DAA/AMES

C/S S0380476

Stanford Applied Psychology Laboratory

Stanford University
Jordan Hall, Bldg. 420
Stanford, CA 94305

(NASA-CR-176964) COGNITIVE AND PERCEPTUAL PRINCIPLES OF WINDOW-BASED COMPUTER DIALOGUES Annual Progress Report, Dec. 1985 - Nov. 1986 (Stanford Univ.) 100 p CSCL 09B

N86-31235

Unclas

G3/61 43020

Status Report for Summer 1985

Bruce D. Verner
Intelligent Systems Laboratory
Xerox Palo Alto Research Center
26 August 1985

This document is a brief summary and status report on my research at ISL/PARC during the summer of 1985. It describes some of the research, software engineering, and other activities I performed for the User System Research (USR) group of ISL. For information on my work as a Research Intern at PARC during the previous summer, refer to the August 1984 technical mini-report entitled "A Brief Summary of My Work on TROOPS."

According to telephone discussions before my return to PARC, my primary project was to be the augmentation of the Input/Output facilities of the Dandetiger computer through electronic hardware and software extensions. Upon my arrival at PARC in early June, Austin Henderson informed me that the information-space flyer project is receiving higher priority and asked me to devote my efforts toward advancing that work. The information-space flyer concept is similar to a flight simulator where one traverses over an information topology (such as a semantic network) rather than a geographical terrain. He explained (as Stuart Card had previously mentioned) that the Human Interface Program at the Microelectronics and Computer Technology Corporation (MCC) has a similar research project in progress and that we want to keep the lead in this area. In addition to advancing the state-of-the-art in user interfaces and information display, our pursuit of this hypermedia project is contributing to the formulation of a general comprehensive theory of visual interfaces. (Hypermedia are presentational media which perform in multidimensional ways. Some of our research is related to the pioneering ideas of Ted Nelson such as the Fantics and Thinkertoys 3-D concepts.) Stu and Austin had completed a considerable amount of work on the underpinnings for a net flyer environment for the Dandeiris 3-D color graphics system, but much remained to be done.

Austin gave me a few pointers to the existing work which consisted of, the IRIS graphics software library for Interlisp-D; the Intermediate-Displayable Network (ID-Net) package for building, manipulating, and accessing object-oriented network data structures; the Organization Chart (Org-Chart) package for generating visual representations of ID-Nets on the Dandetiger; and the Whiteboard (White-Board) package for displaying the outputs of Org-Chart in Euclidean 3-space on the Silicon Graphics IRIS terminal.

After discussing the desired future directions of this research, I proceeded to learn the details of the high-performance, pipelined, custom VLSI-based IRIS graphics processor -- a truly remarkable machine. I supplemented my previous knowledge of computer graphics theory and practice by reading Principles of Interactive Computer Graphics by Newman and Sproull (both formerly of PARC) and various other relevant works. Then, I read the IRIS documentation manuals to become intimately familiar with the functioning of that system. The hardware and software architectures of the IRIS seem to be heavily influenced by the ideas of Newman and Sproull, and thus, I recommend their book to anyone interested in the deeper and more formal concepts underlying this technology. (This impression is not surprising when one considers that Silicon Graphics has several PARC alumni.) Once acquainted with the research vehicles, I began modifying and extending these systems in accordance with Stu's and Austin's requests and suggestions and my ideas.

The first software package I wrote adds a facility that generates and displays function calling path trees on the IRIS in response to the issuance of a Masterscope "Show paths ..." command by an Interlisp-D user. This tool is called Master-Chart and is located in the file {Phylum}<Verner>Lisp>Net>Master-Chart(.dcom). When Master-Chart is loaded, it taps into the Masterscope database by advising the appropriate functions in the Browser package. In response to a user's "Show all paths from aFunction" query, Master-Chart (when enabled by a flag) creates an ID-Network representing the calling paths. Each node in the net contains information about the corresponding function. The resultant network is then displayed on the Dandetiger screen in an Org-Chart browser window for manipulation by the user. With the browser, various extractions (information organizations) and layouts (extraction display formats) may be generated from the information net upon request. The user then selects a menu command to create and display a whiteboard world on the IRIS showing the desired net layout. A flyer may be used to explore the presented relationships between functions. The Master-Chart package, in conjunction with Org-Chart, allows the display of large and complex function interdependencies that exceed the viewing capacity of the existing tools developed for the D-machine screen.

Having completed Master-Chart, I wanted a way to explore the network domain displayed on the IRIS that is easier than the former method of typing coordinate and velocity codes on the keyboard. I set out to design and implement a mouse-driven information space flyer for White-Board. What resulted is a mouse-controlled vehicle that models an inertial mass with thrusters in a frictional medium. The physical dynamics of this explorer are more similar to those of a submarine or heavy aircraft than a spaceship -- the inclusion of friction in the model serves to protect the flyer from wild and difficult to control movements that are a familiar annoyance to users of spacecraft simulators. The mouse controls for this flyer are described in the document, "Summary of Controls for Info-Net Flyer #1."

Lennart Lovstrand has recently suggested and plans to implement what he calls an impulse flyer. I interpret this to be a sort of "hyper-warp" device that does not observe the physical law that the second derivative of the position of a mass must be a continuous function. If the velocity of the flyer is not "everywhere differentiable", then the vehicle is free to make discrete jumps in space or halt in zero time. These behaviors should be carefully controlled to avoid disorienting the user. Still, Lennart's impulse flyer seems an interesting idea and will be useful for comparison with mine.

There are certainly many possible kinds of flyers for our information spaces, and we hope to extract what makes a good flyer from our series of experiments. Toward this end, Stu has suggested the separation of flyers from the rest of the White-Board system to facilitate the rapid interchange of prototype vehicles. In my opinion, this test-bed approach to the White-Board structure is worth pursuing. Perhaps the model for a modular flyer should be a state-machine which interfaces to White-Board through state variables refering to physical parameters such as position, velocity, acceleration, and coefficient of friction. Eventually, the user should have a hanger of craft from which to select according to desired performance properties.

With my completed flyer, I traversed the information space and discovered the need for several refinements to the existing White-Board display paradigms. One problem was that the field-of-view of the flyer could be rotated away from the board -- rendering the network out of view. In this situation, one is left looking into empty space with no orientation clues and little chance of easily finding the whiteboard. To solve this problem, I recalled the relevant theorems of trigonometry and analytic geometry and derived constraint equations on the rotation of the flyer to keep the whiteboard in view. The code for this feature is included in Master-Chart and may be put into effect by setting a global flag.

Another shortcoming is that the "flashlight" (a dot which indicates a point of interest on the whiteboard toward which to gravitate) was of constant size relative to the board. Thus, at farther viewing distances from the board the flashlight would become vanishingly small -- leaving the user confused about the location of the current focus. To remedy this situation, I modified the flashlight program code to make the beam diameter proportional to viewing distance. Now, the flashlight is approximately a constant size relative to the IRIS viewport and always readily visible without being obtrusive. It is interesting to observe that the completion of a first flyer precipitated the discovery of the need for these refinements to White-Board.

All of the previously described work was then demonstrated to Stu and Austin for feedback, and they gave a positive response.

We wanted to demonstrate our hypermedia system to a group of visiting scientists, but it was lacking a major capability -- that of printing text in IRIS displays. Such a feature is necessary for labeling the various network structures in the IRIS world space. Since the need for text printing was immediate, I designed and implemented an object font package which enables the generalized printing of text as graphical objects on the IRIS. During White-Board initialization, this code reads a file of font data (previously generated by Austin) and compiles that representation into a list of IRIS display library commands. When EVALed, this expression builds an object font display-list data structure within the IRIS. A font table of characters, character codes, widths, kearns, and object numbers is created for later use by my printing mechanisms. The compiler-based approach is efficient because the translation of source data to library commands need only be perfomed once. IRIS text display is then available through a transparent programmatic interface. Parameters such as height, position, color, linewidth, and format can be easily modified on a character basis. The Object-Font package was successfully completed and used to label Master-Chart nodes with corresponding function names, to print multiple text lables on Xerox organizaton charts, and to paint numbers on the cards in Lennart's arithmatic problem mockup. The commented code of the object font package is avaliable in the file {Phylum}<Verner>LISP>Net>Object-Font.(dcom).

As requested by Austin, I added to White-Board the capability to display a network of the Xerox organization chart with scaled text labels on each node for post, title, and name. Three different colors are used for the three label types allowing easy identification of information. When enabled by setting a flag, a labeled org-chart will be automatically generated from a text file representation. The default text file name is contained in a global variable which may be reset by the user. Refer to the documentation in Object-Font and Master-Chart for further details.

At one of the discussion sessions, Stu suggested that we try to adapt to our system the powerful "fish-eye view" display paradigm developed by George Furnas at Bell Labs. Furnas had described the advantages of this algorithm during his visit to PARC. I obtained a copy of Furnas' paper on the subject, "The FISHEYE view: a new look at structured files", which I subsequently read. Lennart offered to write a static fish-eye view as another Org-Chart layout option, while I volunteered to create a dynamic-incremental fish-eye view that would reside in White-Board itself.

The existing structure of White-Board had all the arcs and nodes of the net grouped into one IRIS display object -- a configuration unsuitable for the implementation of dynamic fish-eye views. To facilitate the construction of such views, I rewrote and extended much of the White-Board code. One of the major changes was making the net constituents into separate IRIS display objects. Now, each node can be edited to change its scale, color, or other properties independently of other nodes. The node selection behavior in Org-Chart was changed to choose the closest node to a mouse click rather than reporting a miss, and the node selection facility was extended to work on IRIS whiteboards. This fish-eye view test-bed was exercised by adding code that displays the node nearest the flashlight in red at an expanded size. The flashlight is used to select a node on the whiteboard in a manner similar to the use of the mouse cursor in Org-Chart display windows. Text labels on the node are also expanded accordingly. When a new point of interest is chosen, the previously selected node is reverted to its former ordinary appearance and the desired node is highlighted. The documented code for this work is mostly contained in the file {Phylum}<Verner>LISP>Net>Proto-Fish-Eye(.dcom), and other parts of it are in the Master-Chart package. With this solid foundation completed and demonstrated, I was ready to attempt the creation of a full-fledged fish-eye view engine.

Stu suggested that we attend the Siggraph conference in San Francisco to gather information on progress in the field of computer graphics. At that event, we observed many interesting developments; although much of the exibited technology was imitative and uninspired. Silicon Graphics (IRIS) and Lucasfilm (Blitter machine) where among the leaders in hardware. Abel Graphix, some of the smaller companies, and the universities (Caltech, Tokyo, ...) had the best software. The poor quality of the PARC/Apple-like windowing user interfaces that I tried on various machines was disappointing. Obviously, the developers of this hardware and software have ignored the well-refined paradigms stated in the Smalltalk-80 books and fashioned their systems in an inconsistent, inelegant, and ad-hoc manner. Yet this is still progress over the previous conditions, and they will eventually correct their errors. After much manuevering, we managed to obtain tickets for the film and video show -- an enjoyable presentation that spotlighted some of the recent advances in the art and science.

Around this time, we received from Silicon Graphics the new IRIS workstation upgrade for the laboratory. The IRIS workstation has various significant increases in capacity and capability over the old IRIS terminal, including the ability to run the UNIX environment. Our intention is to reimplement the time critical segments of the White-Board main control loop activities in the C language on the IRIS. The Dandetiger will then send to an XNS network daemon (process) on the IRIS any adjustments needed to keep information consistency. We expect this restructuring of our software architecture to yield significant performance gains.

After installing the workstation, we had severe difficulties with the multitasking mode of UNIX. I contacted Silicon Graphics, and with their assistance and Lennart's help isolated the problem to the Ethernet board in the IRIS. After much persistence, the problem was resolved by having SG replace the card. Lennart has configured user accounts for the workstation and we are becoming proficient with the new features of the system.

The IRIS workstation requires the development of new communications and graphics library software in Interlisp-D. This task has been almost finished by Greg Nuynes at Xerox AIS; and he is continuing to debug the IRISlib package. I have sent him "bug reports" describing the details of several fatal problems in the code. He is proceeding to fix these and we await the completion of a fully functioning release so that we can begin use of the new IRIS. The work of Greg and Michel Desmarais (formerly at PARC) on the systems related programming has been essential for our rapid progress.

While Greg perfects his software, I have designed and implemented a dynamic-incremental fish-eye view package for the Dandetiger display. By setting a global flag the output of this package may be viewed on the IRIS. (Testing of the IRIS version awaits completion of debugging of the new IRIS software.) The Dandetiger portions of the package have been tested and successfully generate the intended behavior (see the attached hardcopies). When a point of interest is selected in the Org-Chart display (and the function FEV is called), the node nearest that location becomes the focus of a dynamic fish-eye view which spreads iteratively in a breadth-first fashion through the net. The effect of these transformations is to visually emphasize the focus node and its nearby neighbors while eliding distant nodes. The path from the focus to the root of the net is also spacially emphasized as suggested in the Furnas formal description. Propagation is attenuated through a threshold mechanism to improve time performance. A mouse click may be used to change the focus and a new fish-eye formation will gradually result allowing a brief multi-foveal view. The fish-eye function is highly isolated to permit easy modification or replacement for experimentation. Currently, the function is based on the Furnas formal mathematical definition as stated in his report. The structure of the fish-eye software is general to permit convenient modifications for experimentation with new kinds of views. I have put the annotated code file for the Fish-Eye package in {Phylum}<Verner>LISP>Net>Fish-Eye(.dcom).

While converting the software for the new IRIS, I discovered that the IXLATE translation routine did not implement conversion of the old backslash form of IRIS commands. I made a simple extension to accomplish this in the code, and saved the modified version for future use in {Phylum}<Verner>LISP>Net>IXLATE(.dcom).

At our latest brainstorming sessions we have evolved many ideas for enhancing our hypermedia system including putting a control panel on the IRIS display and tying logical searches of the net to fish-eye views. Stu and Austin suggested devoting the bottom portion of the IRIS screen to something akin to a cockpit panel. One of the controls might be a fill-in form that initiates a logical search for particular information in the net. In the case of org-charts, a typical query could mean "Find and show me the Vice President of Research and emphasize the surrounding posts in the chart back up to the CEO." A fish-eye view would then be initiated with the located node(s) as a focus. These ideas are exciting and very promising directions of exploration. In the remaining week before I return for my senior year majoring in Electrical Engineering at the University of Rochester, I will attempt to make a start in their development.

I would like to thank Stuart Card, Austin Henderson, Tom Moran, and John Brown of ISL for again sponsoring me as a Research Intern, and NASA for granting partial funding for the Dandeiris-Hypermedia project. My activities at PARC with diverse projects, people, and disciplines have been very enjoyable and provided me with a good learning experience. The consulting related to PARC technology transfer, in which I have engaged for Xerox, has contributed to my further development of practical industry-oriented knowledge, skills, and abilities. The combination at PARC of high quality people, excellent facilities, and creative freedom renders the place an extraordinary research environment.

Summary of Controls for Info-Net Flyer #1

Bruce D. Verner
Intelligent Systems Laboratory
Xerox Palo Alto Research Center
17 August 1984

This is a summary of the behavior and operation of the mouse-driven Whiteboard flyer. The flyer is a craft that is used to navigate and explore an information space displayed in 3-D on the IRIS color graphics system. The flyer will gravitate toward a focus (or point of intertest) indicated by a yellow "flashlight" beam displayed on the whiteboard. A user may easily change the focus by aiming the beam with a "gunsight" located at the center of the screen. When the mouse-driven flyer is in use, the old keyboard code controls remain in effect. Although the mouse flyer provides linked control of all essential degrees of freedom, the keyboard may be used for supplemental adjustments. Documentation is also provided for additional keyed commands that were integrated into Whiteboard as an intermediate measure before creation of the mouse flyer.

The mouse on the IRIS is used for the flyer, although with slight modifications to the code the Dandetiger mouse could be the control. All mouse inputs are polled (rather than queued) to provide the most immediate response to the current situation. The effect generated by pressing a mouse button(s) is usually proportional to the interval of time it is held. (Holding down longer increases velocity, continues focus selection, etc ...)

Control Assignment on IRIS mouse:

Mouse Chord			Behavior
LB	MB	RB	Quit the Whiteboard program (exits into LISP)
LB	--	RB	Freeze ONLY Z-axis movement of flyer (Z-axis movement may later be resumed)
--	MB	RB	Set Z-coordinate of focus (POI) to current Z-position of flyer (has effect of halting Z-axis movement)
LB	MB	--	Initiates auto-homing to a viewpoint encompassing the entire whiteboard
LB	--	--	Select a new focus (POI) at the place on the whiteboard in the gunsight (since the flyer is attracted to the focus, changing the POI will affect X and Y-axis motion; just point the flashlight where you want to go!)

--	MB	--	more negative Z-thrust (toward the whiteboard)
--	--	RB	more positive Z-thrust (away from the whiteboard)
mouse X-axis			right-left head rotation (used to look around)
mouse Y-axis			up-down head rotation

[Refer to the program listing for information on new additional keyboard codes.]

COGNITIVE AND PERCEPTUAL PRINCIPLES OF WINDOW-BASED COMPUTER DIALOGUES

Annual Progress Report.

NASA - AMES Grant # NAG 2-269 [illegible]

M. Pavel & S. Card

December, 1985 to November, 1986.

PROGRESS REPORT

COGNITIVE AND PERCEPTUAL PRINCIPLES OF WINDOW-BASED COMPUTER DIALOGUES

Our approach to understanding window-based computer dialogues has two main thrusts: (1) work on the theory of human-computer interaction and (2) exploratory programming of new techniques for graphics dialogues. In this reporting period we were able to construct a system for displaying trees on a simulated "whiteboard" in 3-D space using the project's "DandeIris" (Xerox 1109 DandeTiger Lisp machine linked to a Silicon Graphics Iris 1300 through an ethernet). The tree application chosen was to display an organization chart. The user was able to zoom into a particular node or to "fly" along the surface of the whiteboard. Several algorithms were tried for laying out the tree. It was found that (1) the size of the nodes could not be simply scaled to the size of the whiteboard because very skinny tall nodes would likely result for most whiteboard shapes. (2) If the nodes were scaled so that they had reasonable shapes and separations, then a chart of around a thousand nodes was indistinguishable from a straight horizontal line when the user zoomed back. We therefore tried displaying the nodes in fish-eye form, where the size of the node depends on its distance from some designated focus node and on some specified function for calculating the falloff of interest with distance. When this is done, the user must wait some time before the system can redisplay the tree after he has indicated a new point of focus. We therefore worked out ways to "pulverize" the algorithm, that is to have the fish-eye layout be changed node by node. The major difficulty with the resulting system is response time. An analysis suggested that the problem was in the buffering on both sides of the ethernet. At this point we upgraded to an Iris 2400 workstation, which had the advantage of more graphics primitives.

For the new Iris 2400, we built the DandeIris II, rewriting the Lisp Iris IO functions and adding over a hundred primitives to the Lisp side of the connection. We also programmed a system to compile from the C language version of the graphics primitives into Lisp. We reran our programs in this version, but determined that response time in this version was actually lowered by one graphic primitives Silicon Graphics had discarded from the new library.

Experiments with C versions of our existing code showed that very good response was possible by breaking the system into an animator than runs in C and a network program that runs in Lisp. This version of the system is called DandeIris III. With the help of the C version, we developed a new "impulse flier" that allows the user to zoom in much more rapidly to portions of the tree than was previously possible. We are now implementing this system. We will then have worked out an architecture for an interface building toolkit in the advanced graphics area.

Along with this progress, various theoretical issues are becoming more clear. Particularly interesting is the interaction between window/advanced graphics interfaces and intelligent systems. Application systems can be described by a dialect of Sheridan's supervisory control model we call the "triple agent model." Different sorts of intelligent systems can be got by applying intelligence to different parts of the model. The triple agent model makes it particularly clear that there are many different ways for the user to interact with an system, differing principally according to who approves whose work and who can take the initiative when. This calls out issues of the control structure of dialogues and we have begun thinking about how these relate to window systems under the name of visual discourse studies.

It is clear that for very cognitive tasks such as the so-called "idea processing"/ "idea-browsing" systems now beginning to be seriously researched, advances in windowing techniques along the lines investigated in this project will be important.

Describing the interaction with the user for a graphical user interface is quite difficult. We have therefore analyzed several interface techniques into a small catalogue preliminary to making formal descriptions of them and as an experiment in how to describe these.

Following are a set of papers that elaborate in various ways the themes listed above. In this period, several students worked on the project for varying amounts of time: Michel Desmarais, Roy Nakashima, Lennart Lovstrand, and Bruce Verner.

HUMAN FACTORS AND THE INTELLIGENT INTERFACE

Stuart K. Card
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, California 94304

By now, the average technical conference-goer is used to seeing the term artificial intelligence paired with just about anything. Whereever in the world there is a task that involves the least amount of human thought, someone in that profession (or his boss) has probably thought of applying artificial intelligence to it: *Artificial Intelligence Applications for Business* (the title of a recent book), artificial intelligence and economics (a recent sabbatical in our laboratory), artificial intelligence and library science, artificial intelligence and nursing, artificial intelligence and franchise fast foods, artificial intelligence and human factors. By now, none of these sounds particularly odd, we seem lightyears from the Lighthill report with its skepticism whether AI would ever amount to much. Partly, I suspect, this is because economical computing power has finally arrived at the point where there was about to be a large expansion in the number of computer applications anyway, and artificial intelligence, as used by some people, is really just another name for applications programming in new domains. But in the case of artificial intelligence and human factors, the relationship is really more profound. The most interesting case is where the two come together in the human-computer interface. Here, the boundaries between them break down so fundamentally that a new subdiscipline is emerging. The consequence, I think, is that human factors itself as a discipline will be permanently altered.

Human factors is concerned with influencing the design of machines so that they are better operated by their human users. It began during World War II when substantial advances in the technology of war equipment had resulted in equipment complex enough that only if explicit attention were given to the human would the equipment be usable. For example, in order for a P-51 pilot to reach the lever that retracts the landing gear, he had to put his head down so far in the cockpit that he could not see out the window. This was not a good idea. The maneuver occurred very near the ground, and the resulting air crashes tended to support the idea that more attention placed on understanding the details of the pilot's task would be helpful to airplane design. The upshot of this and like problems was a new systematic consideration of what the human operator could see and operate. Instrument dials began to be made more reliably legible, controls more reachable and less confusing, procedures less error prone. Of course human factors has moved on since then. There are guidelines for displays, there are workload analyses, there are systematic anthropometry tables and computer-generated reach curves. All this is fine help for that original set of human factors concerns that were essentially sensory-motor in nature. But much as that help might be, when applied to current systems it just is not good enough. The technology of machines has taken an even faster leap.

THE CHANGED NATURE OF THE MACHINE

The development of microchip technology has led directly to a set of changes in the machines with which humans interact. Collectively, these changes define a qualitative difference between these machines and those of thirty years ago. The new machines have greater functionality (they try to allow the user to do more things), they address tasks that are more cognitive, they are more complex, and they are more interactive. Each of these characteristics makes more difficult the human factors problem of designing machines operable by their human users.

Increased functionality. The historical increase in machine functionality can be dramatically illustrated for jet plane cockpits. Fig. 1 shows the number of cockpit controls per crew member in U.S. fighter aircraft from World War I on. As can be seen, the increase is roughly exponential. The same picture occurs if instead we consider the number of cockpit displays (Fig. 2). In fact, the trend is close to universal. One could count buttons on food processors, functions on digital watches, or modes on video tape recorders.

My favorite example comes from a talk given a few years ago at the Human Factors Society annual meeting. A thermostat company was designing a new thermostat. During the arab oil embargo, people had been encouraged to turn their thermostats down at night to save energy. Soon there arrived on the market thermostats with two settings, one for day and one for night. Of course, in many households all the members work during the day, and so there is no need to heat the house then either. That led the company to consider another set of settings for the work hours during the day. But people only work five days a week, and so yet another set of settings is required to distinguish between the daytime settings on work days and on weekends. In fact, by the time the company had finished designing its prototype it had a microprocessor, a full QWERTY alphameric keyboard, an LED display, and they were beginning to wonder if people would accept it as a thermostat (it could equally well have passed for a wall-mounted word-processor).

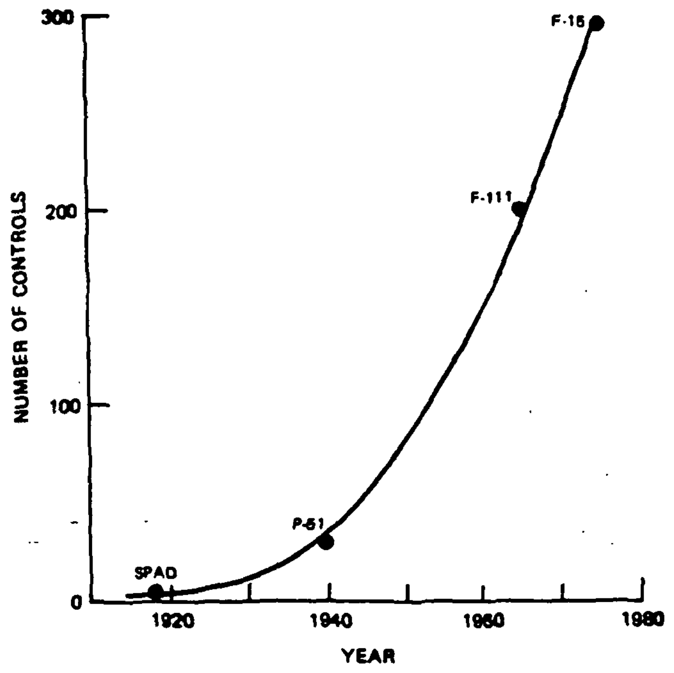


Fig. 1. Number of cockpit controls per crew member in U.S. fighter aircraft, WWI to 1980.

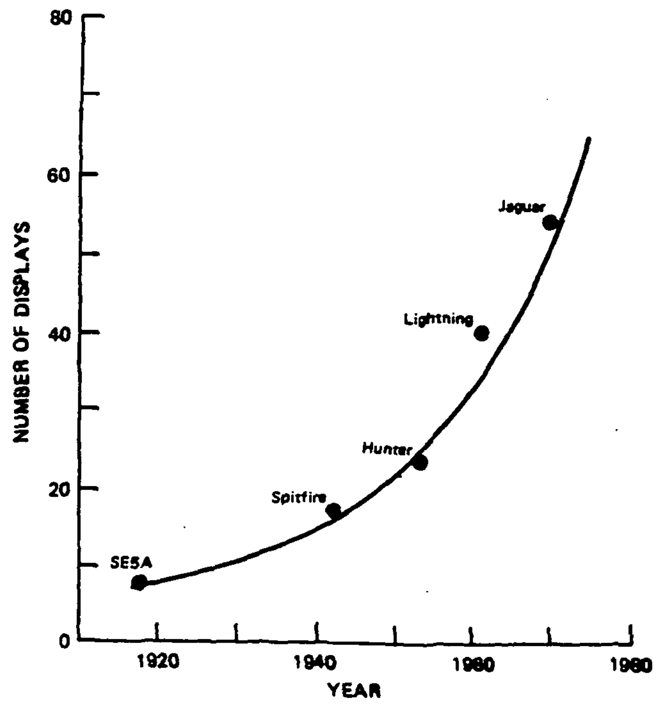


Fig. 2. Number of cockpit displays in British fighter aircraft, WWI to 1980.

And that is only the thermostat. The same garden path of reasoning could almost equally well apply to the refrigerator, the coffee pot, the can-opener, the microwave oven. Of course, it has not escaped us at Xerox that the same evolution can be expected for office computer systems.

More Cognitive Tasks. Computing systems are increasingly being applied to tasks that are more inherently cognitive in nature. Whereas a few years ago it was novel to apply computers to "word processing," now systems are beginning to be applied to so-called "idea processing." John Seeley Brown in my laboratory calls this the trend of moving from the processing of form to the processing of content.

More Complex Applications. The applications we are now attempting to perform with the aid of computers are just plain more complex than those attempted before, especially the applications in daily life. The management of a flexible computer tutor, the automating of office work, the computer-based flight of aircraft, are all very complicated things to do. It is true that aircraft with complicated equipment on board have flown for years, but the way in which aircraft systems were constructed was to produce information in isolation and to make the pilot the integrating element in the system (National Research Council, 1982). Now we are seeking to build systems in which the automation will itself do part of the lower-level decision making and users will interact at a higher level with the system.

More Interactive. Bitmapped displays support a higher bandwidth of interaction between the user and the system than the previous generation of alphanumeric displays. In the Xerox Star system, the memory bandwidth is about 50 MHz. The entire screen is repainted out of memory 39 times/sec. This would swamp most computer memories (Smith, Irby, and Kimball, 1982). The earlier Xerox Alto bitmapped display, in fact, required 60% of the machine processing. The consequence of providing high bandwidth to the user can be seen in the high time-density of user interaction we observe in laboratory videotapes of users. These higher rates of interaction are used to involve the machine more intimately in a coöperative process. As a user moves the mouse, the node of a tree structure displayed on the screen may also move, the links to other nodes turning and stretching so as to maintain the proper connections. This mode of interaction contrasts quite sharply with the turn-taking of older aphanumeric systems.

There is no obvious extension of results that appear in any human factors text that I know that would suffice for doing the human engineering of such systems. Advances in the human engineering of the computing systems we are now building really require additions to the discipline of human factors itself. It has been said that the most progress in science takes place along the boundaries where one discipline meets another. Just as the combination of engineering and psychological concepts laid the basis on which human factors was originally founded, so too it now seems that the absorption of concepts from artificial intelligence, intelligent computer-aided instruction, linguistic discourse theory, interactive computer graphics, and general cognitive science are likely to help lay the basis for human-computer interaction. In order to make this point more concretely, it is useful to examine the notion of the "intelligent interface."

INTELLIGENT INTERFACES

The generic form of human-computer interaction is most evident in the case of an embedded computer, such as a computer used to help control an aircraft. In such a system, the potential for separating one part of the system that (partially autonomously) controls the flight of the airplane and another part of the system that mainly handles the users' flight displays and controls is easy to see. An abstract representation of such a system is represented in Fig. 3 and will be termed the Triple Agent Model. This model is a derivative of Sheridan's "supervisory control" model (Sheridan, 1984). In the diagram, the computing system comprises four parts: the user, a User Discourse Machine interacting with the user, a Task Machine interacting with the task, and the task. In a system having embedded computers, such as an airplane or an underwater robot, both the User Discourse Machine and the Task Machine may be independent computers and may, in fact, be physically separate from each other (e.g., Sheridan and Verplank, 1978). In a personal workstation computer, both could be modules within the same machine. The drive within computer science to separate the code for a "user interface management system" from general application code shows the movement toward the kind of logical separation implied in Fig. 3 (SIGGRAPH, 1983). This model illustrates what Young (1984) has called the Third Law of Expert Systems (he did not propose a First or Second Law):

> *<u>Third Law of Expert Systems</u>. Every expert system has to contain knowledge about two domains, (1) the subject area of the expert system, and (2) how to communicate with the user.*

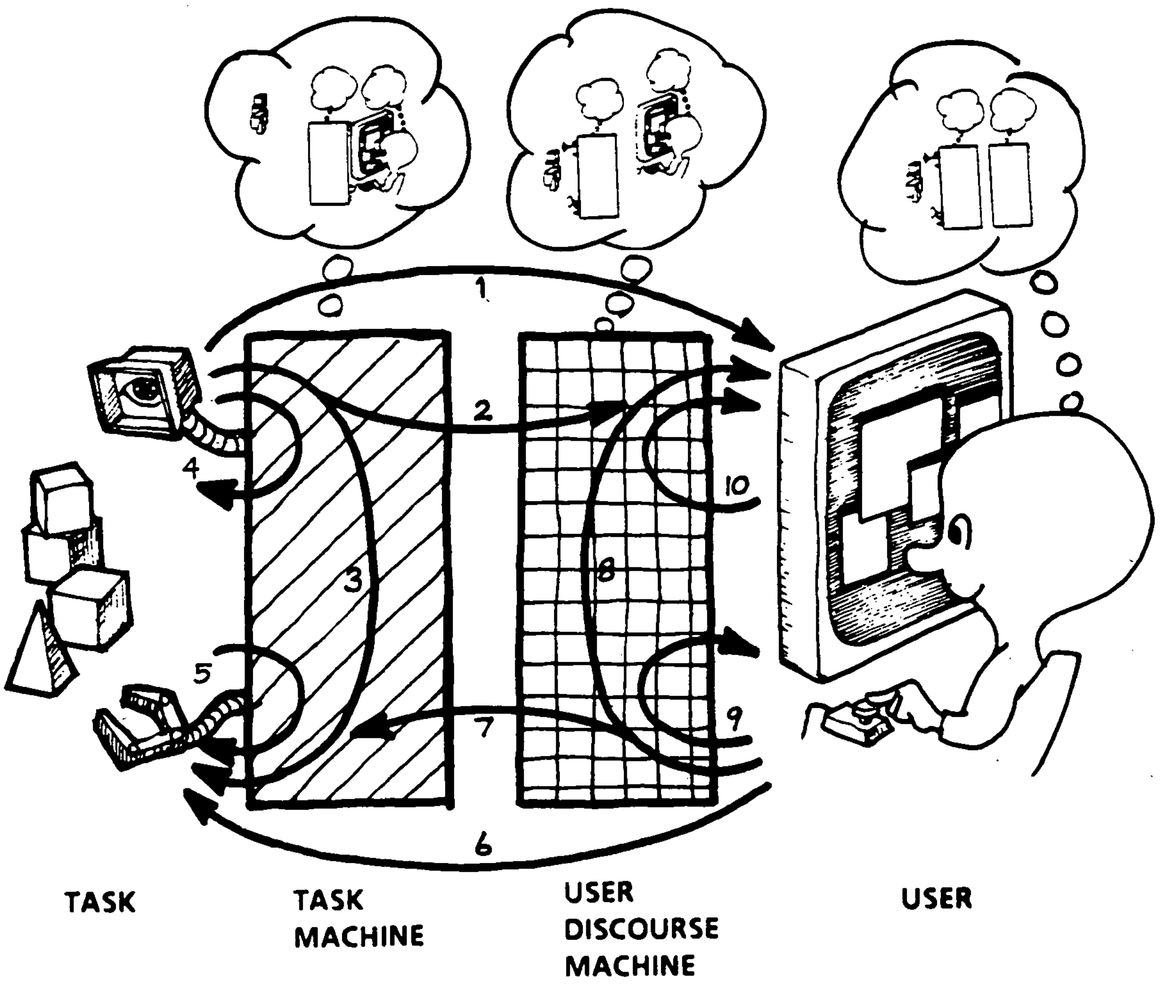


Fig. 3. **Triple Agent Model of Human-Computer Interaction.** Control loops possible: (1) Task is observed directly by human user. (2) Task is observed indirectly through sensors, computers, and displays. This Task Machine feedback interacts with User Discourse Machine feedback. (3) Task is controlled within Task Computer automatic mode. (4) Task is affected by the process of being sensed. (5) Task affects actuators and in turn is affected. (6) Human user directly affects task. (7) Human user affects task indirectly through controls, User Discourse Computer, and actuators. This control interacts with that from the Task Computer. (8) Human user gets feedback from User Discourse Machine. (9) Human user adjusts control parameters. (10) Human user adjusts display parameters. (Inspired by Sheridan, Fischhoff, Posner, and Pew, 1983, Fig. 4–1.)

In fact, as illustrated in the Triple Agent Model, the "Law" is much broader than Young suggested and ought better be called the Third Law of Interactive Application Systems (like Young, we leave the First and Second Laws until some other time):

> *Third Law of Interactive Application Systems. Every interactive application system has to contain knowledge about two domains, (1) the subject area of the application system, and (2) how to interact with the user.*

The term *application system* is taken here very broadly. It includes not only the usual applications of interactive computers (including expert systems applications) but also continuous process control, vehicle control, and embedded systems containing any of these.

There are three agents in Fig. 3 (hence the model's name): two computational agents (the Task Machine and the User Discourse Machine) and the user. Each of these agents potentially can have models of the other agents, including, conceivably, models of the other agents' models (and their models of the other agents' models (and their models of ...)). The existence of a well-developed User Discourse Machine for a system leads to the following observation:

> *Observation 1. Human-computer interaction with a system having a developed User Discourse Machine is less like the use of a tool by a human than it is like the conversation between intelligent agents.*

A partial consequence of Observation 1 is that the potential control and data flows in Fig. 3 are complicated and range all the way from a direct connection betwen the user and the task (the pilot looks out the window and moves the wing control surfaces manually); to subservient interaction in which the machine proposes a course of action then asks the user for approval; to complete automation where the Task Machine acts autonomously, taking action on its own initiative and not informing the user (the lower level control loops for the stabilization of the aircraft in advanced planes). Either the computer or the user can act as slave to the other or can act autonomously.

