SCRIPTING GRAPHICS WITH GRAPHICS: Icons as a Visual Editing Tool

by Dorothy J. Shamonsky
B.F.A., Rhode Island School of Design
Providence, Rhode Island
February 1977

Submitted to the Department of Architecture in partial fulfillment of the requirements of the degree of Master of Science in Visual Studies at the Massachusetts Institute of Technology

June 1984

Signature of the author [illegible]
Dorothy J. Shamonsky, Department of Architecture, May 11, 1984

Certified by [illegible]
Muriel Cooper, Associate Professor of Visual Studies, Thesis Supervisor

Accepted by
Nicholas Negroponte, Chair, Departmental Committee for Graduate Students

SCRIPTING GRAPHICS WITH GRAPHICS:
Icons as a Visual Editing Tool

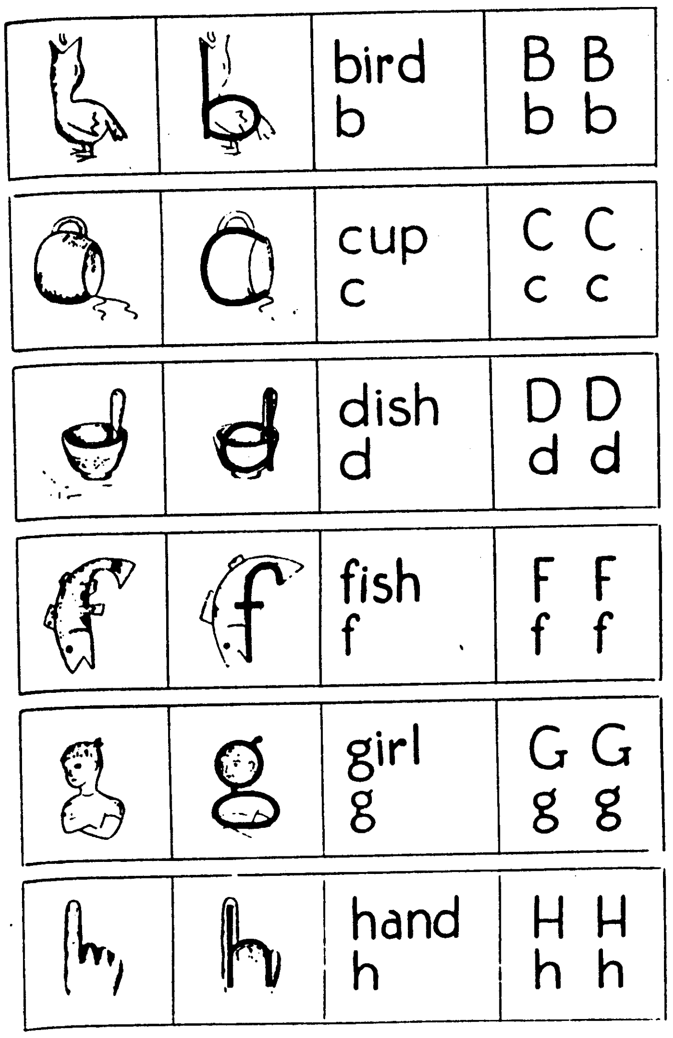
Submitted to the Department of Architecture on
May 11, 1984 in partial fulfillment of the
requirements of the degree of
Master of Science in Visual Studies at the
Massachusetts Institute of Technology

A B S T R A C T

This thesis describes a system for scripting and editing graphic procedures with graphic representations or "icons." The icons are small bitmap images that carry with them information about spatial placement, list placement, and pointers to procedures. Icons have been created and connected to a series of image transformation procedures. Two interactive graphic software packages have been written, one for producing icons and one for scripting with icons. Icons can be created in two ways: (1) making bitmaps from frame buffer images and (2) drawing by grid filling. Scripts can be created in two ways: (1) by storyboarding icons or (2) by playing out a sequence of graphics and recording the list of events. Scripts can be edited, saved, re-edited, and played upon request. The software takes advantage of the ease with which computers can produce simplified, symmetrical, 1-bit images and, by its structure, sets examples of iconic interaction. All interaction is done with puck, tablet, menus, and visual cues. Some prewriting or picture-writing forms are described. A historical overview of computer icons is presented, using several key systems as examples.

Thesis Supervisor: Muriel Cooper
Associate Professor of Visual Studies

"Some centuries later,
in a completely different
part of the world...,
the Aztecs compiled
extraordinary lists
of goods, of offers
and gifts; they drew
exactly what they were,
'6 jaguar skin,
12 feather blankets,
13 jugs of beer,
12 stone knives, etc'.
When some rare bureaucrat
taught the natives to
write, they continued,
in secret, to add a
drawing of the object
to the written word."
[Miranda 1983]



From an old school book.

T A B L E   O F   C O N T E N T S

Abstract 2

Facepage 3

Table of Contents 4

Introduction 5

1.Ø: Prewriting Forms
- 1.1 Introduction 15
- 1.2 Picture-Writing Map 16
- 1.3 Effectiveness & Memory 18

2.Ø: Current Computer Applications of Icons
- 2.1 Introduction 2Ø
- 2.2 Pong 23
- 2.3 Spatial Data-Management 25
- 2.4 Xerox Star 27
- 2.5 Apple Lisa 31
- 2.6 Computer Aided Instruction 33
- 2.7 Visual Perception & Structure 34

3.Ø: Overview of Scripting System
- 3.1 Introduction 38
- 3.2 Icon-Making 38
- 3.3 Icon-Scripting 45

Conclusion 48

Appendix I
- Computing Environment 5Ø

Appendix II - User Scenario
- A. Creating a Script 52
- B. Making Icons 55

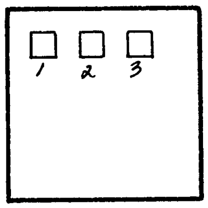
Appendix III
- Program Descriptions 57

Bibliography 66

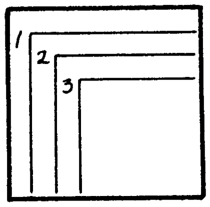
I N T R O D U C T I O N

Suppose you could write a paper, then immediately get an outline of what you have written. You could choose to edit either the paper or the outline. Any change in one would effect the appropriate change in the other. Scripting graphics with icons is similar to this situation. It is a way of visualizing both the actual sequence and the structure of the sequence. Alterations can be made to the structure and the resulting sequence can be viewed, or visa-versa.

Text-editing has acheived a high level of fluidity while the planning of graphical events on computers is done primarily with verbal rather than visual descriptions. Storyboarding is an example of a noncomputer graphical editing device. It consists of an rearrangable series of key images representing scenes. Visual scripts, or storyboards, are used in movies, video tapes, or animation to define the sequence of images. It is a method of planning a film or animation without



structure

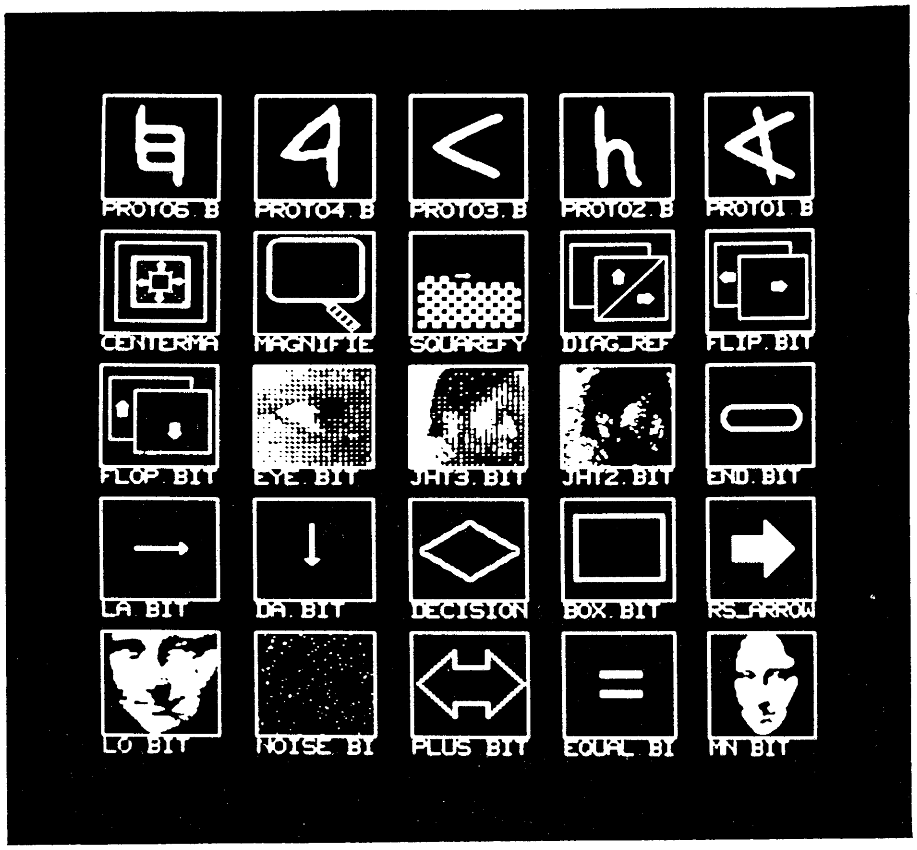


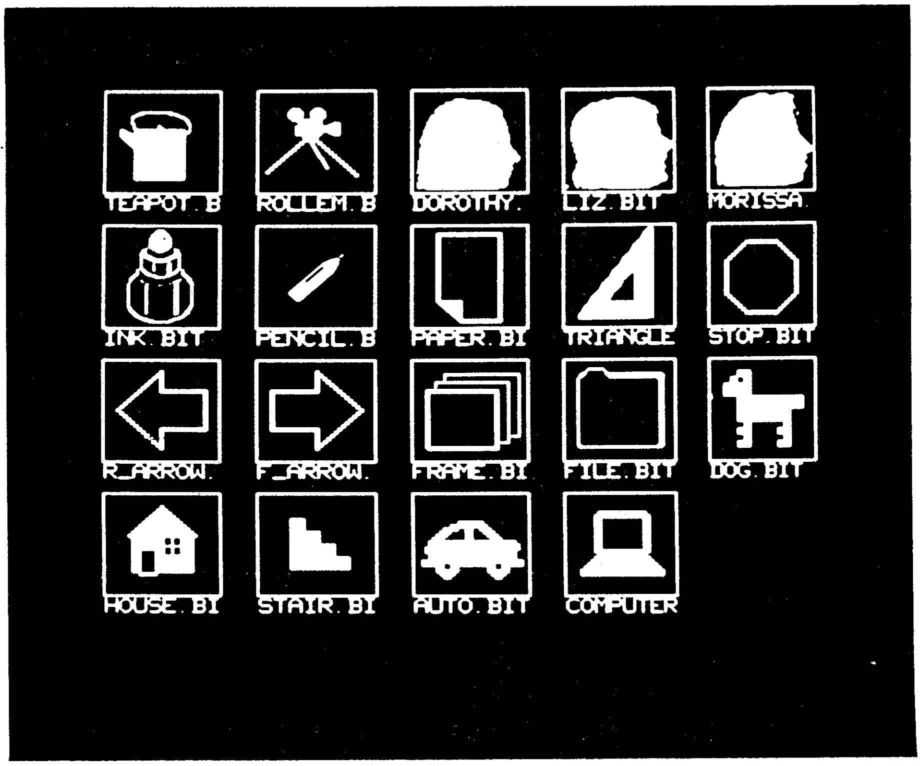
sequence

actually moving cumbersome footage of film or video repeatedly from place to place.

