursor is not over a cutout or layout window), then nothing will be deleted.

When a window's size is being changed, only its rectangular border indicates the new size. When the size of this rectangle becomes quite small (about 1/4 " on either side), the rectangle within the new border becomes solid black instead of transparent. If the SIZE button is released while the rectangle is black, the window will be deleted.

When selecting the name of a book, galley or style sheet in a newly created window, the name currently under the cursor will be complemented when the mouse button is depressed. The book, galley or style sheet (meaning a new book) whose name is complemented when the button is released will be shown to the user in the new window.

COMMAND LANGUAGE

Conventions

It should be possible to specify several Command Languages for any given user's model, just as it should be possible to specify several user's model for any given task description. This section describes a particular implementation of the user's model described above. Most detailed decisions are made in this section.

The command language designer should specify the conventions and philosophy of the proposed command language. He should describe how he intends to use the devices and invoke commands.

The meaning of a command is always affected by the state of objects on the display, but never depends on hidden state set by earlier commands. Each key or chord has a single meaning, except for the mouse button and the CUT/PASTE function key, whose meanings depend on what object is under the cursor.

The keyboard is mostly an editing aid, so the only keys used in the layout task are five function keys and combinations of LOOK with alphabetic keys. The CUT/PASTE function key is the only key that can change the contents of a page or galley. The other function keys are associated with printing, splitting a cutout into two cutouts, or deleting a cutout or a page in a book. The LOOK key in conjunction with an alphabetic key is used to insert a new pattern page into a book.

The keyset keys are used to create and delete windows or to change their contents, size, or position.

The mouse buttons are used to make selections, to slide cutouts, and to choose a document to be displayed.

The position of the cursor is used in certain commands to indicate the current cutout or window.

Action Productions

Note:

Each production is in the form:

Action:
Informal statement
cause (user input and display state) =>
effect (changes to display and program state)

If stylized forms of presentation or abbreviations are used, they should be clearly defined in advance of their use. This section is organized according to the list of actions in the user's model but is organized so as to facilitate cross checking and to expose commonalities.

Sometimes a "cause" is a series of user inputs, such as position the mouse, depress the button, roll the mouse, and release the button. In such cases, the form of the production is:

Action:
Informal statement
cause (user input and display state) =>
1. input #1 =>
effect
2. input #2 =>
effect
...

The formal statements of cause and effect employ certain abbreviations:

CS *is the Current Selection. Its components are:*
CSP*: its page -- if* Nil, *then there is no Current Selection;*
CSR*: its rectangular bounding box in* CSP*;*
CSC*: its contents, a set of complete text lines and illustrations contained in* CSR.

CC *is the Current Cutout -- Nil if there is none.*

CW *is the Current Window -- Nil if there is none.*

Program state changes are in italics.
User input and feedback are in the regular font.

Care should be taken to distinguish user feedback from internal program state changes.

Select Rectangular Area of Page or Galley:

Push the mouse button down at one point in a galley or layout window, draw to another point, and let up to select a rectangle between those two corner points. The feedback is to draw an outline around the rectangle and to complement the items that are entirely within the rectangle, i.e., to show them white on black.

draw from (x_1,y_1) to (x_n,y_n) in CW, which displays page p =>

1. **point** at (x_1,y_1) in CW, which displays page p =>
   If CSP is non-Nil, then complement CSR;
   CSP ← p.