Sheridan and his colleagues (Sheridan and Ferrell, 1967; Sheridan and Verplank, 1978; Sheridan, Fischoff, Posner, and Pew, 1983, Sheridan, 1984) were led to a partitioning of the processing of operator feedback control loops between a task-oriented part and a user oriented part by considerations of how lower-level control tasks could be automated leaving the operator higher-level control tasks, hence their name "supervisory control" as opposed to "manual control," the control paradigm it replaced. But we can come to similar conclusions based on considerations of how computer code that interacts with the user is to be modularized so as to obtain a neat separation from the application code, or based on how to implement user interface management systems (SIGGRAPH, 1983; Hartson, Ehrich, and Johnson, 1984).

We are now ready to observe an obvious, but still fundamental, fact about the application of artificial intelligent techniques to application programs:

> Observation 2. *Intelligence may be applied at different sites of the Triple Agent Model, thereby resulting in different classes of intelligent systems.*

Artificial intelligence can be applied to the Task Machine of the system, or it can be applied to the User Discourse Machine both giving intelligent systems, but of very different sorts. And within these main approaches, different techniques could be emphasized. One system might play on its strength for knowledge representation, for example, whereas another might concentrate on rules for discourse. The difference between intelligence in the Task Machine and intelligence in the User Discourse Machine can be illustrated by the difference between MYCIN (Shortliffe, 1974; Davis, Buchanan, and Shortliffe, 1977) and WEST (Burton and Brown, 1982).

MYCIN: Example of an Intelligent Task Machine

MYCIN is an example of an AI program where the major emphasis is on the Task Machine. MYCIN is an expert system developed to provide advice on the diagnosis and treatment of infectious blood deseases. The program encodes expert knowledge as a set of rules, such as

```
PREMISE:   (SAND    (SAME CNTXT GRAM GRAMNEG)
                    (SAME CNTXT MORPH ROD)
                    (SAME CNTXT AIR ANEROBIC))
ACTION:    (CONCLUDE CNTXT IDENT BACTEROIDES TALLY .6),
```

which roughly translates into English as

```
If:    1) The stain of the organism is gram-negative, and
       2) the morphology of the organism is rod, and
       3) the aerobicity of the organism is anerobic,
then:  there is suggestive evidence (.6) that the identify of the organism is
       baceteroides.
```

The program has no explicit model of medical knowledge. Reasoning is done by backward chaining through its rules, eventually arriving at a diagnosis and suggested treatment. This part of the program, the Task Machine in our terms, has been shown in formal evaluations to perform at expert level (Yu, Buchanan, Shortliffe, Wraith, Davis, Scott, and Cohen, 1979*a*; Yu, Fagan, Wraith, Clancey, Scott, Hannigan, Blum, Buchanan, and Cohen, 1979*b*).

Since MYCIN has to communicate with physicians somehow and since it is important for physicians to be able to understand how the program arrived at its conclusions, it also has a User Discourse Machine portion that could, in addition to asking the user to supply it with various laboratory and other data, respond to queries of why it had reached some conclusion. For example, if the system said:

```
23) IS J. SMITH A COMPROMISED HOST?
```

the user could query,

```
**WHY
```

to which the system would reply

```
[3.0] THIS WILL AID IN DETERMINING WHETHER J. SMITH IS IMMUNOSUPPRESSED.
IF
    [3.1] J. SMITH IS NOT A COMPROMISED HOST
THEN
    IT IS DEFINITE (1.0) THAT J. SMITH IS NOT IMMUNOSUPPRESSED
    [RULE343]
```

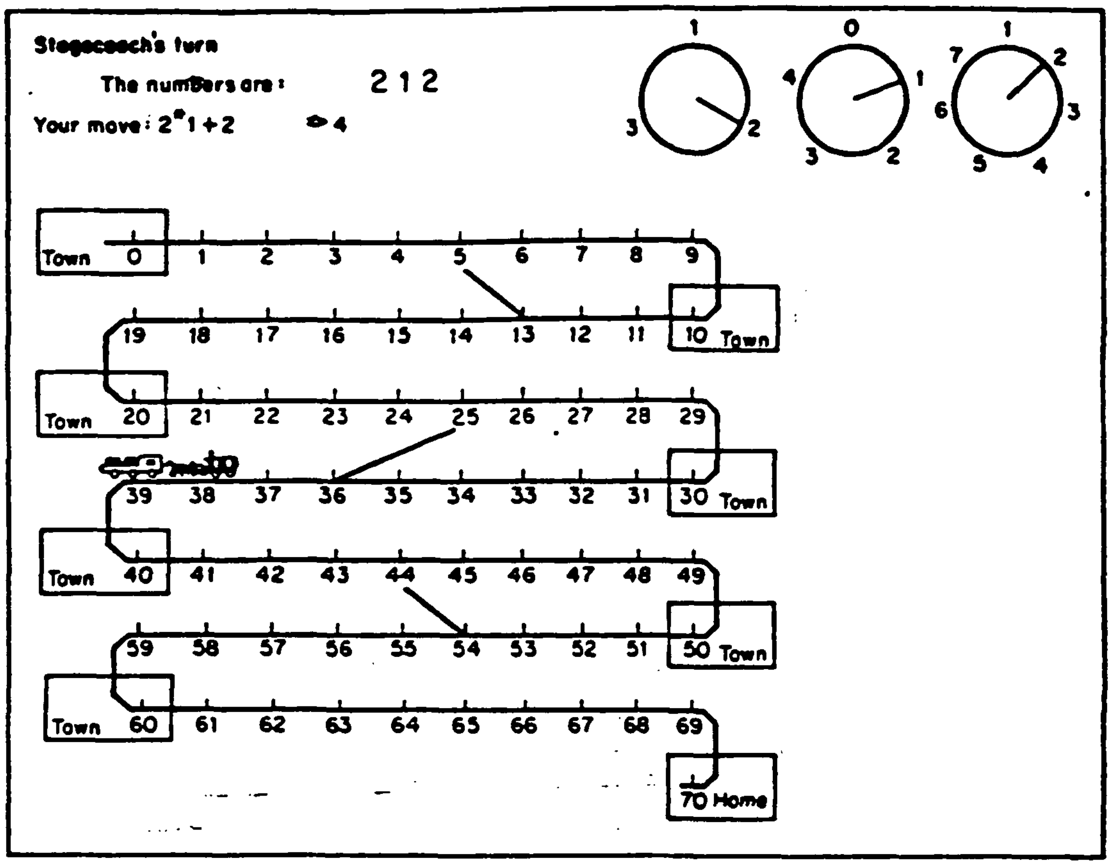
While this attempt at a User Discourse Machine was clever and did make it possible to examine part of the machine's reasoning, there are severe limits to what users could learn about the machine's reasoning that now have become clear in retrospect (Clancy, 1983, Young 1984). For example, the machine can not justify its rules or explain why it considers some hypotheses before others. But the main point for our present concern is that MYCIN is an example of a system where the main effort at intelligence is directed at the Task Machine portion of the system.

WEST: Example of an Intelligent User Discourse Machine

By contrast, in WEST, the main effort at intelligence is directed at the User Discourse Machine portion of the program and relatively little at the Task Machine. WEST is designed to help children practice arithmetic. The program displays a route between two cities (see Fig. 4). On each turn, the program spins three spinners, generating three numbers for a player. The player can combine these numbers using parentheses and the four arithmetic operators. Because there are shortcuts and bonuses for landing exactly on intervening towns, it is often advantageous for the player to combine the digits he spins so as to make a shorter move than the maximum obtainable number. As part of the User Discourse Machine, from time to time, a "coach" pops up and gives unsolicited advice on strategy, or the player may himself request a hint. For example, the coach might say:

> Perhaps you have forgotten that it's OK to rearrange the numbers. You could get to be really good if you tried using other orderings. One good example would be the expression: (2*4) + 1, which would have resulted in a TOWN! YOU would have ended up at 70 with the COMPUTER finishing up the turn at 54.
>
> Would you like to take your turn over?
>
> => | YES | NO |

The User Discourse Machine for WEST addresses the problems of (1) when to break into the game to offer advice and (2) what to say. To do this it uses an "Issues and Examples" paradigm. Issues are characterizations of where the user is weak. Examples are possible moves that could be made on a particular turn. The program accumulates evidence from the


Stagecoach's turn

The numbers are : 2 1 2

Your move : 2*1+2 => 4

Fig. 4. **Game display for WEST.** (Burton and Brown, 1982, Fig. 2.) The gamboard above shows two tokens moving along a number line from starting to ending towns. If the player lands on the beginning of a shortcut, his token is advanced to the end of the shortcut. If the player lands on a town, his token is advanced to the next town.

player's behavior over several turns for which issues characterize his weaknesses. It decides when to break in and gives advice to the user based on which issues it has evidence for and what opportunities for making dramatic points about these issues are presented by the possible moves the current turn provides as examples. If, on previous turns, the player has rather consistently missed short cuts to his disadvantage, then SHORTCUTS becomes an issue. If later, the player has the opportunity to beat the computer by taking a shortcut, but fails to see it, the machine seizes on this example as an excellent and memorable opportunity for coaching the player about shortcuts.

The issues in WEST are elements of the skills required for playing the game, such as SHORTCUTS, PARENTHESES, SUBTRACTION, etc. On each turn, recognizers for each issue examine the player's move to see whether the student could have gotten a better score if he had used the tactic associated with the issue. As a result of these recognizers, the system tallies statistics in its model of the student (Fig. 5). Evaluator routines work on the student model to decide on which issues the system can determine that the student is weak (the evidence on a particular issue may be inconclusive at a given time). Given the issues on which the student can be shown to be weak and given the possible moves available on a given turn as examples, the system proceeds according to a number of principles (Burton and Brown, 1982):

Principle 1. *Before giving advice, be sure the Issue used is one in which the student is weak.*

Principle 2. *When illustrating an Issue, only use an Example (an alternative move) in which the result or outcome of that move is dramatically superior to the move made by the student.*

Principle 3. *After giving the student advice, permit him to incorporate the Issue immediately by allowing him to repeat his turn.*

Principle 4. *If a student is about to lose, interrupt and tutor him only with moves that will keep him from losing.*

Principle 5. *Do not tutor on two consecutive moves, no matter what.*

Principle 6. *Do not tutor before the student has had a chance to discover the game for himself.*

Principle 7. *Do not provide only criticism when the Tutor breaks in! If the student makes an exceptional move, identify why it is good and congratulate him.*

Principle 8. *After giving advice to the student, offer him a chance to retake his turn, but do not force him to.*

Principle 9. *Always have the Computer Expert play an optimal game.*

MOVES	(A + B)*C	(A*B) + C	(A + B)–C	(A*B)/C	A/(B*C)	A*(B–C)	(A*B)–C
Best	1	3	–	–	–	–	–
Good	–	1	–	–	–	–	–
Fair	–	1	–	–	–	–	1
Poor	–	1	–	1	–	–	–
MISS	2	1	1	1	1	1	–

SPECIALS	Took	Missed
Town	3	2
Bump	–	1
Shortcut	–	–

DIRECTION	Good	Poor	Was Best
Forward	5	4	9
Backward	–	–	–

SPIN–ORDER	Good	Poor
Original	2	3
Reverse	–	1
Decreasing	1	–
Increasing	–	–
Other	2	–

PARENTHESES	Need & Use	Use Anyway	No Use	None & & Miss
	1	–	8	3

STRATEGY	Max Delta	Max Dist.	Max Number	Specials	Other
	3	4	2	3	4

Fig. 5. WEST Model of Student.

Principle 10. If the student asks for help, provide several levels of hints.

Principle 11. If the student is losing consistently, adjust the level of play.

Principle 12. If the student makes a potentially careless error, be forgiving. But provide explicit commentary in case it was not just careless.

WEST thus represents the invention of a considerable amount of mechanism in the service of the User Discourse Machine: conversational turn-taking (when it should break in), explanation, diagnosis from user behavior of the state of user knowledge, and even user motivation and emotions. By contrast the Task Machine (computing the best move on each turn) is relatively straight forward.

The application of intelligence to the User Discourse Machine is my provisional choice for what we mean by an intelligent interface. Such systems can come in a variety of forms: intelligent front-ends to powerful, but hard to use, applications programs; advisor programs; coaching programs; programs that allow natural language interaction. Whatever the exact form, the clear intent of all such applications of AI to the User Discourse Machine is to make systems easier for people to use, largely by attempting to deal with the cognitive or linguistic *content* of the communication task with AI tools. But making machines easier to use lies in the center of human factors as a discipline, as well. For this reason, progress on the AI end will not leave human factors unchanged. Either human factors will absorb into itself the concepts from AI that make interactive machines more possible, or it will cease to be the major discipline concerned with this area, or, an interesting possibility, there will emerge a new subdiscipline of human-computer interaction at the intersection of the several disciplines with potent contributions to make in this area. This latter is an attractive possibility because there are other disciplines whose recent results are also important: Interactive computer graphics is concerned with algorithms underlying graphics interfaces, computer science itself is concerned with developing "user management dialogue systems," linguistic discourse analysis is concerned with conceptual analyses of conversational interaction.

PROBLEMS OF THE INTELLIGENT INTERFACE

It must be said that the design of intelligent interfaces is difficult and not well understood. We have said that instead of thinking of a human operating a machine, it is now appropriate to think of communication between intelligent agents. This means the machine no longer does necessarily what it is told to do, but what it thinks the user's *intention* is, or even what it thinks ought to be done if it knows the user's intention is otherwise. I have stressed the impact of developments in artificial intelligence and intelligent tutoring systems may have on human factors. Young (1984) has made the other side of the case, arguing that interactive AI programs need to pay more attention to the human factors of their interfaces. These two arguments are not incompatible. As we have seen, it is possible to build AI systems that concentrate on the Task Machine giving little attention to the human interaction. I will now argue that there are still a considerable number of problems to overcome in the technology of constructing intelligent interfaces. These problems are important limiters of what presently can be done. Work on them forms part of the research agenda for human-computer interaction. Three examples are: the rôle problem, the automation problem, and the communication problem.

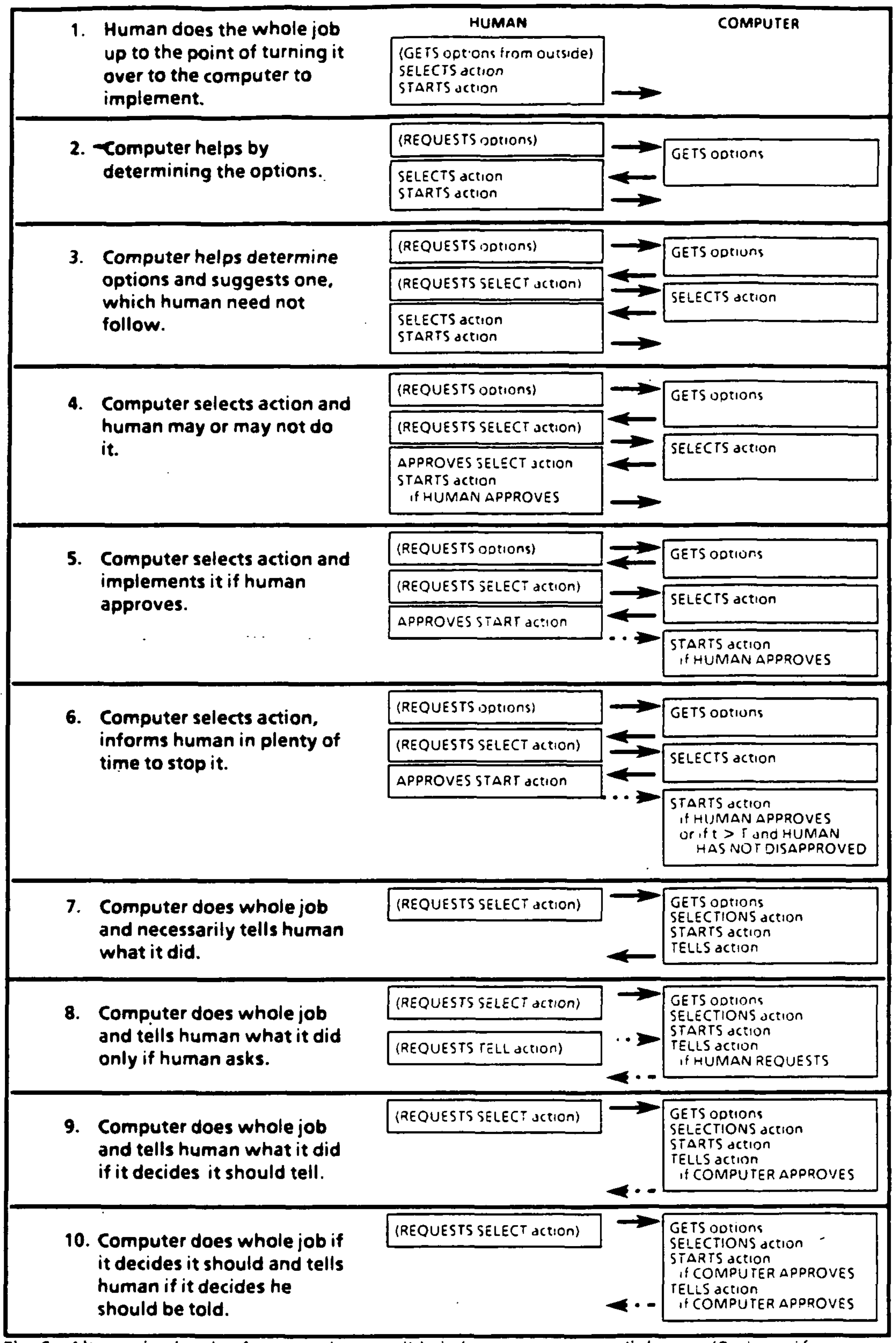
The Rôle Problem. Young (1984) has pointed out that an expert system can be designed to function in a number of rôles vis-a-vis the user. For example, he points out that MYCIN is strongly committed to being the partner who actually makes the decisions. This choice of rôle may influence the acceptance of the program by users quite independently of the programs actual performance expertise. WEST explicitly adopts the rôle of coach; in fact, the research question for WEST is really, how can the rôle of coach be modeled and implemented in a machine? The question of rôle is related to the question of persona. How should the system present itself? Should it have the persona of a person or a tool or an environment? MYCIN and WEST both try to establish personae as artificial people. MYCIN uses natural language dialogue and a heavy dose of medical mannerisms to establish the persona of a consulting physician. WEST uses a small cartoon picture and breezy English dialogue to establish the persona of an athletic coach.

The Automation Problem. There is a long history of problems that have arisen under the name of automation (National Research Council, 1951; National Research Council, 1982), especially in connection with the automation of aircraft control systems, since it was in this area that systems of limited intelligence first occurred. The problem is often stated in the form of, which tasks should the human do, and which the machine? As automation work spread to underwater robots and teleoperation, similar sets of problems began to be studied under the term "supervisory control." In the case of intelligent tutoring systems, AI expert systems, and even the general human-computer interface, we again see a familiar set of problems arising, this time in the context of discrete symbolic systems. The possibilities inherent in the combinations of control loops in Fig. 3 result in a fair-sized design space of automation alternatives. This space has much more structure than the simple division into machine-initiative, user-initiative, and mixed-initiative categories (Carbonell, 1970). One set of possibilities identified by Sheridan and Verplank (1978) is given in Fig. 6. The list runs the gamut from direct control by the human (corresponding to paths 1 and 6 in Fig. 3), to complete control by the machine (path 3 in Fig. 3), to some of the many combinations of other paths.

The list serves to indicate how complex the possibilities are. In fact, even more possibilities are possible that are not on the list: The words *human* and *computer* can be reversed. WEST presents still another possibility, the human selects the option, but the computer gives advice if a sample of the user's decisions suggest it is needed. (MYCIN is an example of option 4 above, the computer selects the action and the human may or may not do it.) If we distinguish the computer (or human) explaining why it did something from mere telling that it did it, we have yet another set of possibilities. This puts us beyond the older form of the automation question, which tasks should the human do and which the machine? and into the much more complex consideration of how should they coöperate? The question is not dissimilar from the question of how can *humans* be organized to coöperate on a task?

The Communications Problem. It is not surprising that many issues of the intelligent interface end up under the category of commications problems. First, there is the issue of trying to find a framework of interacting with the user that is open, but natural and predictable. This desire leads naturally to attempts to apply and generalize linguistic discourse theory so that it applies to computer interfaces. For example, Grice (1975) has put forward the notion that human conversation takes place in a number of "conversational moves," each taking the conversation to a new state in the discourse. Grice formulates a number of maxims and defines inappropriate conversational moves as moves that violate these maxims. Reichman-Adar (1984) has taken this approach further and attempted to describe, using an augmented transition-state network, a way in which conversational moves may be mechanized. Such discourse notions as conversational focus; control of turn-taking; opening, narrowing, and closing a conversational context; time scale; and deictic (pointing) reference need to be examined in the context of computer-based dialogue.

Second, there needs to be more work on modeling the dynamic, communication-induced, changes in the user. In order to carry on an intelligent conversation with the user the system must be able to diagnose in an approximate way the user's state of knowledge about the topic of conversation (WEST spends considerable effort attempting to do this) and the system must contain the implementation of a theory of explanation.



	HUMAN	COMPUTER
1. Human does the whole job up to the point of turning it over to the computer to implement.	(GETS options from outside) SELECTS action STARTS action	
2. Computer helps by determining the options.	(REQUESTS options)	GETS options
	SELECTS action STARTS action	
3. Computer helps determine options and suggests one, which human need not follow.	(REQUESTS options)	GETS options
	(REQUESTS SELECT action)	SELECTS action
	SELECTS action STARTS action	
4. Computer selects action and human may or may not do it.	(REQUESTS options)	GETS options
	(REQUESTS SELECT action)	SELECTS action
	APPROVES SELECT action STARTS action if HUMAN APPROVES	
5. Computer selects action and implements it if human approves.	(REQUESTS options)	GETS options
	(REQUESTS SELECT action)	SELECTS action
	APPROVES START action	STARTS action if HUMAN APPROVES
6. Computer selects action, informs human in plenty of time to stop it.	(REQUESTS options)	GETS options
	(REQUESTS SELECT action)	SELECTS action
	APPROVES START action	STARTS action if HUMAN APPROVES or if t > T and HUMAN HAS NOT DISAPPROVED
7. Computer does whole job and necessarily tells human what it did.	(REQUESTS SELECT action)	GETS options SELECTIONS action STARTS action TELLS action
8. Computer does whole job and tells human what it did only if human asks.	(REQUESTS SELECT action)	GETS options SELECTIONS action STARTS action TELLS action if HUMAN REQUESTS
	(REQUESTS TELL action)	
9. Computer does whole job and tells human what it did if it decides it should tell.	(REQUESTS SELECT action)	GETS options SELECTIONS action STARTS action TELLS action if COMPUTER APPROVES
10. Computer does whole job if it decides it should and tells human if it decides he should be told.	(REQUESTS SELECT action)	GETS options SELECTIONS action STARTS action if COMPUTER APPROVES TELLS action if COMPUTER APPROVES

Fig. 6. Alternative levels of automation possible in human-computer dialogue. (Redrawn from Sheridan and Verplank, 1978)

Third, the way in which communication with the user interacts with the constant visual presentations available to him needs to be better understood. The graphics display allows the system to make a conceptual model compelling to the user and to give memory support for the components of that conceptual model. Furthermore, the user can specify by pointing or gesture things to the system that would otherwise be difficult to express.

HUMAN FACTORS AND ARTIFICIAL INTELLIGENCE

Communications with future systems will surely be more subtle than with today's systems. Mouse-pointings, gestures, command buttons, typed commands, voice, voice intonations, and eye gaze are all input modes within the current state-of-the-art. Explanation, diagnosis, and negotiated information retrieval are some of the modes of interaction desired. The human engineering of these systems will only be possible by collecting the results from sciences outside the traditional sphere of human factors.

Both human factors and artificial intelligence have disciplinary interests in human-computer interaction. In artificial intelligence this interest has been expressed in work on the intelligent interface. Theoretical work in the related disciplines of intelligent computer-aided instruction, computer graphics, discourse linguistics, cognitive psychology, and computer science also look relevant.

Just as the combination of engineering and psychological concepts laid the basis on which human factors was originally founded, it may now be appropriate to attempt to lay a new base for human-computer interaction based on these cognitive and computer sciences above. This could be done within the human factors discipline, but it could also occur largely outside of it, in a new subdiscipline of human-computer interaction. Whichever is the final choice, AI and the intelligent interface are sure to have permanent affects on human factors as a field of study.

REFERENCES

Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Carbonell, J. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems. MMS-11* (4).

Clancey, W. J. (1983). The epistemology of a rule-based expert system–A framework for explanation. *Artificial Intelligence 20*, 215–251.

Davis, R. Buchanan, B. and Shortliffe, E. H. (1977). Production rules as a representation for a knowledge-base consultation program. *Artificial Intelligence 8*, 15–45.

Grice, H. P. (1975). Logic and conversation. In P. Cole and J. Morgan (eds.), *Syntax and Semantics*, 41–58. New York: Academic Press.

Hartson, H. R.; Ehrich, R. W.; Johnson, D. H. (1984). The management of dialogue for human-computer interfaces. Department of Computer Science, Virginia Tech, Blacksburg, Virginia, April.

National Research Council (1951). *Human Engineering for an Effective Air-Navigation and Traffic-Control System*, (edited by P. M. Fitts). Washington, D. C.: National Academy Press.

National Research Council (1982). *Automation in Combat Aircraft*. Washington, D. C.: National Academy Press.

Reichman-Adar, R. (1984). Extended person-machine interface. *Artificial Intelligence 22*, 157–218.

Sheridan, T. B. and Ferrell, W. R. (1967). Supervisory control of manipulation. In *Proceedings of the 3rd Annual Conference on Manual Control*, 315–323. NASA SP-144.

Sheridan, T. B. and Verplank, W. L. (1978). Human and computer control of undersea teleoperators. M. I. T. Man-Machine Systems Laboratory Report.

Sheridan, T. B.; Fischhoff, B.; Posner, M.; and Pew, R. W. (1983). Supervisory control systems. In National Research Council, *Research Needs for Human Factors*. Washington, D. C.: National Academy Press.

Sheridan, T. B. (1984). Supervisory control of remote manipulators, vehicles and dynamic processes: Experiments in command and display aiding. In *Advances in Man-Machine Systems Research*, Vol. 1, 49–137, JAI Press Inc.

Shortliffe, E. H. (1974). MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection. Ph.D. Dissertation, Medical Information Sciences, Stanford University; also: *Computer-Based Medical Consultations:* MYCIN. New York: Elsevier, 1976.

SIGGRAPH (1983). Graphical input interaction technique (GIIT): Workshop summary. *Computer Graphics 17* (1), 5–30.

Smith, D. C.; Irby, C.; and Kimball, R. (1982). The star user interface: an overview. National Computer Conference, Young, R. M. (1984). *Human interface aspects of expert systems.* MRC Applied Psychology Unit, Cambridge, August.

Yu, V. L.; Buchanan, B. G.; Shortliffe, E. H.; Wraith, S. M.; Davis, R.; Scott, A. C.; and Cohen, S. N. (1979*a*). Evaluating the performance of a computer-based consultant. *Computer Programs in Biomedicine 9*, 96–102.

Yu, V. L.; Fagan, L. M.; Wraith, S. M.; Clancey, W J.; Scott, A. C.; Hannigan, J. F.; Blum, R. L.; Buchanan, B. G.; and Cohen, S. N. (1979*b*). Antimicrobial selection by a computer–a blinded evaluation by infectious disease experts. *Journal of the American Medical Association 242*, 1279–1282.

A CATALOGUE OF USER/COMPUTER INTERACTIONS

prepared by

Roy Nakashima

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA

TABLE OF CONTENTS

INTRODUCTION

The technology of bit-mapped graphics and mouse pointing devices has made possible the development of a mode of interaction which is clearly different from interfaces based on teletypewriter technology. This new interaction style relies heavily on the use of graphic symbols such as icons, metaphors such as windows, and command selection by pointing at menus. Recently, the public's acceptance of the Apple MacIntosh personal computer has made it clear that this new style of user-interface design is becoming more universally accepted (Markoff, 1984). Systems based on this style have been in use at PARC since 1974. Many years of independent development has resulted in a surprising amount of diversity among them. New systems are continually being developed through a combination of innovation and the borrowing and adapting of ideas from existing systems.

Designers of new user interfaces face several problems, mostly related to communication. One of these problems is maintaining an awareness of what has already been designed. Ignorance of existing design solutions can result in wasted effort in "reinventing the wheel". Deviations from existing conventions can create compatibility problems with other systems. On the other hand, exposure to the work of other designers can act as a stimulus to creativity.

The second problem is that designers need to be able to communicate the stylistic aspects of a proposed user interface design without having to implement it on a live system. There are several reasons why this is useful. During the early stages of a project, it is desirable to generate and evaluate a large number of designs economically. The generation of a large quantity of preliminary design ideas increases the probability that a good idea will be found. The performance of at least preliminary evaluation of ideas or concepts before implementing them can reduce the amount of effort wasted pursuing inappropriate design directions. Once an idea or concept has passed preliminary evaluation, they must be communicated to programmers for implementation. All the stylistic, interactive aspects of the design must be described completely, yet concisely. After a system has been implemented, designers may want to document what has been done. Documentation specialists may wish to describe the operational aspects of the implemented system to novice users. Clearly, a method of efficiently communicating the stylistic aspects of user interface designs would be valuable to many computer professionals.

The third problem is that there does not yet exist a comprehensive, generic vocabulary of user interface techniques. Because of this, it is difficult for system designers from different companies or backgrounds to converse or discuss user interfaces without having to refer to specific products. It is hoped that, by cataloguing the many techniques which have been employed to date and determining a taxonomy for these techniques, we can move one step closer to the development of a generic vocabulary. Other possible benefits of such a taxonomy include the development of a methodology for selecting the best interaction technique for a given application and a directed exploration of the design spaces which are found to be underrepresented in the taxonomy.

This document contains a partial catalogue of interactive graphic techniques presently used to implement user interfaces in a variety of commercial and research computer systems, including Interlisp-D, Star, Smalltalk, Cedar and Apple MacIntosh. It combines text and graphics in an innovative way to provide an effective means of communicating the interactive, stylistic aspects of window system designs, both existing and proposed. It has also proven to be a useful tool for documenting user/computer dialogues for further study by interaction languate analysis.

The catalogue was compiled during the summer of 1984 while the author was employed as a summer research intern at Xerox Palo Alto Research Center. It is by no means complete and is intended mainly as a demonstration of an effective communication and documentation methodology. The author wishes to thank Stuart Card and Bill Verplank for their guidance and the initial idea of writing this document.

DESCRIPTION OF THE CATALOGUE

Organization

The catalogue was organized around relatively high level goals such as "copying a document" or "moving an icon" because it was felt that the quality of the user interface is determined more by the way in which elemental acts combine to form a command sequence than by the elemental acts themselves. The approach was taken that the best user interfaces are designed so that the *sequence* of events or actions required to specify a particular command form a gestalt which enables the user to fuse them into a single act.

Presentation

The catalogue has both graphic and written content. Although it is obvious that graphic content is necessary to communicate the graphic quality of an interface, supplementary written explanations are also required to keep the overall format as concise and economical as possible. A picture is not always worth a thousand words and it is sometimes easier to communicate a message with words than with pictures.

The catalogue shows both sides of the human/computer dialogue. The idea of conversational interaction with computers has been around for some time (Nickerson, 1981). In order to accurately depict and evaluate the quality of this collaboration, it must be possible to show with equal clarity and detail the actions and responses of both the user and the computer.

Grain of Analysis

The temporal grain of analysis was selected to match the experiences of the user. Each command sequence was broken down into a number of sequential frames, similar to those used in comic books or cinematographic story boards. One frame was provided for each *distinct* visual event experienced by the user. E.g., if two visual events occur so close together in time that they are perceived to occur simultaneously, then both of these events are depicted in a single frame.

The spatial grain of analysis was selected to match the visual characteristics of the user. It was assumed that the user's attention would be focussed on a relatively small portion of the screen, therefore in most cases only a fragment of the screen is depicted in each frame. Judgements regarding frame content were based purely on the subjective experiences of the author. For example, when describing the operation of the Star system, the function key pad is shown each time a function key must be pressed because the author never acquired enough proficiency with the system to memorize the position of the keys and manipulate them by touch, whereas the mouse is not shown each time a mouse button is pressed because the author never felt the need to look at the mouse.

The narrative content of the each of the frames is numbered to correspond to the exchange of conversational control between user and computer. Each numbered narrative segment describes the events which take place during one complete exchange of conversational control. Each narrative segment begins with a decision or task to be performed by the user and ends with the response provided by the system. The narrative segment begins at the point in time just after conversational control has been transferred from computer to user, includes the events preceeding and following the transfer of control back to the computer and ends at the moment the computer relinquishes control again.

As might be expected, there are a few exceptions to the rules described above. In order to accurately depict the subtleties of mouse techniques, it was necessary to choose a finer grain of temporal analysis during mouse buttoning activities. Although most users experience the clicking of the mouse button as a single event, in some cases it was necessary to illustrate this action with two frames in order to show events occurring while the button was being held down. Occasionally it was necessary to provide separate frames for visual events which occurred simultaneously, but were spatially separated. This can be justified by considering that only one of them is in the user's field of view and is obviously perceived first. The other is seen only when the user shifts his gaze and is therefore perceived as temporally separated from the first visual event, even though the two events occurred simultaneously. When this type of situation is presented to the user, it is depicted using two frames instead of just one. During cursor positioning tasks, the user is obtaining essentially continuous feedback from the computer throughout the performance of the task. Strictly speaking, there are hundreds of interactions occurring between user and computer. However, in order to keep task descriptions as concise as possible, and because it can be argued that the task is experienced as a single event, throughout this catalogue, cursor positioning tasks are illustrated with a single frame.

NOTES

1. Because of problems of compatibility of screen graphics data between the many different computer systems, it was decided that traditional graphic design techniques would be the most efficient way to generate this document (as opposed to more advanced electronic publishing methods). These include the creation of original artwork by hand, the use of variable reduction photocopiers to produce repetitive images of the required size, modification of photocopied images by hand to save time using opaque white paint and black felt tip markers, cutting and pasting photocopied images.

   On the other hand, it was found that the use of hardcopied bitmap representations can greatly speed up the process of screen depiction, especially when the images are repetitive or finely detailed such as pop-up menus, reverse video images, or window contents. If possible, hardcopy bitmaps should be printed at larger than full scale, since most of the frames are depict screen images at greater than full scale and because reducing photocopiers are easier to get access to than enlarging photocopiers.

2. Commercially available dry transfer media such as "Zip-a-tone" should be used to render "gray" textures used in backgrounds or scroll bars. Hardcopy of the actual bitmap used for the texture can also be used. Gray felt tip markers are not recommended for this purpose since they do not reproduce well.

3. When composing frames, the same care and principles should be followed as in composing a painting or photograph. Whenever possible, contextual cues should be included in the frame to provide the observer with some sense of the scale and screen location of the depicted view. The cursor and the edge of the screen are examples of good contextual cues.

4. Although the technique used in this catalogue is an effective means of communicating the interactive and graphic aspects of direct manipulation window systems, it has the disadvantages of being time consuming and requiring a certain amount of artistic skill, Clearly, a more efficient, "short-hand" version of this methodology would be useful to computer professionals as a quick and informal way of recording observations of user/computer interactions.

   One example of how this might be accomplished is shown in Appendix A. In this example, provided by Stuart Card, a keystroke model is used to record interactions between user and computer. Once the interactions have been reduced to this form, they can be further studied using interaction language analysis. These analyses allow researchers to make more meaningful comparisons between window systems and may lead to new insights into the nature of interaction languages.

These notations can be generated either by direct observation of a user or from inspection of the diagrammatic representations of user/computer interactions contained in this catalogue.

In the example, P refers to a cursor positioning task, K refers to a keystroke, M3 refers to the third mouse button.

REFERENCES

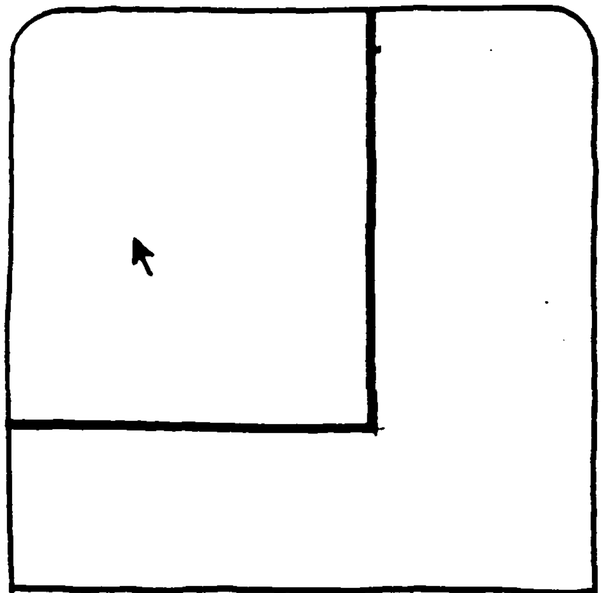
Markoff, J. 1984. "Five Window Managers for the IBM PC." *BYTE*, vol. 9, no. 9.

Nickerson, R. S. 1981. "Some Characteristics of Conversation", in B. Shackel, ed., *Man-Computer Interaction: Human Factors Aspects of Computers and People.* Rockville, Maryland: Sijthoff & Noordhoff.