This thesis describes a project that is conceptually similar to a visual editing tool such as a storyboard. Icons are used as a means of defining and moving graphic procedures around an interface space when those procedures can not otherwise be "handled" except as program names. The icons are small, bitmap images that represent graphic procedures and also carry with them vital information about themselves, such as their number in a sequence, and position on the screen.

The software system allows the user to create a script of graphical events using icon representations of those events. Traditionally, scripts are lists of information or, more often, lists of directions: do this, then do this, then do this, etc. In this case the icons are synoptic stories of events and are a way of visualizing the structure of a sequence by creating a visual script. A puck and tablet is the primary method of interaction, and a menu is used to access scripting procedures. Icons can be created, saved, and arranged into a script.

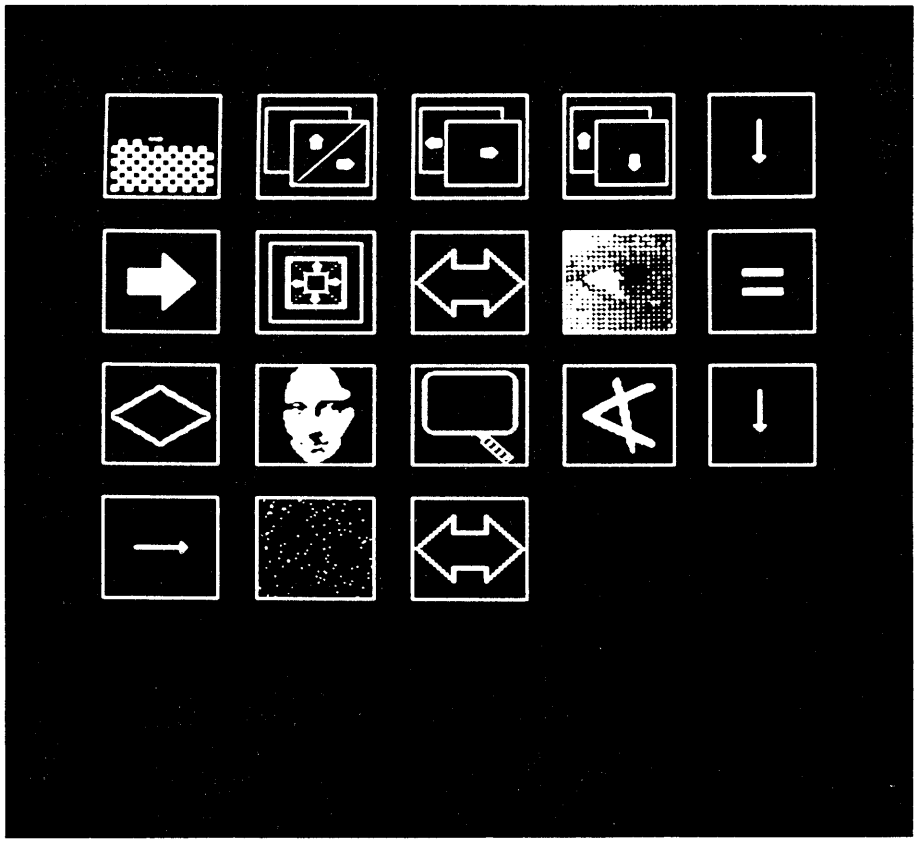




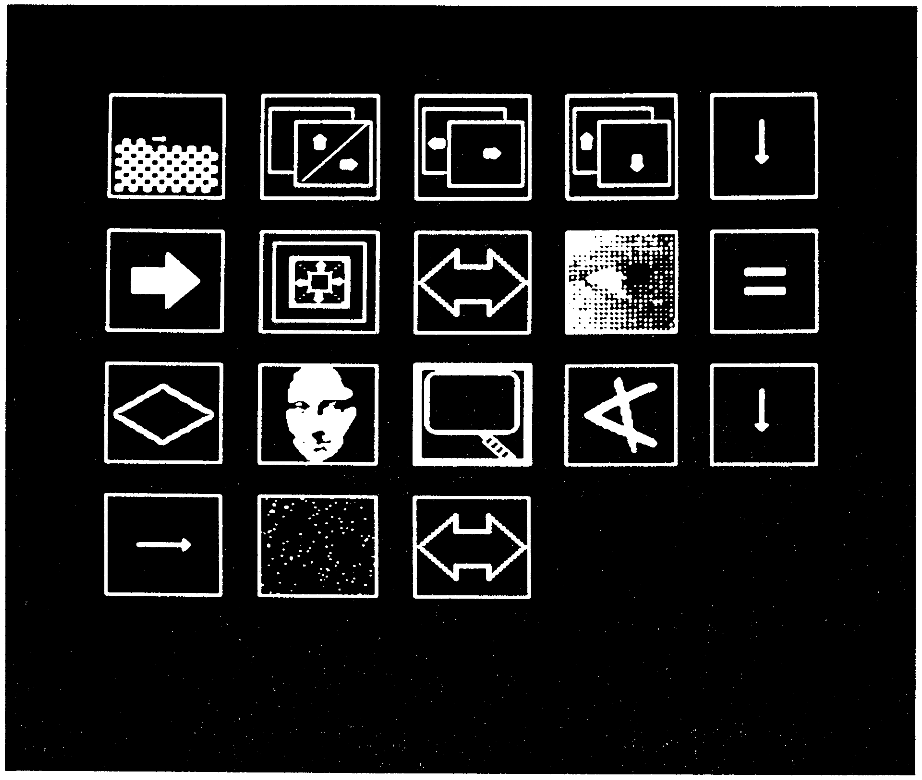
The contents of the directory of saved icons.

The script can be edited, saved, and re-edited. The editing capabilities consist of inserting, deleting, and rearranging the sequence. While "playing" the script is displayed and visual indications are made of the place in the script.

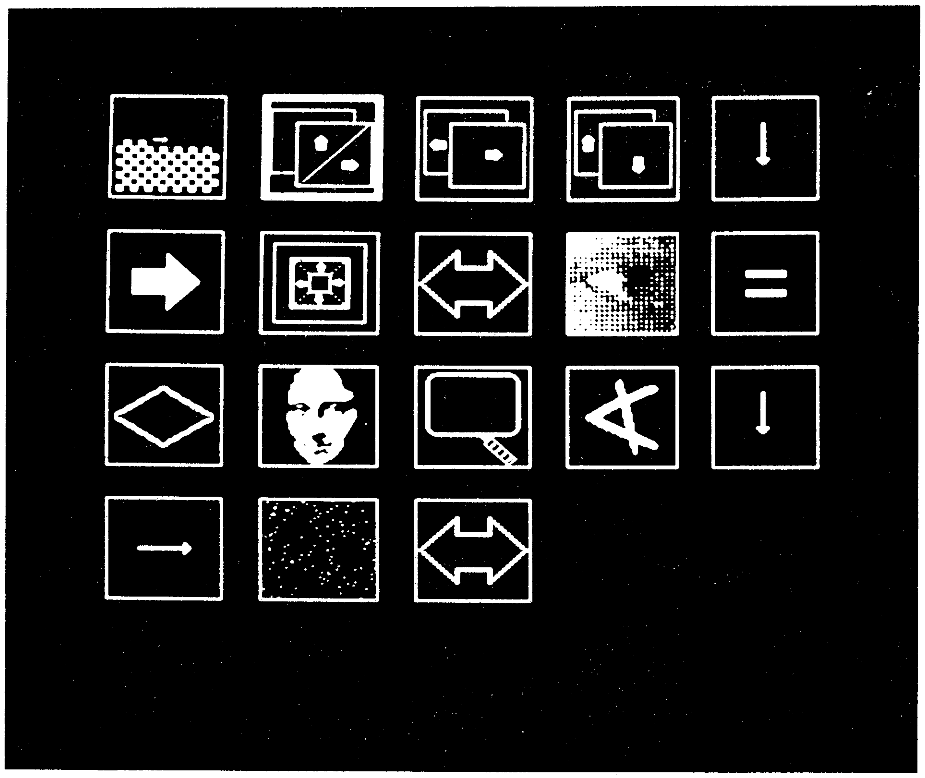
The storyboard capability:



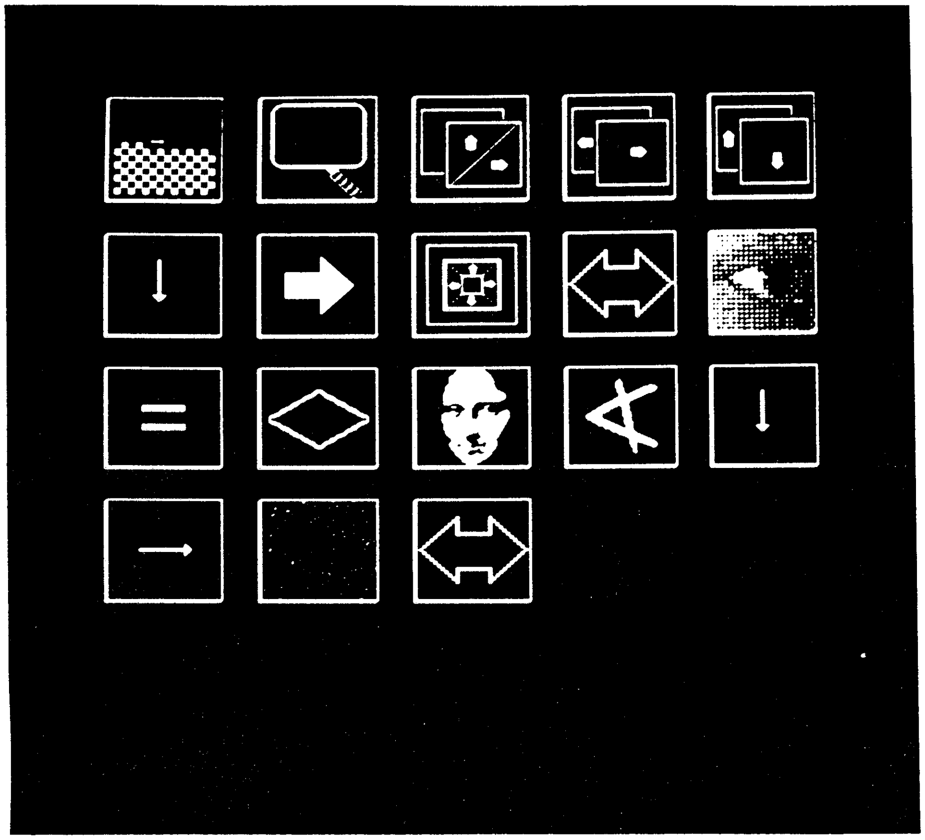
The sequence to be rearranged.



Place this one (the highlighted box)...



in front of this one (the highlighted box).



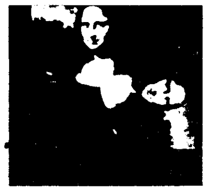
The new sequence redisplayed.

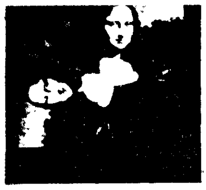
The ability to script graphics with icons is useful in storyboarding digital effects for television, film, and video. It is the natural computer counterpart to traditional storyboarding. Similarly, scripting with icons would be useful in planning computer animations.

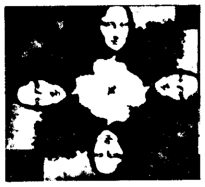
Text-editing capabilities are used to plan the text of electronic books but the illustrations, frequently being little animation sequences themselves, are still almost completely dependent and non-computer planning techniques. The process of planning an electronic document or book with many illustrations, some animation or video, and needing consistency among themselves, would benefit from a planning tool such as scripting with icons.