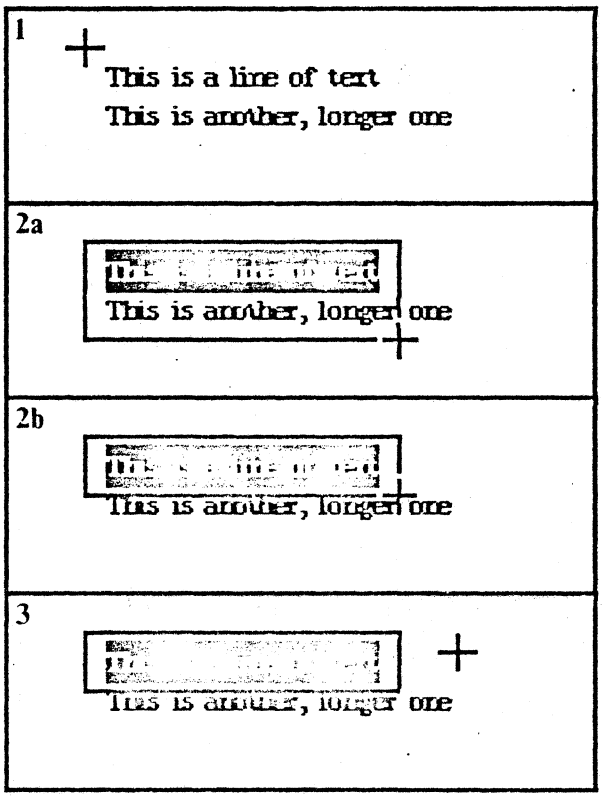
2 (repeatable, i=2..n). roll mouse to (x_i,y_i) in CW =>
   Erase outline of CSR;
   Complement items completely within CSR;
   CSR ← (x_1,y_1,x_i,y_i);
   Draw outline of CSR;
   Complement items completely within CSR.

3. release button =>
   CSC ← items completely within CSR.

The action productions should be presented in the order established by the user's model as an aid to cross checking.

The bold text is an English description of the command and is presented as an aid to the reader.

draw and point were defined in the Terminal Description.



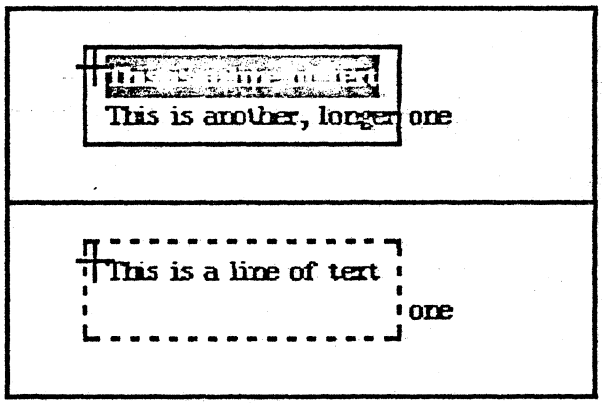
Illustrations used in this section should give an accurate picture of what the user will do and see. Illustrations should be provided wherever the designer deems them necessary. Our sample provides only a few as examples.

Cut Selection:

Depress CUT/PASTE when the cursor is over a window with a selection in it to decomplement that selection and move its contents into a new cutout. The material being cut is not moved on the screen; only its appearance changes. The Galley closes up the hole left by the cut material, as though it had never been there.

Depress CUT/PASTE when the cursor is over CSR =>

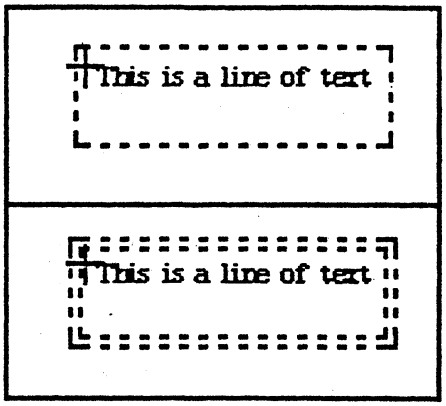
Complement CSC;
Erase outline of CSR;
Erase from CSR any partial text lines or illustrations not among CSC;
Create a new Cutout c over CSR;
CC ← c;
Transfer CSC from CSR to CC;
(If CW is a galley window, the removal of CSC from the galley will cause the galley to close up to eliminate the hole. The user may not see this, however, since it may be obscured by the new cutout.)
CSR ← CSC ← CSP ← Nil;
Outline CC with dashed lines;



Select cutout:

The Current Cutout "CC" is the one the mouse is over. This only applies to visible portions of cutouts. Selecting it causes it to be "on top of" other windows/cutouts it overlaps.

position over Cutout c =>
CC ← c;
Replace border by double dashed border.
Regenerate CC's contents to make it overlay other windows and cutouts it overlaps.