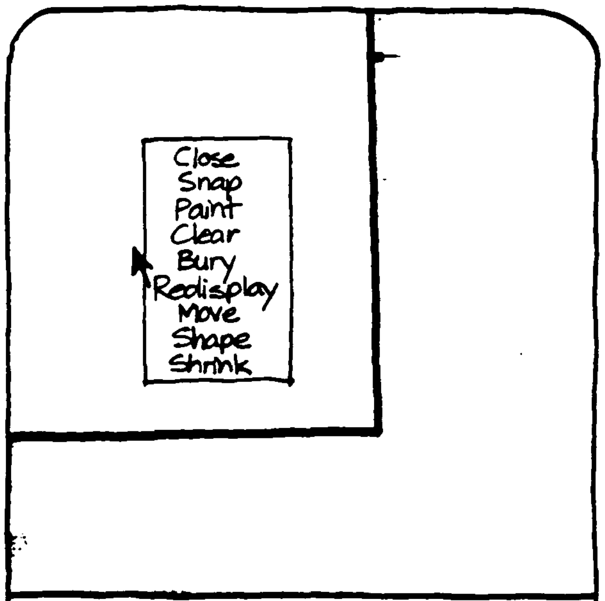
WINDOW OPERATIONS

TASK: MOVING A WINDOW PAGE 1 OF 2

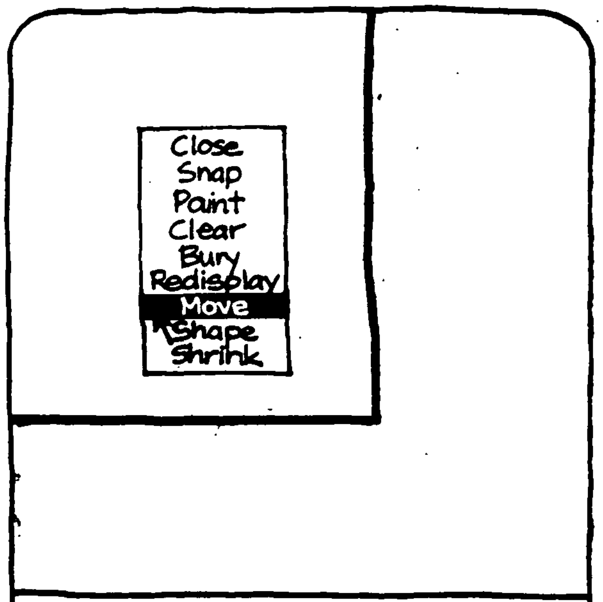
SYSTEM: INTERLISP-D



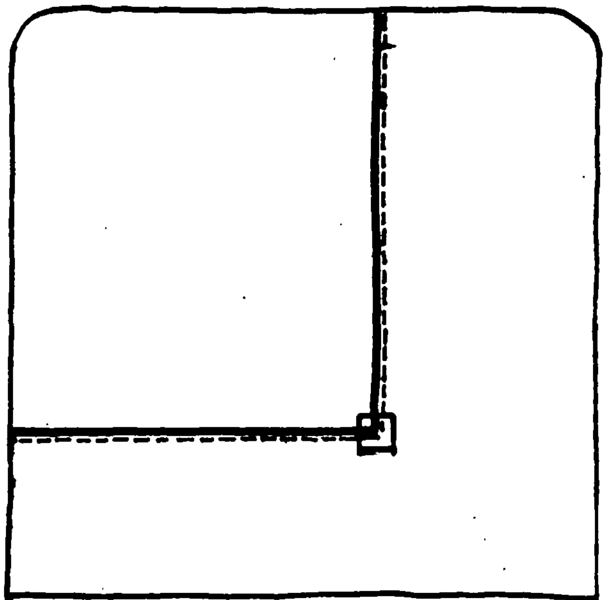
1. Position cursor anywhere in window.



2. Press and hold right mouse button. (Menu pops up)



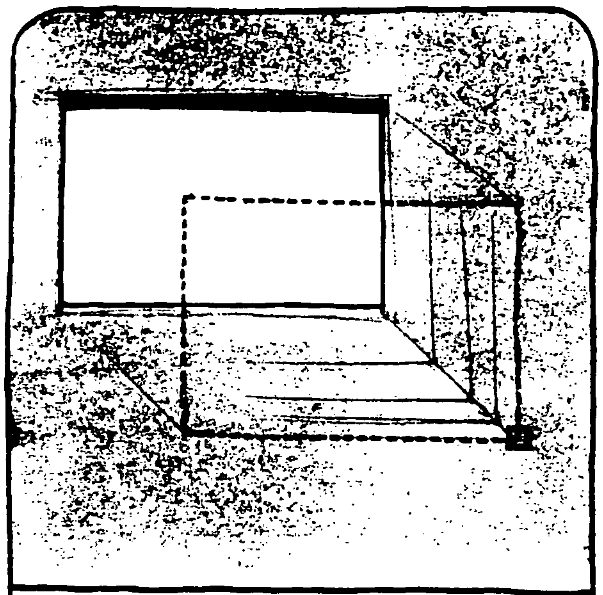
3. Position cursor on "Move" command. ("Move" reverses video.)



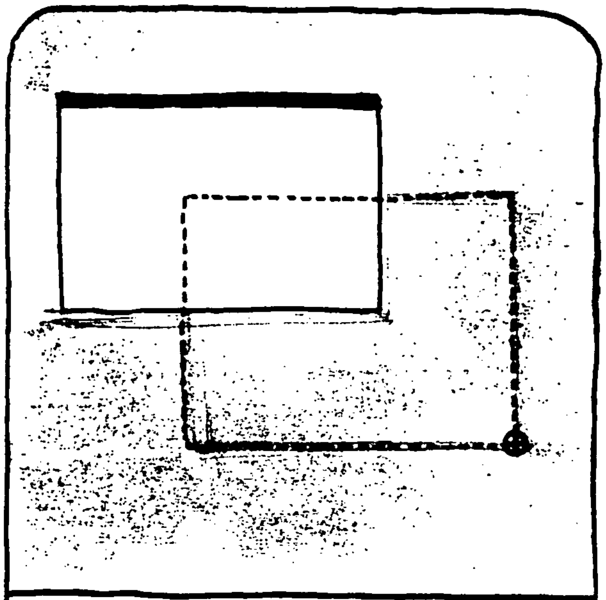
4. Release right mouse button. (Menu and cursor disappear. Square cursor appears at corner of window. Dashed line box appears around window.)

TASK: MOVING A WINDOW

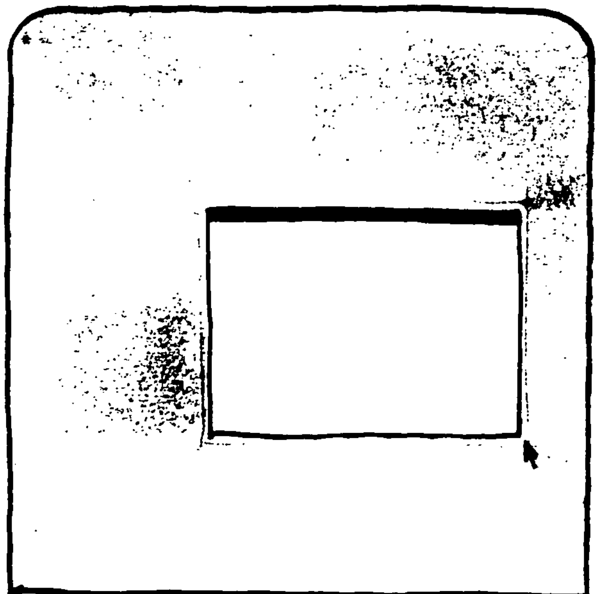
SYSTEM: INTERLISP



5. Move cursor to desired location of window corner. (Dashed line box follows movement of cursor.)



6. Press left mouse button. (As button is pressed, cursor changes to cross-hair shape.)

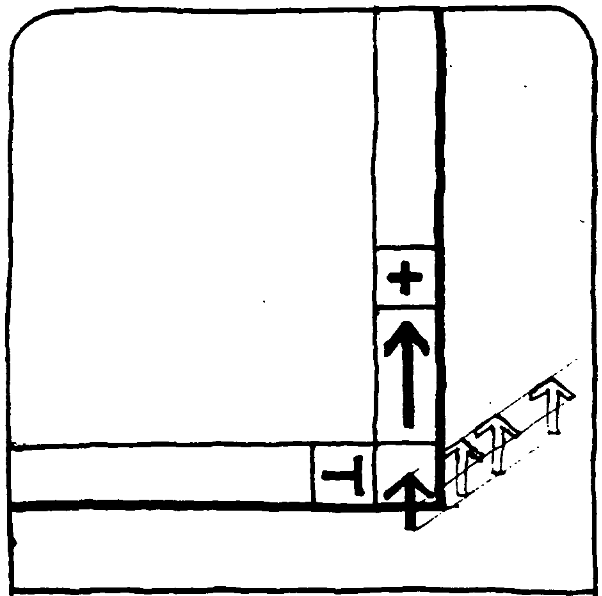


6. continued. (When button is released, window disappears and reappears in new location. Cursor reverts to normal shape.) <END>

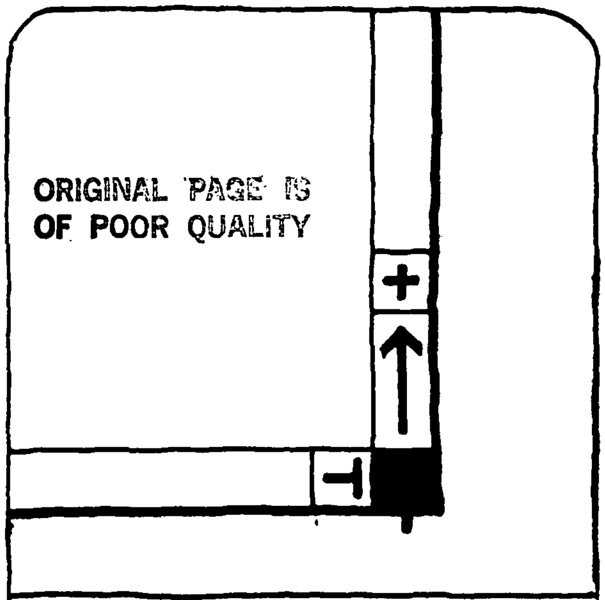
ORIGINAL PAGE IS
OF POOR QUALITY

TASK: ADJUSTING WINDOW SIZE/SHAPE PAGE 1 OF 2

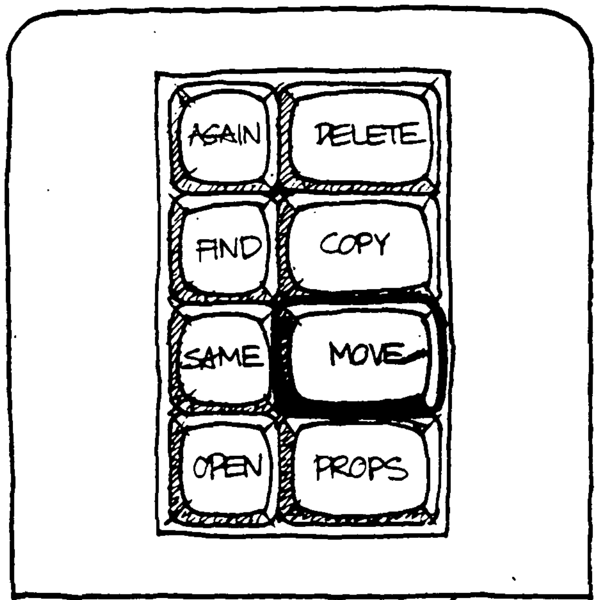
SYSTEM: STAR



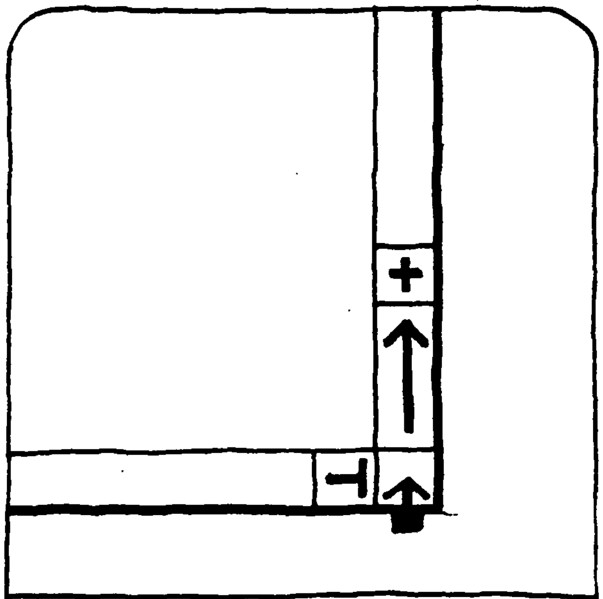
1. Position cursor on box in lower right corner of window.



2. Press left mouse button. (Box reverses video.)



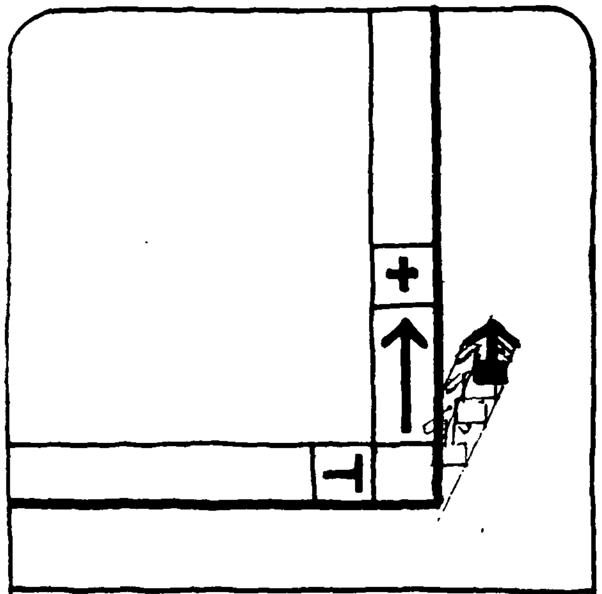
3. Press "MOVE" button. (Message appears in prompt window: "Please indicate a destination with either button.")



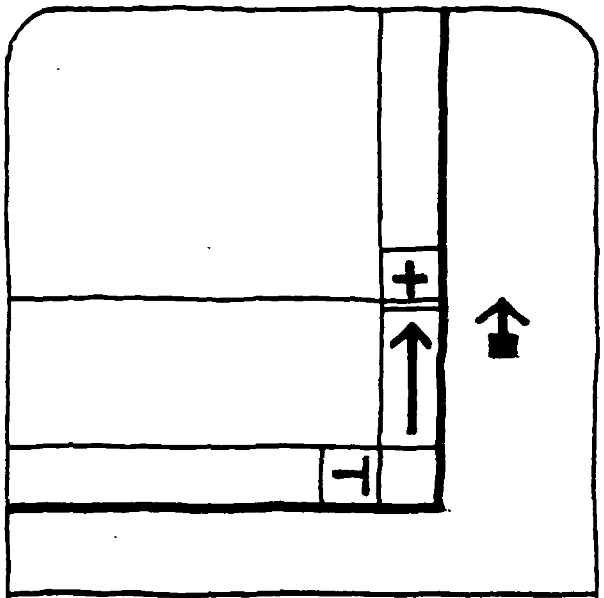
3. continued. (Box reverts to normal video. Cursor changes shape.)

TASK: ADJUSTING WINDOW SIZE/SHAPE PAGE 2 OF 2

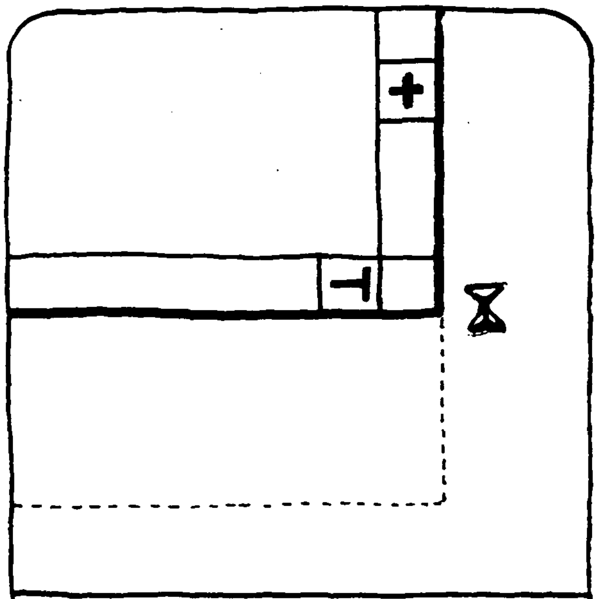
SYSTEM: STAR



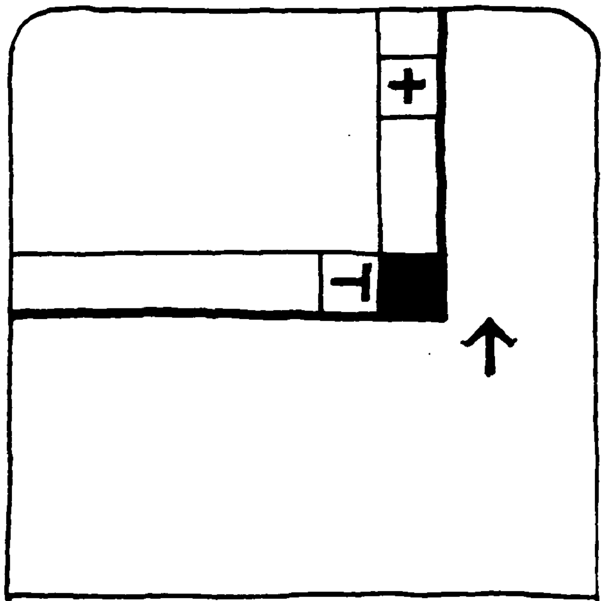
4. Move cursor to desired position of bottom of window frame. (Top and sides of window frame cannot be adjusted.)



5. Press either mouse button. (While button is held, horizontal line appears in window and follows vertical movement of cursor.)



5. continued. (When button is released, cursor changes to hour glass shape. Bottom of window frame disappears and reappears in new position.

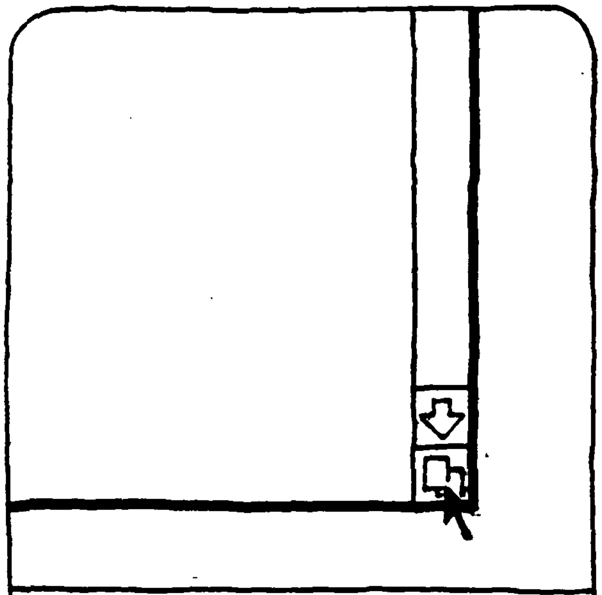


5. Continued (Cursor reverts to normal shape. Box reverses video.)

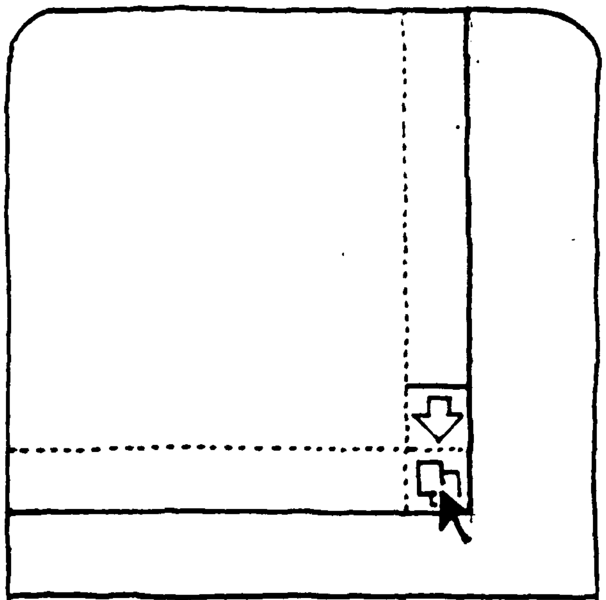
<END>

TASK: ADJUSTING WINDOW SIZE/SHAPE

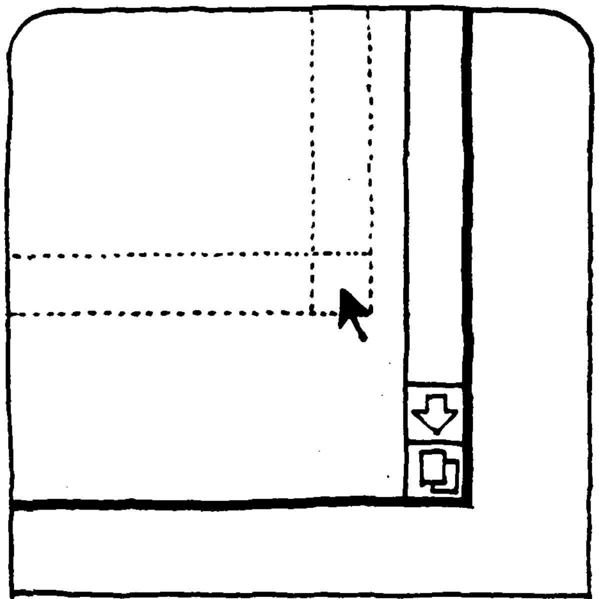
SYSTEM: MACINTOSH



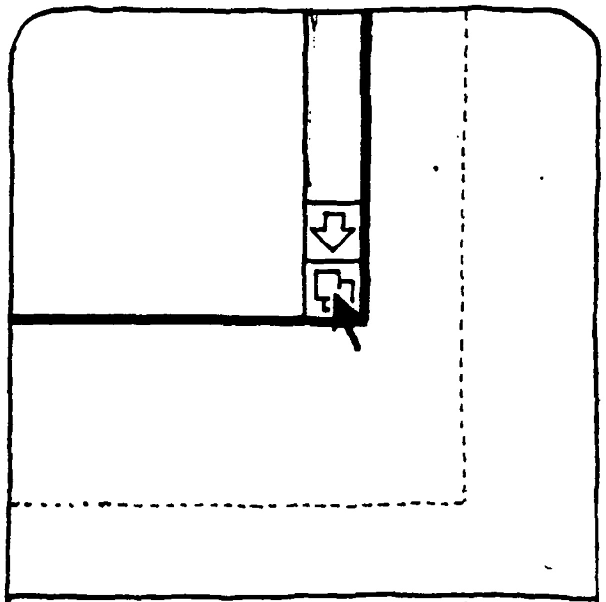
1. Position cursor on "Grow Region" in lower right corner of window.



2. Press and hold mouse button.

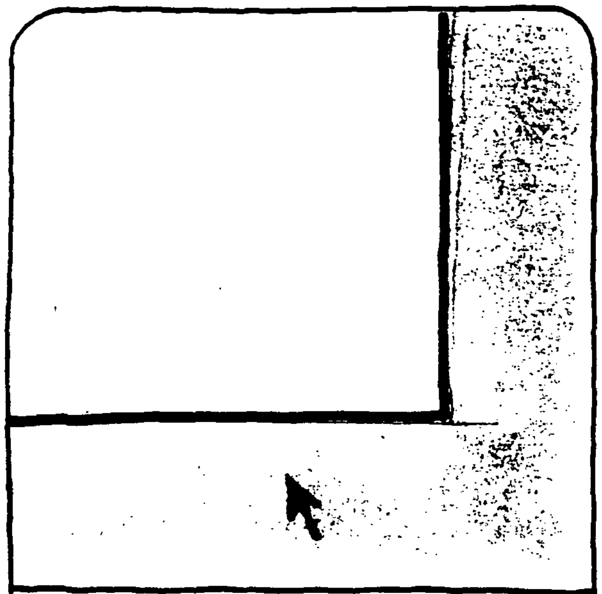


3. Move cursor to desired position of lower right corner of window frame. (Upper left corner remains fixed.)

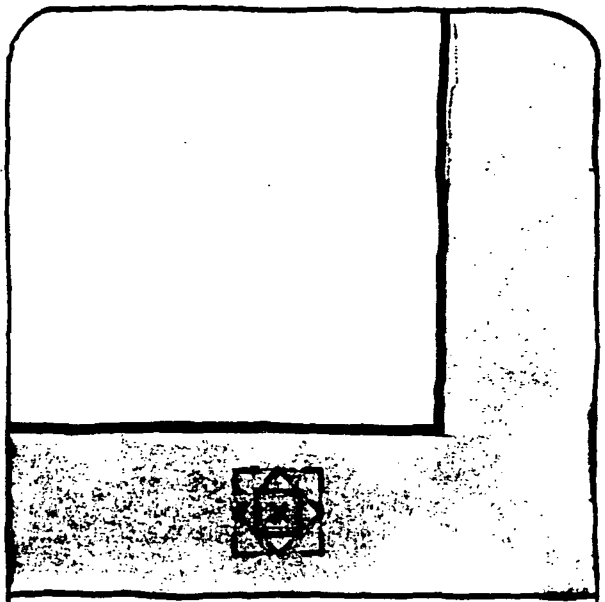


4. Release mouse button. (Bottom and right side of window frame disappear and reappear in new position.) <END>

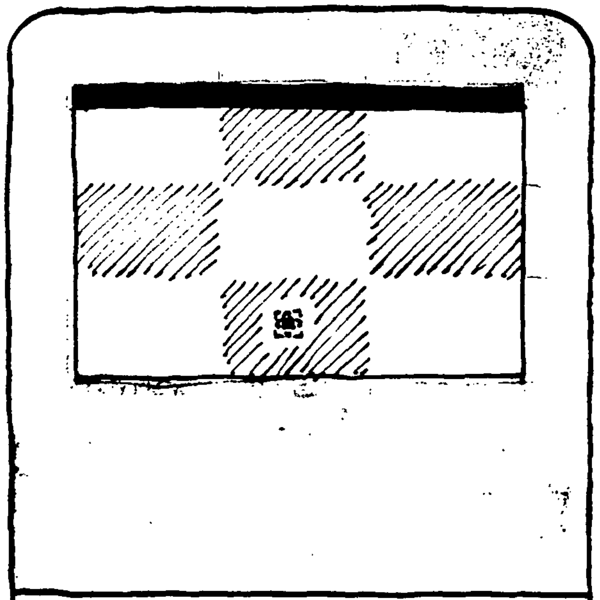
TASK: ADJUSTING WINDOW SIZE/SHAPE PAGE 1 OF 2
SYSTEM: MOVE-WINDOWS (METHOD I)



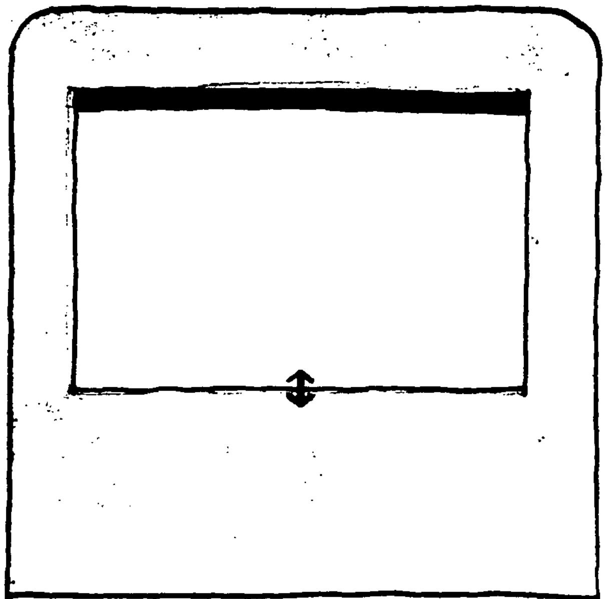
1. Position cursor on background.



2. Press left or middle mouse button. to enter "Move-Windows" mode. (Cursor changes shape.)



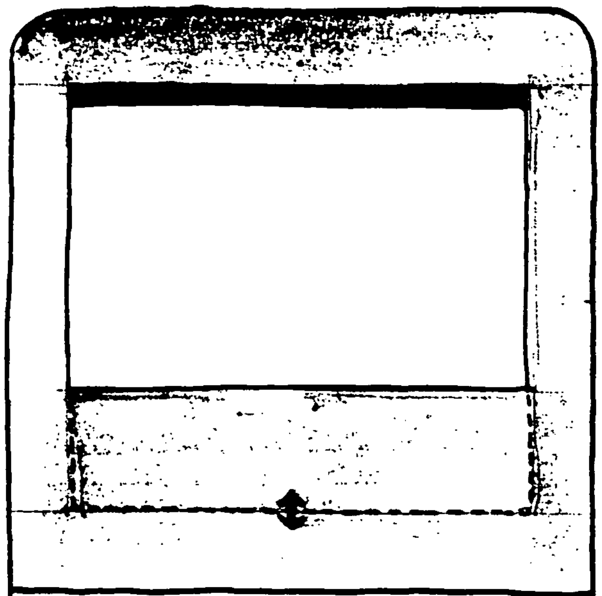
3. Position cursor within one of the four regions indicated above. (These are normally not visible.)



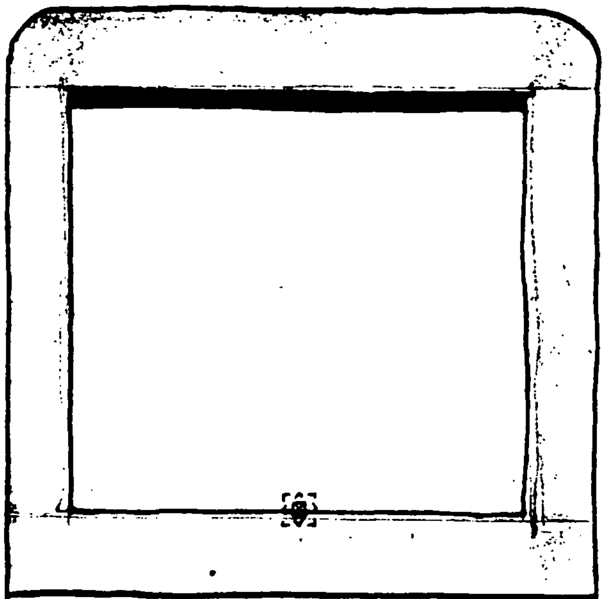
4. Press and hold left mouse button. (Cursor disappears and reappears with new shape at nearest edge of window.)

TASK: ADJUSTING WINDOW SIZE/SHAPE PAGE 2 OF 2

SYSTEM: MOVE-WINDOWS (METHOD I)



5. Position cursor in desired location of window edge. (Outline of window follows vertical movement of cursor.)

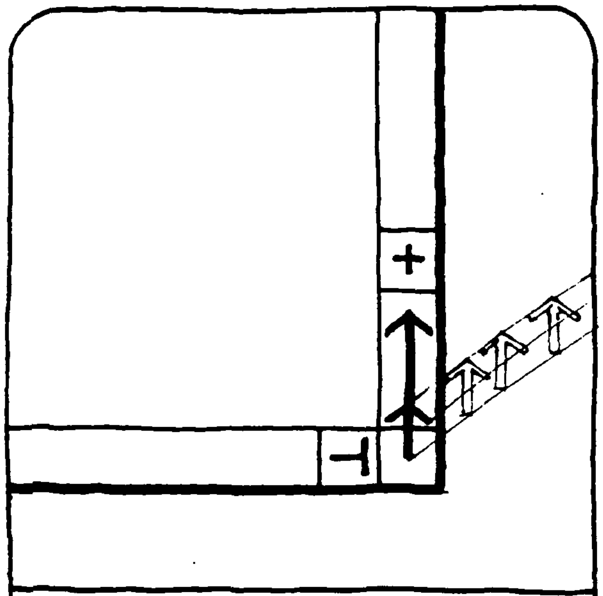


6. Release mouse button. (New window edge appears. Cursor reverts to previous shape.

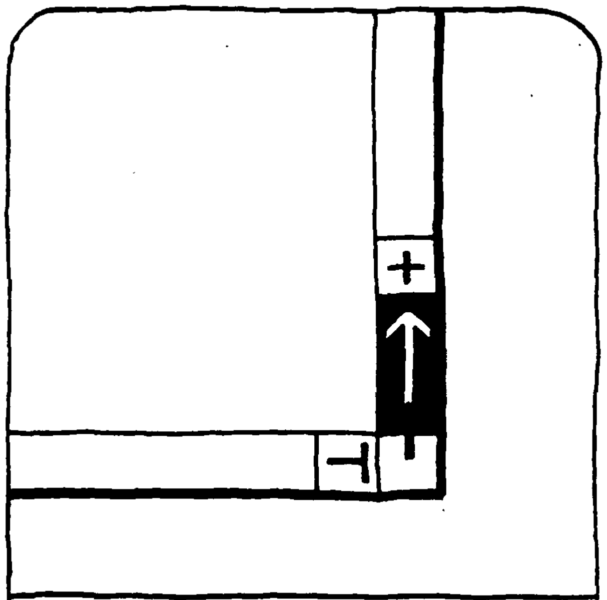
< END >

TASK: SCROLLING

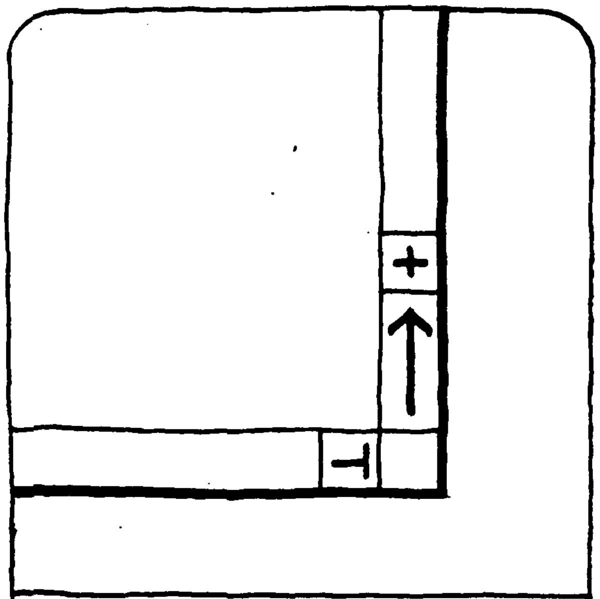
SYSTEM: STAR



1. Position cursor on scroll region.



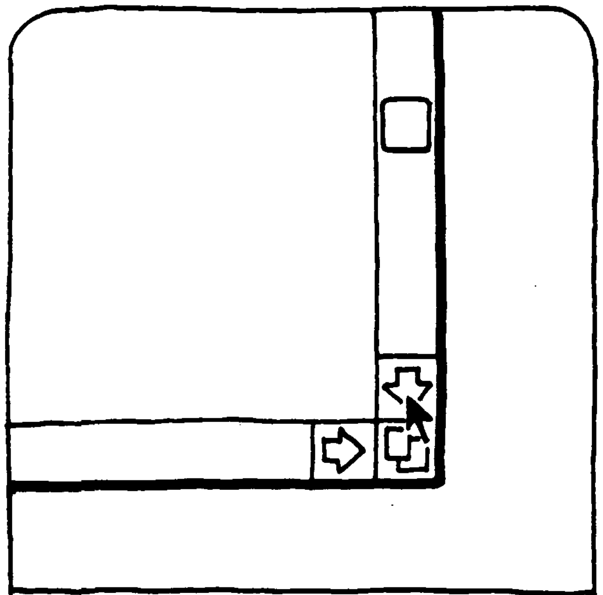
2. Press and hold left mouse button. (Scroll region reverses video. Window contents appear to move upwards.)