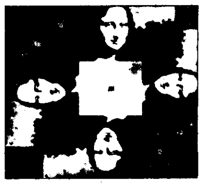
Defining and structuring graphic events on computers with descriptive words, prevents us

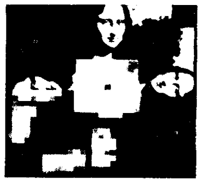
Original image


1. diagonal reflect

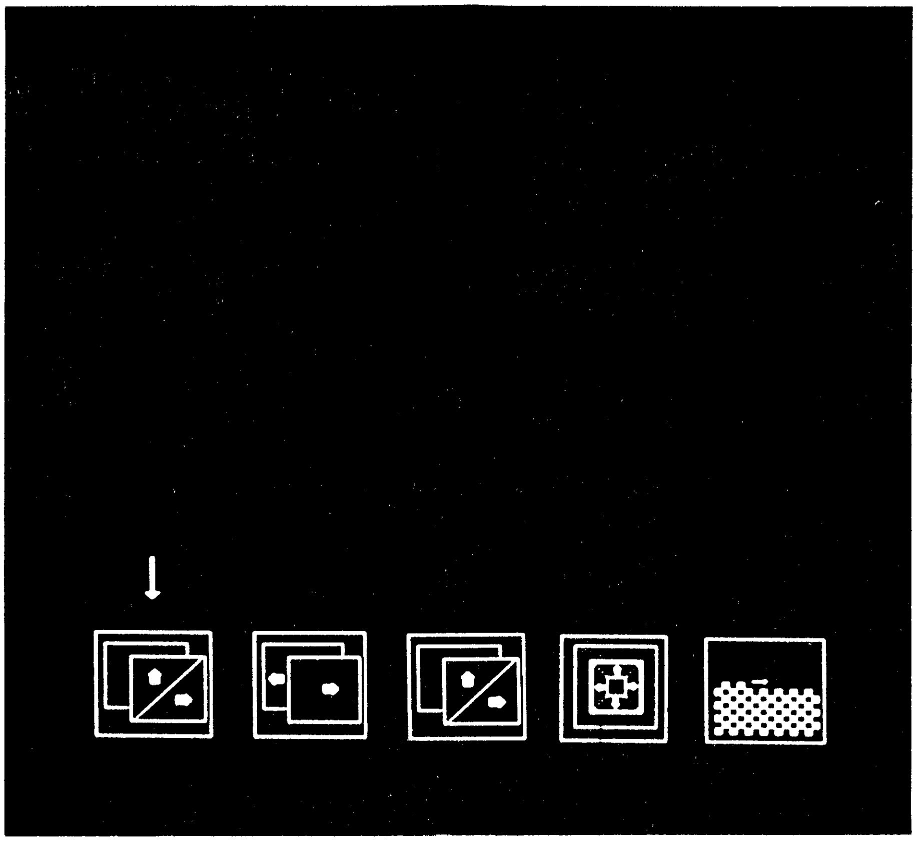

2. flop


3. diagonal reflect


4. magnify center


5. squarify

A final script being played.


1. 2. 3. 4. 5.

from using the organizational and spatial skills we usually employ in solving visual problems. There are subtle ways that artists and designers have learned to organize and define structure in any design problem or art project. The sequence of drawings in a sketchpad or a scribbled page of thumbnail sketches are often the unconscious methods we use to create a visual structure. These methods, often spatial, allow us to put perspective on a task, keep track of its parts, and visualize the final result.

Computers have eliminated many of these common structural aids we depend upon because of the one-dimensional expression of information in most interfaces. In the past, hardware limitations presented real restrictions to the use of graphics in interface design Now graphic designers and artists are being called upon to apply their expertise to interface design, incorporating traditional method of information display and inventing new ones.

"Icon" has become the accepted "computereze" term to describe graphical representations on a computer of a real person, object, or function. The term "icon," by definition, is an image, portrait, or semblance, especially of a saint. It comes from the Greek word which means"to be like" or "likeness". The implication is that of representing life, especially human. [Klein 1967]

Computer icons often act as representations. A representation simply stands for something. Used on commercial systems, icons frequently represent persons, things, functions, and modes. In the same way that a logo represents a company, there is a "user" icon on the Xerox Star, for example, that represents that user. The cursor on the Apple Lisa represents what mode of the computer you are in (an arrow for selecting from a menu, a paint pail for painting, etc.).

The general distinction between an icon and any other image, large or small, on a computer is that an icon carries with it information about what it represents. Icons are live pictures. You can "pick" an icon and something happens. You can "open" an icon and get to its contents. You can combine two icons and they modify each other.

The root of "icon" is also related to the

Lithuanian word meaning "to occur" or "to come true" [Klein 1967]. This makes computer icons ironically appropriate to represent functions. Function representation, although going back centuries, is most familiar to us in the form of the push button and the on/off switch. Icons that represent functions are akin to push buttons, but on a system like Lisa, the function icons are taking us farther away from the cryptic push-button mode and closer to the real event. To throw something away on the Lisa the user "picks up" the object/icon, "drags" it to the trash can/icon and "lets go" of it. The object then disappears.

1.Ø: PREWRITING FORMS

1.1 Introduction

Telling stories or recording events with pictures is the most ancient form of writing. Our alphabet is just a sequence of highly reduced pictures, or signs, whose designations we have memorized. The history of the alphabet is one of simplification of the its form to accomodate increased complication of ideas. There are infinite numbers of possible pictures compared to the twenty-six characters of our alphabet. Some of the earliest alphabets, direct decendants from picture-writing forms, had as many as 5ØØ characters.

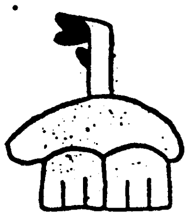
Cave painting from Altamira, Spain.

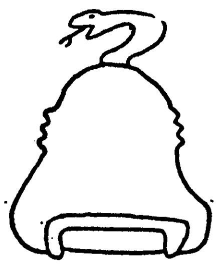
Chronical of events used by Dakota Indians.

A fully developed system of writing or sign/ speech communication has evolved in the last five thousand years. By comparison to the time spent in the evolution of humans as tool-makers, writing is a recent development. Drawing and painting was practiced for thousands of years by the races of the Magdalenian culture before writing emerged. As sophisticted as those images are, they lack any clear sequence or storyline. Specific information communicated represents the first stage of writing and is lacking here. [Diringer 1968]

There are severe limitations on the use of pictures for communication. A person who has never previously seen an image or had it explained to him/her may find it hard to interpret. And apart from the inherent difficulty or impossibility of illustrating many concepts at all, it is possible to forget what a picture means or symbolizes.

a


b


c

Aztec phonograms of town names.

1.2 Picture-writing Map

A simple scheme of the different types of writing:

1. Picture-writing: a synoptic view of a scene or event.
2. Pictograms: separation of the different items in the picture and portrayal by a number of distinct signs, each standing directly for the object shown.
3. Ideograms: as in (2) except here the connection between the object shown in the sign and the idea portrayed is indirect working by suggestion.

In all stages (1) to (3) the writing is independent of the spoken language.
4. Phonograms: the sign stands for a sound, and so is bound up with the language. Signs are of three sorts, being the equivalent (1) of an indefinite number of sounds and especially of the whole of the word (mono or polysyllabic); (2) of only one sound, the syllable; (3) in the alphabet, of either a vowel sound, or of a consonant that is not properly a sound by itself. [Moorhouse 1953]

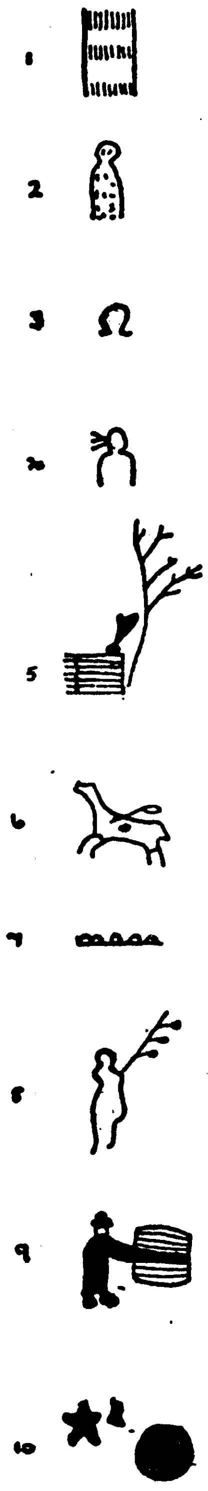
Between stage (1) and the rest there is a big gap since it is only in retrospect that we can see that picture-writing is a system of writing at all. The transition between the other stages was gradual when it was accomplished by a single people making the transition on their own. The Egyptians are a good example, and in such a case the relics of a more primitive stage are apt to remain long after they are necessary. On the other hand, we find a sharply marked transition where there is borrowing from the writing system of another people. The borrowers tended to take what they found valuable and discard the rest, being unaffected by the demands of tradition.

Detail from the Book of the Dead of Tuthmosis III, New Kingdom, about 1450 B.C.

1.3 Effectiveness & Memory

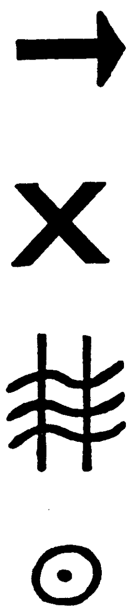
Where picture-writing has been the only means of communication it was used primarily for important tasks such as recording laws and treaties. If the meaning of this record is forgotten it is lost forever. The French explorer, Cadillac, reported that an Indian chief told him that his tribe had an illustrated collar which they had received from the Iroquois, but that "the old men had forgotten what it said". A problem of this sort was normally prevented by performing certain rituals. The "records" were kept together in a safe place under the charge of a "keeper" who had a good memory. On a regular basis the tribe would meet, pass around the belts or other object from hand to hand, and repeat in unison the official version of the events that were recorded. [Moorhouse 1953]

It seems, then that the efficacy of picture-writing as a self-contained and unassisted means of communication depended on the maintenance of an unbroken tradition and on the existence of good memories. The memory of illiterate people is often more highly developed. The advent of writing proper caused



The calendar of Lone Dog, Dakota Indian.

a relaxation in the cultivation of memory, which was at first regarded as a regrettable loss. In New Zealand the Maoris opposed the introduction of writing because they feared the effect on the loss of memory. Caesar (De Bello Gallico, Book vi.14) gives this as one reason why the Druids would not put down their religious lore in writing. The question is also considered in Plato's Phaedrus (para. 275), where he suggested the the use of writing makes it easier to recall facts to mind when necessary, but destroys the true and·intimate knowlege that comes from perfect memorization.



Boy Scout trail signs: this way, not this way, over water, gone home.

2.0: Current Computer Applications of Icons

2.1 Introduction

It is impossible to perceive with our senses how a computer functions as one can with a car or a toaster. The only way to understand a computer is by descriptions, models, and representations. Until recently the only means of communication at the computer interface were words; the vocabulary required was highly specialized. The kinds of knowledge necessary were not like any previous experiences; the time required to become minimally competent was not worth the investment to a layperson. Thus, computers have remained mysterious and intimidating to much of the population.