Slide Cutout:

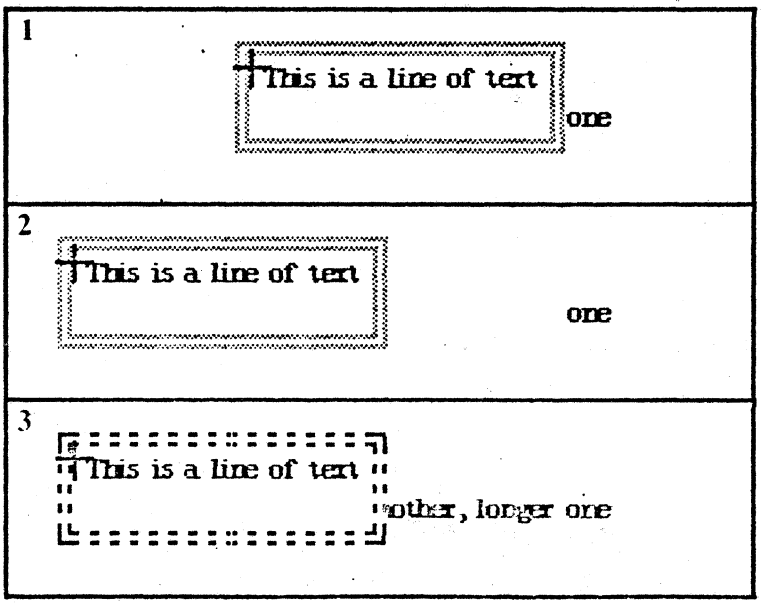
Position cursor over a cutout (select it), depress the mouse button, roll the mouse a distance, and release the button, to slide the cutout the corresponding distance in the indicated direction. At least 1/2" of each dimension is kept on the screen.

draw from (x_1,y_1) in CC to (x_n,y_n) on screen =>

1. **point** at (x_1,y_1) in CC, whose boundary is (x_l,y_t) to (x_r,y_b) =>
   Change CC's outline to dotted lines.

2 (repeatable, i=2..n). roll mouse to (x_i,y_i) on Screen such that $MIN[|x_i-x_l|, |x_i-x_r|, |y_i-y_t|, |y_i-y_b|] \geq 1/2"$ =>
   Erase the dotted outline;
   Redraw the dotted outline offset from its previous position by $(x_i-x_{i-1},y_i-y_{i-1})$.

3. release button =>
   Change the dotted outline to dashed lines;
   Erase CC;
   Regenerate any windows under CC;
   Change CC's position to the position of the dotted outline;
   Regenerate CC.

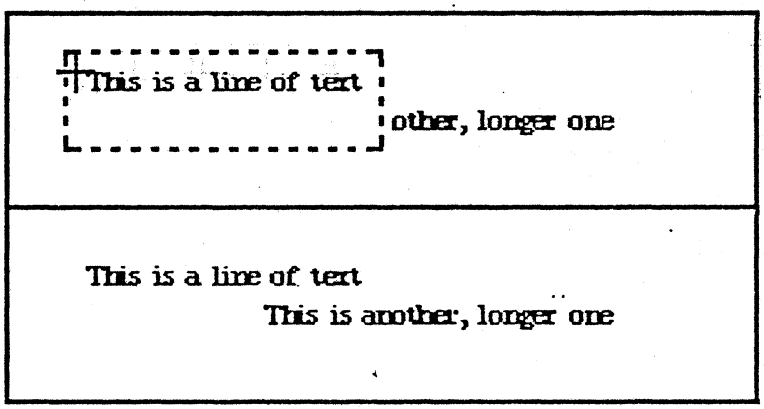


Paste Cutout:

Depress CUT/PASTE when the cursor is over a cutout to transfer its contents to the page beneath and destroy the cutout. If the layout window is scaled down (see Change Window Size) the contents of the cutout are scaled down in the process of transfering it to the page. The correspondence between the upper left corner of the cutout and the position it overlies on the page is maintained, but other correspondences may change because of the scaling during the paste operation.

Depress CUT/PASTE when CC subtends Rectangle (x_1,x_2,y_1,y_2) of Page p and when no text line or illustration in CC overlaps a text line or illustration in p or extends beyond the boundaries of p (when possible scaling is taken into account -- see Change Window Size) =>

Erase CC's outline;
CSP ← p;
CSR ← (x1,x2,y1,y2);
Transfer CSC from CC to CSP;
Destroy CC;
CC ← Nil;
Regenerate in CSR any partial text lines or illustrations not among CSC;
Complement CSC and outline CSR.



