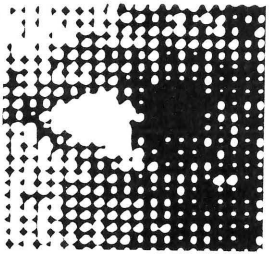
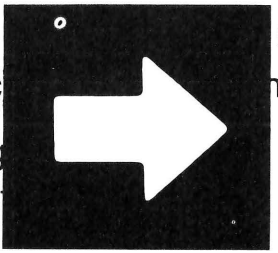




"A well designed system makes

"everything relevant to a task visible on the screen."

Scripting Graphics With Graphics: Icons as a Visual Tool

Dorothy Shamonsky

Abstract

This paper describes a system for scripting and editing graphic procedures with graphic representations or *icons*. The icons are small bit-map images that carry with them information about spatial placement, list placement, and pointers to procedures. Two interactive graphic software packages have been written, one for creating icons and one for scripting with icons. Icons can be created in two ways: (1) making bit-maps from frame buffer images and (2) drawing by grid filling. Icons have been created with pointers to a series of image transformation procedures. 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, played, and re-edited. All interaction is done with puck, tablet, menus, and visual cues. A historical overview of computer icons is presented, using several key systems as examples.

Introduction

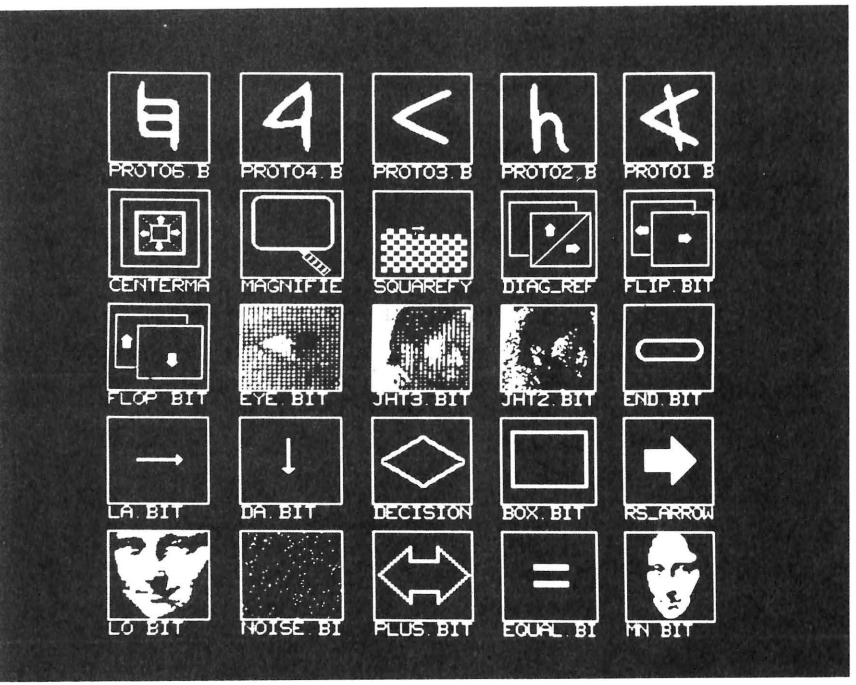
Suppose you could write a paper, then immediately get an outline of what you have written. You could choose to edit either the paper or the outline. Any change in one would effect the appropriate change in the other. Scripting graphics with icons is similar to this situation. It is a way of visualizing both the actual sequence and the structure of a sequence of visual events. In other words, it is a visual editing tool.

Text-editing on computers has achieved a high level of fluidity, while creating graphical events on computers, such as computer animations, still depends primarily on verbal rather than visual planning methods. Visual planning for videodiscs and electronic books depends largely on traditional methods of book design with pencil or paper. This can be awkward and ineffectual when what you, as a visual designer, are planning for is electronic, intangible, and has an immense number of variables. The computer tools have not caught up with the possibilities of projects that the technology affords.

A storyboard is an example of a non-computer visual editing tool. It consists of a rearrangable series of key images and is used in film, video, and animation to plan the story or define the sequence of visual events. It provides a method of planning a film or animation without actually editing cumbersome footage of film or videotape repeatedly from one position to another.