Ideas for more accessible interfaces originated a number of years ago. In the early sixties Ivan Sutherland wrote a doctoral thesis on his graphical interface called Sketch-pad in which he states:

"Heretofore, most interaction between men (sic) and computers has been slowed down by the need to reduce all communication to written statements that can be typed; in the past, we have been writing letters to, rather than conferring with, our computers." [Sutherland 1963]

Using a lightpen to input data, Sutherland created an interface to produce graphics that worked primarily by pointing. Although not quite like drawing with pencil on paper, it used embedded visual knowledge and was more direct than verbal descriptions of graphics.

Most interactions with computers at that time were with keyboard input to a CRT (cathode ray tube) screen or printer. Advances in interface hardware have made more direct manipulation possible. These include not only the light-pen, but the tablet and puck, joystick, touch-sensitive screen, and mouse. Douglas Englebart of S.R.I. first experimented with a mouse in the early seventies, although it is only within the last couple of years that it is incorporated in systems as often as the keyboard.

Image-making capability has also improved. With more powerful computers, high resolution and many colors are the norm rather than the exception. It is not surprising that images on computers are as important and popular as they are in the rest of life. What is surprising is how crucial they are becoming to creating believable illusions at the interface. Graphics are becoming the primary tool for creating clear conceptual spaces at a computing interface, and "virtual" objects and functions are being described by visible movable images.

The use of graphical representation on computers is a natural evolution of the need to understand certain opaque information. The graphics are often iconic rather than literal because they need to be only models of reality. Visually simple, they often carry with them large amounts of information about their own behavior in order to act true to their representative reality. Icons are like a black box abstraction and as long as we, the user, can read the label we can work with the icon as we would its reality.

Recent studies have shown that certain interface features evoke a positive response in users. Those interfaces are often ones that incorporate graphics to create conceptual worlds. Features that appear most significant to this positive reaction are: (1) visibility of object of interest, (2) replacement of complex command language syntax by direct manipulation of the object, (3) actions caused by the manipulation occur within an understandable space, and (4) reversiblility. [Shneiderman 1983]

2.2 Pong

The earliest systems that embodied these attributes were video games. Pong came on the market in the mid-seventies. It requires about 3Ø seconds of observation to become a competent novice. Its concept is simple: two rectangles acting as paddles are manipulable with two knobs, and a white spot acting as the ping pong ball ricochets off the boundaries of the play area. The rules are similar to those in the game of ping pong and the basic principles of play can be learned in a few minutes by watching the on-line automatic demonstration. Competency of a novice can be acquired within a few minutes but reaching advanced skill levels can take hours of play. Thus, although the rules are simple, there is a challenge to the alternative skill levels.

What is satisfying about the game was the directness of control; the "paddle" responds instantaneously to any movement of the knob. With the simplest graphical elements, lines and rectangles, a convincing metaphorical space is created with metaphorical objects that behave in predictable ways. The field of activity in most video games, and objects within that field, are a viable abstraction of reality. The actions or skills necessary to play are already part of our vocabulary. The commands are physical actions: button pushes, joystick motions or knob rotations, whose results appear immediately on the display.

Several prominent people within the field have defined what is appealing about these kinds of systems. "What you see is what you get" or "WYSIWYG" is a phrase used by Don Hatfield of IBM to describe direct manipulation. Alan Kay, first at Xerox Parc and now at Atari, claims direct manipulation works because "language is just an extension of pointing". Ted Nelson, author of Computer Lib talks about the principal of "virtuality," a believable representation of reality whose space and objects act in believable ways [Nelson 1980]. Chris Rutkowski, on the subject of the computer/human interface, explains the concept of "transparency" of tool so that the user is able to apply his intellect directly to the task [Rutkowski 1982].

2.3 Spatial Data-Management

Between 1976 to 1978 the Architecture Machine Group at M.I.T. developed a system of organizing and referencing data called Spatial Data-Management (SDMS). It is designed on the concept of reference by place rather than by name. It depends upon the creation of a plausible "virtual" space at the computer interface, together with a way of getting around in that space quickly and easily.

"It is surprising how pervasive the underlying notion of spatiality is, even in the symbolic modes of thought. Consider a file of three-by-five notecards, organized alphabetically, but without letter tabs showing. "Filed under the letter R" translates into a tactile estimate of how far down the row of cards we must reach...Thus, "R" is somehow a relative distance, as well as a letter. There is more spatiality implicit in what is ordinarily thought of as symbolic retrieval than we may realize." [Bolt 1979]

The spatial world of SDMS consists of a single plane called "Dataland"; a metaphorical desktop. On that plane/desk, in fixed positions, rest objects: letters, telephone list, reports, etc.

"The person who uses this desk has organized the layout of items in a more or less systematic way. He or she refers to them constantly throughout the working day:

reaching in that direction, that far, up, down, to the right, to the left. Through this activity, a mental image of the layout of the desk is elaborated in the "minds eye." Additionally, through constant tactile interaction with the items, reaching for and touching them, a "motor memory" of where things are arises as well. A script for the act of retrieval becomes encoded into the musculature, as it were, according to where the item is located." [Bolt 1979]

It is presented to the user in two ways: in its entirety in an "aerial," top-on view displayed on one monitor, called the "world view" monitor, and simultaneously, a small subsector of the surface is displayed on a ten-foot screen, vastly enlarged and with considerable gain in detail. The relationship between these two views is like a mapping key which places a particular image within a larger picture. The "world view" monitor serves as a navigational aid to the user in getting around Dataland. The large screen functions as a "window" and as a "magnifying glass" into Dataland. The large display of any portion of Dataland is so "close up" that the user would get lost easily if there were not always a map on view of the entire Dataland world. A small, highlighted, "you-are-here" rectangle on that display shows the user at all times the position of the large-screen window on Dataland.

Movement through Dataland is with a joystick and touch-sensitive display; the sensation is of flying and hovering over a landscape at different heights. You can move your position but the landscape and objects within the space are fixed.

The space is illusory while most of the objects were photographic images (although a few graphic representations were used, eg. the images of a calculator and slide rack). The objects are displayed in more or less detail depending on your position as the user. The effect was a kind of automatic iconization of the objects.

2.4 Xerox Star

After many years of research and an earlier release of the experimental Altos computer, Xerox released the 8010 Star Information System in the spring of 1981. It was a new personal computer designed for offices, and was intended for business professionals who handle information. The hardware was modeled after the experimental Xerox Alto computer and consisted of a processor, a bit-mapped display screen having a 72-dots-per-inch resolution, a keyboard, and a mouse pointing device.

Xerox devoted about thirty work-years to the design of the Star user interface. The paramount concern was to define a conceptual model of how the user would relate to the system.

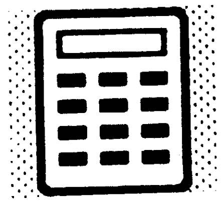
"A user's conceptual model is the set of concepts a person gradually acquires to explain the behavior of a system, whether it be a computer system, a physical system, or a hypothetical system. It is the model developed in the mind of the user that enables that person to understand and interact with the system...The Star designers devoted several work-years at the outset ot the project discussing and evolving what we considered an appropriate model for an office information system: the metaphor of a physical office." [Smith, Irby, Kimball and Harslem 1982]

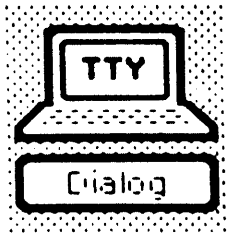
Xerox took the desktop metaphor a step farther and created a metaphorical physical office space, with a network of terminals and communications capabilities between users.

"The designer of a computer system can choose to pursue familiar analogies and metaphors or to introduce entirely new functions requiring new approaches. Each option has advantages and disadvantages. We decided to create electronic counterparts to the physical objects in an office: paper, folders, file cabinets, mail boxes, and so on - an electronic metaphor for the office. We hoped this would make the electronic "world" seem more familiar.
...A physical metaphor can simplify and clarify a system. In addition to eliminating the artificial distinctions of traditional computers, it can eliminate commands by taking advantage of more general concepts."
[Smith, et.el. 1982]

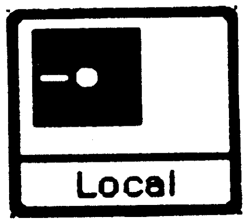
Instead of moving through a fixed landscape, the user can manipulate the graphic representations of objects, or "icons" as Xerox called them, with a mouse device. Icons could be moved through the metaphorical space and their contents accessed. Unlike SDMS, all information was not visible by zooming in closer. Instead the icons were true to their definition in that they were a minimum representation of the underlying story or data.

Xerox icons

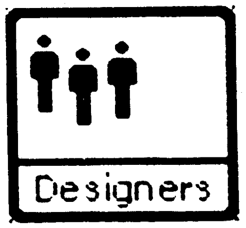

calculator



TTY emulator



floppy disc



user group

Every user's initial view of Star is the "Desktop," which resembles the top of an office desk, with surrounding furniture and equipment. It represents your working environment where your current projects and accessible resources reside. On the screen are displayed pictures of familiar office objects: documents, folders, file drawers, in-baskets, and out-baskets. You can "open" an icon to use or work with what it represents. This enables you to read documents, inspect the contents of folders and file drawers, see what mail you have received, etc. When opened, an icon expands into a larger form called a "window," which displays the icon's contents.

Icons are classified as data icons and function icons. Data icons represent objects upon which actions are performed. Currently the three types of data icons are documents, folders, and record files. Function icons are of many types with more being added as the system evolves: file drawers, in- and out-baskets, printers, floppy-disk drives, calculators, etc. In general, anything that can be done to one data icon can be done to all.

The Desktop is the principal Star technique for realizing the physical office metaphor. The icons on it are visible, concrete embodiments of the corresponding physical object. Star users are encouraged to think of the objects on the Desktop in physical terms.

"Star is the first computer system designed for a mass market to employ icons methodically in its user interface. We do not claim that Star exploits visual communication to the ultimate extent; we do claim that Star's use of imagery is a significant improvement over traditional human-machine interfaces." [Smith, et.el. 1982]

The model of a physical office space provides a simple base from which learning can proceed in an incremental fashion. You are not exposed to entirely new concepts all at once. Much of your existing knowledge is embedded in the base. "A well designed system makes everything relevant to a task visible on the screen." [Smith, et.el. 1982]

2.5 Apple Lisa

Apple released Lisa on the market at the end of 1982. It is a personal computer that embodies many ideas used in the Star but emphasizes the individual user. More graphics are used, both for operations and available as tools to the user. The graphic style is more popular, less clean or slick, "friendlier", and direct manipulation techniques as physical actions are emphasized. Greg Williams, senior editor of BYTE Magazine, in describing his encounter with a Lisa, gives an insight into its appeal:

"With a few movements of the mouse... I "tear off" a sheet of Lisa Graph "paper" (thus activating a program called Lisa Calc and displaying an empty grid on the screen) and give it the heading "Annual Sales".... At this point, I can simply print the graph or save it for inclusion with my report, but I'm not satisfied with the way it looks. I then use the mouse to "cut" the graph from the Lisa Graph paper and put it in a temporary storage place called the clipboard. I can then "throw away" the Lisa Graph "paper" I was using. My next step is to "tear off" a sheet of Lisa Calc "paper" and paste my "Annual Sales" bar chart from the clipboard onto it." [Williams 1983]

Metaphors of tangible objects being physically moved with actions described as "cutting," "pulling," and "dragging." The cursor performs psuedo actions upon tangible objects. Picking with the mouse button has been expanded to more sensual and demanding tasks. These are all, of course, illusions to expand the real world model.