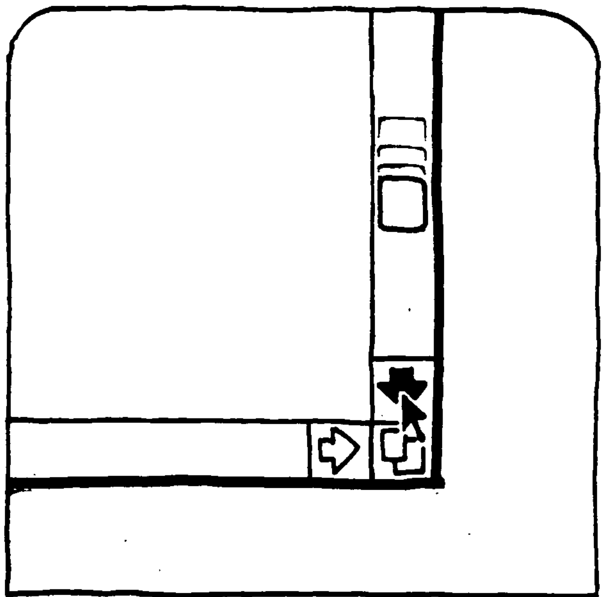
3. Release mouse button when desired portion of document appears in window. (Window contents stop moving.) ⟨END⟩

TASK: SCROLLING

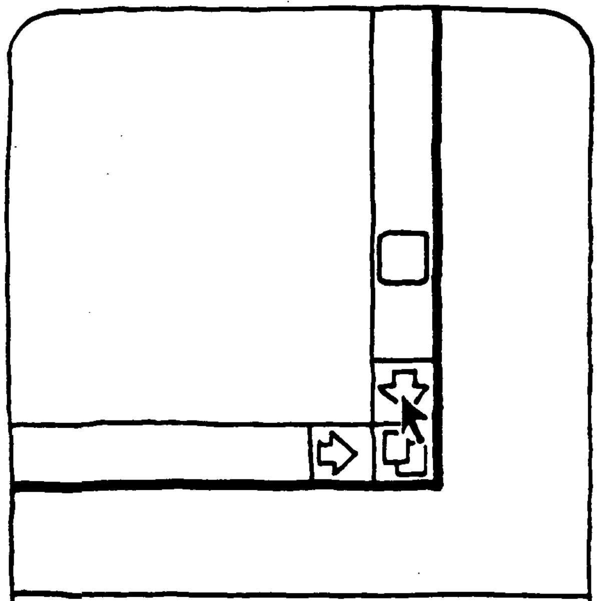
SYSTEM: MACINTOSH



1. Position cursor on scroll region.



2. Press and hold mouse button. (Arrow in scroll region turns black. Window contents move upward. White box moves downwards.)



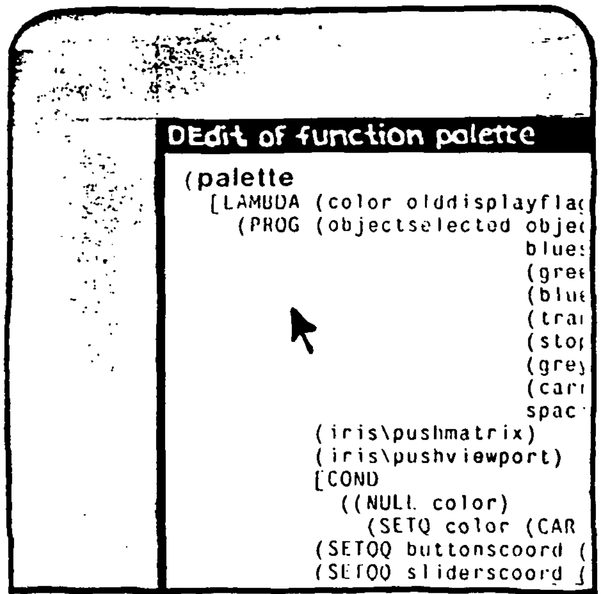
3. Release mouse button when desired portion of document appears in window.

⟨END⟩

TASK: SCROLLING PAGE 1 OF 2

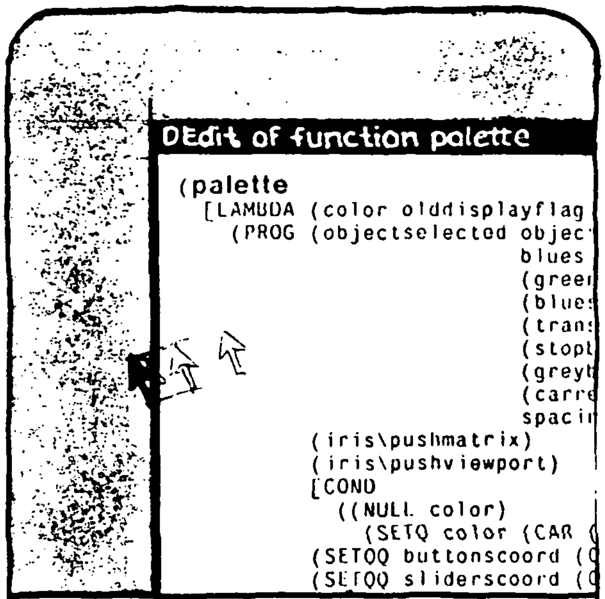
SYSTEM: INTERLISP-D

ORIGINAL PAGE IS
OF POOR QUALITY



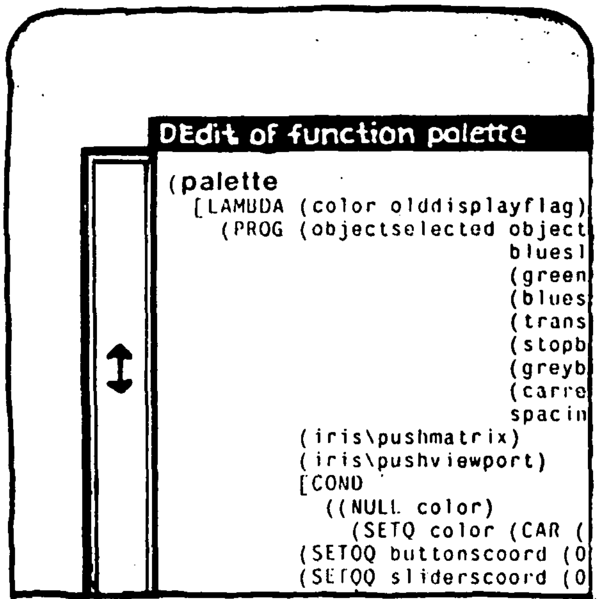
```
(palette
  [LAMBDA (color olddisplayflag
    (PROG (objectselected objec
                          blues
                          (gree
                          (blue
                          (tran
                          (stop
                          (grey
                          (carr
                          spac
          (iris\pushmatrix)
          (iris\pushviewport)
          [COND
            ((NULL color)
              (SETQ color (CAR
          (SETQQ buttonscoord (
          (SETQQ sliderscoord (
```

1. Position cursor anywhere within window of interest.



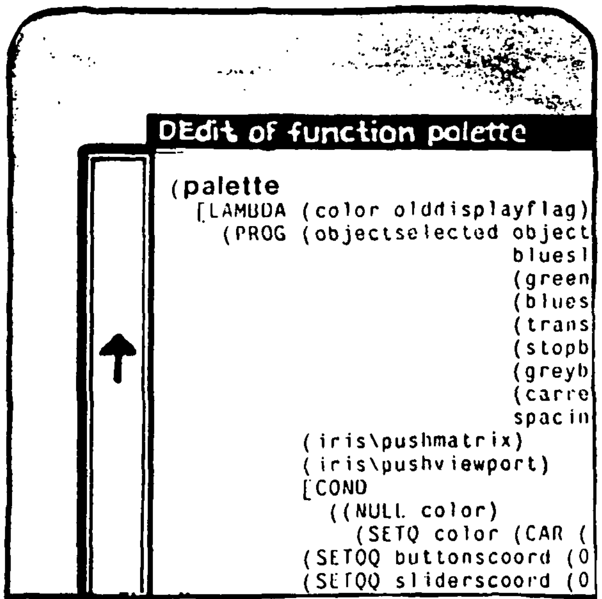
```
(palette
  [LAMBDA (color olddisplayflag
    (PROG (objectselected objec
                          blues
                          (green
                          (blues
                          (trans
                          (stopb
                          (greyb
                          (carre
                          spacin
          (iris\pushmatrix)
          (iris\pushviewport)
          [COND
            ((NULL color)
              (SETQ color (CAR (
          (SETQQ buttonscoord (0
          (SETQQ sliderscoord (0
```

2. Move the cursor until it just crosses the left border of the window.



```
(palette
  [LAMBDA (color olddisplayflag)
    (PROG (objectselected object
                          bluesl
                          (green
                          (blues
                          (trans
                          (stopb
                          (greyb
                          (carre
                          spacin
          (iris\pushmatrix)
          (iris\pushviewport)
          [COND
            ((NULL color)
              (SETQ color (CAR (
          (SETQQ buttonscoord (0
          (SETQQ sliderscoord (0
```

2. Continued. (Scroll bar appears. Cursor changes shape.)

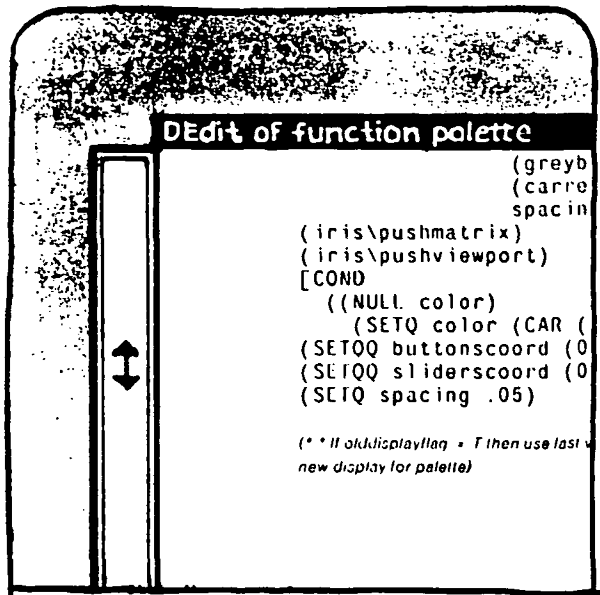


```
(palette
  [LAMBDA (color olddisplayflag)
    (PROG (objectselected object
                          bluesl
                          (green
                          (blues
                          (trans
                          (stopb
                          (greyb
                          (carre
                          spacin
          (iris\pushmatrix)
          (iris\pushviewport)
          [COND
            ((NULL color)
              (SETQ color (CAR (
          (SETQQ buttonscoord (0
          (SETQQ sliderscoord (0
```

3. Click left mouse button. (As button is pressed, cursor changes shape.)

TASK: SCROLLING

SYSTEM: INTERLISP-D



```
                         (greyb
                         (carre
                         spacin
(iris\pushmatrix)
(iris\pushviewport)
[COND
  ((NULL color)
    (SETQ color (CAR (
(SETQQ buttonscoord (0
(SETQQ sliderscoord (0
(SETQ spacing .05)

(* * If olddisplayflag = T then use last v
new display for palette)
```

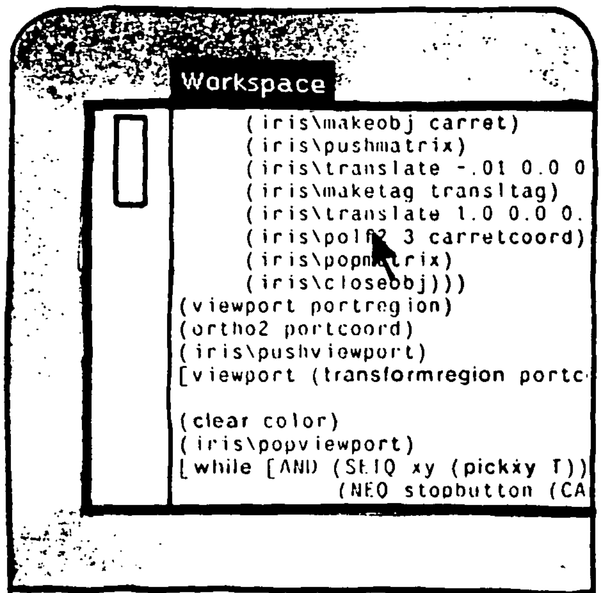
3. Continued. (As button is released, cursor resumes previous shape. Window contents disappear and reappear with line of text nearest the cursor located at top of window.

⟨END⟩

TASK: SCROLLING (FORWARDS)

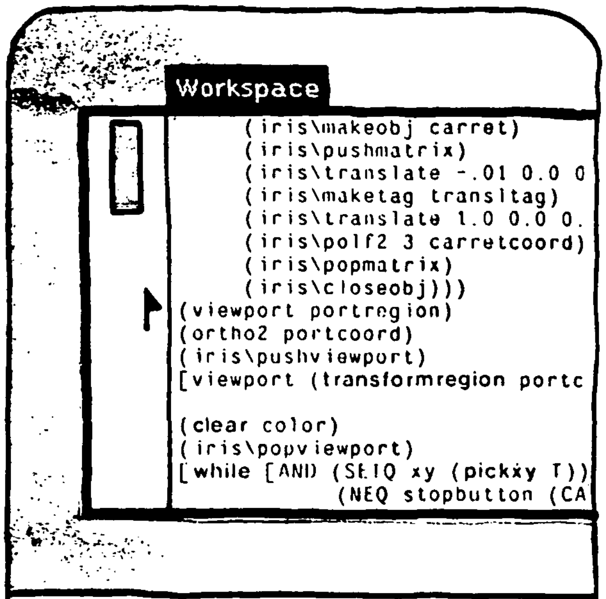
SYSTEM: SMALLTALK

ORIGINAL PAGE IS
OF POOR QUALITY



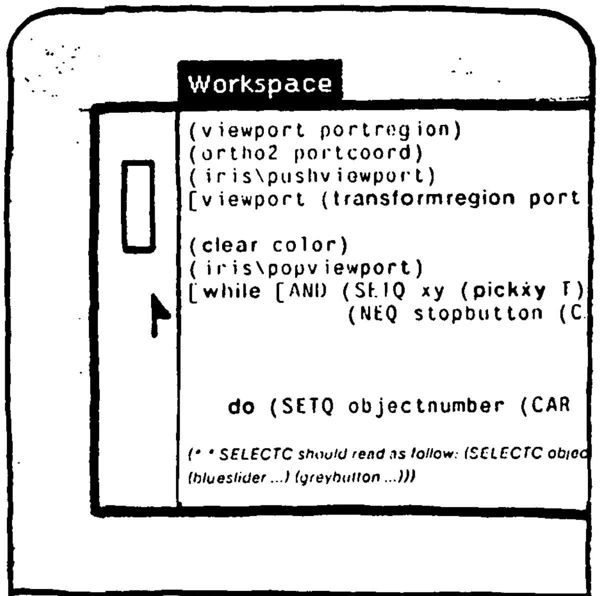
```
        (iris\makeobj carret)
        (iris\pushmatrix)
        (iris\translate -.01 0.0 0
        (iris\maketag transltag)
        (iris\translate 1.0 0.0 0.
        (iris\polf2 3 carretcoord)
        (iris\popmatrix)
        (iris\closeobj)))
(viewport portregion)
(ortho2 portcoord)
(iris\pushviewport)
[viewport (transformregion portc

(clear color)
(iris\popviewport)
[while [AND (SETQ xy (pickxy T))
                (NEQ stopbutton (CA
```

1. Position cursor anywhere within window of interest. (Scroll bar appears on left side.)



```
        (iris\makeobj carret)
        (iris\pushmatrix)
        (iris\translate -.01 0.0 0
        (iris\maketag transltag)
        (iris\translate 1.0 0.0 0.
        (iris\polf2 3 carretcoord)
        (iris\popmatrix)
        (iris\closeobj)))
(viewport portregion)
(ortho2 portcoord)
(iris\pushviewport)
[viewport (transformregion portc

(clear color)
(iris\popviewport)
[while [AND (SETQ xy (pickxy T))
                (NEQ stopbutton (CA
```

2. Position cursor within right third (vertically) of scroll bar. (Cursor changes to upward pointing arrow.)



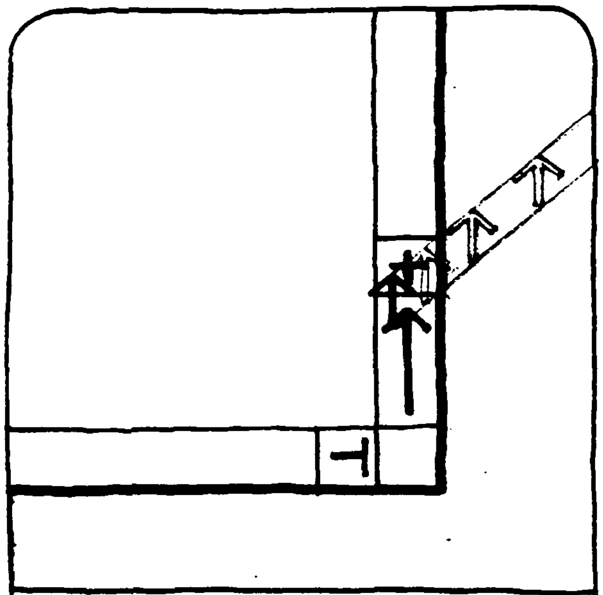
```
(viewport portregion)
(ortho2 portcoord)
(iris\pushviewport)
[viewport (transformregion port

(clear color)
(iris\popviewport)
[while [AND (SETQ xy (pickxy T)
                (NEQ stopbutton (C


   do (SETQ objectnumber (CAR

(* * SELECTC should read as follow: (SELECTC objec
(blueslider ...) (greybutton ...)))
```

3. Click left mouse button. (Window contents disappear and reappear with line of text nearest the cursor located at top of window.)

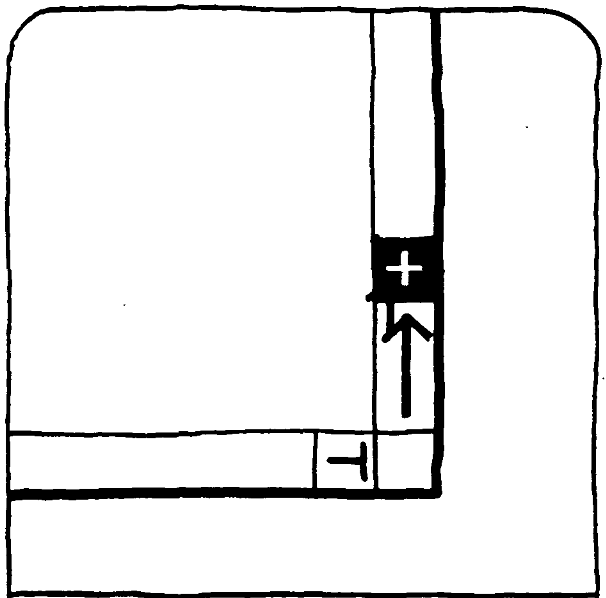
<END>

TASK: PAGING

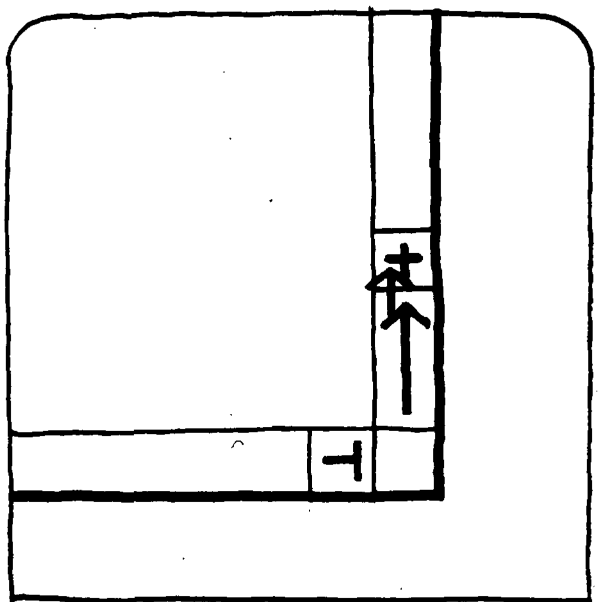
SYSTEM: STAR



1. Position cursor on ⊞ region.



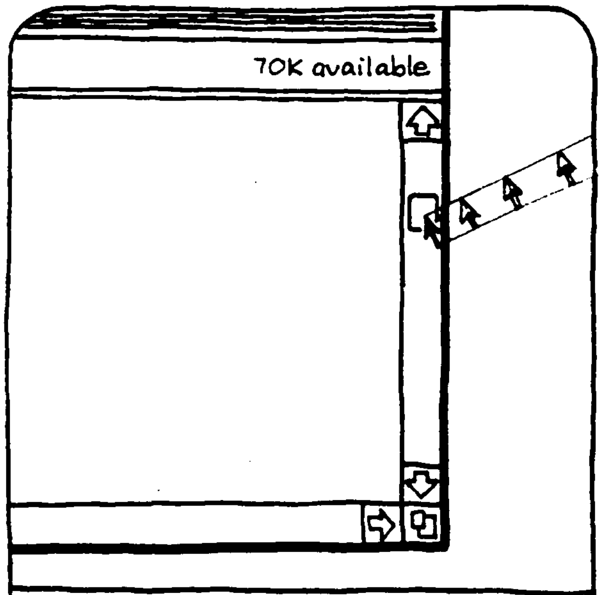
2. Press left mouse button. (While button is held ⊞ region reverses video.)



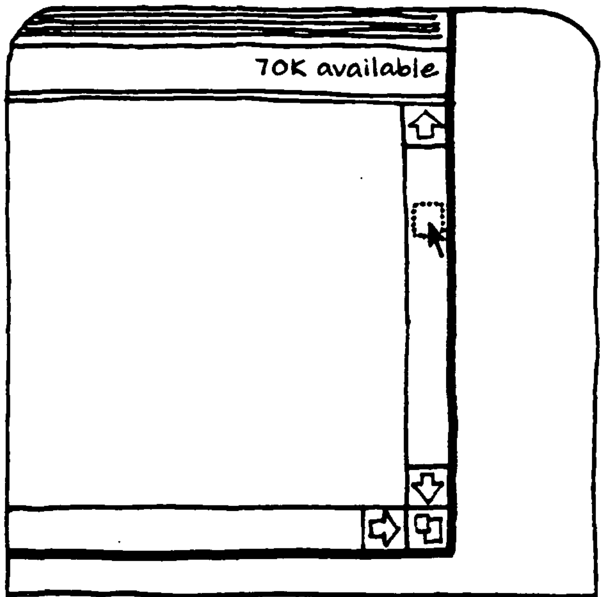
2. Continued. (When button is released, ⊞ region reverts to normal video. Top of next paginated page is placed at top of window. ⟨END⟩

TASK: THUMBING

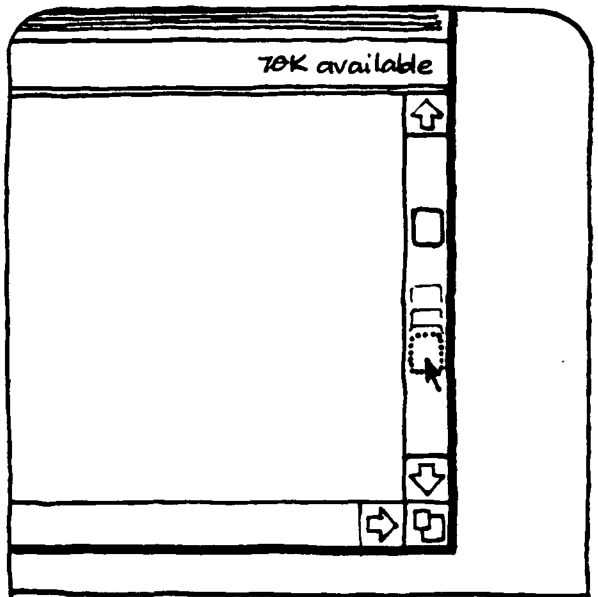
SYSTEM: MACINTOSH



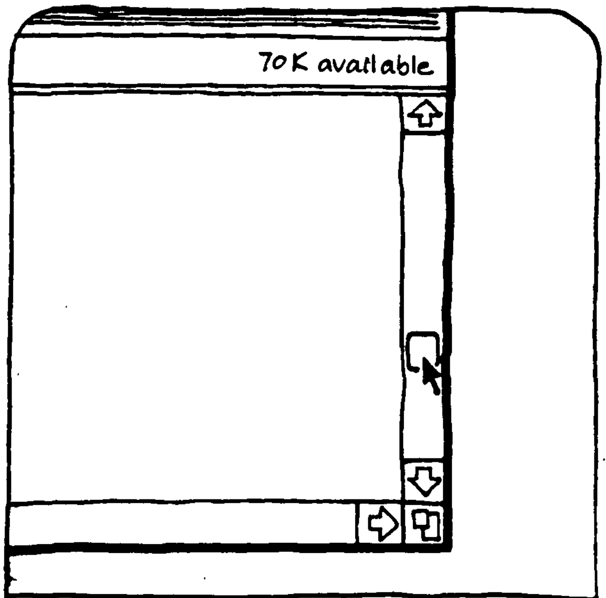
1. Position cursor on white box in scroll bar.



2. Press and hold mouse button.



3. Move cursor to position on scroll bar corresponding to desired portion of document. ("Ghost image of box follows cursor.)



4. Release mouse button. (Original white box disappears and reappears in new position. Desired portion of document is displayed in window.)

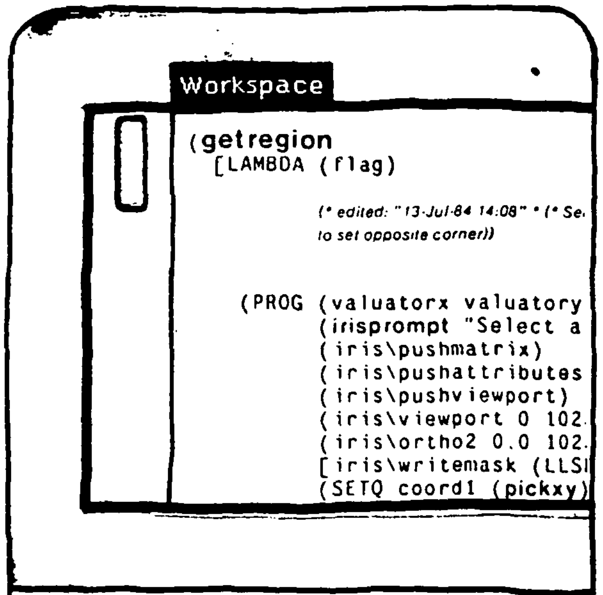
⟨END⟩

TASK: THUMBING

SYSTEM: SMALLTALK

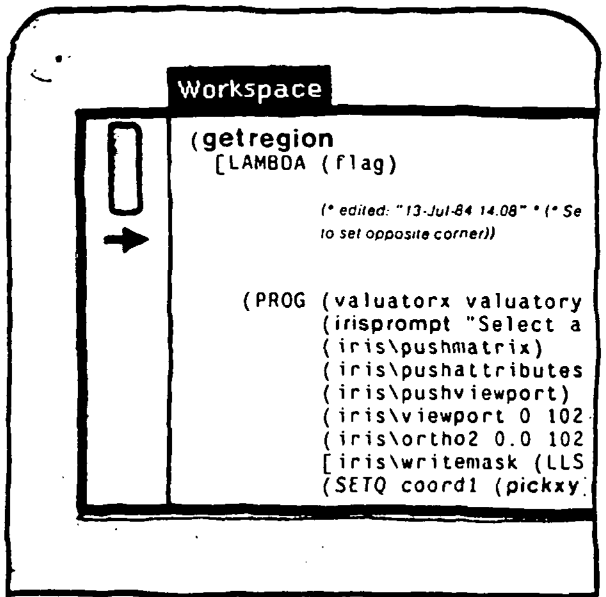
ORIGINAL PAGE IS OF POOR QUALITY

PAGE 1 OF 2



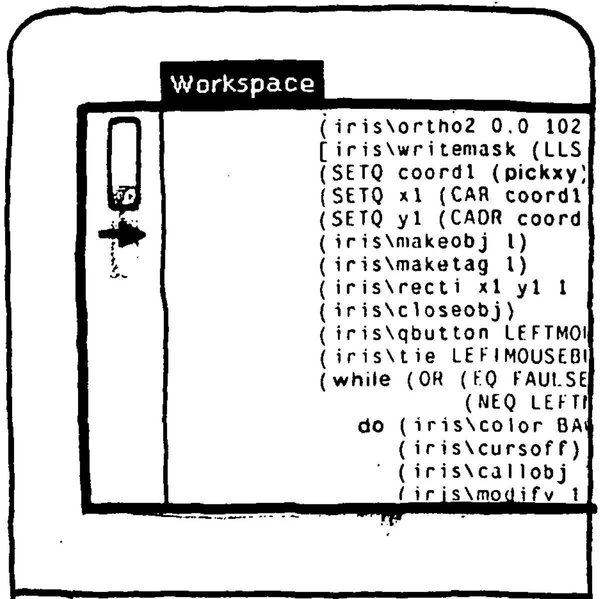
```
(getregion
  [LAMBDA (flag)
          (* edited: "13-Jul-84 14:08" * (* Se
          to set opposite corner))

    (PROG (valuatorx valuatory
          (irisprompt "Select a
          (iris\pushmatrix)
          (iris\pushattributes
          (iris\pushviewport)
          (iris\viewport 0 102.
          (iris\ortho2 0.0 102.
          [iris\writemask (LLSI
          (SETQ coord1 (pickxy)
```

1. Position cursor anywhere within window of interest. (Scroll bar appears on left side.)



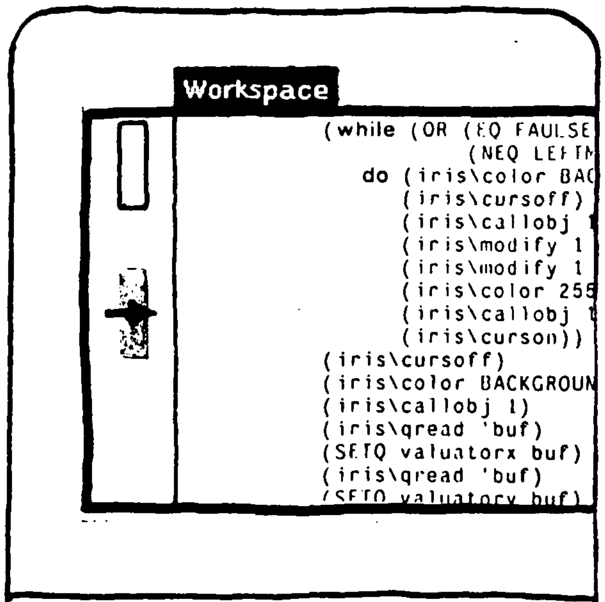
```
(getregion
  [LAMBDA (flag)
          (* edited: "13-Jul-84 14:08" * (* Se
          to set opposite corner))

    (PROG (valuatorx valuatory
          (irisprompt "Select a
          (iris\pushmatrix)
          (iris\pushattributes
          (iris\pushviewport)
          (iris\viewport 0 102
          (iris\ortho2 0.0 102
          [iris\writemask (LLS
          (SETQ coord1 (pickxy
```

2. Position cursor within middle third (vertically) of scroll bar. (Cursor changes to horizontal arrow.)



```
(iris\ortho2 0.0 102
[iris\writemask (LLS
(SETQ coord1 (pickxy)
(SETQ x1 (CAR coord1
(SETQ y1 (CADR coord
(iris\makeobj 1)
(iris\maketag 1)
(iris\recti x1 y1 1
(iris\closeobj)
(iris\qbutton LEFTMOI
(iris\tie LEFTMOUSEBI
(while (OR (EQ FAULSE
                (NEQ LEFTI
   do (iris\color BA
      (iris\cursoff)
      (iris\callobj
      (iris\modify 1
```

3. Press and hold left mouse button. (Grey box moves to cursor. Window contents change accordingly.)

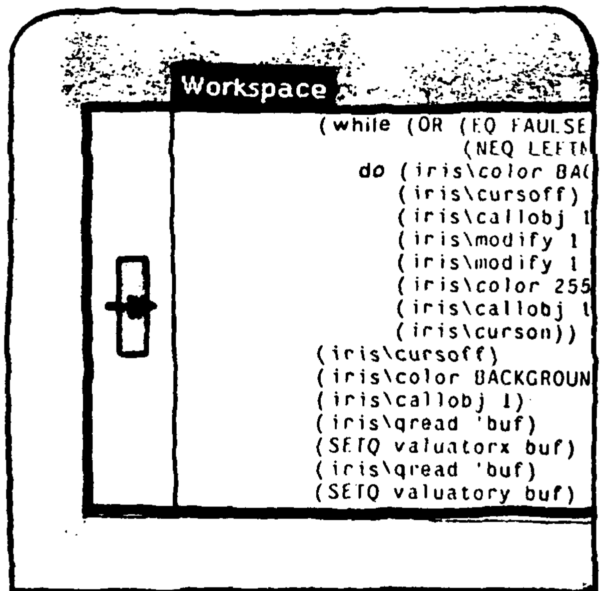


```
(while (OR (EQ FAULSE
               (NEQ LEFTM
   do (iris\color BAC
      (iris\cursoff)
      (iris\callobj 1
      (iris\modify 1
      (iris\modify 1
      (iris\color 255
      (iris\callobj 1
      (iris\curson))
(iris\cursoff)
(iris\color BACKGROUN
(iris\callobj 1)
(iris\qread 'buf)
(SETQ valuatorx buf)
(iris\qread 'buf)
(SETQ valuatory buf)
```

4. Move cursor up or down scroll bar. (Gray box tracks cursor movement. Window contents move rapidly in opposite direction.)

TASK: THUMBING PAGE 2 OF 2

SYSTEM: SMALLTALK



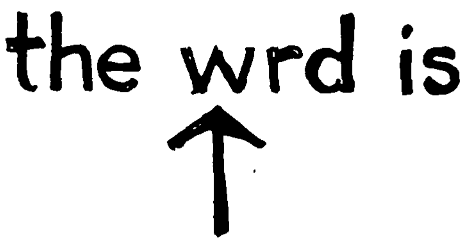
```
(while (OR (EQ FAULSE
            (NEQ LEFTM
    do (iris\color BAC
       (iris\cursoff)
       (iris\callobj 1
       (iris\modify 1
       (iris\modify 1
       (iris\color 255
       (iris\callobj 1
       (iris\curson))
(iris\cursoff)
(iris\color BACKGROUN
(iris\callobj 1)
(iris\qread 'buf)
(SETQ valuatorx buf)
(iris\qread 'buf)
(SETQ valuatory buf)
```

5. Release mouse button. (Box outline disappears and reappears around gray box.)

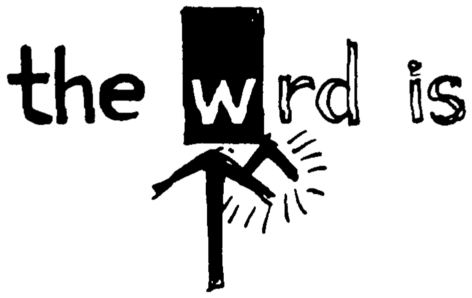
ORIGINAL PAGE IS
OF POOR QUALITY

TASK: INSERTION POINT SELECTION

SYSTEM: STAR



1. Position cursor on character immediately preceding desired insertion point.

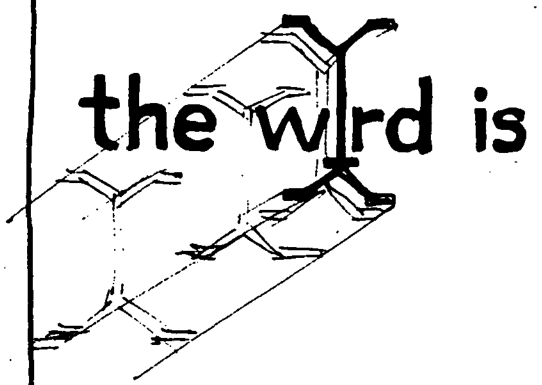


2. Press mouse button. Character reverses. Flashing caret appears to right of ~~charact~~ character

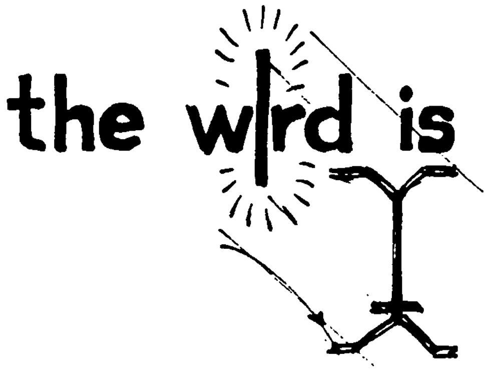
<END>

TASK: INSERTION POINT SELECTION

SYSTEM: MACINTOSH



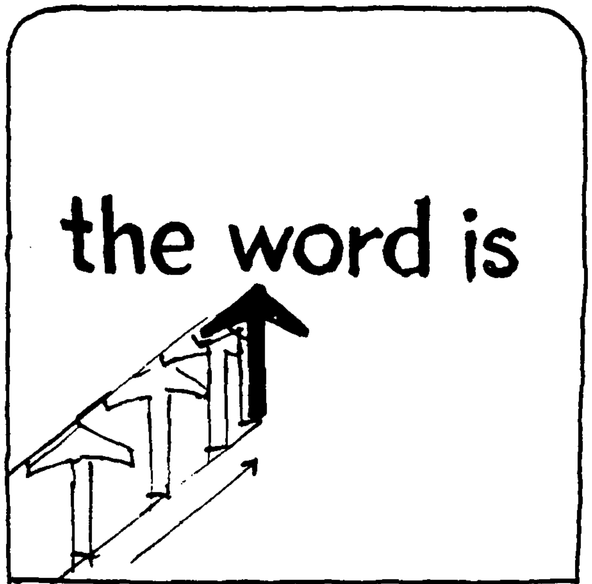
1. Position cursor on desired insertion point. Press mouse button.