Split Cutout:

Position over a cutout and depress and release the SPLIT key to form two cutouts from one. A horizontal line will appear at the break location while the SPLIT key is depressed. The break will occur at the horizontal position indicated by the horizontal line which will have been rounded up or down to the nearest boundary of an illustration.

1. **Position** to (x_1,y_1) in cutout c on the screen and depress SPLIT =>

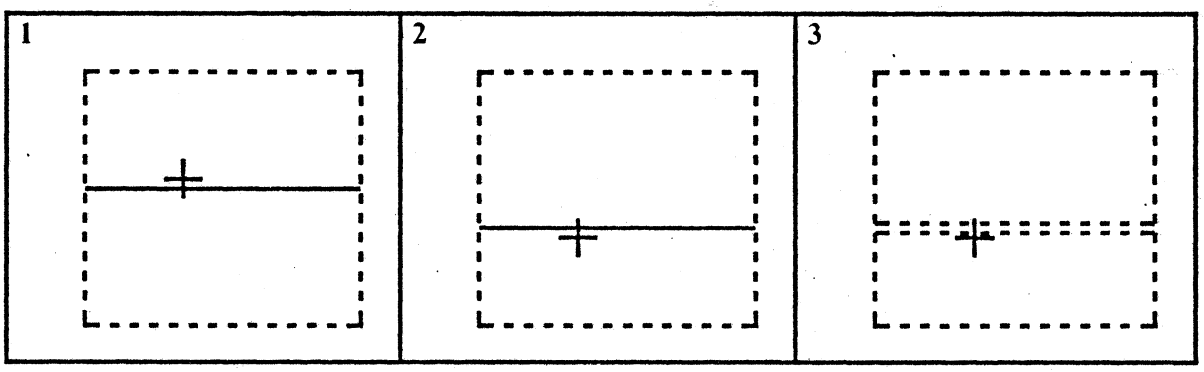
   Round (y_1) to an appropriate position y between lines and/or illustrations;
   Draw a bold horizontal line across c at y to indicate the split location.

2. (repeatable, i=0..n) roll mouse to (x_i,y_i) on screen over a cutout c_i =>

   Erase previous horizontal line;
   Round (y_i) to an appropriate position y between lines and/or illustrations;
   Draw a bold horizontal line across c_i at y to indicate the split location.

3. release SPLIT while cursor is over cutout c' with a bold horizontal line at position y in c' =>
   CC ← c';
   Adjust bottom boundary of CC to by y;
   Create new cutout with contents of CC from y down;
   Erase bold horizontal line;
   Erase border from CC;
   Draw borders around CC and new cutout.

If the **SPLIT** key is released under any other condition, it is a null operation. More generally, whenever anything happens that is not covered by a production, it is a null operation.



Delete Cutout:

Position over a cutout and depress and release the DEL key to delete the cutout and its contents.

1. **Position** over a cutout c on the screen and depress **DEL** =>

    Turn rectangle of c gray;

2. (repeatable, i=0..n) roll mouse to (x_i,y_i) on screen over a cutout c_i =>

    Turn rectangle of c_i gray;

3. release **DEL** while over cutout c' =>

    CC ← c';
    Erase CC from screen;
    Delete contents of CC;
    CC ← NIL.

Create Window:

Depress the NEW keyset key, roll the mouse between two points on the screen, and release the key to create a new window with corners at those two points. A list of documents and other things appears in the window, all of which comprise a menu for selecting the contents of the "new window" (see Select Book or Galley and Create Book below). No cutting or pasting may be performed on this window until a menu item is selected, although other operations may be performed to other windows and cutouts.

Hold down Keyset NEW while rolling the mouse from (x_1,y_1) to (x_n,y_n) =>

1. depress keyset NEW with cursor at (x_1,y_1) =>

    Create a New Window nw;
    CW ← nw;
    CW's upper left ← CW's lower right ← (x1,y1);
    Outline CW with dotted lines (it will have zero size at first).

2 (repeatable, i=2..n). roll mouse to (x_i,y_i) on Screen =>

    Erase the dotted outline;
    Redraw the dotted outline with the nearest corner to (x_i,y_i) relocated at (x_i,y_i).

3. release keyset NEW while the dotted outline circumscribes a rectangle each dimention of which is larger than 1/4" =>

    Set CW's position to the position of the dotted outline;
    Change the dotted outline to dashed lines;
    Display a list of first lines of style sheets (for *Create Book*), books, and galleys (for *Select Book or Galley*) in CW.