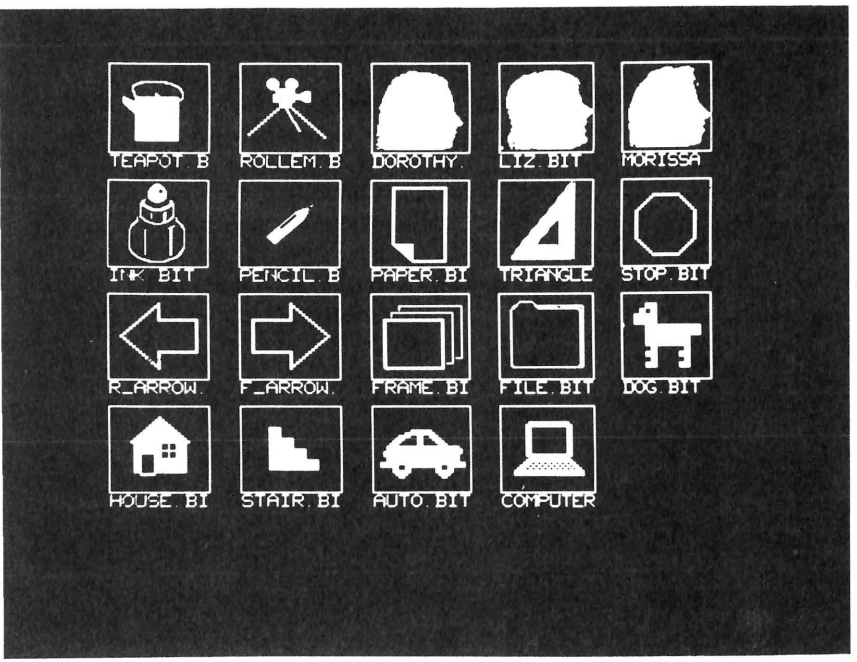
Visible Language XIX 2 (Spring 1985), 226-242