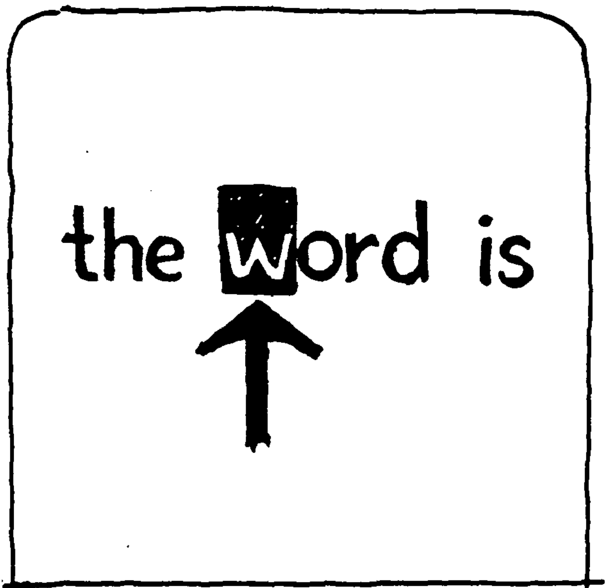
2. Flashing vertical bar appears in desired insertion point.

<END>

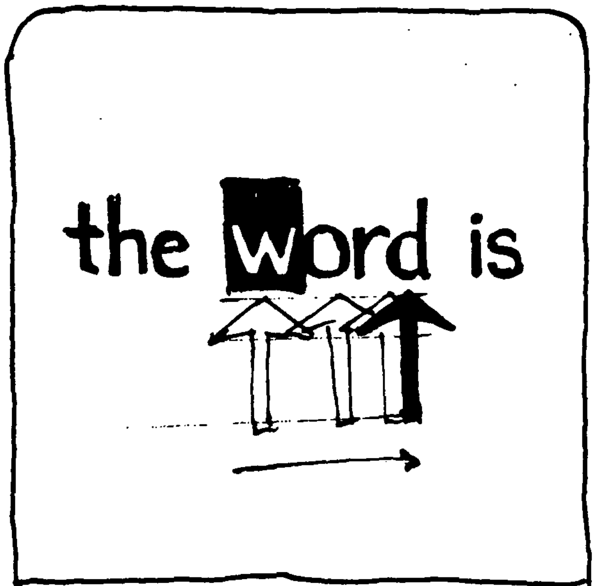
TASK: WORD SELECTION
SYSTEM: STAR (METHOD I)



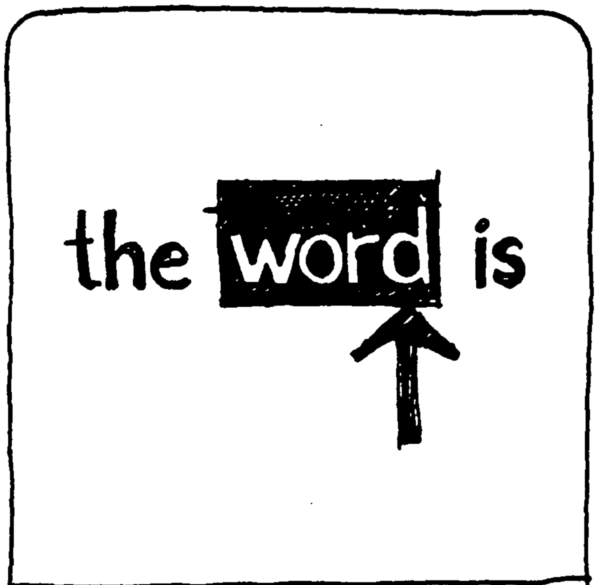
1. Position cursor under first letter in word.



2. Press left ~~cursor~~ mouse button. [Reverse video.]



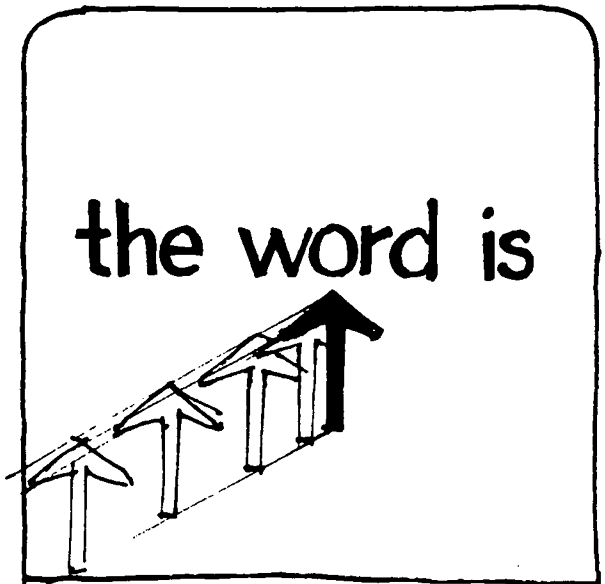
3. Position cursor under last letter in word.



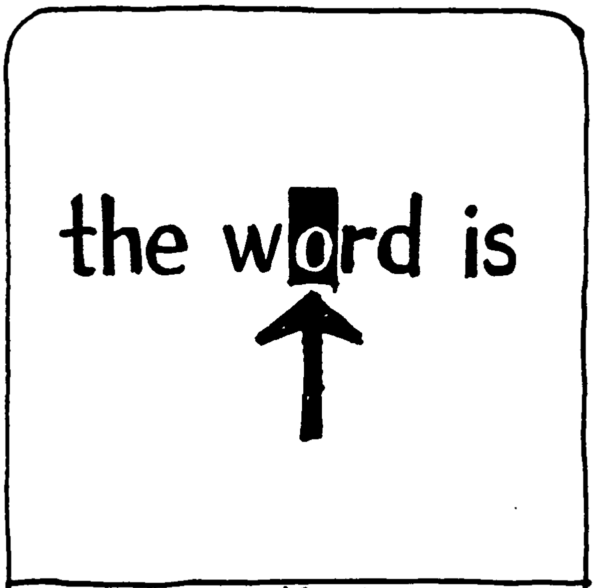
4. Press right ~~cursor~~ mouse button. [Reverse video.]

<END>

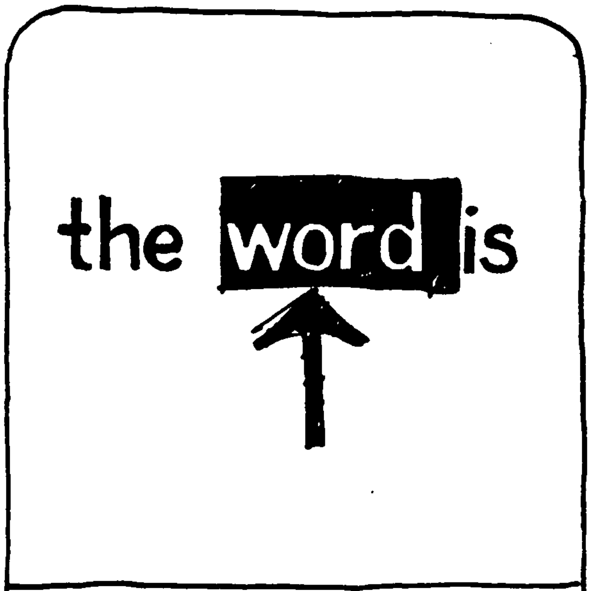
TASK: WORD SELECTION
SYSTEM: STAR (METHOD II)



1. Position cursor under any letter in word.



2. Press left ~~cursor~~ mouse button. [Reverse video.]



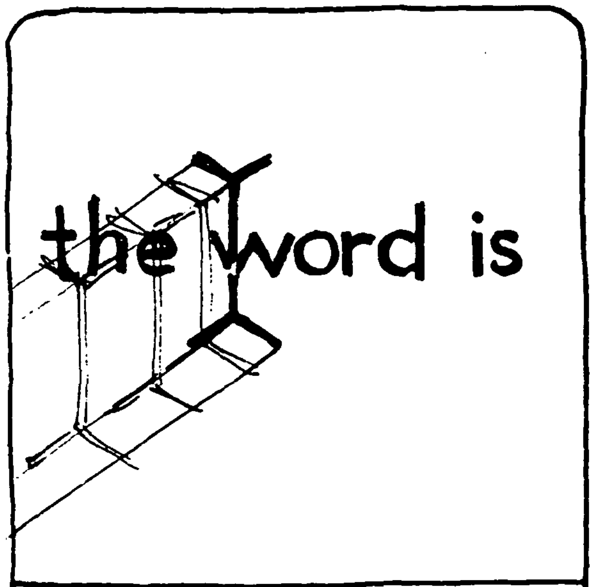
3. Press left ~~m~~ mouse button again

[Reverse video]

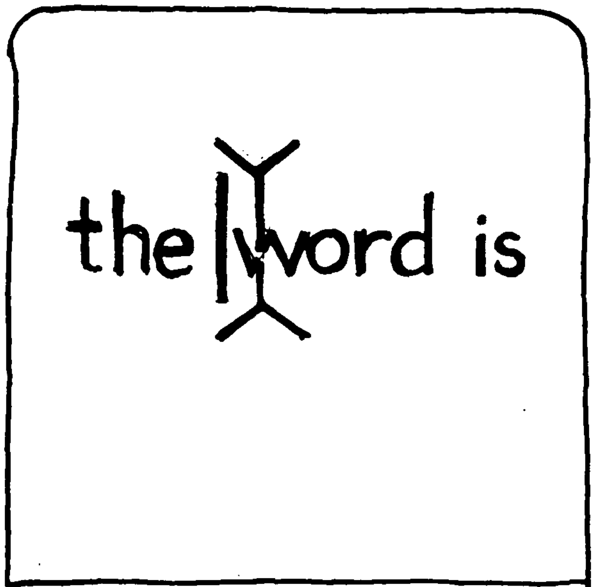
Note that space following word is also selected.

<END>

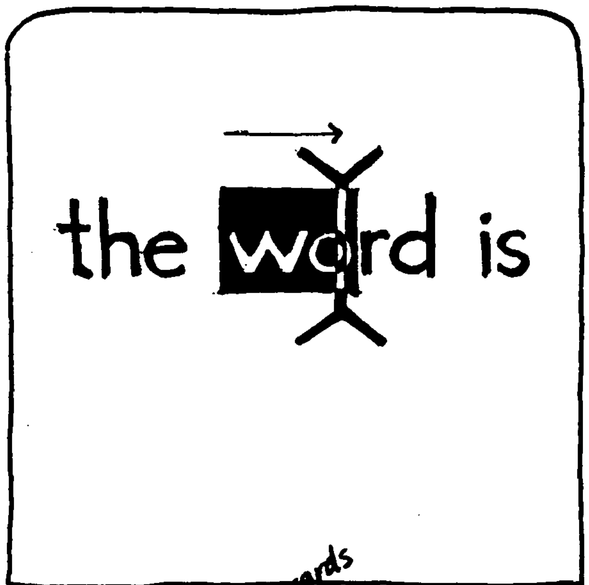
TASK : WORD SELECTION
SYSTEM : MACINTOSH



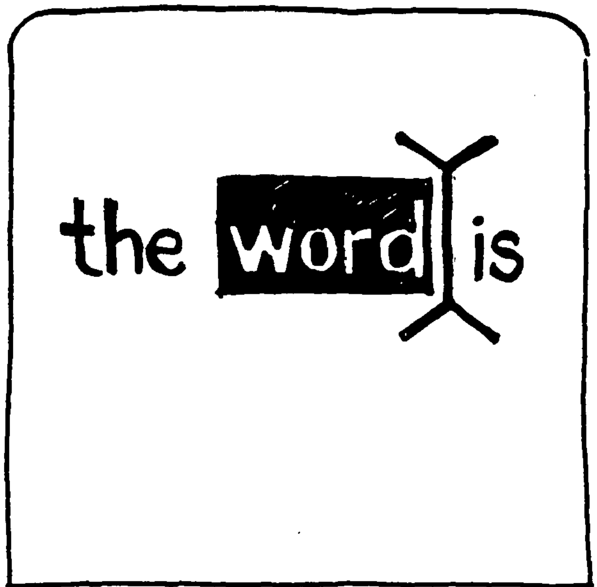
1. Position cursor to left of center of first letter in word.



2. Press and hold mouse button. [illegible] Vertical bar appears to left of first letter.



3. Move cursor towards end of word. Vertical bar disappears. Selected characters are reversed.

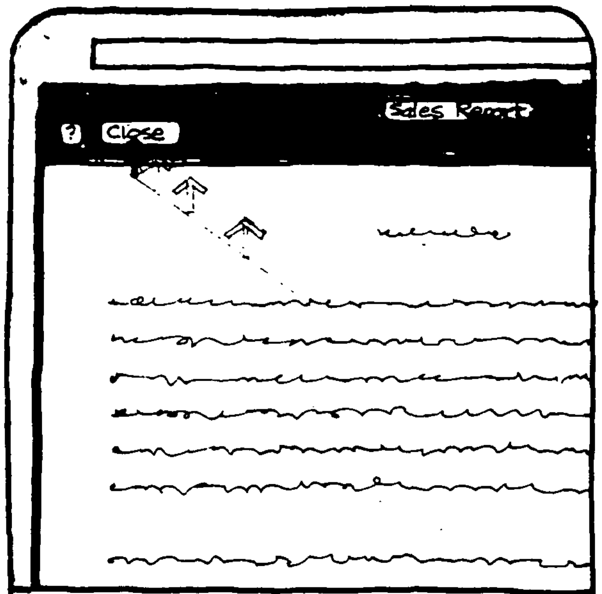


4. Release mouse button when entire ~~wor~~ word is reversed.

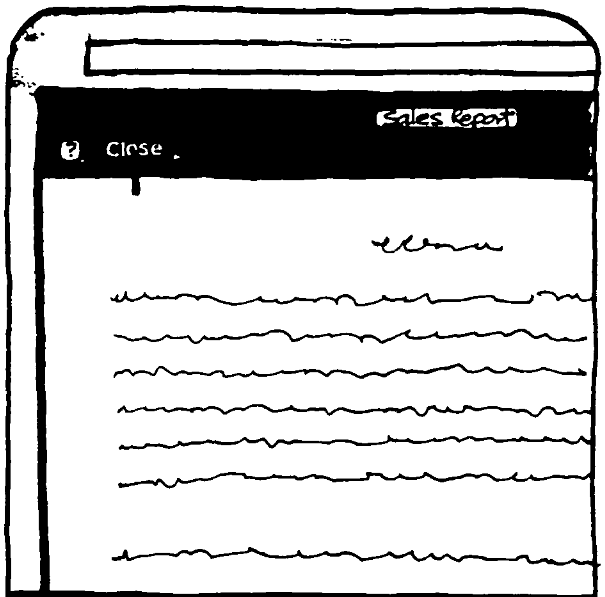
<END>

TASK: CLOSING A WINDOW

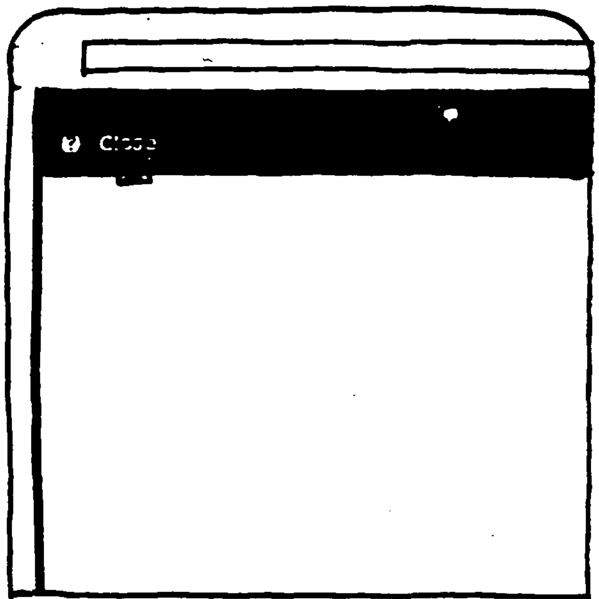
SYSTEM:



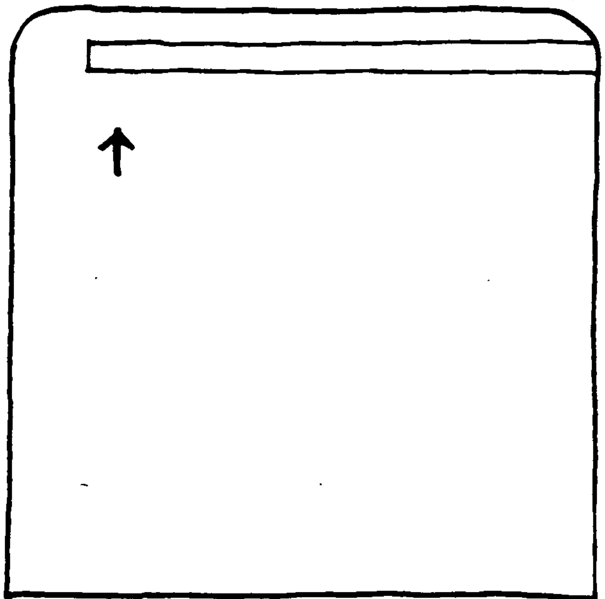
1. Position cursor on the "CLOSE" box in the top left corner of the window.



2. Press left mouse button. As button is pressed, "CLOSE" box reverses video.



2 (continued) As button is released, cursor changes to hour glass. Title box shrinks.



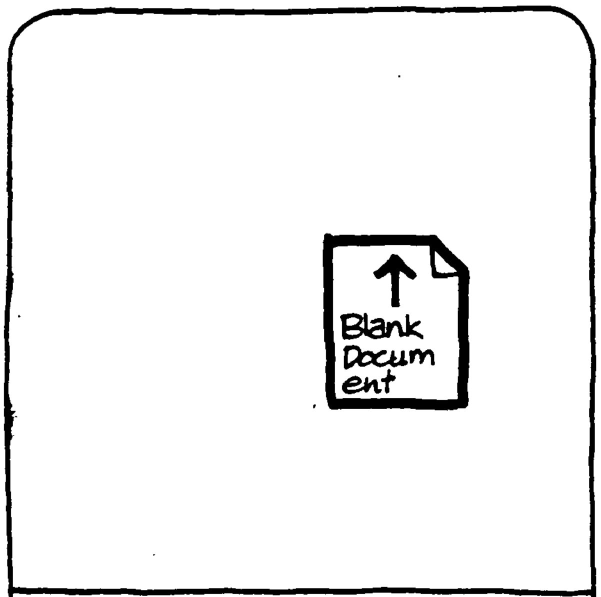
2. (continued.) Window disappears. Cursor reverts to normal shape.

<END>

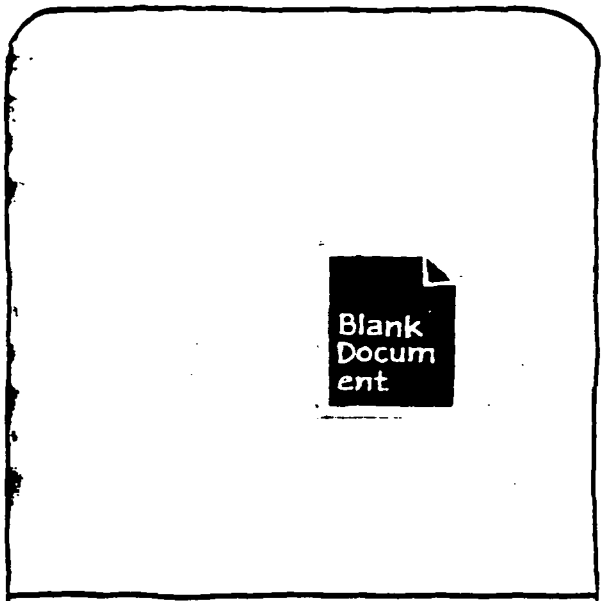
ICON OPERATIONS

TASK: COPYING A DOCUMENT PAGE 1 OF 2

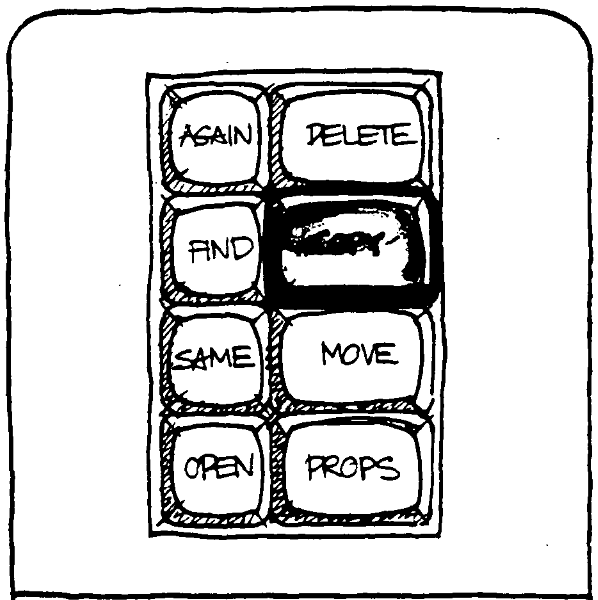
SYSTEM: STAR



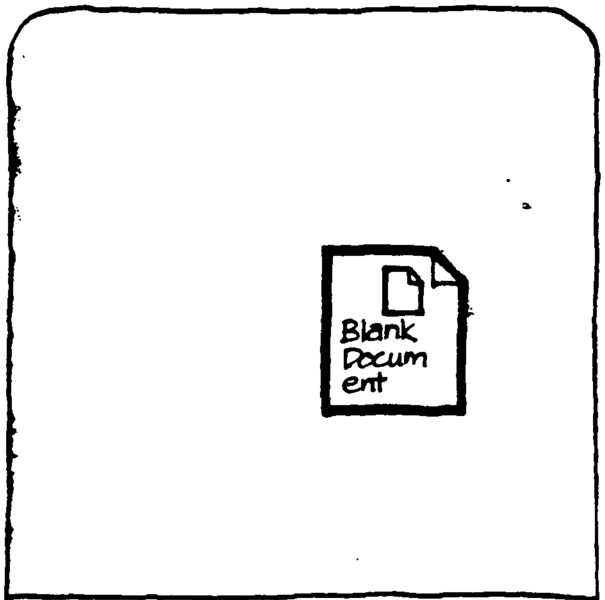
1. Position cursor on desired document icon.



2. Press left mouse button. (Icon reverses video.)



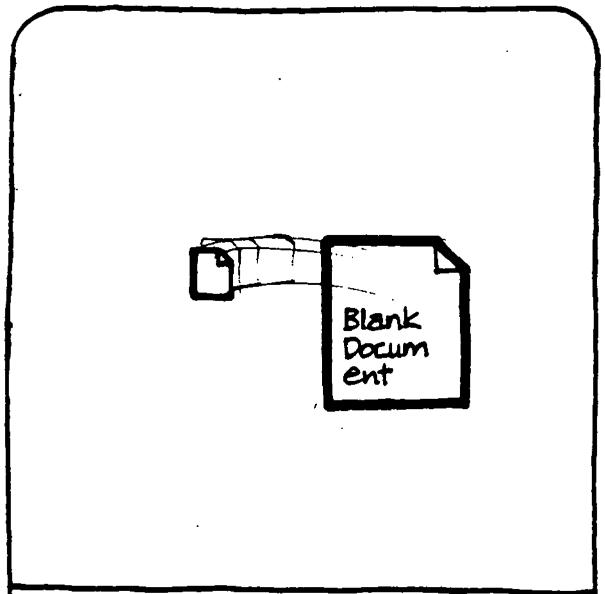
3. Press "COPY" button. (Message appears in prompt window: "Please indicate a destination with either button.")



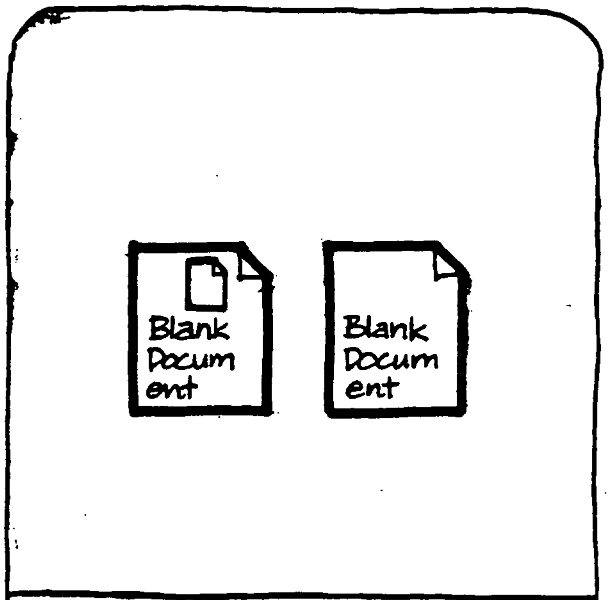
3. continued. (Icon reverts to normal video. Cursor chan into document mini-icon shape.

TASK: COPYING A DOCUMENT PAGE 2 OF 2

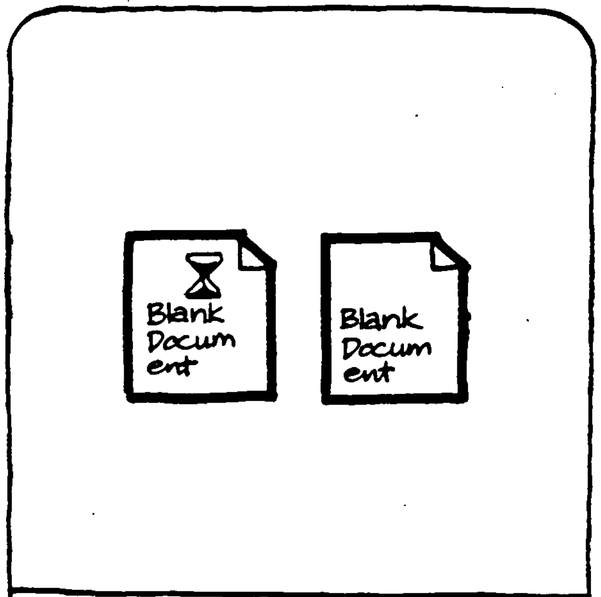
SYSTEM: STAR



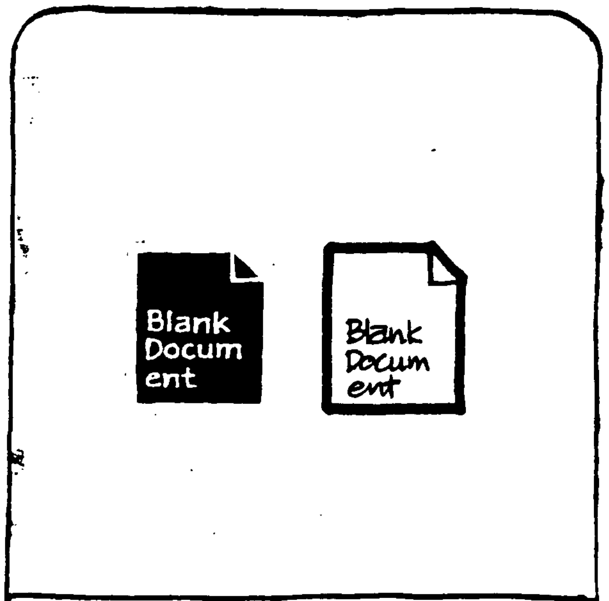
4. Position cursor in desired location.



5. Press either mouse button. (While button is held, document icon appears behind cursor.)

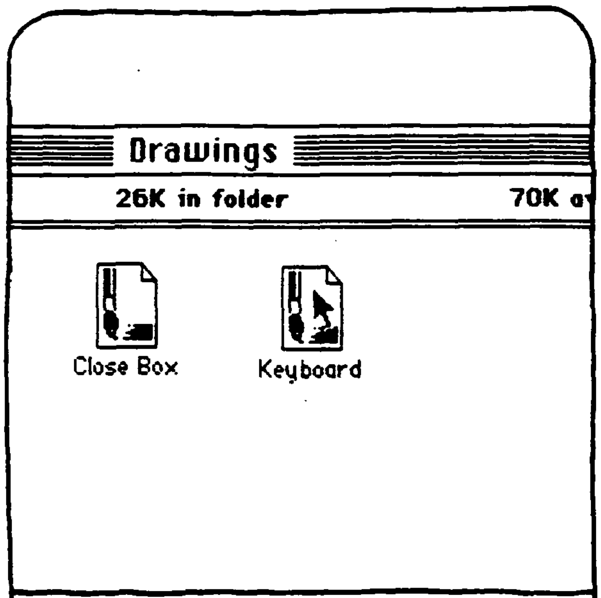


5. continued. (When button is released, cursor changes to hour glass shape.)

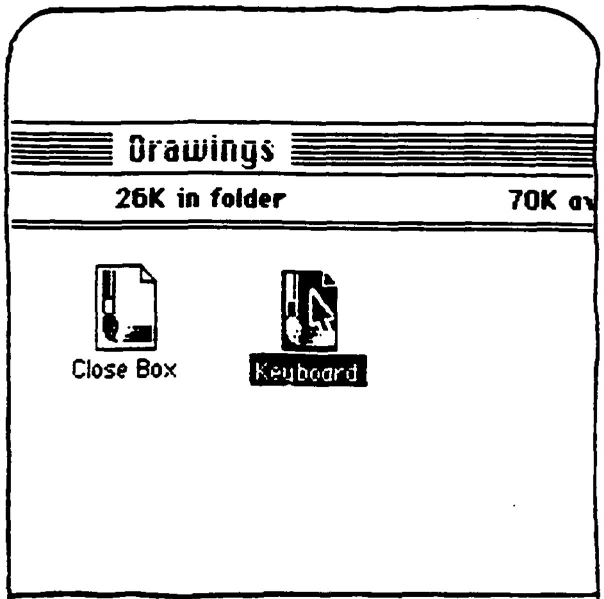


5. continued. (After two seconds, icon copy reverses video. Cursor reverts to normal shape.) <END>

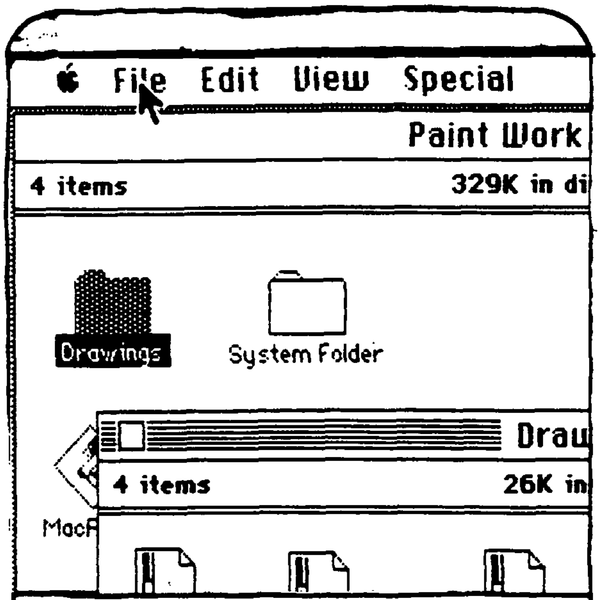
TASK: COPYING A DOCUMENT PAGE 1 OF 2
SYSTEM: MACINTOSH



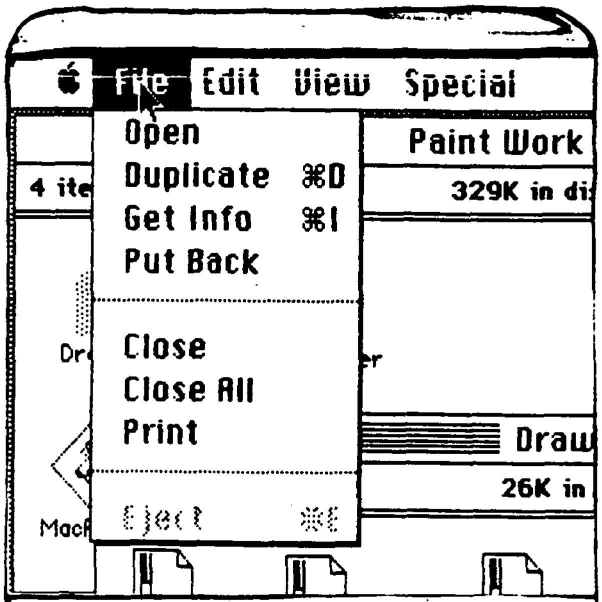
1. Position cursor on desired document icon.



2. Press mouse button. (Icon reverses video.)



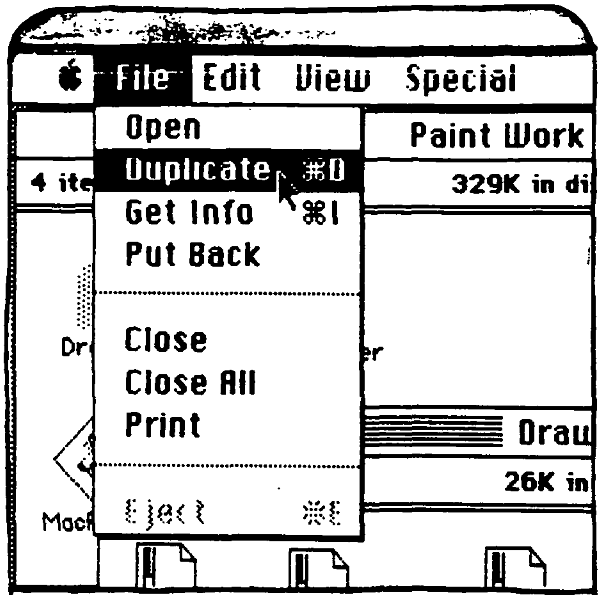
3. Position cursor on "File" menu title.



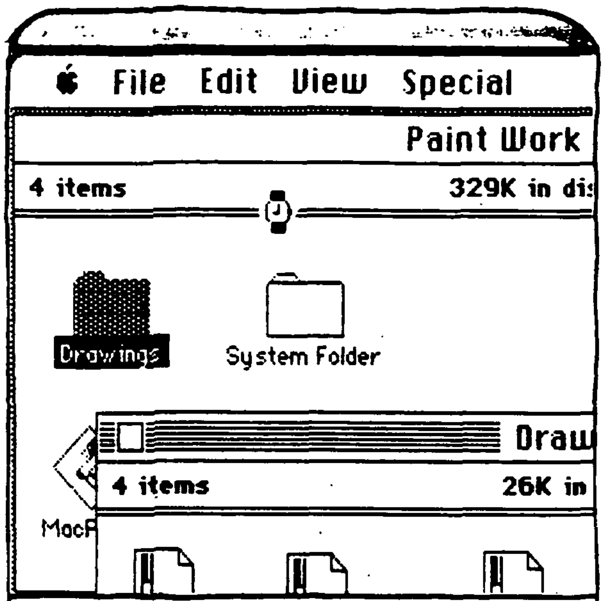
4. Press and hold mouse button. ("File" reverses video. Menu appears.)