When you turn on the Lisa system, the screen is empty except for the presence of several icons. The Lisa also depends on the metaphor that the video display is a desktop, while the icons are objects on the desktop. Each peripheral connected to the Lisa: floppy and hard disks, printers, and other peripherals connected by the interface cards, is represented on the desktop by either an icon (if it is not in use) or a rectangular area called a window (if it is available for use). Each file is represented by an object of some sort: a report, a tool, or a document. Objects can be grouped together in folders, which are also treated as objects.

"The desktop metaphor does two things for you. It helps you to remember certain operations because they make sense in the context of the object-related icons. Second, it draws on your general knowledge of office supplies and how they are used. These elements help Apple achieve its objective of creating a system that people can learn to use some aspect of in under 30 minutes." [Williams 1983]

The Lisa file system makes you feel as if you are actually moving and changing objects, not merely manipulating abstract data. In other words, the "data-as-concrete-object" metaphor demystifies the computer by transforming data into physical objects that behave in a predictable and reasonable way.

2.6 Computer Aided Instruction

In a world of "virtual objects" residing in "virtual spaces" icons have become a tool for creating "virtual worlds." Icons often act as objects and possess most of the characteristics associated with the real object. In this way icons act as models or they are the components to create models.

"The aim of a model is, of course, precisely not to reproduce reality in all its complexity. It is rather to capture in a vivid, often formal, way what is essential to understanding some aspect of its structure of behavior...A model is always a simplification, a kind of idealization of what it is intended to model." [Weizenbaum 1976]

In CAI (Computer Aided Instruction) systems, icons are frequently used to represent component "parts" that can experimentally be put together to create larger systems. A project, called Steamer, was done at Bolt Baranek and Newman Inc., in Cambridge, MA, to teach about U.S. Naval steam propulsion systems. A library of icons was created to represent the component parts. Each icon carries with it the information about what it represents. Therefore it can interpret user input to affect the state of the modeled component. For instance, touching a gauge causes it to produce a value corresponding to the location of the touch on the gauge face. The icons then act as models of parts. In combination, they create a model of a system and display the characteristics of that system. [Stevens, Roberts and Stead 1983]

One of the benefits of "virtuality" is that the user, in a short period of use, develops a clear conceptual image or model of the entire system in his/her mind. It is exactly because systems such as Star and Lisa are designed as models of real situations and events that occur are true to that model, that those systems are easy to learn. This doesn't detract from the usefulness or sophistication of the tool; they are, in fact, more useful.

2.7 Visual Perception

Part of the success of "virtual representations of reality" is that they capitalize on the capabilities of human visual perception.

"Vision is the most common and possibly the most important means by which people access their personal data bases. Clues as to the location of information in the space may take on such visual aspects as position, shape, color or texture."[Bolt 1979]

Recent research has shown that an image can be comprehended within a shorter span of time by the human brain, and it is more enduring. Visual/ spatial information is handled by the right hemisphere of the brain while the left hemisphere comprehends language. The right hemisphere proceses in a parallel form, able to take in large amounts of information simultaneously; the left processes in a serial form, one piece of information at a time. Visual information is stored by the brain in the long-term memory while verbal information is stored in the short-term memory.

Many studies have analyzed the short-term memory and its role in thinking. Two conclusions stand out: (1) conscious thought deals with concepts in the short-term memory, and (2) the capacity of the short-term memory is limited. When everything being used in a computer system is visible, the display screen relieves the load on the short-term memory. Thinking becomes easier and more productive. [Haber and Wilkinson 1982]

"The particular strength of the human visual system is its ability to encode multidimensional information rapidly and enduringly. By taking advantage of this strength, computer scientists can provide one of the most effective means of communicating between a computer and a person." [Haber and Wilkinson 1982]

Icons sometimes act as maps, presenting overviews as opposed to underviews. They use the spatial ability of humans.

"The well-evolved human ability to organize information spatially remains essentially untapped in the realm of computer-based information handling. Typically, in such systems, retrieval on a symbolic of name basis is the norm, and must be the norm because the conventional keyboard interface is too limited a channel, the wrong mode and medium, to begin to offer the user a direct, palpable sense of spatiality." [Bolt 1979]

Visual programming languages and program visualization are two areas of current research where icons are used to illustrate a structure. Brown University has been working on BALSA (Brown Algorithm Simulator and Animator), part of a interactive teaching system. "Tinker" is a program visualizer by Henry Lieberman, research scientist at the Artificial Intelligence Laboratory at M.I.T. J.C.R. Licklider, professor of computer science at M.I.T., and Don Hatfield of IBM are working on visual programming languages.

At times icons act as metaphors. They create an insight/outlook about an object within a context that did not exist or was not obvious before the existence of the icon. Anyone who has worked on a metaphorical desktop will forever after perceive his/her own desktop in a different way. We have been made glaringly aware that the organization of our lives is primarily spatial and visual. We perceive a real distincion between objects and the space they occupy. In a similar way that computers themselves have become metaphors for human and mind, and have altered the way we perceive ourselves.

"Often the heuristic value of a metaphor is not that it expresses a new idea, which it may or may not do, but that it encourages the transfer of insights, derived from one of its contexts, into its other context... We can say in anticipation that the power of a metaphor to yield new insights, depends largely on the richness of the contextual framework it fuses, on their potential resonance. How far that potential will be realized depends, of course, on how profoundly the participants in the creative metaphoric act can command both contexts."[Weizenbaum 1976]

3.Ø: Overview of Scripting Software

3.1 Introduction

My goal was to implement a system that would enable a user to interactively script a series of graphic procedures which could be played upon request. I wanted the scripts to be easy to edit, and direct manipulation be the major mode of interaction.

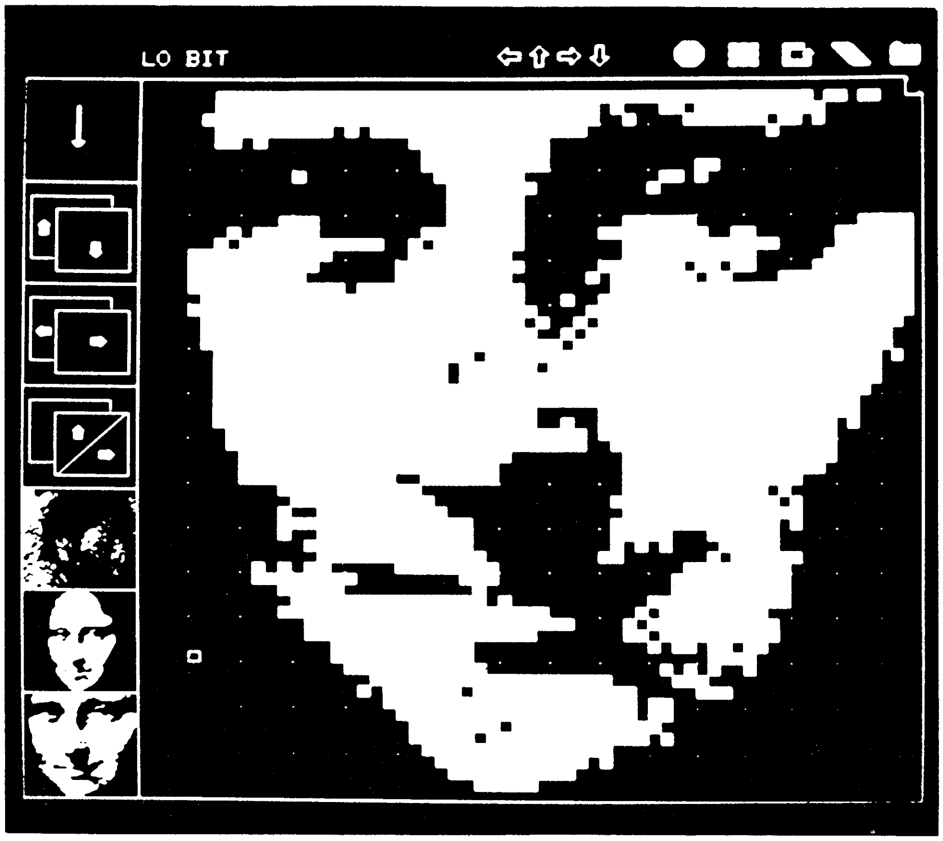
The graphic procedures are programs that currently reside on our computer system. Included are image transforms, digital effects, and stills for video or animation. The procedures are represented by symbols or "icons," which have been designed specifically for that purpose. To implement icon scripting I first needed to create a system to make 1-bit icons.

3.2 Icon-Making

The icons are 1-bit images that are made and moved on the overlay planes unless resolved to red, green, and blue values with 24-bits of information, (rgb). Saving, loading, making bitmaps, scaling, listing, deleting, displaying the contents of the icon directory, and resolving to rgb are the available functions. They are accessed through a menu. [*] Many command level functions that support the icon making process are included in this menu.

The software is organized into three catagories: making, saving, and using. Making and using remain open-ended domains to which an infinite variety of additions can be made. The connecting link between those domains is the storage system consisting of two discrete programs, "icon_save" and "icon_load", and a directory where icons can be saved, >u>icon>**. "icon_save" and "icon_load" initiate segments and call low level routines to read from and write to the overlay planes. Each icon is saved in an individual segment. Subdirectories exist in >u>icon to save catagories of icons. This saves searching through screens of icons to find the ones you want.

Currently, icons can be produced by two methods: making bitmaps from frame buffer images (these have been digitized with a video camera or created with other graphic software) and drawing by grid-filling. Bitmaps are

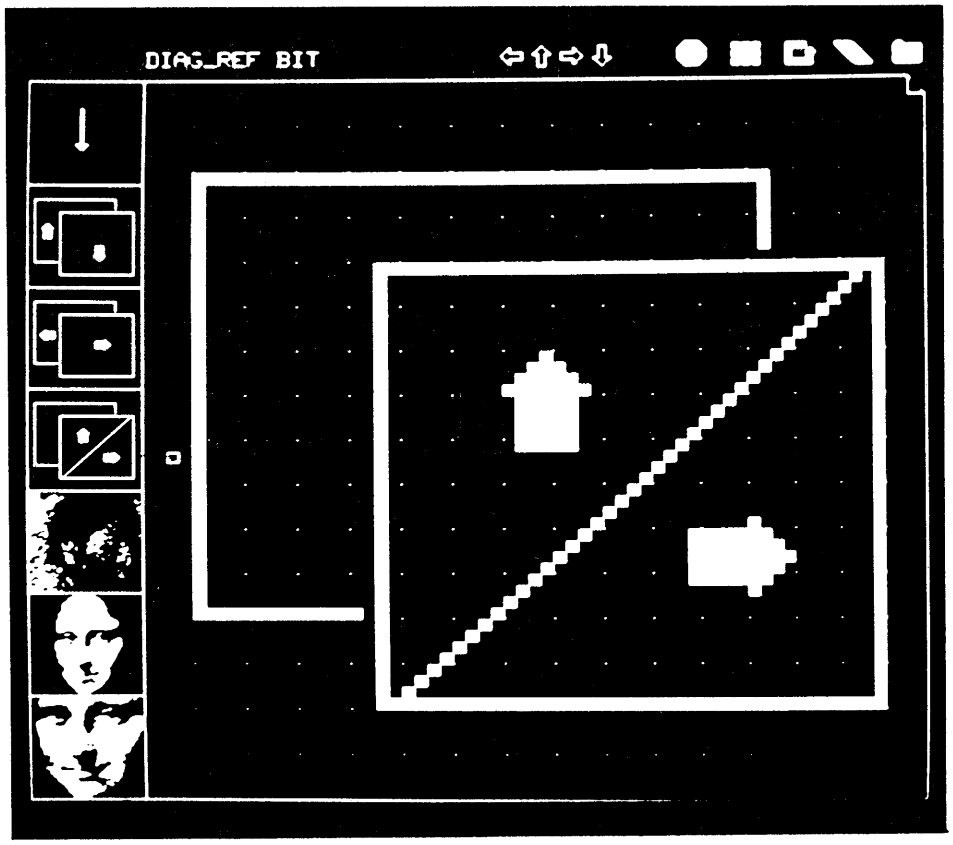


Bitmap.

images containing 1-bit of information for every pixel. In other words, the pixel is either on or off. They are made by defining a range of rgb value and reading every pixel to see if it falls within that range, in which case it is "turned on" or colored. These are full or close to full screen processes. Therefore, scaling procedures are included which allow for both size and proportion changes.