Note that the user's model Create Window action is being satisfied by this command and the Select Book or Galley and Create Book command below. This is perfectly fine but should be made clear to the reader.

The left-hand/right-hand coordination needed here may be a problem for users. Some tests should be done to better understand this issue.

Change Window Size and Delete Window:

Depress the SIZE keyset key, roll the mouse from a point in a window to some other point, and release the key, to change the size of the window. At each point during the roll, the corner of the window nearest to the cursor goes to the cursor. If any side of the new rectangle is less than 1/4", the rectangle becomes black. If the SIZE key is released when the rectangle is black the window will be deleted. For layout and new windows, the entire page or list of books, galleys, and style sheets, respectively, is always shown in the window. Thus, if such a window is created small or made small, its contents are scaled

Note that the command language designers chose to satisfy both user's model actions with one command.

down as best it can and will be simulated when necessary. The contents of galley windows is not scaled as the galley window size changes -- the user just sees less material in the window.

Hold down Keyset SIZE while rolling the mouse from (x_1,y_1) in CW to (x_n,y_n) on the screen. CW has dimentions (x_l,y_t) to (x_r,y_b) =>

1. depress keyset SIZE with cursor at (x_1,y_1) =>

   Change CW's outline to dotted lines;

2. (repeatable, i=2..n). roll mouse to (x_i,y_i) on Screen such that MIN[$|x_i-x_l|$, $|x_i-x_r|$, $|y_i-y_t|$, $|y_i-y_b|$] > 1/4" =>

   Erase the dotted outline;
   Redraw the dotted outline with the nearest corner to (x_i,y_i) relocated at (x_i,y_i);
   DeleteWindow ← *FALSE.*

2. (repeatable, i=2..n). roll mouse to (x_i,y_i) on Screen such that MIN[$|x_i-x_l|$, $|x_i-x_r|$, $|y_i-y_t|$, $|y_i-y_b|$] < 1/4" =>

   Erase the dotted outline;
   Redraw the dotted outline with the nearest corner to (x_i,y_i) relocated at (x_i,y_i);
   Fill the new outline with black;
   DelteWindow ← *TRUE.*

The start-continue-stop (1-2-3) steps may involve more than one production for each step.

3. release keyset SIZE with *DeleteWindow = FALSE* =>

   Set CW's position to the position of the dotted outline;
   Change the dotted outline to solid lines;
   Regenerate windows that were hidden by old CW location and are now uncovered;
   Regenerate CW.

3. release keyset SIZE with *DeleteWindow = TRUE* =>

   Erase CW;
   Regenerate windows that were hidden by CW;
   Destroy CW;
   CW ← *NIL.*

Move Window:

Depress the MOVE keyset key, roll the mouse from a point in a window to any other point, and release the key, to move that window the corresponding distance in the indicated direction.

Hold down Keyset MOVE while rolling the mouse from (x_1,y_1) in CW, whose boundary is (x_l,y_t) to (x_r,y_b), to (x_n,y_n) on the screen =>

Note the similarity between this and the Slide Cutout command. Whether the two concepts should be combined or kept separate is unclear and would warrant some user tests.

1. depress keyset MOVE with cursor at (x_1,y_1) =>

   Change CW's outline to dotted lines.

2 (repeatable, i=2..n). roll mouse to (x_i,y_i) on Screen such that MIN[$|x_i-x_l|$, $|x_i-x_r|$, $|y_i-y_t|$, $|y_i-y_b|$] > 1/2" =>

   Erase the dotted outline;
   Redraw the dotted outline offset from its

previous position by $(x_i-x_{i-1}, y_i-y_{i-1})$.

3. release keyset MOVE =>

Change the dotted outline to dashed lines;
Erase CW;
Regenerate any windows under CW;
Change CW's position to the position of the dotted outline;
Regenerate CW.

Select Layout or Galley Window:

The Current Window "CW" is the one the mouse is over. Selecting it causes it to be "on top of" other windows/cutouts it overlaps.

position over Layout or Galley Window w =>

CW ← w;
Replace border by double border. Regenerate CW's contents to cause it to overlay other windows and cutouts it overlaps.