TASK: COPYING A DOCUMENT PAGE 2 OF 2

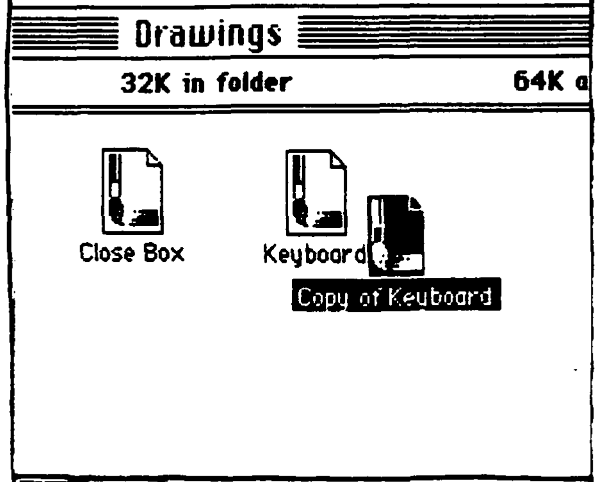
SYSTEM: MACINTOSH



5. Position cursor on "Duplicate" command. ("Duplicate" reverses video.)



6. Release mouse button. (Menu disappears. Cursor changes into watch shape.)



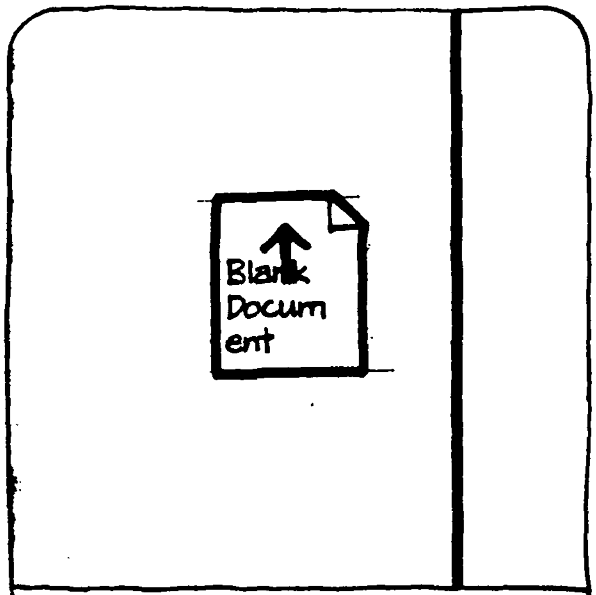
6. continued. (After a few seconds, icon reverts to normal video. New icon appears ~~to~~ in reverse video to lower right of original. Cursor reverts to normal shape) <END>

<END>

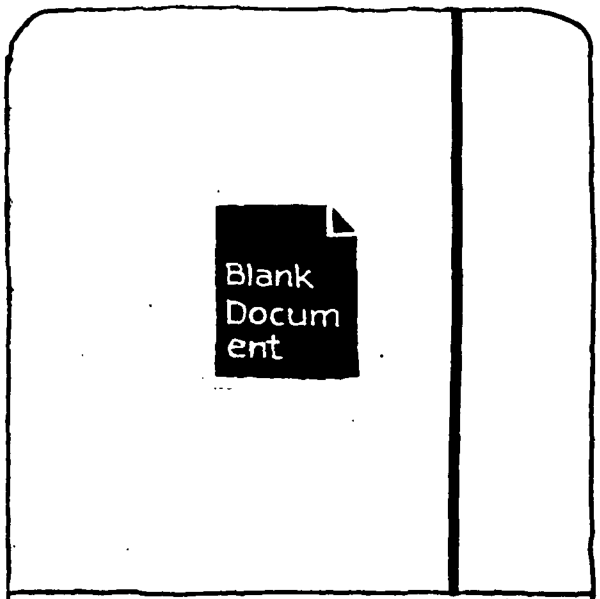
ORIGINAL PAGE IS OF POOR QUALITY

TASK: RENAMING A DOCUMENT PAGE 1 OF 4

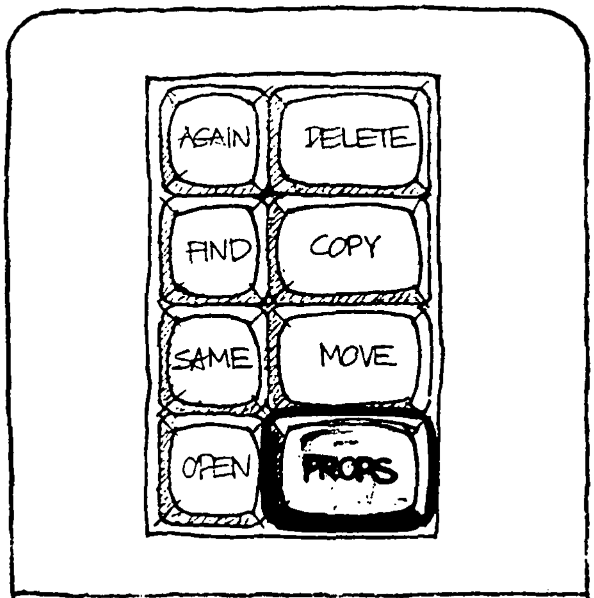
SYSTEM: STAR



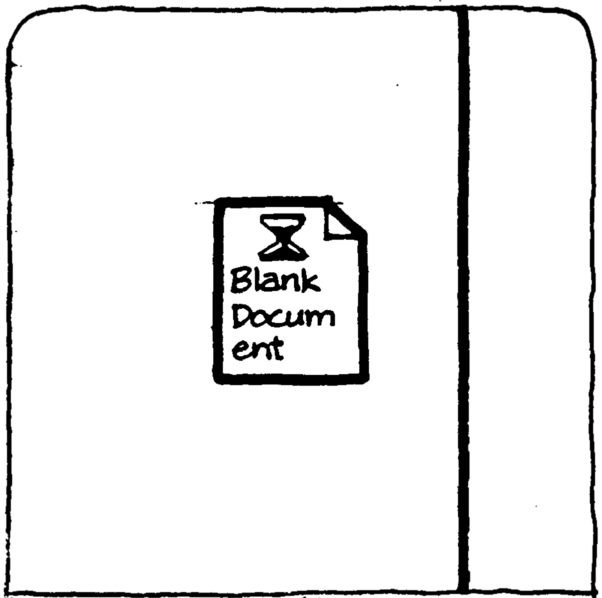
1. Position cursor on desired document icon.



2. Press left mouse button. (Icon reverses video.)



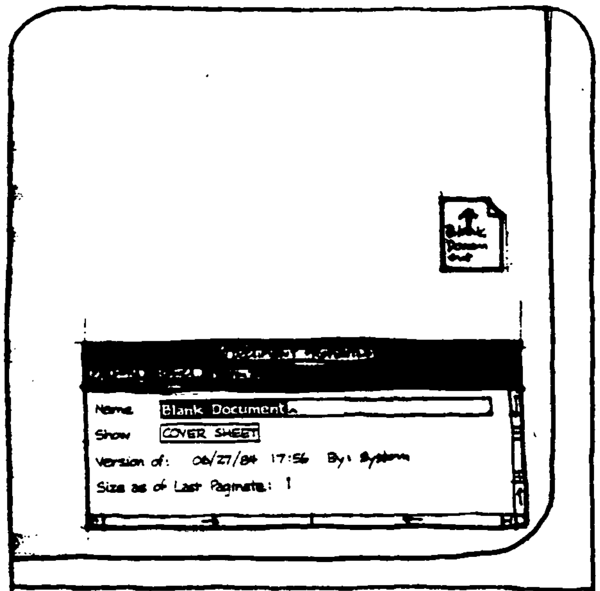
3. Press PROPS button.



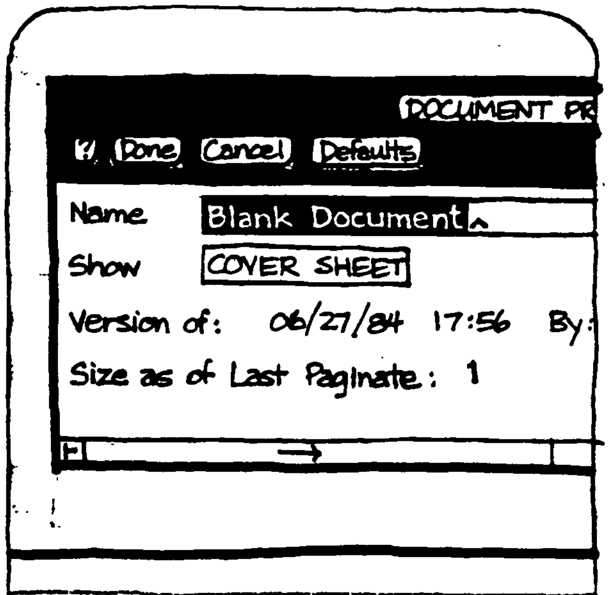
3. continued. (Icon reverts to normal video. Cursor changes to hour glass shape.)

TASK: RENAMING A DOCUMENT PAGE 2 OF 4

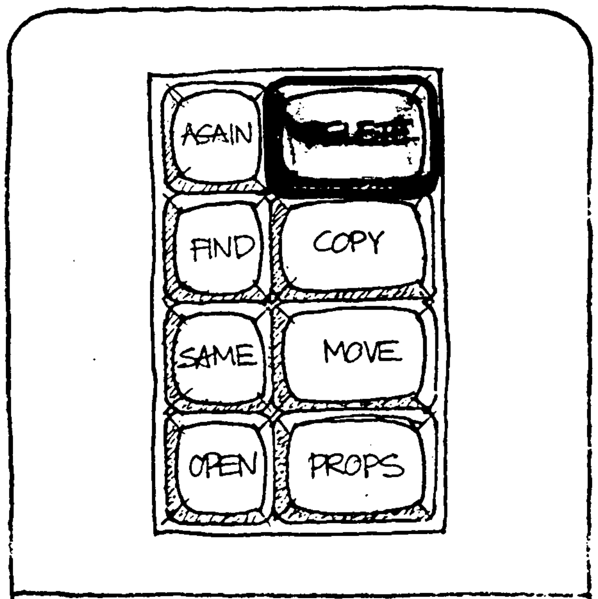
SYSTEM: STAR


Blank Document
Name Blank Document
Show COVER SHEET
Version of: 06/27/84 17:56 By: System
Size as of Last Paginate: 1

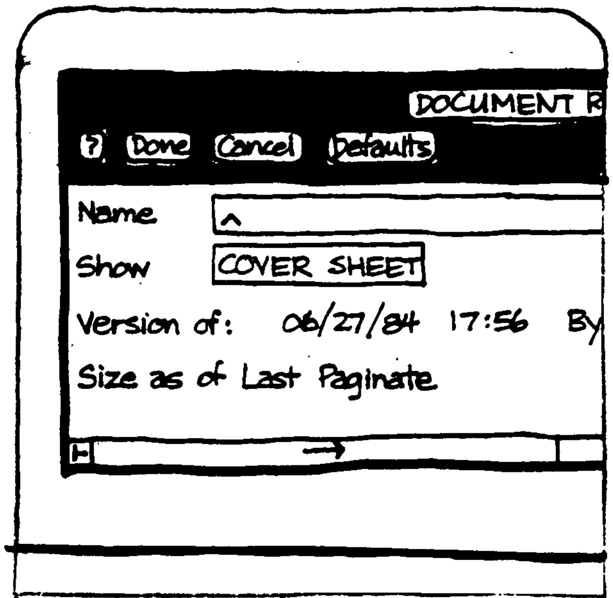
4. (Properties window appears in bottom right corner of screen. Cursor reverts to normal shape.)



4. continued. (Document name appears in reverse video and is followed by blinking caret.)



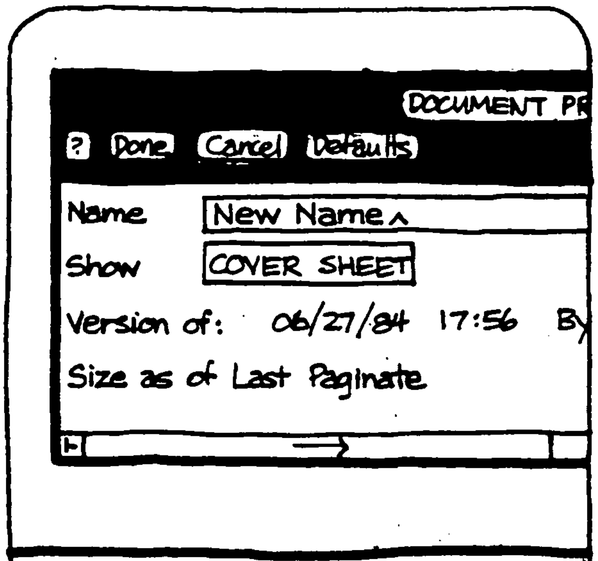
5. Press "DELETE" button.



5. continued (Document name is deleted.

TASK: RENAMING A DOCUMENT PAGE 3 OF 4

SYSTEM: STAR



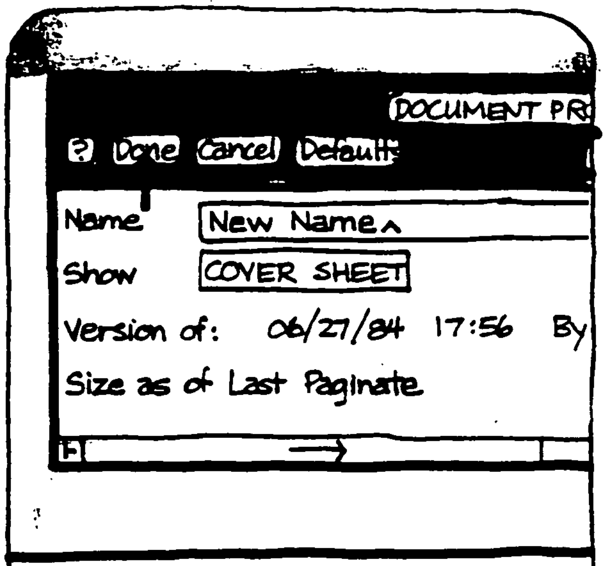
Name New Name^

Show COVER SHEET

Version of: 06/27/84 17:56 By

Size as of Last Paginate

6. Type in new name on keyboard. (New name appears in Properties Window.)



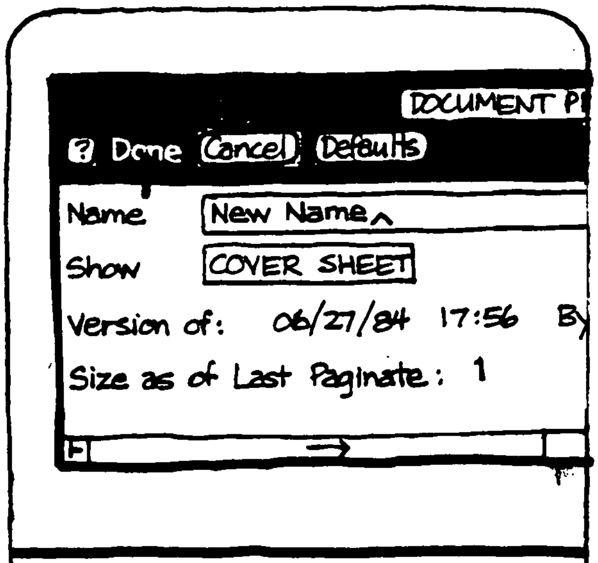
Name New Name^

Show COVER SHEET

Version of: 06/27/84 17:56 By

Size as of Last Paginate

7. Position cursor on "DONE" box.



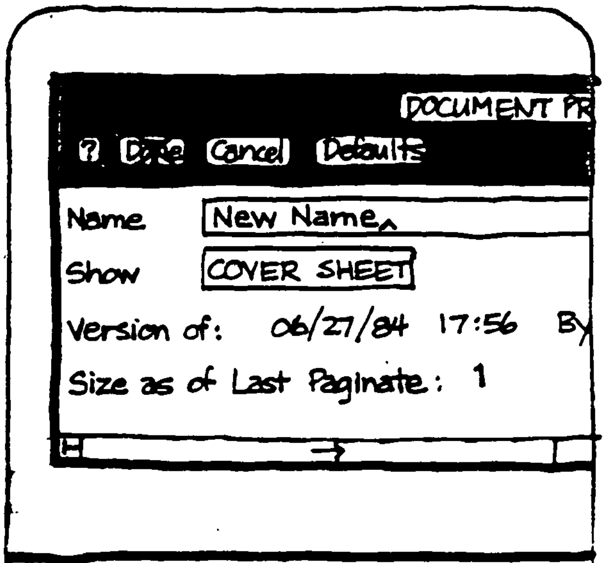
Name New Name^

Show COVER SHEET

Version of: 06/27/84 17:56 By

Size as of Last Paginate: 1

8. Press either mouse button. (As button is pressed, "DONE" box reverses video.)



Name New Name^

Show COVER SHEET

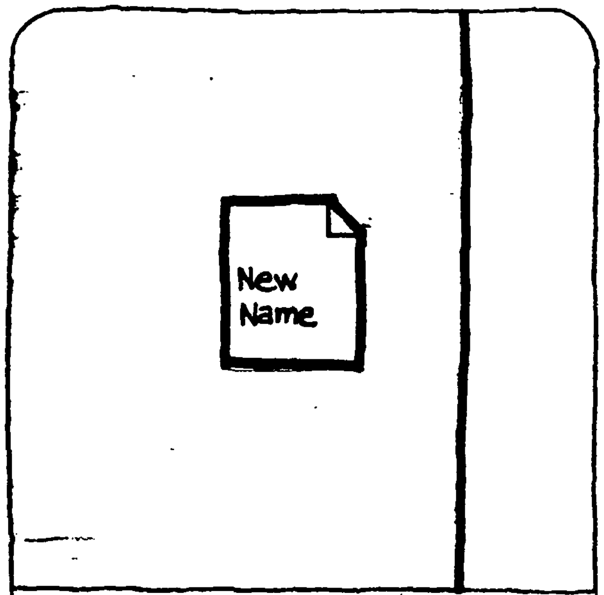
Version of: 06/27/84 17:56 By

Size as of Last Paginate: 1

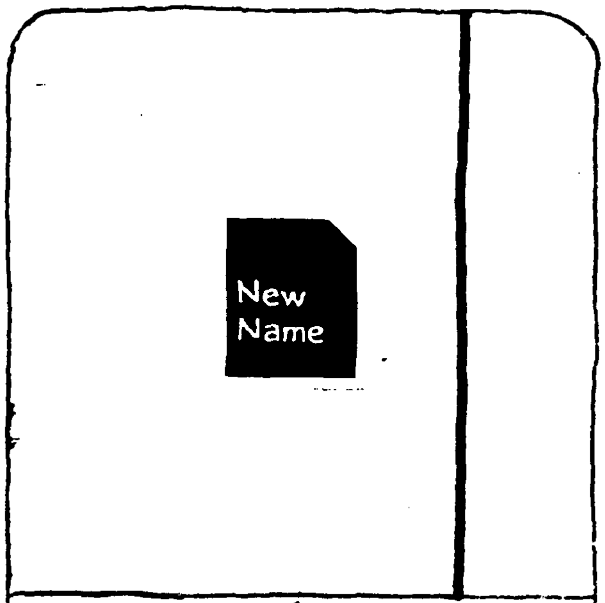
8. continued. (As button is released, cursor changes to hour glass shape.)

TASK: RENAMING A DOCUMENT PAGE 4 OF 4

SYSTEM: STAR



8. continued. (Properties window disappears. New name appears in document icon.)

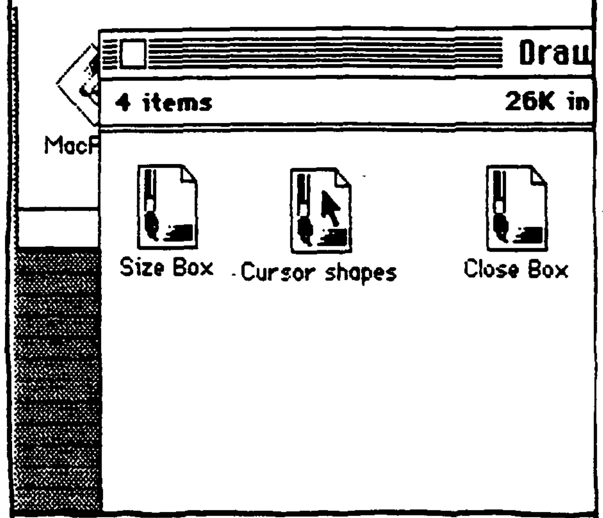


8. continued. (Icon reverses video. Cursor reverts to normal shape.)

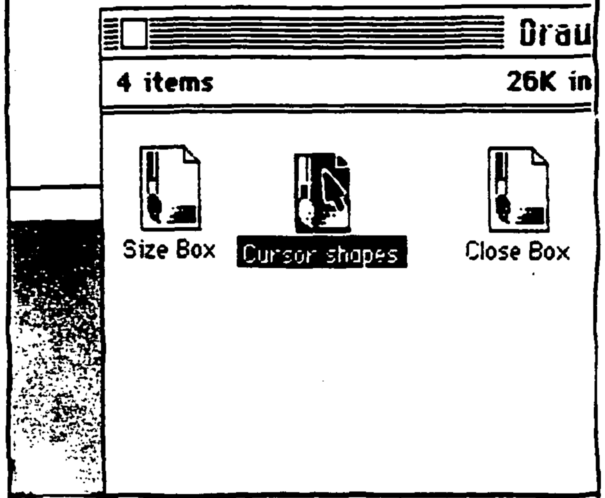
<END>

TASK: RENAMING A DOCUMENT

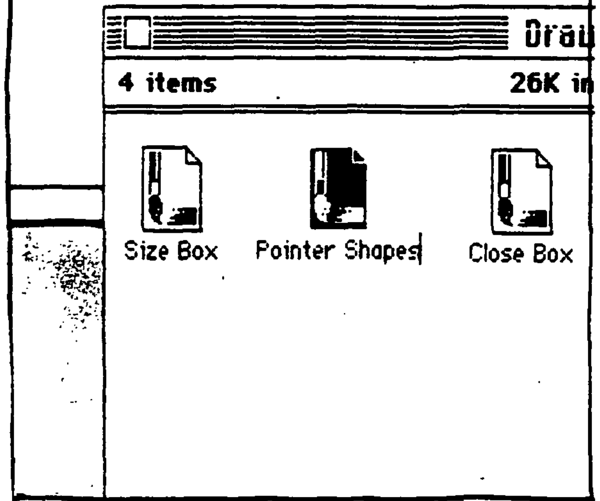
SYSTEM: MACINTOSH



1. Position cursor on desired document icon.



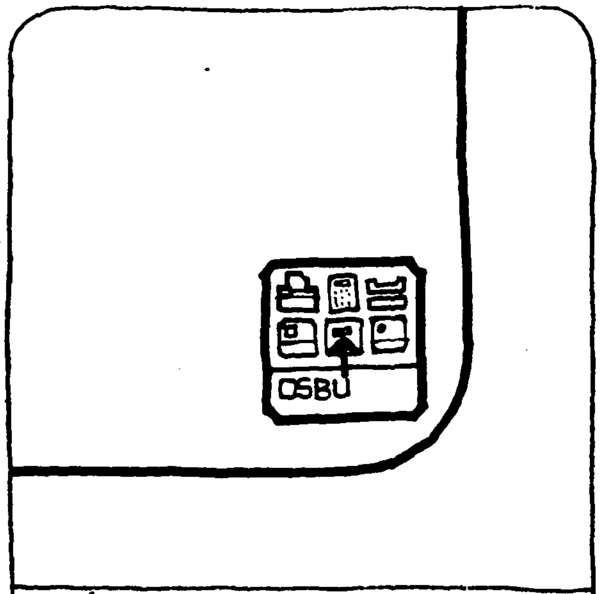
2. Press mouse button. (Icon reverses video.)



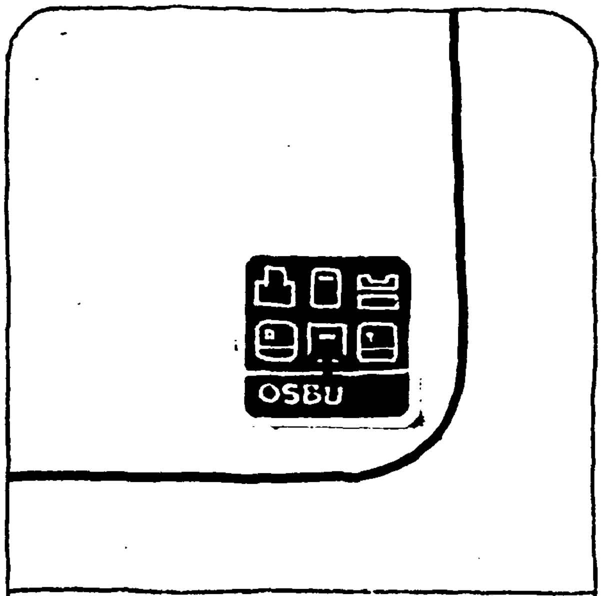
3. Type new name on keyboard. (Old name is deleted at first keystroke and new name appears under icon.)

<END>

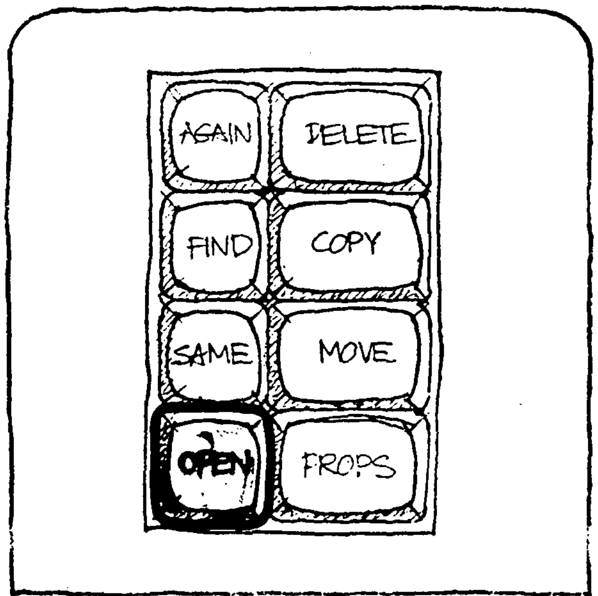
TASK: CREATING A NEW DOCUMENT PAGE 1 OF 6
SYSTEM: STAR



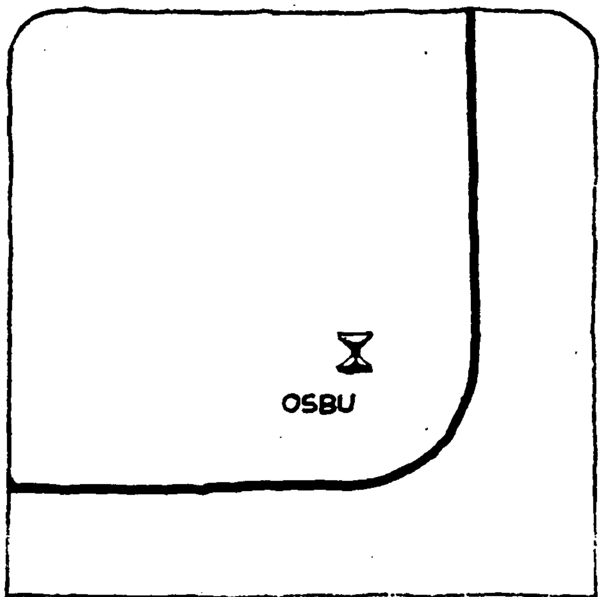
1. (If Blank Document icon is on desktop, then skip to Frame 15.) Position cursor on Directory icon.



2. Press left mouse button. (Icon reverses video.)



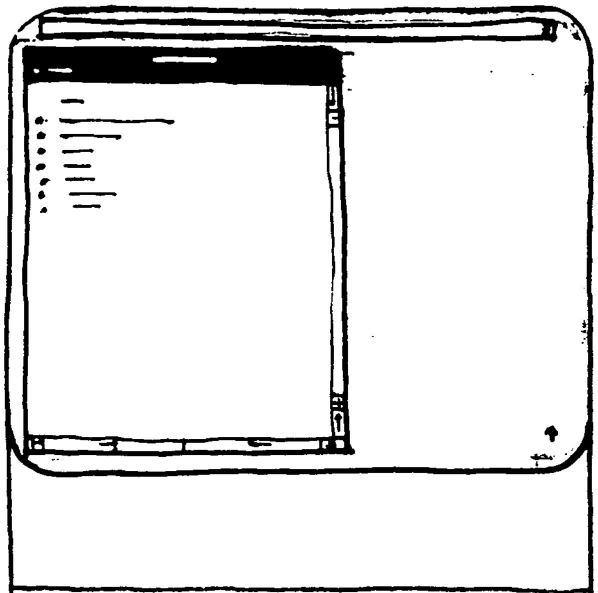
3. Press "OPEN" button.



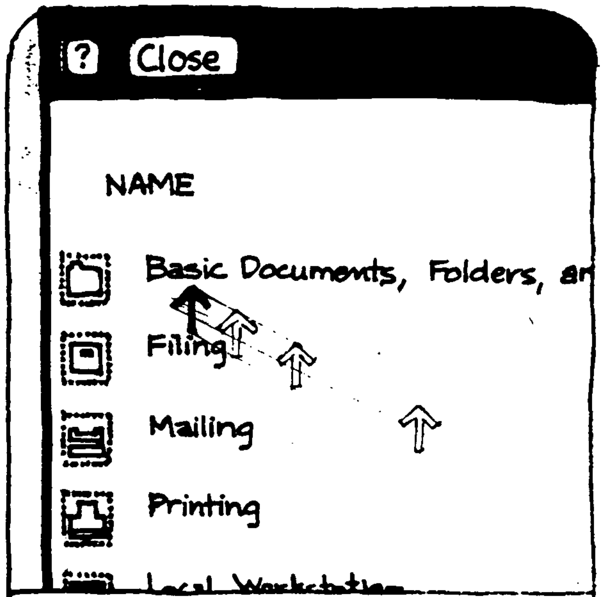
3. continued. (Cursor changes to hour glass shape. Icon disappears, leaving "hole".)

TASK: CREATING A NEW DOCUMENT PAGE 2 OF 6

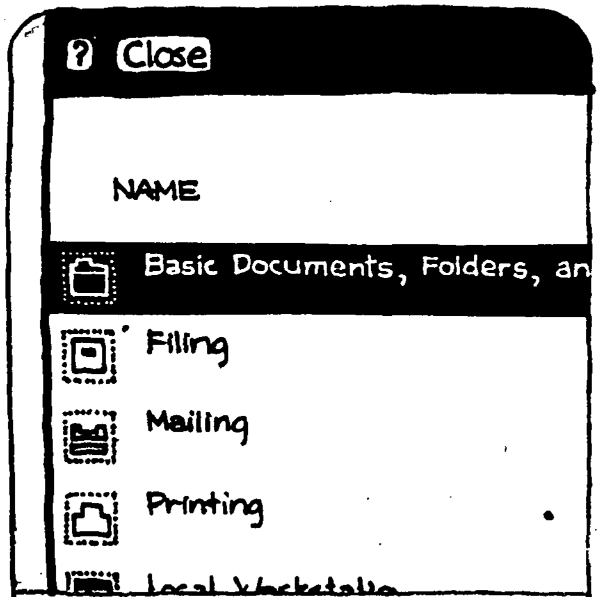
SYSTEM: STAR



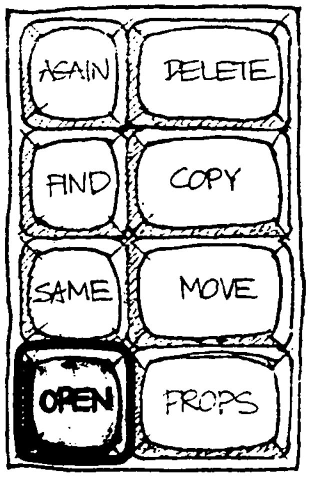
3. continued. (Window appears on left side of screen. Cursor reverts to normal shape.)



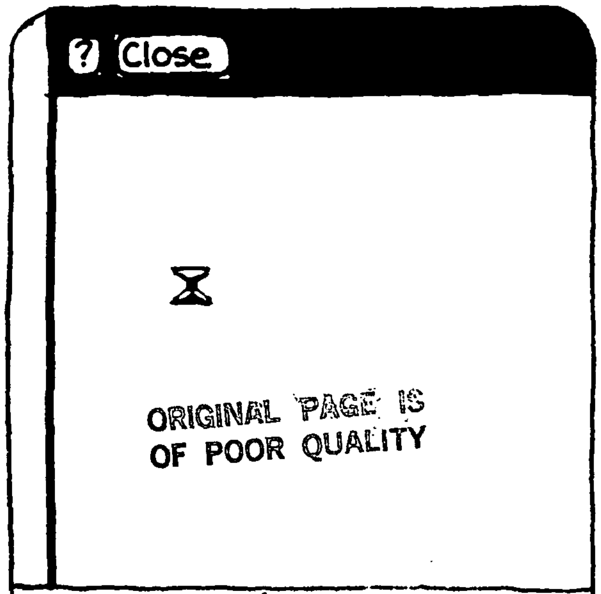
4. Position cursor on line labelled "Basic Documents, Folders and Record Files."



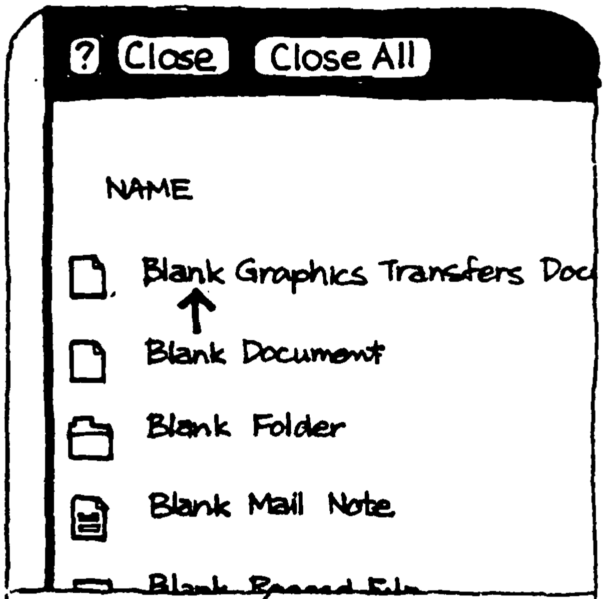
5. Press left mouse button. (Line reverses video.)



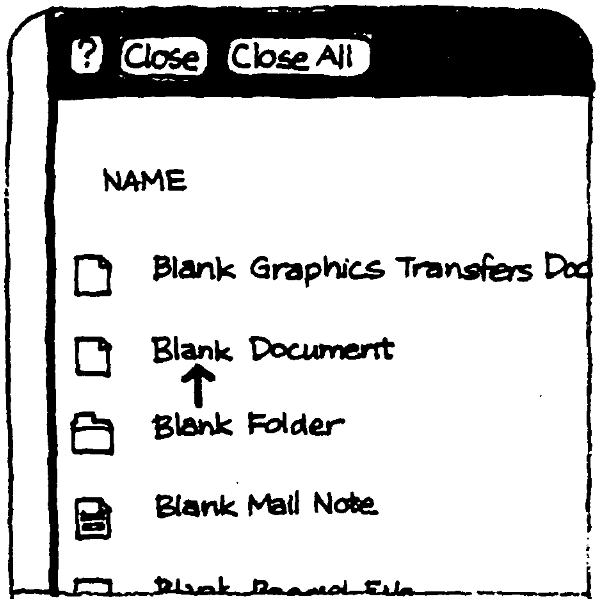
6. Press "OPEN" button.



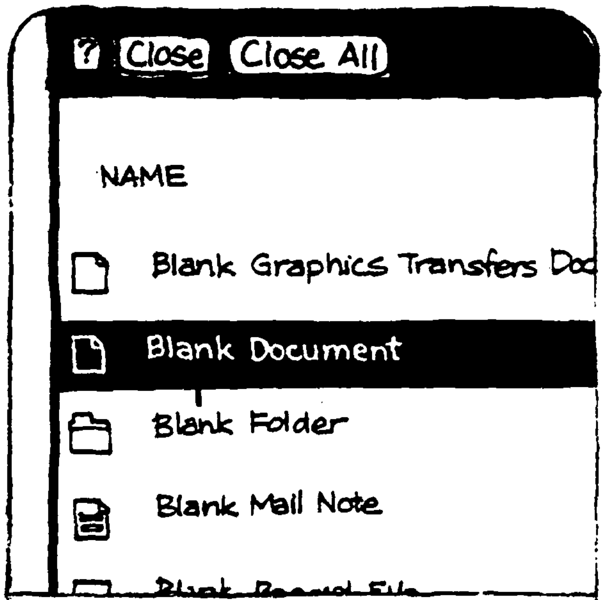
6. continued. (Cursor changes to hour glass shape. Window contents disappear.)



6. continued. (Window heading changes. New window contents appear. Cursor reverts to normal shape.)

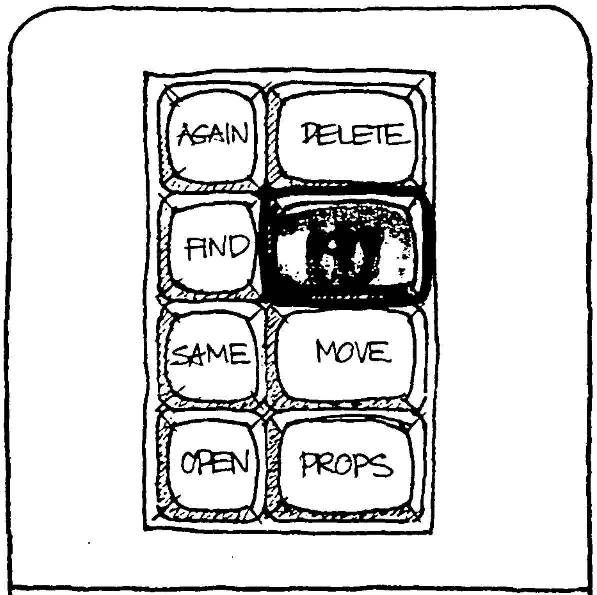


7. Position cursor on line labelled "Blank Document".

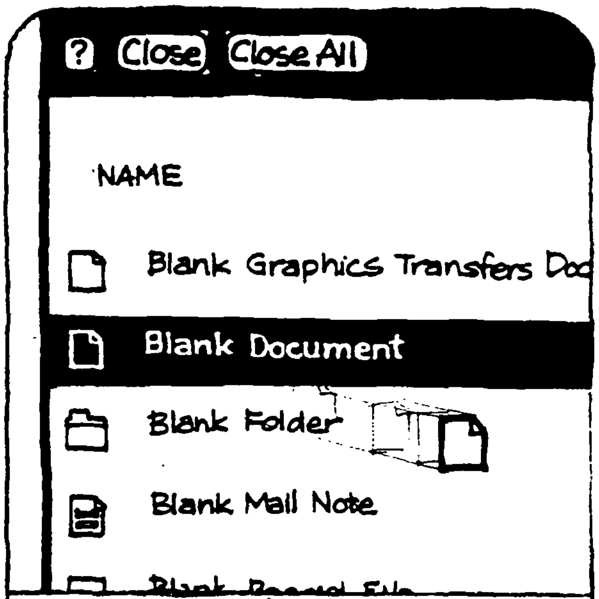


8. Press left mouse button. (Line reverses video.)

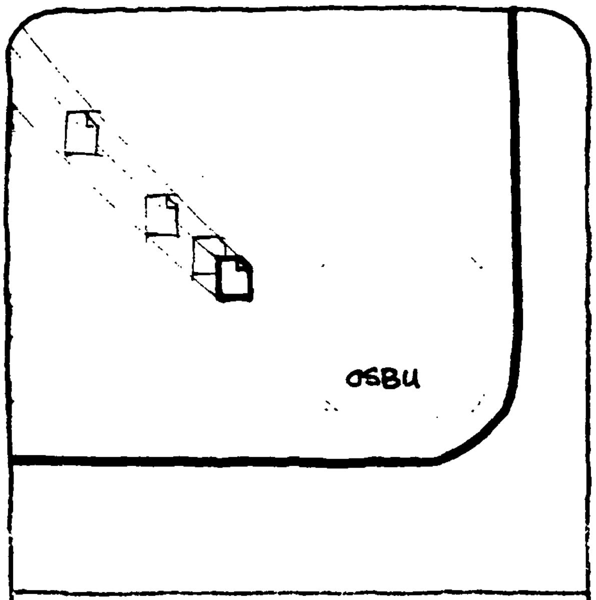
TASK: CREATING A NEW DOCUMENT PAGE 4 OF 6
SYSTEM: STAR



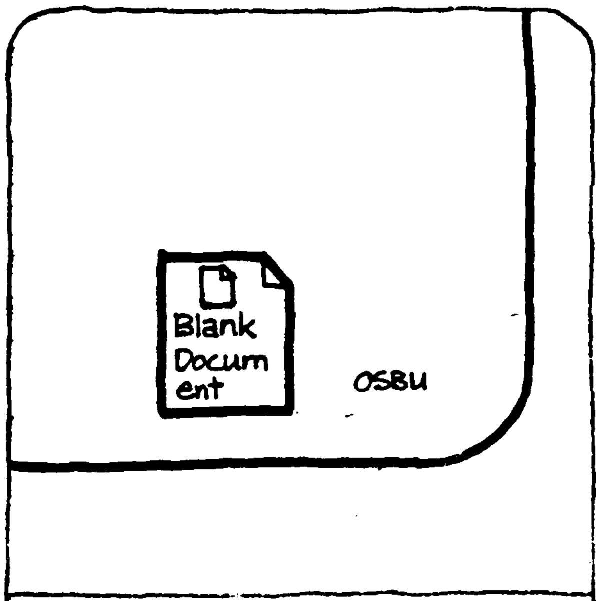
9. Press "COPY" key



9. continued. (Cursor changes shape. Message appears in prompt window: "Please indicate a destination with either button.")