There are no restrictions to use just the icon-making software to create 1-bit icons. Any picture making ability already existing on the computer could be used. Conversely, any image on the computer could be iconized using this software. Applications of icons is equally as open-ended, with software written for scripting with icons (see section: Icon Scripting).

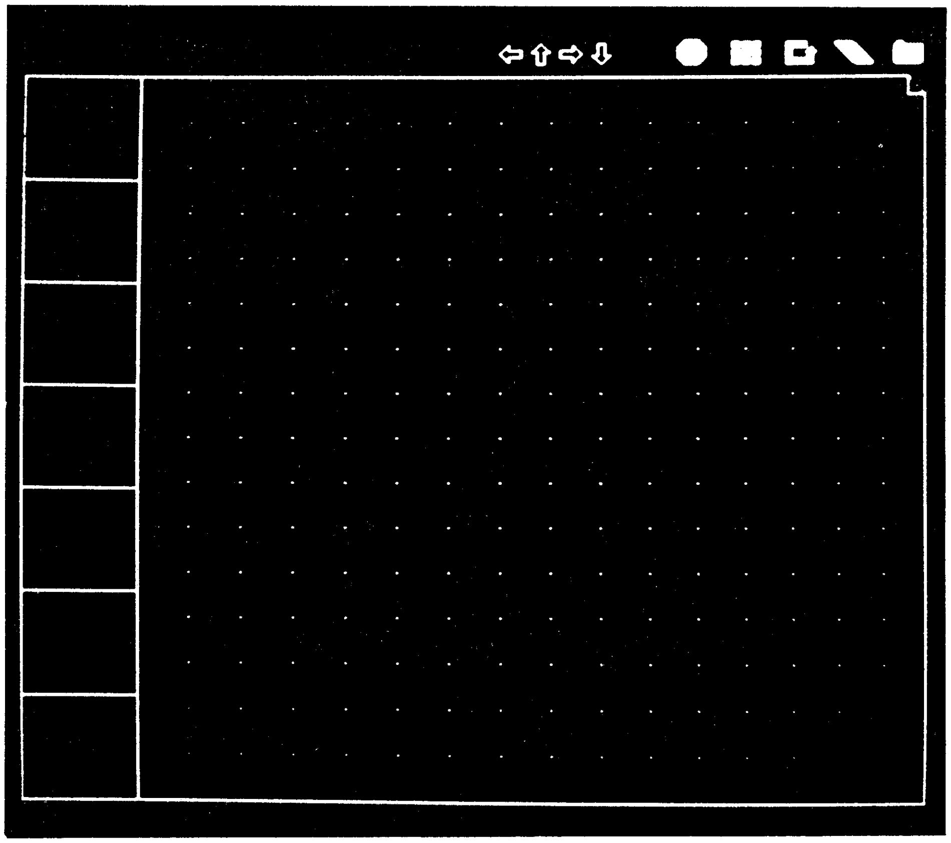
Icon editing is done with a drawing tool which works by filling squares on a grid. Previously made icons can be edited or new images can be produced. All activity occurs on the overlay planes, leaving the frame buffer free for reference images; an image can be traced on the overlay plane from an image in the frame buffer.



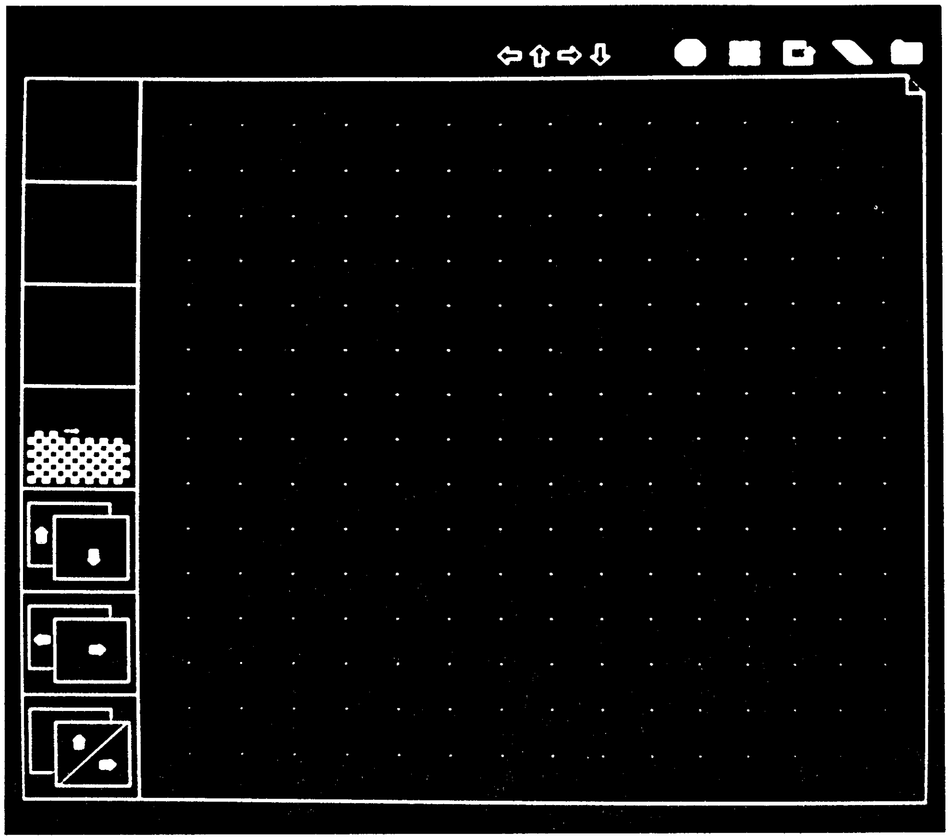
Grid-filling.

The screen is divided into three areas: drawing or work area, display area of up to seven icons for comparison or temporary storage, and an area for function icons. Drawing, erasing, storing, moving, starting over, and returning to contents are the available functions, and are represented and accessed by icons displayed accross the top of the screen above the drawing area.

The process of editing begins with a display of an icon directory on the upper monitor. The user picks, using the tablet and puck, the images to be modified. Up to seven



drawing area.

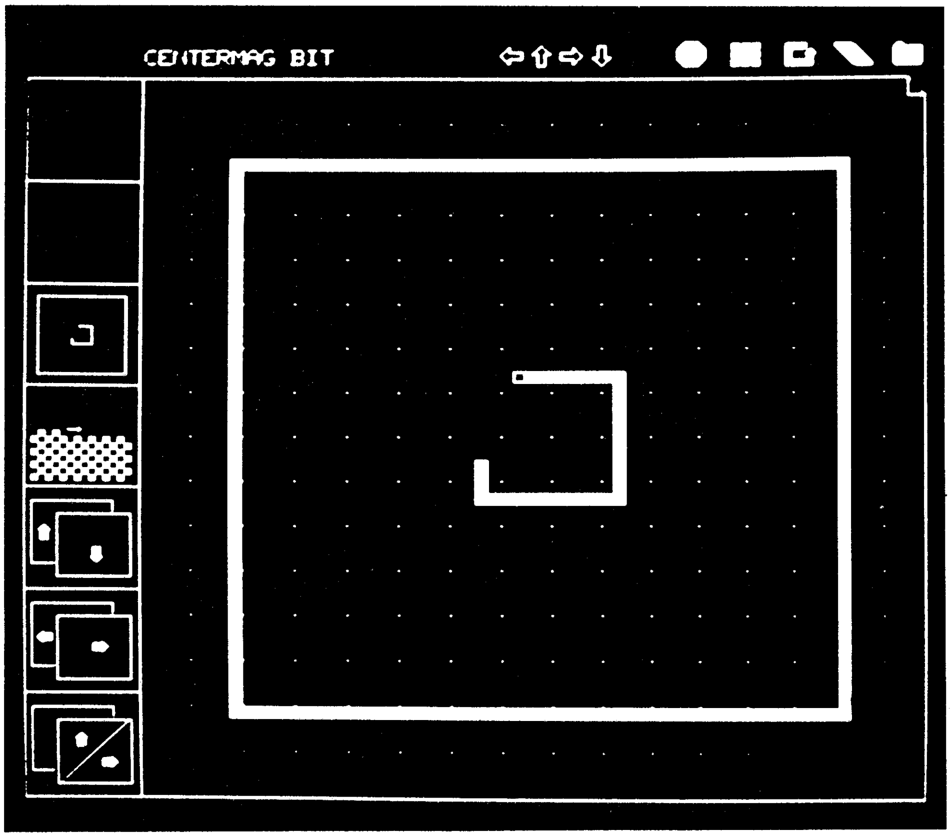


Choosing icons to edit.

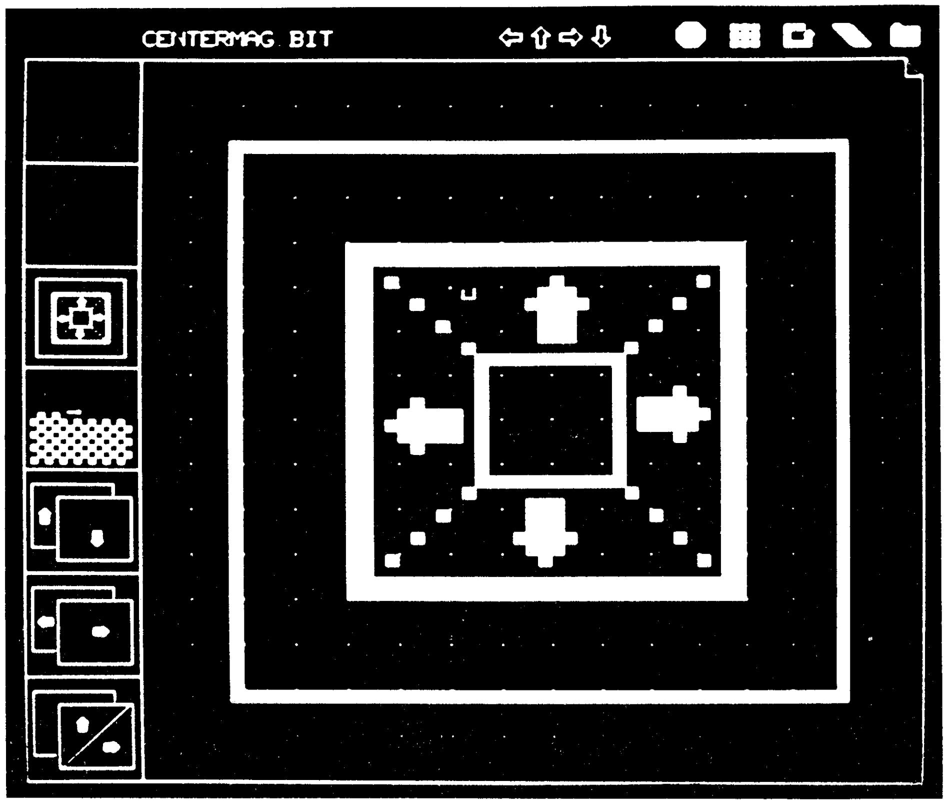
icons can be chosen. As they are picked, each icon appears in a slot on the lower monitor. After the choosing process is complete, attention is directed solely to the lower monitor.