Turn Page Forward:

Depress the FORWARD keyset key to flip the Current Layout Window to the next page. The page will advance each second as long as the key is depressed. When the back cover is reached paging will cease.

Depress Keyset FORWARD when CW is a Layout Window and CW's page is not the back cover =>

Note the use of covers in the Turn Page commands.

Once each second until back cover is reached or FORWARD is released:

CW's page ← CW's page's successor;
Regenerate CW.

Turn Page Backward:

Depress the BACKWARD keyset key to flip the Current Layout Window to the previous page. The page will advance each second as long as the key is depressed. When the front cover is reached paging will cease.

Depress Keyset BACKWARD when CW is a Layout Window and CW's page is not the front cover =>

Once each second until front cover is reached or BACKWARD is released:

CW's page ← CW's page's predecessor;
Regenerate CW.

Create Page:

Depress LOOK-*, where * is some alphabetic key associated with a pattern page in the style sheet, to insert a new page of that pattern after the page in the Current Layout Window. The back cover of the book lists the pattern pages and its corresponding alphabetic key. The Current Layout Window is advanced to the new blank pattern page.

Note that the designers could have used the alphabetic keys by themselves (without the LOOK key). This was avoided because of the tendency for users to want to type text while learning the system -- thus avoiding unexpected system behavior.

Depress LOOK key, then an alphabetic key associated with the desired pattern page pp, when CW is a Layout Window displaying page p =>

Insert a new page np after p;
np's pattern ← pp;
CW's page ← np;
Regenerate CW.

Delete Page:

Depress and release DEL to delete the page in the Current Layout Window. The Current Layout Window is advanced to the next page in the book.

1. Depress DEL when CW is a Layout Window displaying page p =>

make CW gray.

2. (repeatable i=0..n) roll mouse so it is over a layout window CW displaying page p =>

make CW gray
CW ← NIL.

2. (repeatable i=0..n) roll mouse so it is NOT over a layout window =>

make CW normal.

3. release DEL when CW is a Layout Window displaying page p =>

remove gray from CW;
CW's page ← page following p;
Delete page p from p's book;
Regenerate CW.

Print Page:

Depress PRINT PAGE to print a page of a book, a portion of a galley, or a cutout.

Depress **PRINT PAGE** when positioned over a Layout Window displaying Page p =>

Print p.

Depress **PRINT PAGE** when positioned over a Galley Window =>

Print portion of the galley being shown in the window.

Depress **PRINT PAGE** when positioned over a cutout =>

Print the contents of the cutout.

Scroll Forward:

Depress the FORWARD keyset key to scroll the Current Galley Window to the next line. The scroll will repeat once each half second until the last line of the galley is reached or FORWARD is released.

Note that the same keys are used in the turn Page and Scroll commands (see Input Cross Reference section below).

Depress Keyset **FORWARD** when CW is a Galley Window and CW's last line has a successor =>

Once each half second until CW's first line is galley's last line or until **FORWARD** is released:

CW's first line ← CW's first line's successor;
Regenerate CW.

Scroll Backward:

Depress the BACKWARD keyset key to scroll the Current Galley Window to the previous line. The scroll will repeat once each half second until the first line of the galley is reached or BACKWARD is released.

Depress Keyset **BACKWARD** when CW is a Galley Window and CW's first line has a predecessor =>

Once each half second until CW's first line is galley's first line or until **BACKWARD** is released:

CW's first line ← CW's first line's predecessor;
Regenerate CW.

Select Book or Galley and Create Book:

A newly created window (see Create Window command) has a list book, galley, and style sheet names. Pick one of the book or galley names to view the corresponding book or galley in the window. Point at one of the style sheet names to create a new book with that style and display it in the window. Text that is pasted into the title region of the front cover will appear in *New Window's* list of book names.

Pick the name of a book, galley, or style sheet in a new window nw =>

1. While the cursor is over a newly created window nw, depress the mouse button =>

CW ← nw;

Complement the book/galley/style sheet name nearest the cursor.

2. (repeatable for i=0..n) roll mouse within a newly created window =>

CW ← nw;
Complement the book/galley/style sheet name nearest the cursor.

3. release the mouse button while the cursor is within a newly created window CW and the name of a book b is complemented =>

CW's type ← layout;
CW's page ← front cover of b;
Regenerate CW.