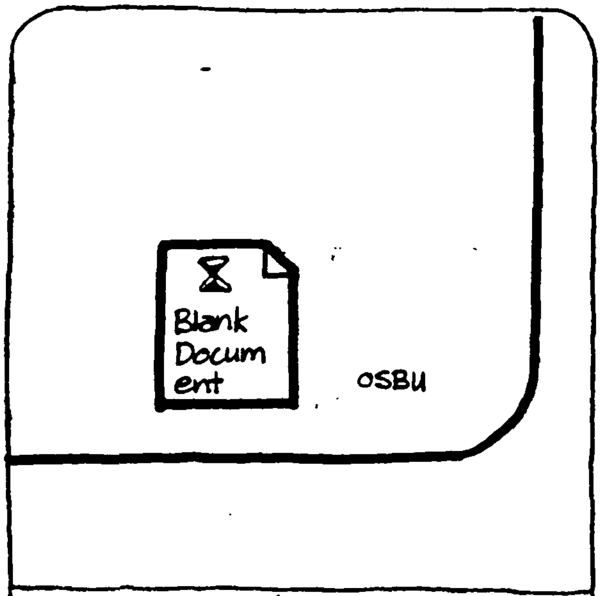


10. Position cursor in desired location.

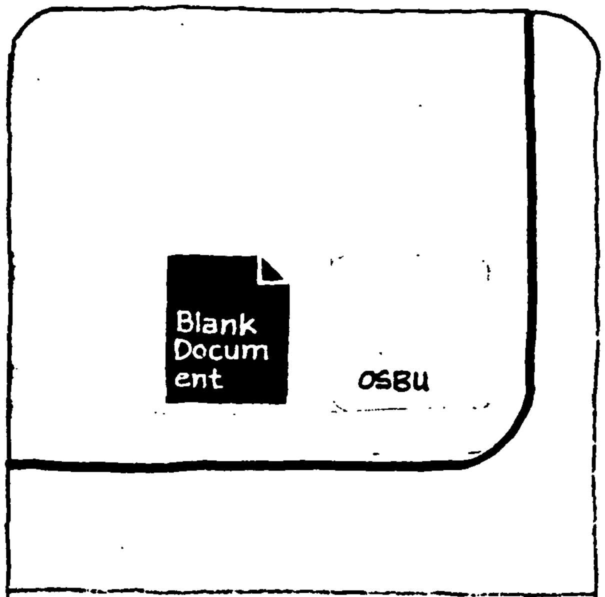


11. Press either mouse button. (While button is held, icon appears behind cursor.)

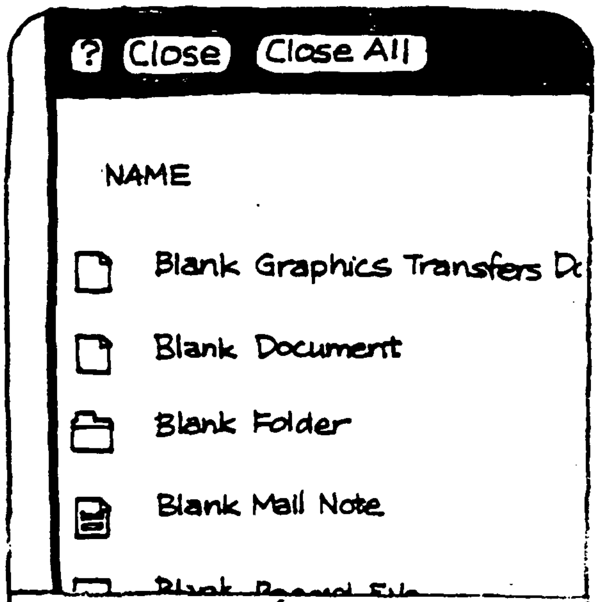
TASK: CREATING A NEW DOCUMENT PAGE 5 OF 6
SYSTEM: STAR



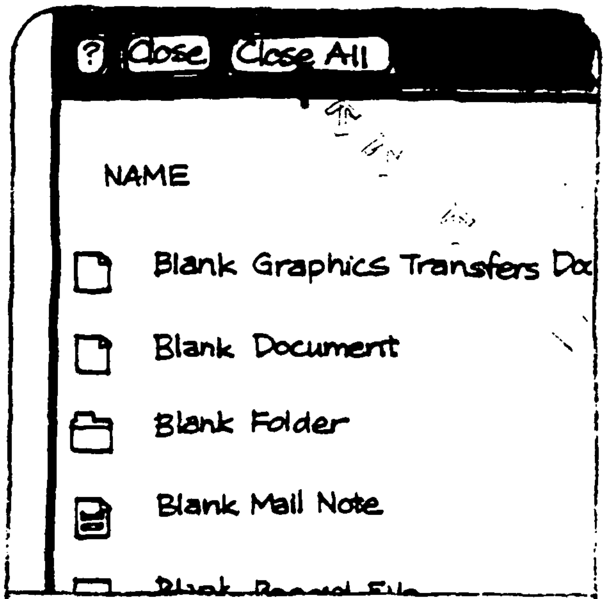
11. continued. (When button is released, cursor changes to hour glass shape.)



12. After 2 seconds, icon reverses video. Cursor reverts to normal shape.

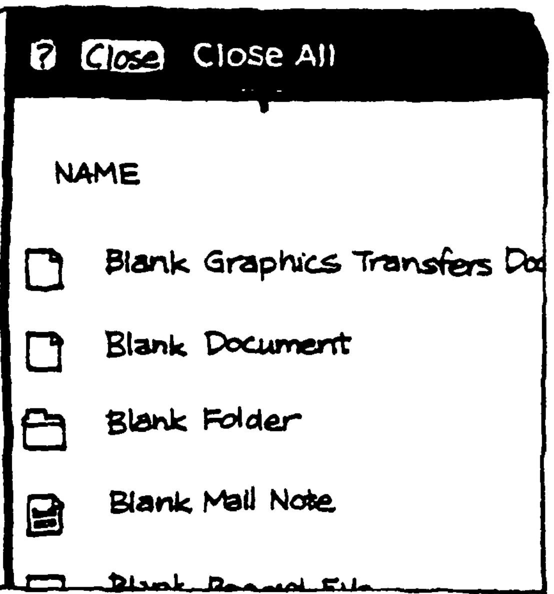


12. continued. (Line reverts to normal video.)

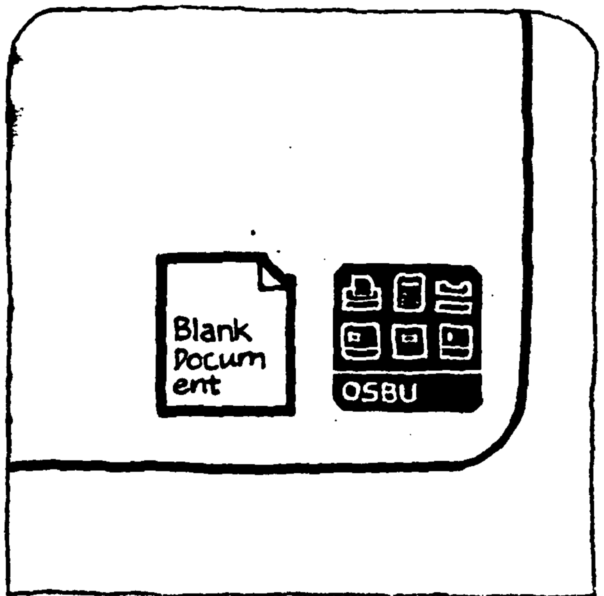


13. Position cursor on "Close All" box.

TASK: CREATING A NEW DOCUMENT PAGE 6 OF 6
SYSTEM: STAR



14. Press either mouse button. (As button is pressed, "Close All" box reverses video.)



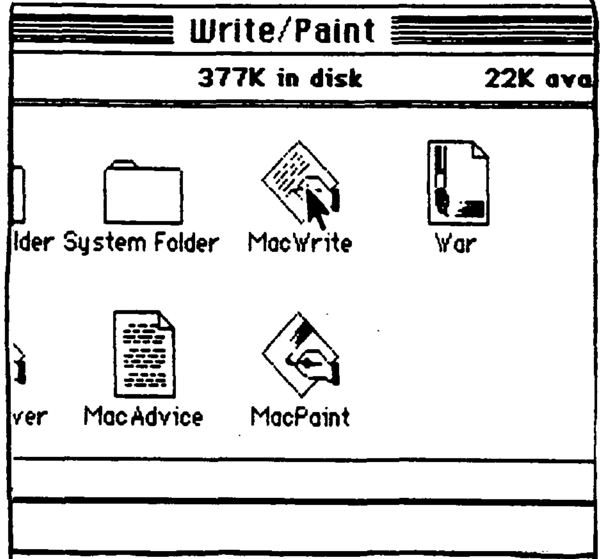
14. continued. (As button is released, window disappears, Directory icon reappears in reverse video. Document icon reverts to normal video.)

15. Make a copy of the Blank Document and rename it. (See Copying a Document and Renaming a Document.)

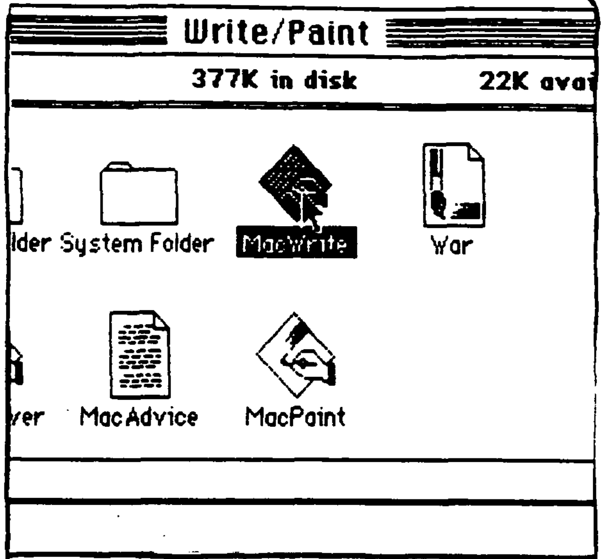
<END>

TASK: CREATING A NEW DOCUMENT PAGE 1 OF 3

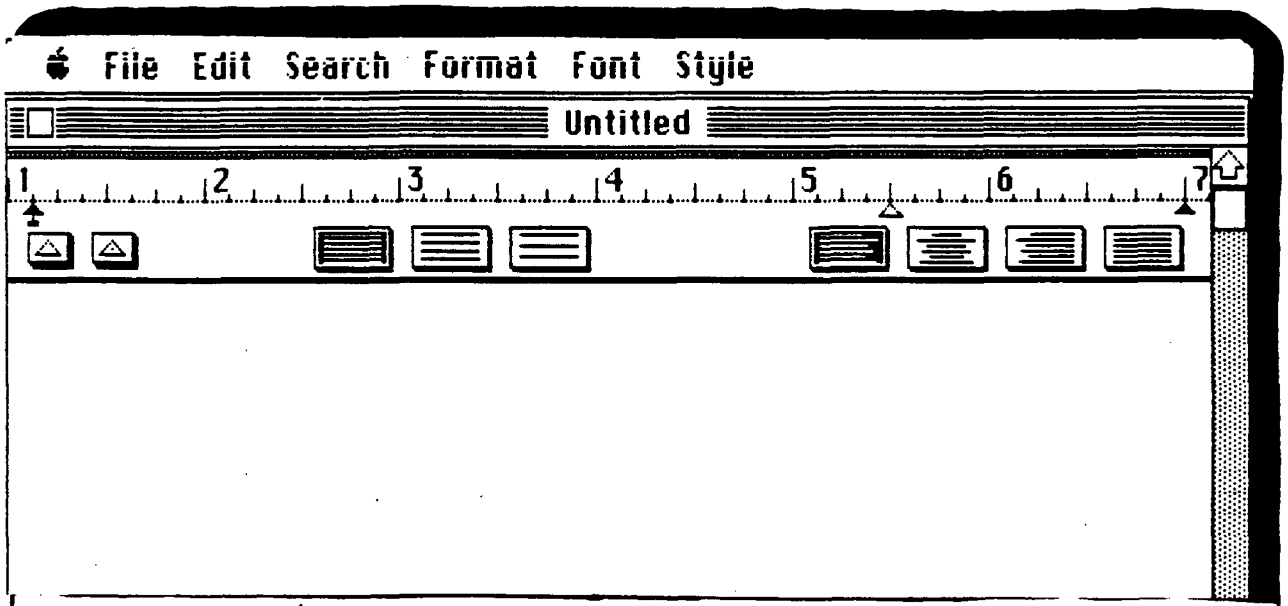
SYSTEM: MACINTOSH



1. Position cursor on desired application icon.

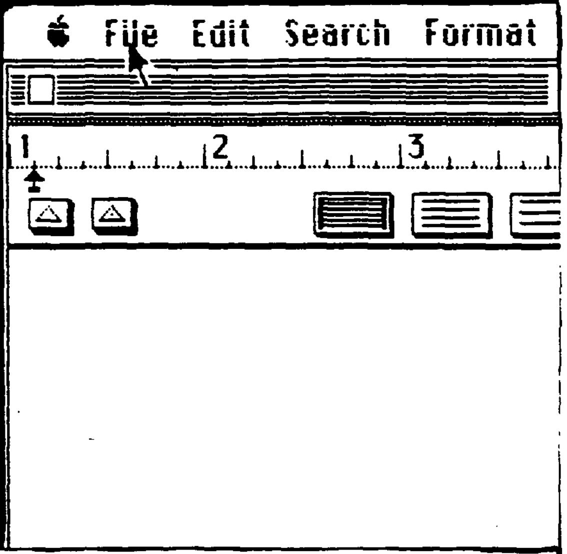


2. Press the mouse button twice. (On the first press, icon reverses video.)

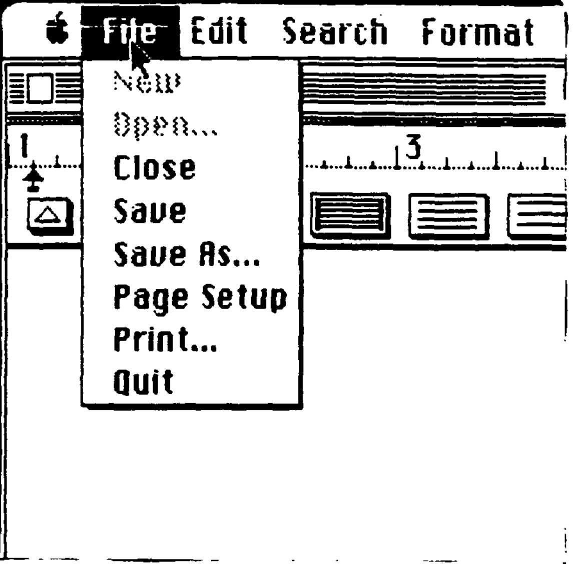


2. continued. (On the second press, icon grows into application window.)

TASK: CREATING A NEW DOCUMENT PAGE 2 OF 3
SYSTEM: MACINTOSH

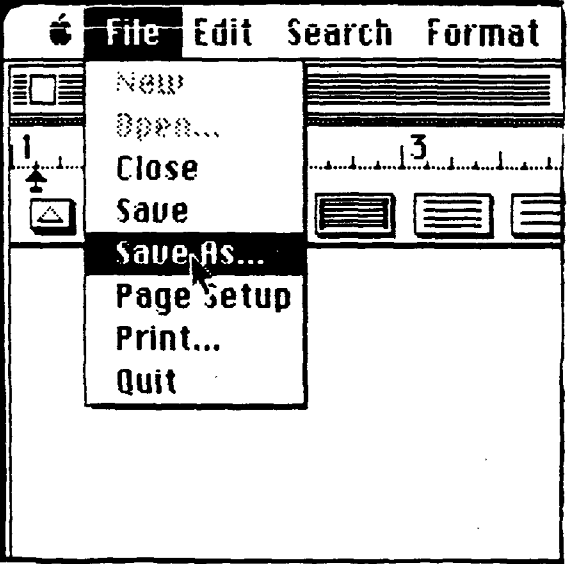


3. When finished with the application, position cursor on "FILE" label.

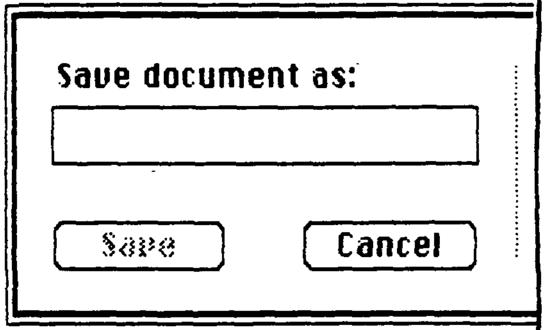


4. Press and hold mouse button. ("FILE" reverses video. Menu appears.)

ORIGINAL PAGE IS
OF POOR QUALITY



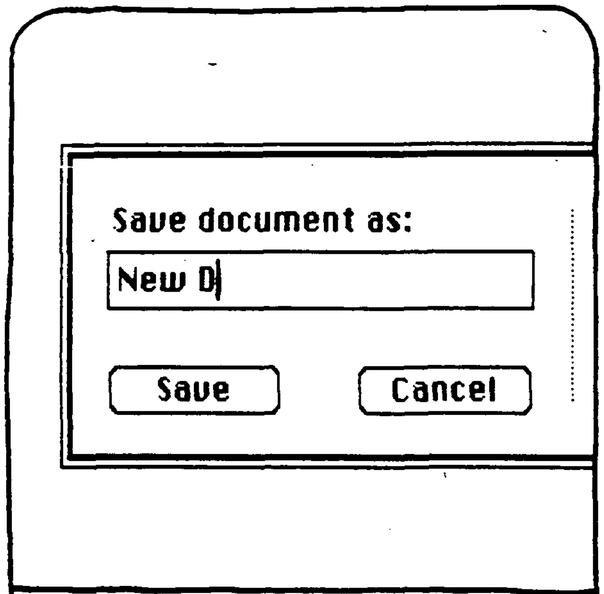
5. Position cursor on "SAVE AS..." command. ("SAVE AS..." reverses video.)



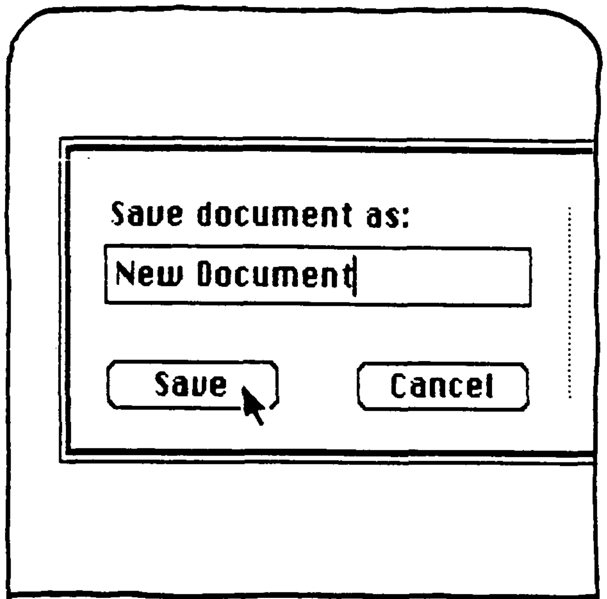
6. Release mouse button. (Menu disappears. Prompt window appears in front of application window.)

TASK: CREATING A NEW DOCUMENT PAGE 3 OF 3

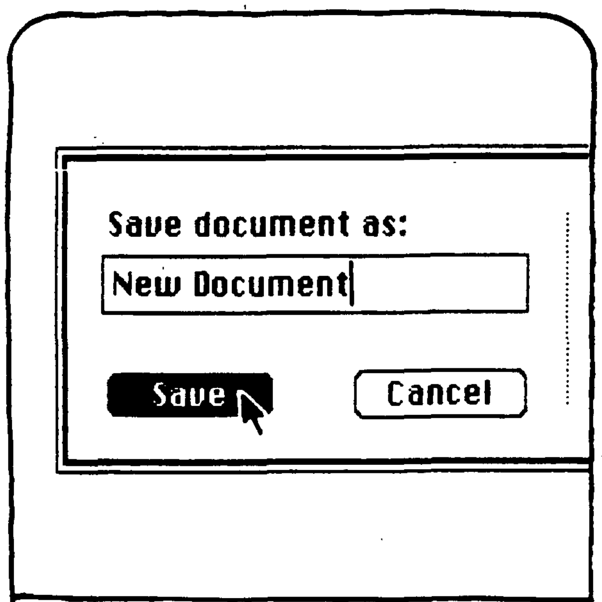
SYSTEM: MACINTOSH



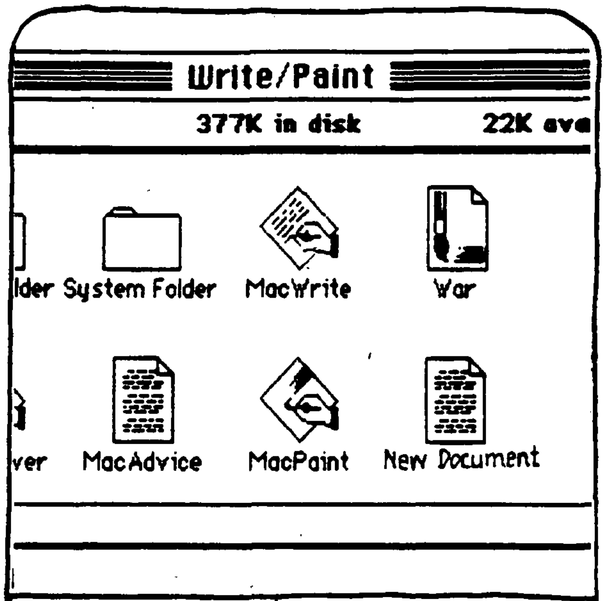
7. Type document name on keyboard. ("SAVE command turns black at first keypress. Document name appears in prompt window.



8. Position cursor on "SAVE" command.

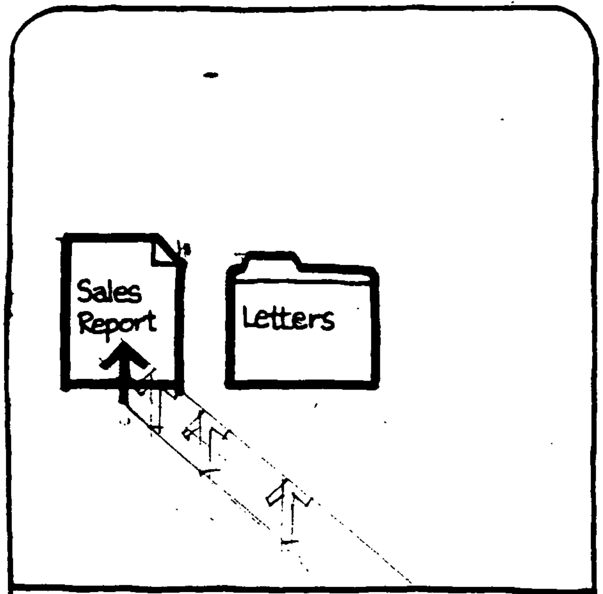


9. Press mouse button. ("SAVE reverses video.)

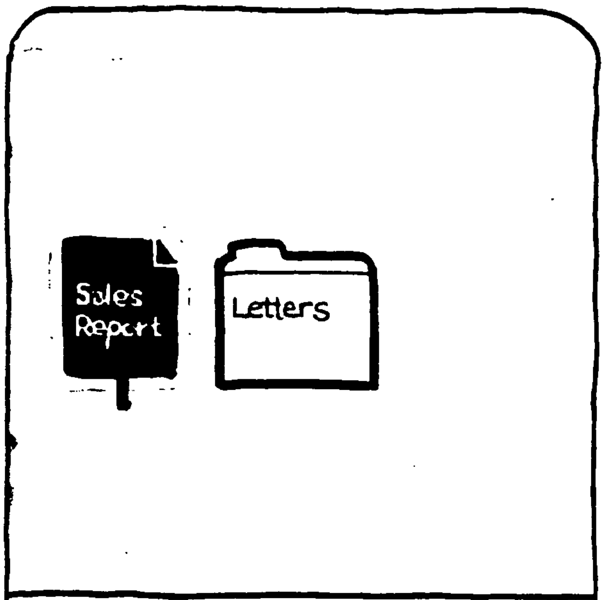


9. continued (Application and Prompt windows disappear. New icon appears in same window as application.)

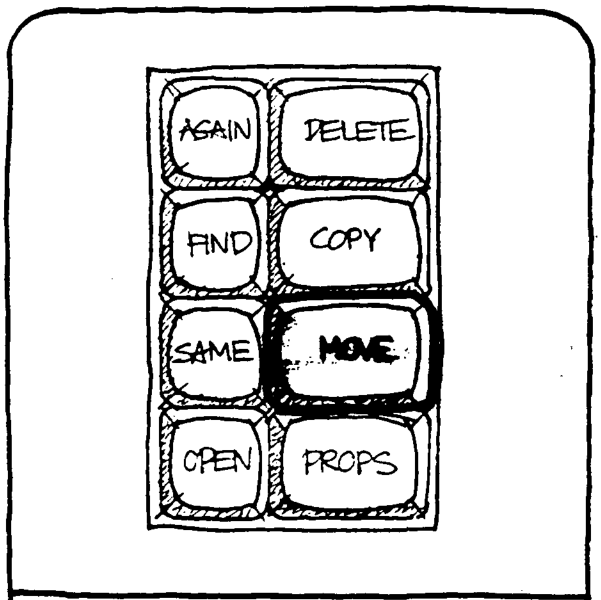
TASK: MOVING AN ICON PAGE 1 OF 2
SYSTEM: STAR



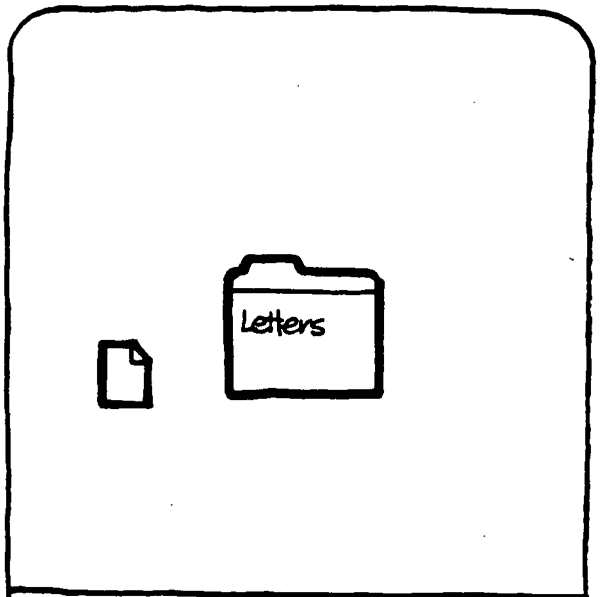
1. Position cursor on desired icon.



2. Press left mouse button. Icon reverses video.



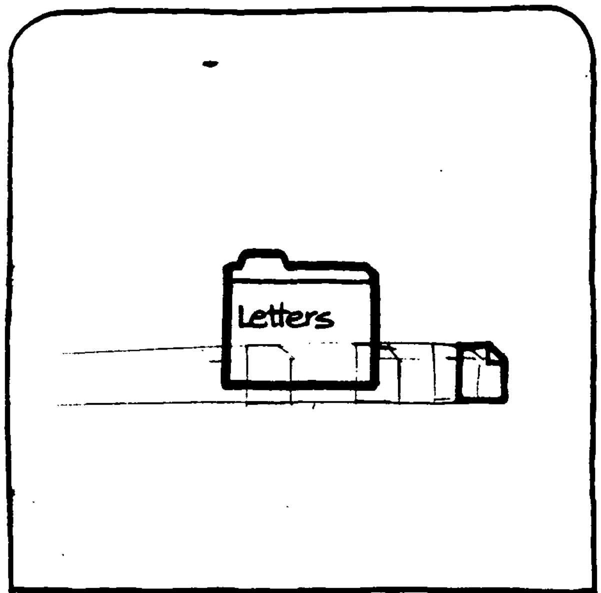
3. Press "MOVE" button. Message appears in prompt window: "Please indicate a destination with either button."



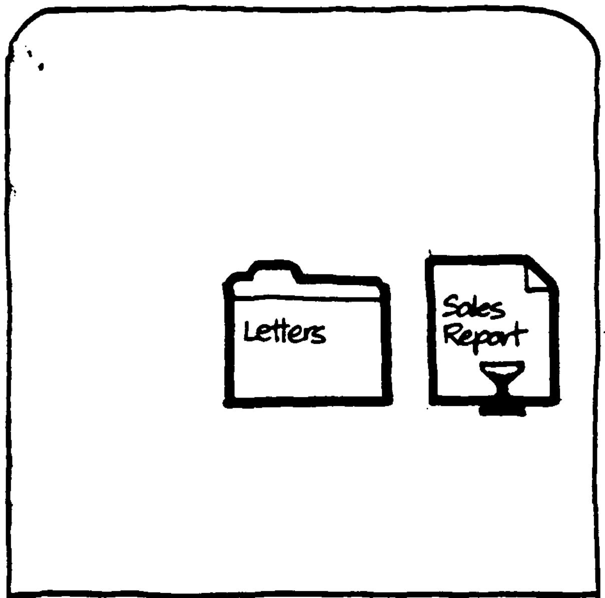
3. (continued) Icon disappears. Cursor changes into small version of icon.

TASK: MOVING AN ICON (continued) PAGE 2 OF 2

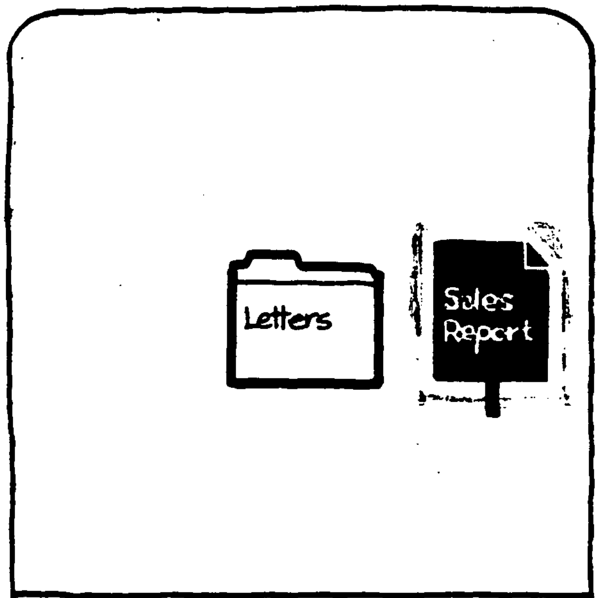
SYSTEM: STAR



4. Position cursor at desired destination.



5. Press either mouse button. Icon appears in desired position. Cursor changes to hourglass shape.

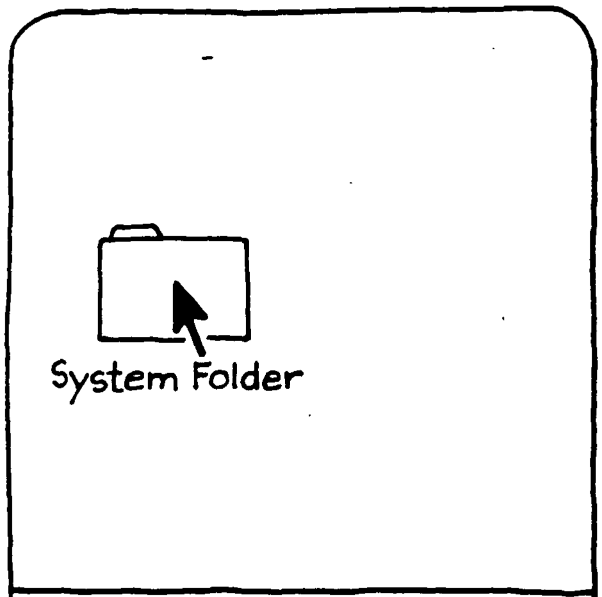


6. Icon reverses video. Cursor reverts to normal shape.

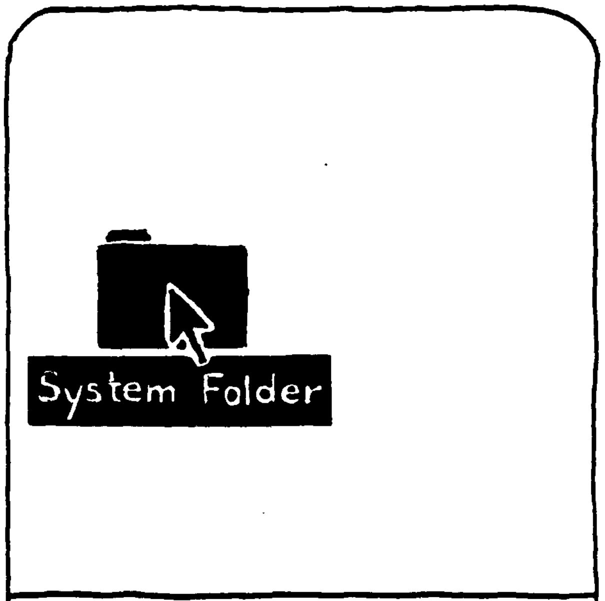
<END>

TASK: MOVING AN ICON

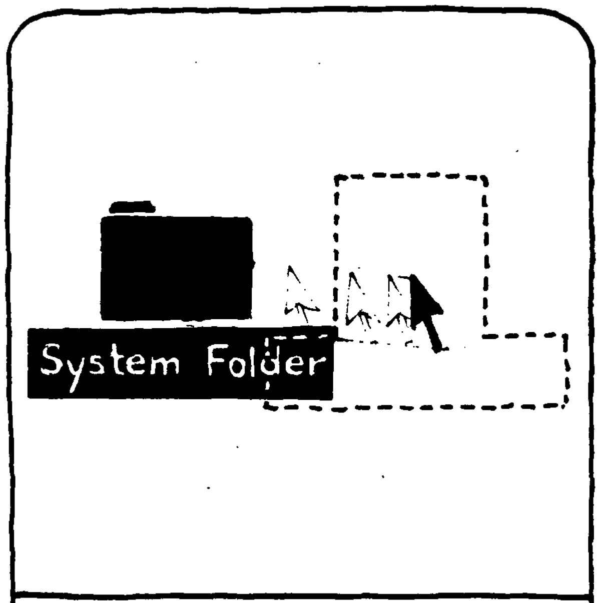
SYSTEM: MACINTOSH



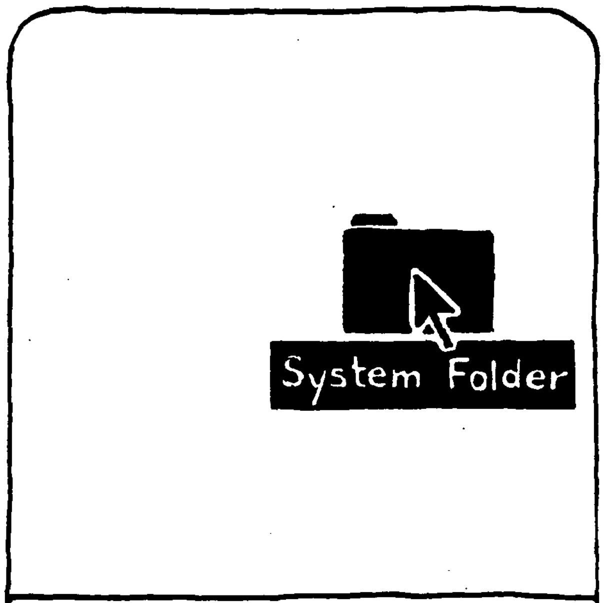
1. Position cursor on desired icon.