To modify an icon the user picks it with the cursor; it is drawn, enlarged, in the drawing area. To draw a new icon the user picks a blank square. The user can either draw or erase squares at three module sizes: 7 x 7, 14 x 14, or 28 x 28 pixels. Each pixel in the icon is expanded to a 7-pixel square in the enlarged version. By drawing with the 7 x 7 module size, 1-pixel changes are being made at the icon size. Any changes in the enlarged image occur simultaneously in the icon.

The image can be moved either up, down, left, or right, and resaved in its edited form. Erasing is done by toggling the erase icon, on or off. It is possible to start an image over by touching the startover icon. The user can return to the contents of the directory by touching the return icon. Upon saving any modified iamge, its file is updated with the new version in the directory >u>icon.



Starting a new icon.



The finished icon.

3.3 Icon-Scripting

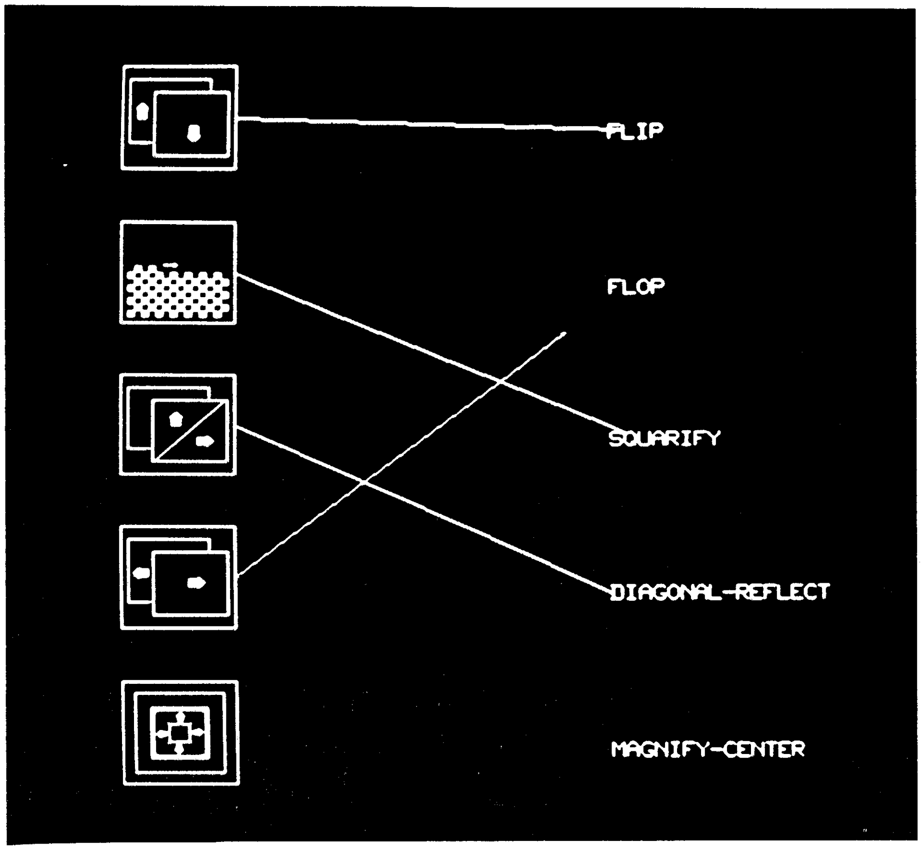
Scripting demands a method to keep track of a sequence of "things", in this case 1-bit images. I started with some software written by Professor Ron MacNeil (and Richard Mylnarik, M.I.T. '84) which keeps track of an order of generic rectangles by using doubly linked lists. by adding to the software and inserting icons into this package in place of rectangles, I created a storyboarding system.

"Storyboarding" is a term used in movie-making to describe the visual planning of the sequence of scenes. Traditionally done on paper, it consists of a series of images representing a number of key frames that can be arranged and rearranged. It acts as a visual map of the movie. My storyboard allows the user to pick, with the puck and tablet, from the contents of the directory >u>icon>** displayed on the upper monitor. As the images are picked, they are displayed on the lower monitor in the order chosen.

Once the choices are made, the icons can be pointed to and moved to a new location. The sequence is then updated to the new order. When a desired sequence is reached, it can be saved, tied to procedures, displayed on the upper monitor and played out upon request. Chosing procedures is done by pointing with the cursor to a displayed icon and drawing a connecting line to the name of a procedure.

Storyboarding was first useful within itself as an planning device for animation, video, or video discs. This storyboarding capability afforded a way to plan sequences interactively.

Making and using icons are separate entities. In a later version these could be tied together in an inherent way. Currently, the user has the choice as to the level of involvement.



Connections.

* A menu package was written by Hennigan '83. It allows a tree-structured access path to individual programs.

C O N C L U S I O N

There are several directions in which this project could be carried farther. One is to extend it as a more highly interactive and fluid editing tool. New features could be incorporated such as automatic icon generation and animated icons that mock the procedure to which they point. A second direction could be toward greater capabilities of information and data display. In networks of computers, for example, in the context of an electronic bulletin board or help directories, icons could tell synoptic stories or illustrate a structure of events. A third direction is to expand the concept of visual notation into something that approaches a visual language.

Language is, among other things, a symbolic code by which messages are transmitted and understood, by which information is encoded and classified, and through which events are announced and interpreted. Like language, culture is a symbolic code through which messages are transmitted and interpreted. But more than that, a culture is a set of conceptions of, and orientations to the world, embodied in symbols and symbolic forms. Through the adoption of and adherence to particular concepts of and orientations to reality, human beings actually create the worlds within which they live, think, speak, and act. This is the process of world construction, or the social construction of reality.

Configuration of environments and worlds of interfaces is a construction of a concept of reality. Alan Kay refers to the interface world as the "myth of the computer". Much care and responsibility is necessary in designing computer interfaces because they become models for our own reality. Like language, they become the model and teacher of our children. Human beings from many disciplines should have the opportunity to contribute their particular expertise to ensure a balanced and broad-based result.

APPENDIX I: The Computing Environment

HARDWARE

The center of the computer system is a Perkin-Elmer 3220 32-bit mini-computer with 512K of core memory. Disk storage is a 300 megabyte "trident-type" drive with a high I/O bandwidth. Images are displayed with a Grinnell GMR-270 frame buffer that has 512 x 512 pixel resolution. Each pixel contains 27-bits of color information, 8 bits each of red, green and blue.

There are also three overlay planes referred to as planes eight, nine and ten, each with 1-bit of color information. Each plane can be enabled or disabled individually. When enabled, a plane may be either opaque white or one of six transparent colors.A second black and white monitor with the same resolution is used as an alternate display for plane ten. This allows uninterrupted image making to occur on the color monitor with simultaneous menu display on the b&w monitor.

The input devices include a Summagraphics Bit Pad tablet with a 4-button puck, and a vidicon surveillance camera which feeds non-composite video signals to the frame buffer.

SOFTWARE

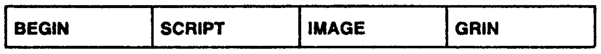
The operating system, MagicSix, supports a multi-user environment with a tree structured file and directory system, dynamic linking, and a process stack. The language is pl/1, a subset of PL/1, and supports structures, signals, pointers, and "initiated segments," allowing the user to structure core memory. Both MagicSix and pl/1 were developed at the Architecture Machine Group, M.I.T.

Assembly language library routines exist providing the low-level file system and graphics code called by pl/1 programs. The frame buffer has its own assembly language interface. Code for the frame buffer, included as ASCII text in pl/1 programs, is sent to and interpreted by the frame buffer at run-time. A complete library of graphic routines has been developed for the frame buffer.

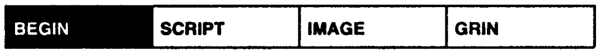
Appendix II - User Scenario

A. Creating a Script

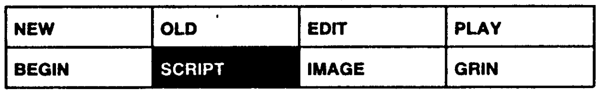
To begin creating a script with the icons the user types "IS" (Icon Scripting) to invoke the menu. The initial menu is displayed below.



"BEGIN" initializes the pointers, clears the overlay planes, draws a palette at the top of the screen, and allows the user to choose a background color.



"SCRIPT" is the root to all the scripting procedures.



"NEW" is used to create a new script. It requests the name of an icon subdirectory (>u>icon>**) and displays the contents of it on the top monitor (plane 1Ø). Using the z-button on the puck, the user chooses any number and sequence of icons. As they are choosen the icons are displayed on the lower monitor on plane 8 consecutively from upper left to lower right. The 1-button exits from the process.

NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

"EDIT" accesses the procedures to modify the new script.

INSERT	DELETE	STORYBOARD	CONNECTIONS
NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

"INSERT" lets the user insert icons from contents into a sequence. Using the z-button, the user first points to the new icon, and second, points to the one behind which to insert it.

INSERT	DELETE	STORYBOARD	CONNECTIONS
NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

"DELETE" eliminates icons from the sequence. It works in a similar way to INSERT.

INSERT	DELETE	STORYBOARD	CONNECTIONS
NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

"STORYBOARD" allows the user to rearrange the icons in a script. Using the z-button, the user first points to the icon to be moved, and second, points to the icon in front of which the first will be moved. The sequence is redisplayed starting from the first choosen icon and proceeding up or down the list to the second choosen icon. This procedure can be repeated indefinitely until a desired sequence is achieved. The 1-button exits the procedure. As each change is made the data is updated to the new sequence.

INSERT	DELETE	STORYBOARD	CONNECTIONS
NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

Once a desired sequence is achieved, "CONNECTIONS" displays the script on the left side of the lower monitor. Procedure names are displayed on the right side of the screen. Connections between icon and procedure can be made by drawing a line from one to the other.

INSERT	DELETE	STORYBOARD	CONNECTIONS
NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

"PLAY" displays the script on the upper monitor, highlighting the active icon, while the graphic events play on the lower monitor.

NEW	OLD	EDIT	PLAY
BEGIN	SCRIPT	IMAGE	GRIN

B. Making Icons

"IMAGE" is the root to icon-making and picture-making with icons.

ICON	PICTURE	LIST	RESOLVE
BEGIN	SCRIPT	IMAGE	GRIN

"ICON" is the path to all the icon-making support programs.

MAKE	SCALE	SAVE	LOAD
ICON	PICTURE	LIST	RESOLVE
BEGIN	SCRIPT	IMAGE	GRIN

"MAKE" gives the user the option of creating a bitmap or going into the icon-editor to edit existing icons or draw a new one. (The icon-editor is described in chapter 3).