3. release the mouse button while the cursor is within a newly created window CW and the name of a galley g is complemented =>

CW's type ← galley;
CW's first line ← g's first line;
Regenerate CW.

3. release the mouse button while the cursor is within a newly created window CW and the name of a style sheet ss is complemented =>

Create a new Book b with style sheet ss;
Add b to the list of Books that is diaplyed in new windows;
Insert front and back covers into b (front cover has title region and time of creation;
back cover is list of pattern pages from ss for use in Create Page);
CW's page ← front cover;
Regenerate CW.

Print Book, Galley, or Cutout:

Depress PRINT to print the book or galley in the Current Window.

Depress **PRINT** when CW is a Layout Window showing a page from book b =>

Print all pages of the book b.

Depress **PRINT** when CW is a Galley Window showing a portion of galley g =>

Print g, inserting page breaks as necessary.

Depress **PRINT** when CC is a cutout c =>

Print all of c.

On:

Turn the On/Off switch to the On position.

On and Off should probably have been part of the user's model.

On/Off switch moved to On position, some user disk mounted and disk up to speed =>

Restore state that was saved on user disk when system was last turned off;
Regenerate display as per saved state.

Off:

Turn the On/Off switch to the Off position.

On/Off switch moved to Off position =>

Save the state of the system, including the state of the display on the user's disk.
Turn the display black.

Input Cross Reference

Mouse Buttons

The mouse makes a selection when pointed in an old window, slides a cutout when pointed in a cutout, and chooses a document from a list when pointed in a new window. The buttons are used interchangeably.

For each input device (key, button, pointing device, menu item, and so forth), this section should list the effects it can have under different circumstances.

Keyboard keys (alphabetic keys used only with **LOOK**; numeric and punctuation keys unused)

PRINT
prints book, galley, or cutout, depending on position of mouse.

PRINT PAGE
prints page of book, portion of galley, or cutout, depending on position of mouse.

CUT/PASTE
PASTE when the cursor is over a cutout; CUT when the cursor is over a page with selection in it, and a no-op otherwise.

SPLIT
splits the cutout at the horizontal position of the cursor.

DEL
If mouse is over a cutout, delete it; if over a page in a layout window, delete the page from the book.

LOOK a (an alphabetic key)
If the mouse is over a layout window, insert a new page in the book following the currently shown page using pattern page a for the new blank page.

Keyset keys

Key Name	Chord	Function
BACKWARD	(xoooo)	SCROLL BACKWARD*, PAGE BACKWARD*
MOVE	(oxooo)	MOVE-WINDOW**
NEW	(ooxoo)	CREATE-WINDOW**
SIZE	(oooxo)	GROW-WINDOW**
FORWARD	(oooox)	SCROLL FORWARD*, PAGE FORWARD*

* Scrolling applies to galley windows, paging to layout windows.

** Used in conjunction with mouse movement.

Output Cross Reference

List all the forms of display feedback, control information, and data presentation that the user might see on his screen and indicate for each the possible interpretations for it.

Note that the cursor symbol can change to indicate different system states. The designers of this particular command language merely chose not to do this.

Cursor: A constant symbol is used throughout.

Solid outlines around rectangles containing text and/or illustrations: Galley and layout windows.

Dashed outlines around rectangles containing text and/or illustrations: Cutouts

Dotted outline: A moving or growing window or a sliding cutout.

Double dashed outline around selected cutout.

Double solid outline around selected window.

Bold horizontal line within a cutout: Feedback for a Split Cutout command.

A gray rectangle with solid outline: A page in a layout window that is about to be deleted.

A gray rectangle with a dashed border: A cutout that is about to be deleted.

A narrow black rectangle with dotted outline: a window that is about to be deleted.

A solid outline around a rectangle, perhaps with some complemented text and/or illustrations within it, all within a window: A sellection being specified for use in a Cut Selection command.

Complemented text and/or illustrations within a window: A selection to be used with a Cut Selection command or a name of a book, galley, or style sheet in a newly created window (only while cursor is over the window and the mouse button is depressed).

Text and/or illustrations within a window: A portion of a galley (each line of text and each illustration is numbered), a portion of a layed-out page of a book, A list of books, galleys, and style sheets in a newly created window, or a book cover.