2. Press and hold mouse button. Icon reverses video.



3. Move cursor to desired position. "Ghost" image follows movement of cursor.

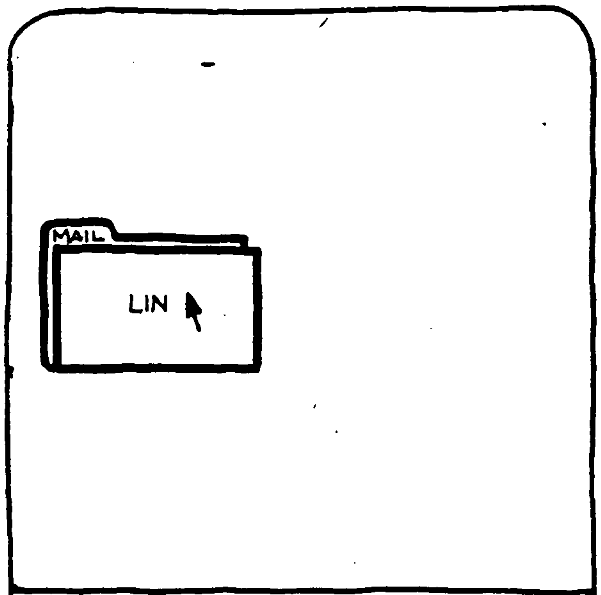


4. Release mouse button. Icon disappears and reappears in position of "ghost".

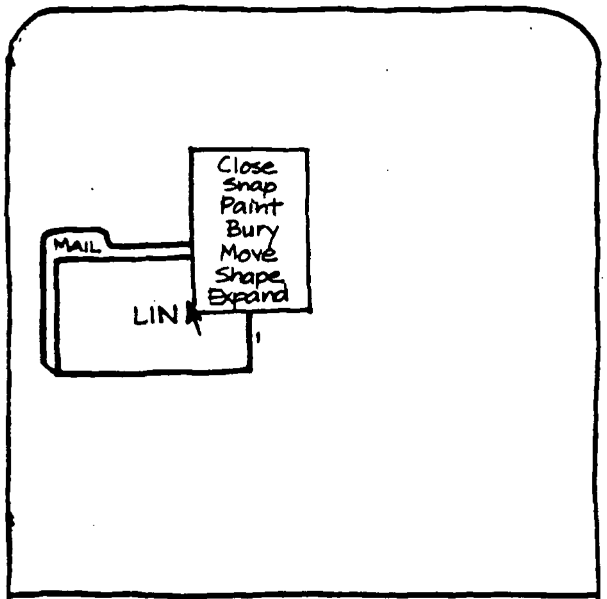
<END>

TASK: MOVING AN ICON PAGE 1 OF 2

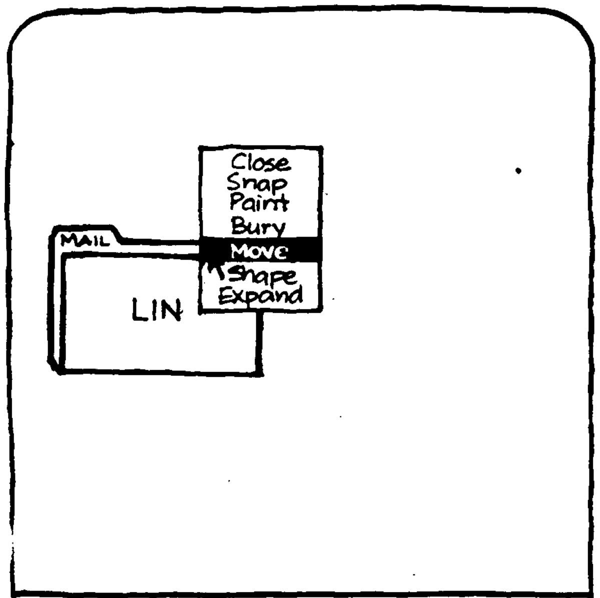
SYSTEM: INTERLISP-D



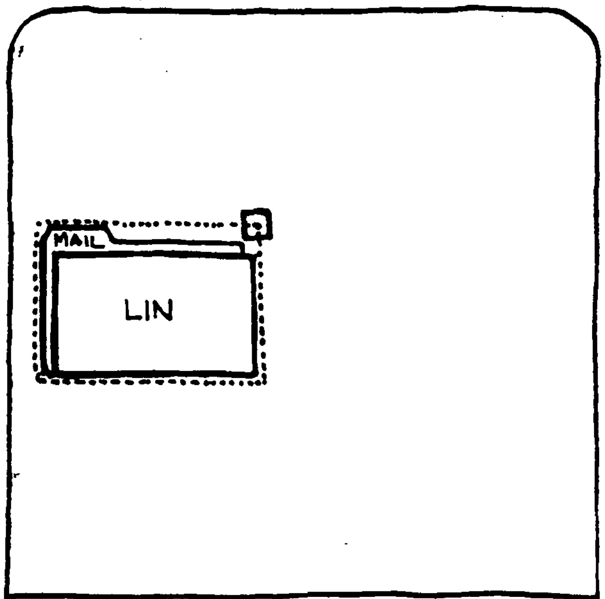
1. Position cursor on desired icon.



2. Press and hold right mouse button. (Menu pops up.)



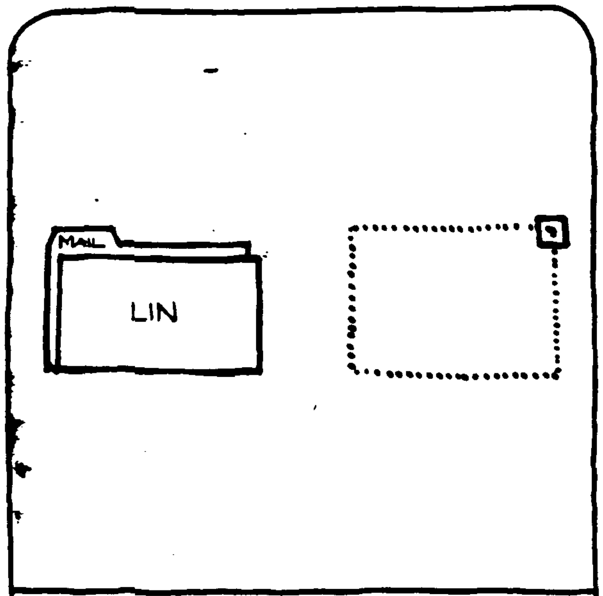
3. Position cursor on "Move" command. ("Move" reverses video.)



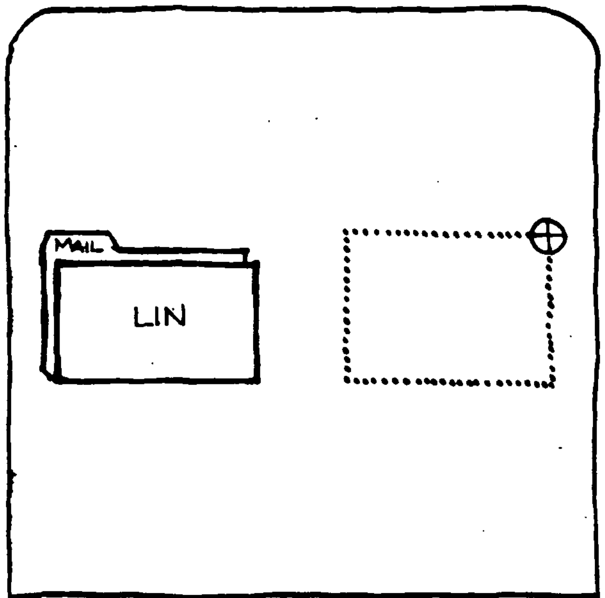
4. Release right mouse button. (Menu and cursor disappear. Box appears around icon. Square shaped cursor in top right corner.)

TASK: MOVING AN ICON

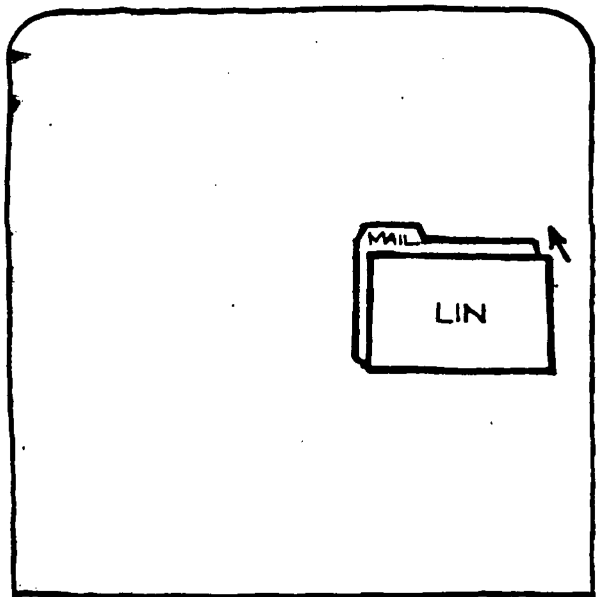
SYSTEM: INTERLISP-D



5. Use cursor to "drag" box to desired location.



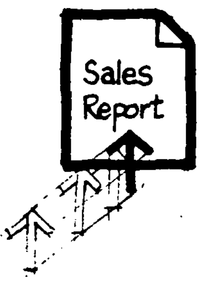
6. Press left or middle mouse button. (As button is pressed, cursor changes to cross shape.



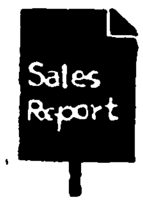
6. continued. (As button is released, icon disappears and reappears in new location. Cursor reverts to normal shape.)
<END>

TASK: OPENING AN ICON PAGE 1 OF 2

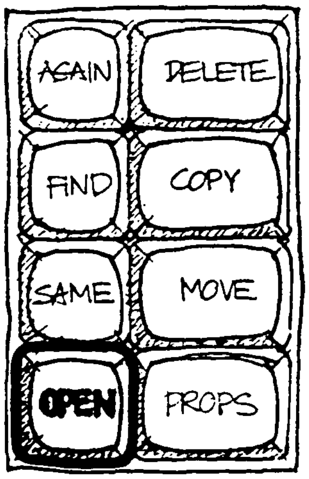
SYSTEM: STAR



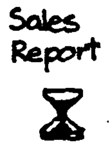
1. Position cursor on desired icon.



2. Press left mouse button. Icon reverses video.



3. Press "OPEN" button.

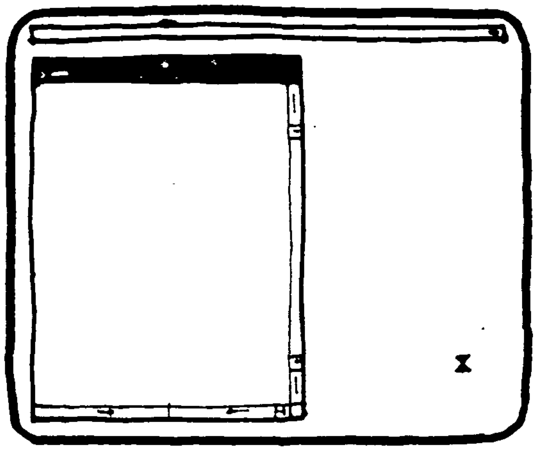


3. (continued) Icon disappears, leaving "hole". Cursor changes into hour glass.

TASK : OPENING AN ICON

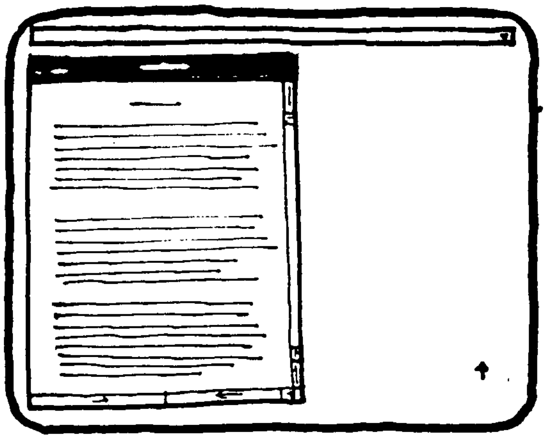
SYSTEM : STAR

PAGE 2 OF 2



3. (continued) Empty window appears on left side of screen.

4. Cursor reverts to normal shape.



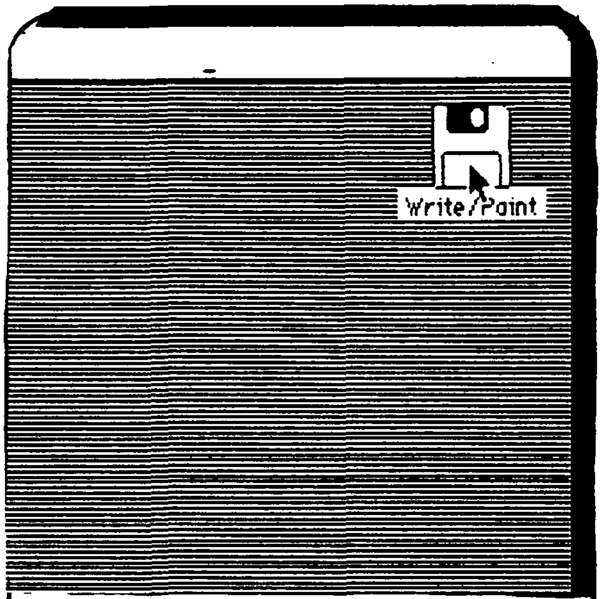
4. (continued) Contents of icon appear in window.

< END >

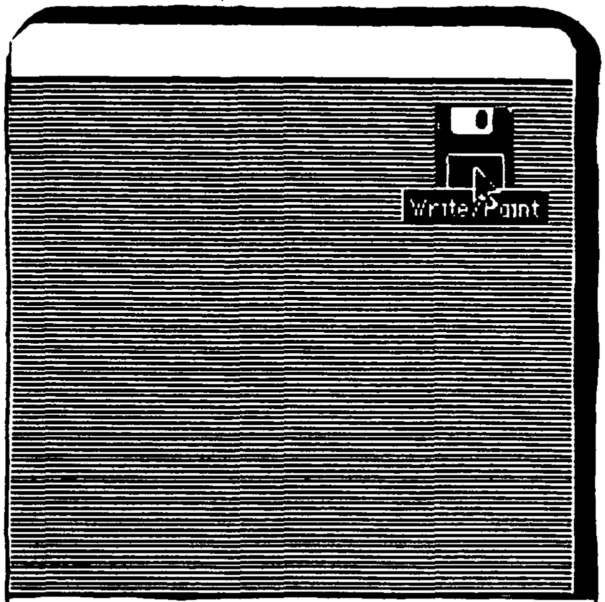
54.

TASK: OPENING A WINDOW PAGE 1 OF 2

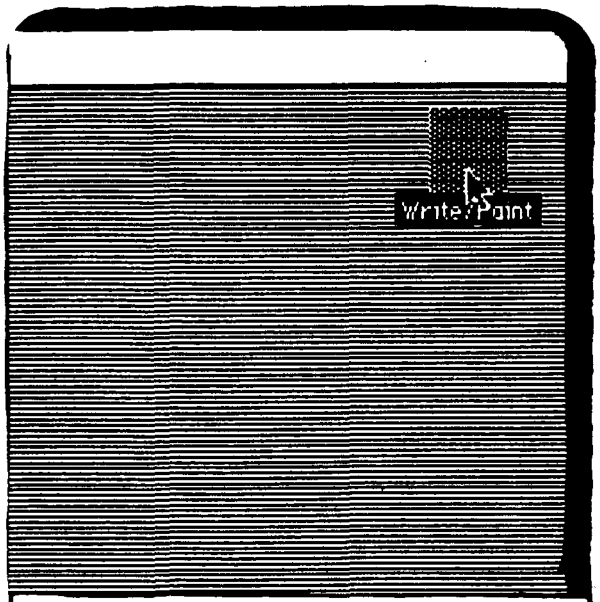
SYSTEM: MACINTOSH



1. Position cursor on icon representing desired information



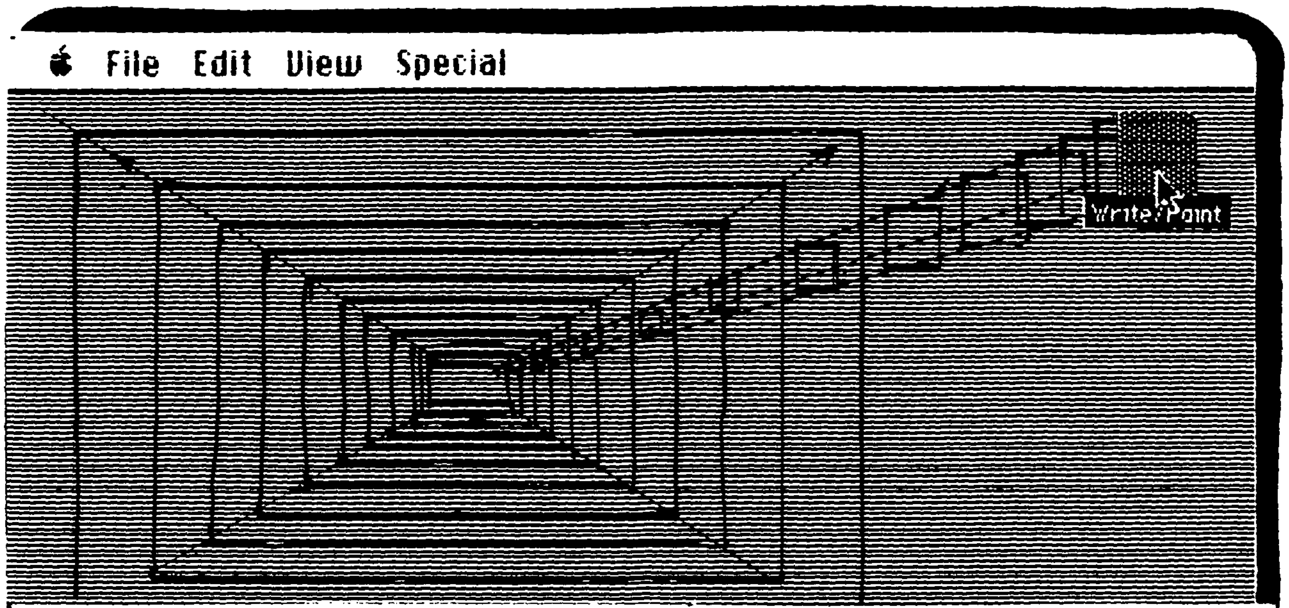
2. Press mouse button twice. (After first press, icon reverses video.)



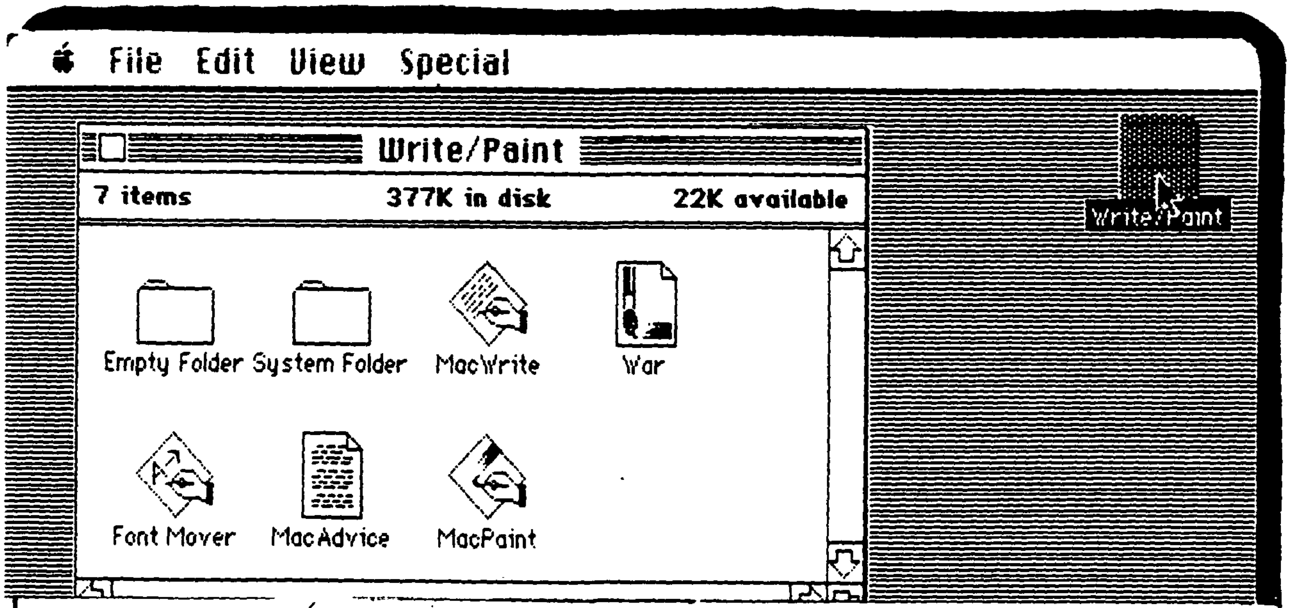
2. continued. (After second press, icon turns dark gray.)

see next frame.

TASK: OPENING AN ICON PAGE 2 OF 2
SYSTEM: MACINTOSH



2. continued. (Outline of box detaches from icon, shrinks and moves towards center of screen, then grows to size and shape of window.)



2. continued. (Window appears.)

<END>

APPENDIX A

WINDOW DESIGNS
84.0816 S. Card

(USER)	(SYSTEM)

1. Open Window (= Expand Icon)

Star:

P[icon]	Cursor arrow follows mouse.
K[M3]	Icon reverse video.
K[OPEN]	Icon turns into hole;
	Cursor ← Hourglass;
	Empty window appears
	Cursor ← Normal arrow;
	Window fills

Macintosh:

P[icon]	Cursor arrow follows mouse.
K[M1]	Icon reverse video.
K[M1] (ra)	Icon turns dark gray;
	Copy of icon outline shrinks to dot as it moves to center;
	Dot expands to window.

Lisp:

P[icon]	Cursor arrow follows mouse.
KD[M3]	Menu pops up.
P2[Expand]	Reverse video bar follows mouse.
KU[M3]	Window appears.

--or--

P[icon]	Cursor arrow follows mouse.
K[M2]	Menu expands.

2. Close Window (= Shrink to Icon)

Star:

P[CLOSE]	Cursor arrow follows mouse.
K[M1]	"CLOSE" box reverses video;
	Cursor ← Hourglass;
	Titlebox shrinks;
	Window disappears and icon outline fills.

Lisp:

P[window]	Curor arrow follows mouse.
KD[M3]	Menu pops up.
P2[Shrink]	Reverse video bar follows mouse.
KU[M3]	Window disappears;
	Icon appears.

3. Move window.

Star:

P[lrsq]	Curor arrow follows mouse.
K[M1]	lrsq box ← reverse video.
K[MOVE]	Instruction message appears in message window;
	lrwq box ← normal video;
	Cursor ← arrow with box;
P3[loc]	Cursor boxarrow follows mouse.
KD[M1]	Bottom of box dotted line follows mouse.
KU[M1]	Cursor ← hourglass;
	Bottom of box reappears in new position;

```
                  Cursor ← arrow;
                  lrsq box ← reverse video (to indicate selected).

Macintosh:
     ?

Lisp:
     P[window]    Curor arrow follows mouse.
     KD[M3]       Menu pops up.
     P2[Move]     Reverse video bar follows mouse.
     KU[M3]       Cursor ← Square;
                  Dotted window outline appears;
     P3[loc]      Dotted line box follows movement of cursor.
     KD[M1]       Cursor ← Crosshair.
     KU[M1]       Full window appears in outine;
                  Cursor ← Arrow.

4. Adjust size of window.

Star:
     P[lrsq]      Curor arrow follows mouse.
     K[M1]        lrsq box ← reverse video.
     K[MOVE]      Instruction message appears in message window;
                  lrwq box ← normal video;
                  Cursor ← arrow with box;
     P3[loc]      Cursor boxarrow follows mouse.
     KD[M1]       Bottom of box dotted line follows mouse.
     KU[M1]       Cursor ← hourglass;
                  Bottom of box reappears in new position;
                  Cursor ← arrow;
                  lrsq box ← reverse video (to indicate selected).

Lisp:
     P[window]    Cursor arrow follows mouse.
     KD[M3]       Menu pops up.
     P2[Shape]    Reverse video bar follows mouse.
     KU[M3]       Cursor ← Arrow and pointything;
                  Small dotted rectangle outline appears;
     P3[loc]      Dotted line box follows movement of cursor.
     KD[M1]       Cursor ← LRCorner.
     P3[loc]      LR Rectangle corner follows mouse.
     KU[M1]       Blank window appears, size of rectangle
                  Window fills in;
                  Cursor ← Arrow.

-- Alternative to last action --

     KD[M3]       Cursor ← Tongs.
     KU[M1]       Cursor ← Open tongs.
     P3[loc]      Tongs follow mouse.
     KD[M1]
     KU[M3]       Cursor ← UL corner.
     P3[loc]      UL corner of box follows mouse.
     KU[M1]       Blank window appears, size of rectangle
                  Window fills in;
                  Cursor ← Arrow.


Macintosh:
     P[lrsq]      Curor arrow follows mouse.
     KD[M1]       Dotted line appears on bottom and right side.
     P3[loc]      Dotted bottom and right follow mouse.
     KU[M1]       Bottom and right side of window fram disappears
                   and reappers in new position.
```

Layout 4: The Fisheye View – A Short Description

By: Lennart Lovstrand
Filed in: {Phylum}<Lovstrand>Chart>Layout4.tedit
Date: 24-Jul-85 15:47:30

The Layout 4 displays an IDN tree in a fisheye view. It scales all nodes according to their distance from a chosen point or level of focus, keeping the proportions of each node's box constant. In order to closer describe how the scaling is performed, a piece of mathematical framework is needed. Assume a world concisting of a two-dimensional plane where a tree is being displayed as a collection of rectangles and connecting lines. Besides its "natural" vertical order ranging from root to leaf, it is also considered to have a horizontal order – here represented by *branch numbers* – ranging from a node's leftmost branch to its rightmost. We are now ready to define our basic entities:

A *branch number* is a non-negative integer, representing the nth leftmost branch relative to some node in the tree, with 0 denoting the far leftmost branch on each level.
A *node position* is a sequence of *branch numbers*, representing a specific node in the tree. The node is found by walking along each of the branches relative to the root of the tree, with the empty sequence denoting the root of the tree.
The *level* of a *node position* is the length of its sequence, with 0 denoting the root level.
Finally, we need the function *least upper bound (lub)* of two nodes, which have the normal meaning of denoting the closest node that is "above or equal" to both of them.

This framework lets us to continue by defining something more useful, such as the *walking distance* (d_w) from one node to another, informally described as the number of levels to pass in going from one of the nodes via the level beneith their *lub* to the other node. We also want the *branching distance* (d_b), which is the absolute difference between the two node's branch numbers relative to their *lub*:

$$d_w(a, b) = level(lub(a, b)) - 1 - level(a) + level(lub(a, b)) - 1 - level(b);$$
$$d_b(a, b) = |\, branch(lub(a, b), a) - branch(lub(a, b), b)\, |$$

In Layout 4, the sizes of the displayed boxes have been calculated as $s_v^{d_w} * s_h^{d_b}$, i.e. the vertical scaling factor (s_v) raised to the power of the walking distance (d_w) multiplied by the horizontal scaling factor (s_h) raised to the branching distance (d_b); all measured from the focus to each of the nodes. This formula makes nodes appear exponentially smaller the further they are away from the focus. A small example might give a better idea of what is being performed:

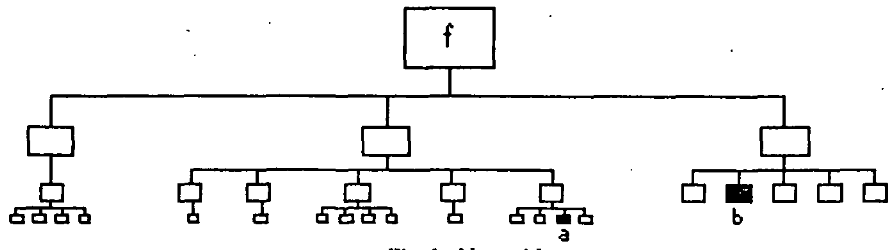


Fig. 1: Normal Layout

Figure 1 shows an "normally" formatted tree, where each node is scaled to half of it's parents size (scaling factors being s_h=0.7, s_v=0.5; the *f* in the figure markes the focus; nodes *a* and *b* are for future reference). The view was actually produced using Layout 4 with the FOCUS property set to 0 (the root). Now we continue to something more spectacular in figure 2.

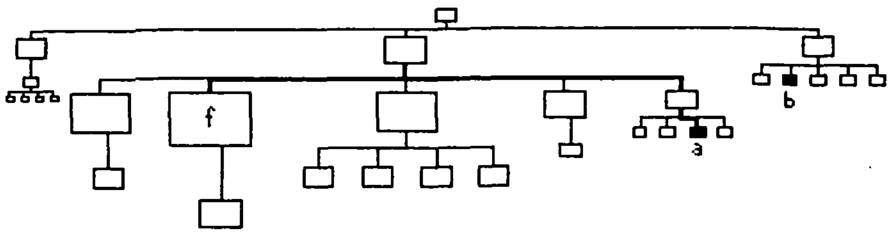


Fig. 2: Fisheye Layout (Subtree in Focus)

Here, the point of focus have been set to (1 1), but with no change in scaling factors. [Remember: Branch numbers start with zero.] The walking distance between the focus and node *a* (path being boldfaced) is 0 (*f* lies on the level under their *lub*)+1 (*a* is one level further down), while the branching distance is 3 (*a*'s branch is three branches away from *f*'s branch). This makes the ratio between the size of the node in focus and the size of node *a* be $0.5^{0+1} * 0.7^3 = 0.1715$. Node *b*'s size is similarily calculated as $0.5^{1+1} * 0.7^1 = 0.175$, making it appear about same the size of node *a*. Using Layout 4, one may also select a complete lvel of the tree as focus by setting the FFOCUS property to a single number instead of a list (mathematically by keeping the *branching distance* constant at 0). Previously this was done to produce the "normal" kind of view in figure 1; now we set it to a couple of levels further down the tree, giving a resulting view that in like figure 3.

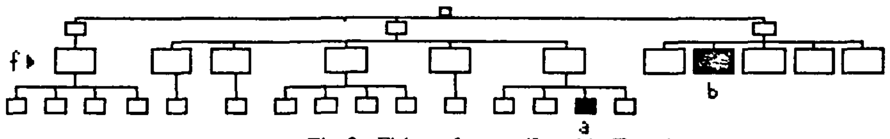


Fig. 3: Fisheye Layout (Level in Focus)

Here, the focus has been set to level 2, while still keeping the scaling factors to their original values. Usually, this method does not give a very good display unless it is applied to a level with a modest amount of nodes.

It is also possible to change the view by adjusting the horizontal and vertical scaling factors. If you, for example, would like to <<emphasize>> the path from the root to the focus while almost ignoring the surrounding nodes, setting sh to a high value (close to one) and sv to a low (round zero) will <<accomplish>> something like this. Following are a couple of different samples produced using Layout 4 with the differing parameters attached to each figure. They are all produced using the ORG.FAKE.A.CHART function of ORG-CHART with the parameters *max node* count set to 80, *average branching factor* set to 3, and *substructure probability* set to 0.8).

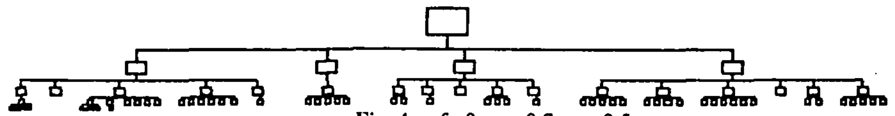


Fig. 4a: $f=0$; $s_h=0.7$; $s_v=0.5$

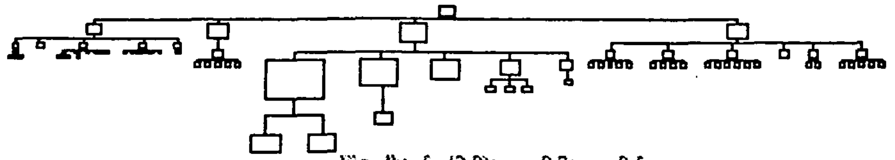


Fig. 4b: $f=(2\ 0)$; $s_h=0.7$; $s_v=0.5$

C - 2

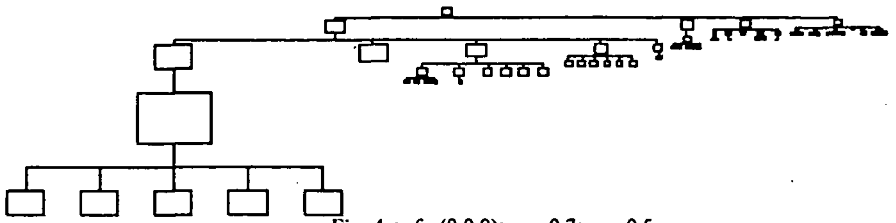


Fig. 4c: $f=(0\ 0\ 0)$; $s_h=0.7$; $s_v=0.5$

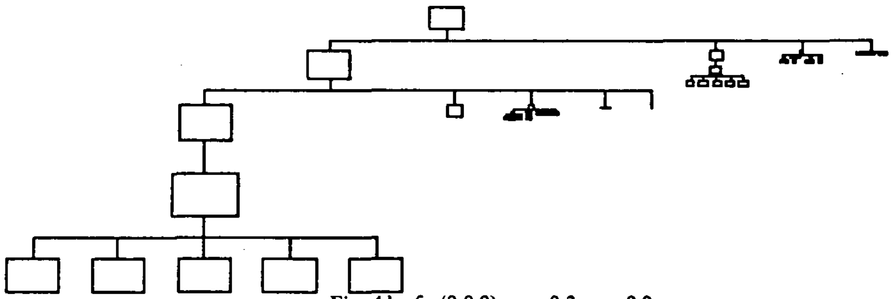


Fig. 4d: $f=(0\ 0\ 0)$; $s_h=0.3$; $s_v=0.8$

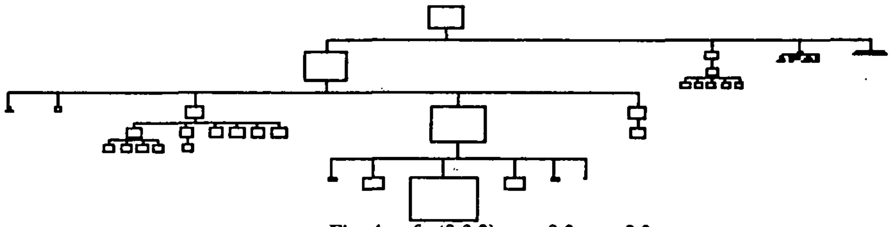


Fig. 4e: $f=(0\ 3\ 2)$; $s_h=0.3$; $s_v=0.8$

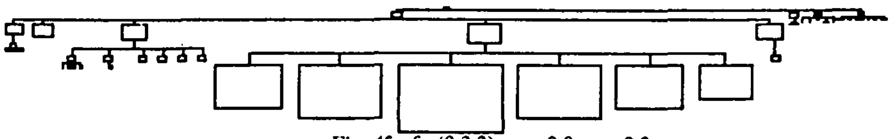


Fig. 4f: $f=(0\ 3\ 2)$; $s_h=0.8$; $s_v=0.3$

The Properties

Layout 4, as well as the other layouts of ORG-CHART, is driven by network properties, put on the root. The most interesting ones are the following:

Name	Dfltval	Description
PCT.BORDER.SPACE	0.1	same as Layout 3
HORIZONTAL.SCALING.FACTOR 0.5	see above	
VERTICAL.SCALING.FACTOR	0.7	see above
BOX.HEIGHT/BOX.WIDTH.RATIO 0.67	same as Layout 3	
SIBLING.SPACE/BOX.WIDTH.RATIO	0.5	same as Layout 3
FAMILY.SPACE/BOX.WIDTH.RATIO	1.0	same as Layout 3
FOCUS	0	the node (LISTP) or level (SMALLP) of
focus		

<<motivationer, applikationer?>>
<<mer volym>>
<<perhaps including multiple focuses>>