GRID	BITMAP		
MAKE	SCALE	SAVE	LOAD
ICON	PICTURE	LIST	RESOLVE
BEGIN	SCRIPT	IMAGE	GRIN

APPENDIX III: Programs Descriptions

Icon-Making Programs:

bitmap

Creates a bitmap within user established parameters from a frame buffer image. Parameters are threshhold value between Ø and 255, a range value from the threshhold, and dither factor between Ø and 1. bitmap_save is an entry point in bitmap and calls a system routine that saves a bitmap in the directory > pix>bitmaps. bitmap_load is another entry point that calls a system routine to load a bitmap from >pix>bitmaps. bitmap_list is a third entry point that calls a system routine to list the file contents of >pix> bitmaps.

icon_scale

Scales 1-bit images by two methods: full_page_scale is an entry point and scales a full page 1-bit image to icon-size (64x64 pixels). define_icon is another entry point and lets the user define a portion of a 1-bit image to be an icon, allows for proportion changes and displays the coordinates. It automatically scales the defined image area to 64x64 pixels.

oscale

A simple scaling procedure for a 1-bit image that lets the user change proportions and scale to any size.

icon_save

Saves an icon in the directory >u> icon. It initiates a segment, prompts for a name, and call a system routine to write to the segment. The image to be saved is defined interactively with a displayed box (64 x 64 pixels), moved with tablet and puck.

icon_load

Loads an icon from the directory >u> icon. It initiates a segment, prompts for a name, and calls a system routine to read from the segment. Placement is with the tablet and puck.

icontents

Displays the contents of the directory >u>icon on the upper monitor. The user can delete, choose, position, and resolve icons to the frame buffer. If the directory is too large to be displayed completely on one screen, the user can cycle through screens of images.

icon_list

Calls a system routine to list the contents of the directory >u>icon.

icon_delete

Deletes segments from the directory >u>icon.

clear_planes

Clears the overlay planes individually. clear_8, clear_9, and clear_1Ø are entry points.

pix

Calls the system routines to load and list the directory >pix. pix_load and list_pix are entry points.

icon-editor is a package of programs for producing and editing icon-images by filling in grid modules, bitmaps created in the above program can be edited to cleaner images. The programs included:

main

The base program. It initializes the data and calls all other programs.

grid_draw

Displays the chosen icons, a drawing area, and iconic representations of the utilities to choose from: save, erase, start over, go back to contents, quit, move. It lets the user edit the chosen icons or create a new one. It draws simultaneously large and icon-size.

open_seg

Creates and initiates a segment in which to save the data.

close_seg

Terminates a segment.

icon_load2

Initiates a segment and loads icons from the directory >u>icon. It calls a system routine to write 1-bit images to the overlay planes.

icon_save2

It initiates a segment and saves icons in the directory >u>icon. It prompts for an icon name and calls a system routine to read 1-bit images from the overlay planes.

icon_move

Moves the icon and the enlarged image up, down, left, or right. move_up, move_down, move_left, and move_right are entry points.

icontents2

Displays the contents of the directory >u>icon on the top monitor and lets the user pick icons to edit. Displays the names of each icon underneath the image.

template

Called by grid_draw. It draws the template that defines the drawing area, grid, and icons to call the utilities.

Scripting Programs

is.menu

The menu through which all programs are accessed.

glstr.incl.pll

The data structure for groups, elements, and pointers. The root is the only global variable. It keeps track of the group's linked list structure by pointing to the head group, tail group, and current working group. In addition, it keeps track of the current marked element, saved element, background color, storyboard status, and area into which all this data is stuffed. Each member of the group structure keeps track of its two sister members, next_gp and prev_gp, and the first, tail, and current working elements which are its responsibility. The element structure is at the end of this chain. Each element has to keep track of its sisters, next and prev, and also keeps track of where it is at any point in time reletive to the origin with x_left and y_bot, and what color it is.

startup5

Does all the setup work: making static variables, creating the first dummy group and element pointers, initializes the data in current group and element, and allocating the space.

select_color5

Allows the user to pick colors with the cursor from any place on the screen. It pans the screen down to the palette which is written at top of screen, waits until z-button push, and stores the color under cursor in the structure.

select_bkgnd_color5

Allows the user to select a background color from any place on the screen. It pans the screen down to the palette which is written at top of screen, waits until z-button push, and stores the color under cursor in the structure. Called by startup5 tpalette5

Draws a palette of a color spectrum and greyscale at the top of the frame buffer; it must be panned down to pick from it. Called by startup5

icontents5

Displays the contents of the directory >u>icon>** and adds elements chosen.

icontents5Ø

Displays the contents of the directory >u>icon>** and inserts elements into the list.

icontents6

Displays the contents of >u>icon>** on plane 1Ø and adds elements chosen.

icontents6Ø

Displays the contents of >u>icon>** on plane 1Ø and inserts elements chosen.

icon_load5

Initiates a segment and loads an icon calling oswrit2.

search_el_windows5

Searches through the element list looking at the window locations setting the ovelay plane and marker when a position match is found.

marked_to_saved5

Changes the "marked" element to a "saved" element.

delete_saved_el5

Deletes an element marked "saved" from the list.

put_saved_under5

Adds the element marked "saved" into the list under the element "marked".

erase_all_overlay5

Runs thru the list of elements starting at head-list and redraws them using the backgroud color. The companion process is called redisplay_all.

put_under_element5

Called by insert_element. The only difference is marked vs curr. It sends back to glstruct the actual position of the window.

erase_overlay_fr_marker5

Starting at the marked element, it travels through the list erasing all overlays.

excise_element5

Searches the element windows to mark on for removal, erases all deletes the element from the list and redisplays the current list.

delete_marked_el5

Searches the element windows to mark on for removal, erases all deletes the element from the list, and redisplays the current list.

insert_under_marked5

Allocates some space, inits the variables, and inserts pointers.

redisplay_fr_marker5

Starting at the marked element, it travels through the list redrawing all elements at their proper positions with their proper colors, etc.

redisplay_except5

Starting at the first element pointed to by head-list , travel through the list redrawing all elements at their proper positions with their proper colors except the marked one.

redisplay_all5

Runs through the list of elements starting at head-list and redraws them using the backgroud color. The companion process is called erase_all5.

erase_all5

Runs thru the list of elements starting at head_list and redraws them using the backgroud color. the companion process is called redisplay_all.

move_placed_element5

Searches the element windows until the z-button push and then removes the marked element,and with put_el moves that element around the overlay, erasing top ones until a z-button push places the icon.

resolve5

Writes an image from an overlay plane to a chosen rgb value.

add_element5

Adds an element at the end of the list. It sets up the element pointers: make a new element, select the color, select the size put it where you want.

add_el_at_tail5

Allocates some space and sets the variables.

set_group_overlay5

Starting at the marked element, it travels through the list erasing all overlays.

prev_group5

Bumps the current group backward one notch.

next_group5

Bumps the current group forward one notch.

move_group5

Updates the groups offset numbers storing the initial position on z-button push, and then erases all members and redisplays them anew, and also scales the elements around the center of gravity of group.

add_gp_at_tail5

Allocates some space and sets the variables.

rearrange5

Calls all the procedures neccessary to rearrange elements in a list.

reassign5

Reassigns the values of the x_left and y_bot for placement.

temp_values5

Changes the values of the x,y placement coordinates to a vertical display for connections.

untemp_values5

Changes the values of the x,y placement coordinates to for a script.

connections5

Displays the storyboarded icons and a list of procedures to connect them.

save_values5

Opens a segment and saves the informations about a script in that segment.

make_script5

Initializes a segment in which to save a script and reads the values of icon name and corresponding procedure name.

play_script5

Initializes the segment and plays the script.

get_old_script5

Prompts for a name of an old script, initializes the file, and sets the pointers.

B I B L I O G R A P H Y

Moshe J. Augenstein & Aaron M. Tenenbaum, DATA STRUCTURES AND pl/I PROGRAMMING, Prentice-Hall, Inc., New Jersey, 1979.

C.K. Bliss, SEMANTOGRAPHY (A Non-alphabetical Symbol Writing), vols. 1, 2, 3, The Institute for Semantography, Sydney, Australia, 1946.

Margaret Boden, ARTIFICIAL INTELLIGENCE AND NATURAL MAN, Basic Books, Inc., New York, 1977.

Dr. Richard A. Bolt, SPATIAL DATA-MANAGEMENT. MIT, 1979.

David Diringer, THE ALPHABET: A KEY TO THE HISTORY OF MANKIND, Hutchinson, London, 1968.

David Entwistle, "The Use of Graphics in Computing: A Personal View from the Inside", INFORMATION DESIGN JOURNAL, 3/3, 1983.

Ralph Norman Haber and Leland Wilkinson, "Perceptual Components of Computer Displays", IEEE COMPUTER, May 1982.

Carl G. Jung, ed., MAN AND HIS SYMBOLS, Doubleday & Company, New York, 1964.

Gyorgy Kepes, ed., SIGN, IMAGE, SYMBOL, George Brasiller, New York, 1966.

Dr. Ernest Klein, A COMPREHENSIVE ETYMOLOGICAL DICTIONARY OF THE ENGLISH LANGUAGE, vols. I & II, Elsevier Publishing Company, Amsterdam, 1967.

Kenneth N. Lodding, "Iconic Interfacing", IEEE, pp. 11-20, March/April 1983.

Alan MacDonald, "Visual Programming", DATAMATION, Vol. 28, no. 11, Oct. 1982, pp. 132-140.

Aaron Marcus, "Designing Iconic Interfaces", NICOGRAPH'83, pp. 103-123.

MINI-MICRO SYSTEMS, "English University Developes Unix Icon Interface", December 1983, p. 93.

Santiago Miranda, "Storia di un Progetto - The Design of the Consolles of the New Olivetti Office Machines", DOMUS, no. 638, April 1983.

A.C. Moorhouse, THE TRIUMPH OF THE ALPHABET (A History of Writing), Henry Schuman, New York, 1953.

Ted Nelson, "Interactive Systems and the Design of Virtuality", CREATIVE COMPUTING, Vol. 6, no. 11, Nov. 198Ø, pp. 56-6Ø , & Vol. 6, no. 12, Dec. 198Ø, pp. 94-1Ø2 .

THE OXFORD ENGLISH DICTIONARY, Oxford University Press, Oxford, 1933.

David F. Rogers & J. Alan Adams, MATHEMATICAL ELEMENTS FOR COMPUTER GRAPHICS, McGraw-Hill Book Company, New York, 1976.

Chris Rutkowski, "An Introduction to Human Applications Standard Computer Interface, Part 1: Theory and Principles, BYTE MAGAZINE, pp. 291-31Ø, October 1982.

Ben Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages", IEEE COMPUTER, August 1983, pp. 57-69.

Dr. David Canfield Smith, Charles Irby, Ralph Kimball & Eric Harslem, "The Star User Interface: An Overview., Proc. of National Computer Conference; 1982 June 7-1Ø, Houston, pp.515-528.

Dr. David Canfield Smith, Charles Irby, Ralph Kimball, Bill Verplank & Eric Harslem, "Designing the Star User Interface", BYTE MAGAZINE, pp. 242-282, April 1982.

Albert Stevens, Bruce Roberts and Larry Stead, "The Use of a Sophisticated Graphics Interface in Computer-Assisted Instruction", IEEE COMPUTER, March/April 1983.

Ivan Edward Sutherland, "SKETCHPAD, A MAN-MACHINE GRAPHICAL COMMUNICATION SYSTEM, Phd Thesis, M.I.T. 1963.

Luc Vanmalderen and Claude M. Begue, "How to Build a Specific Code", INFORMATION DESIGN JOURNAL, 3/2, 1982.

Joseph Weizenbaum, COMPUTER POWER AND HUMAN REASON, W. H. Freeman and Co., San Francisco 1976.

Greg Williams, "The Lisa Computer System", BYTE MAGAZINE, February 1983, pp. 33-5Ø.

Thomas A. Wadlow, "The Xerox Alto Computer", BYTE MAGAZINE, September 1981, pp. 58-68.