Author's address: 21 Spring Hill Terrace, Somerville, Massachusetts USA 02143



1
The contents of the directory of saved icons.

1a

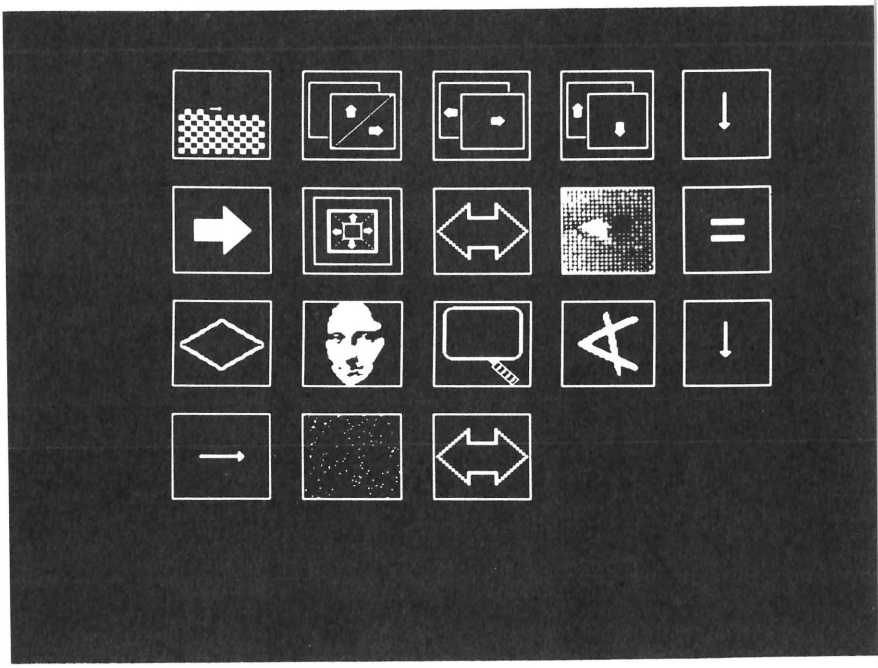


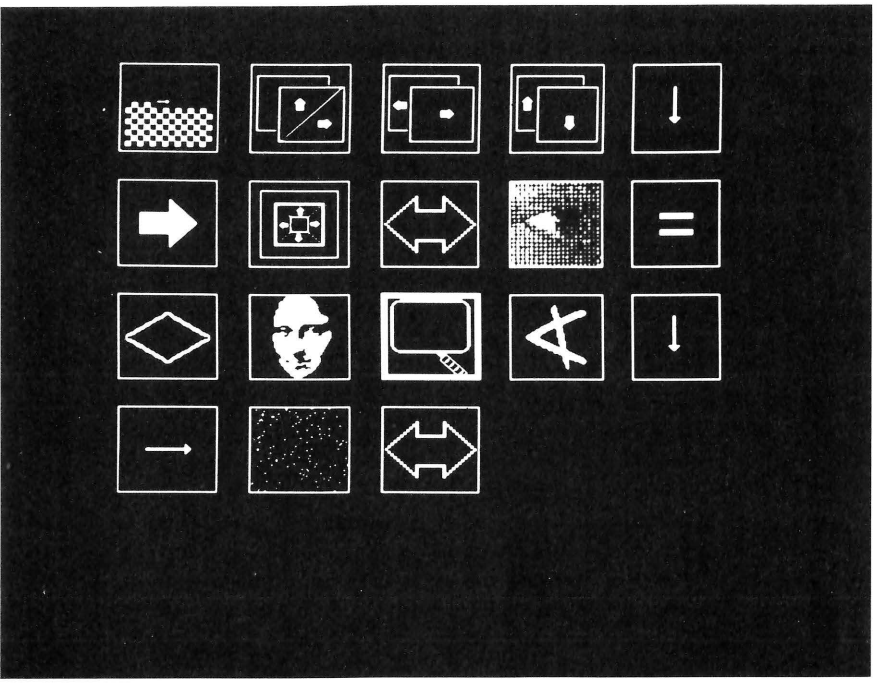
1b

Icon Scripting is conceptually similar to a storyboard (**Figure 1**). Icons ■ are used to define and rearrange graphic sequences around the computer interface *space* when those procedures can not otherwise be *handled* except as names (**Figure 2**). The icons are small (approximately one inch square), bitmap images (1 bit or one color) that represent, in this particular case, simple graphic transformations, such as reflection of one part of the screen to another.

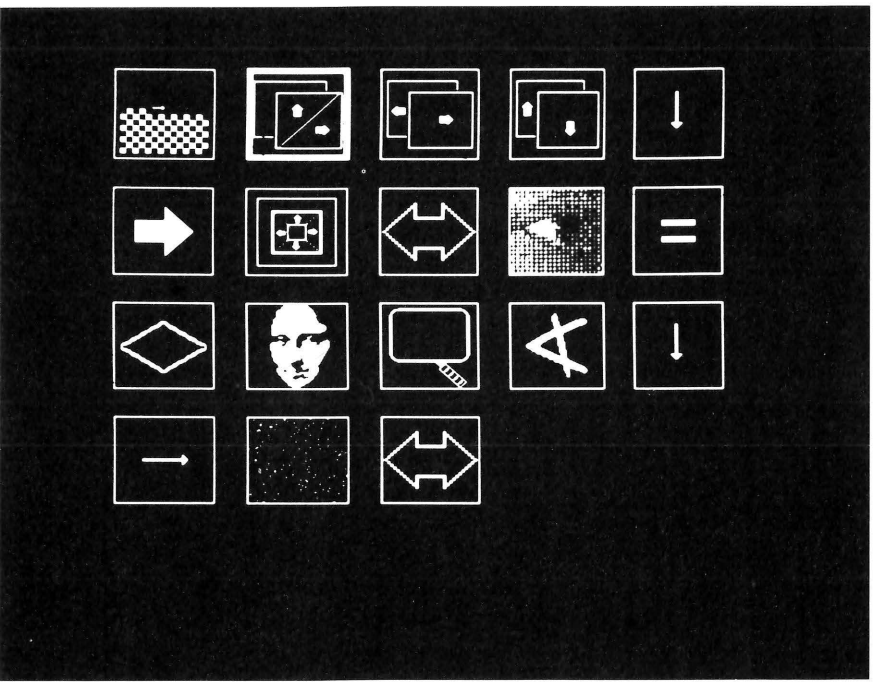
2
The storyboard capability:

2a
The sequence to be rearranged.



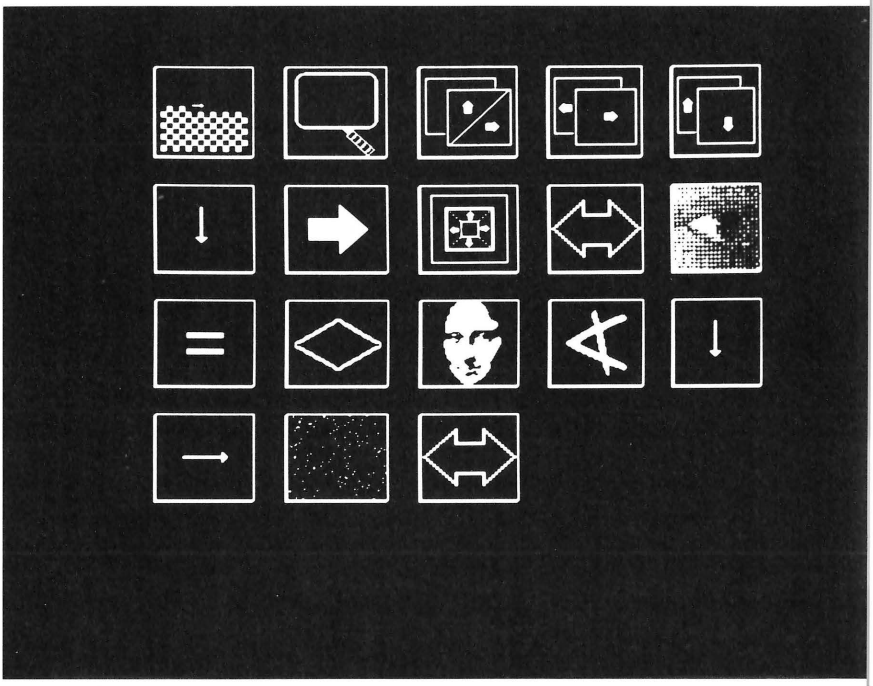


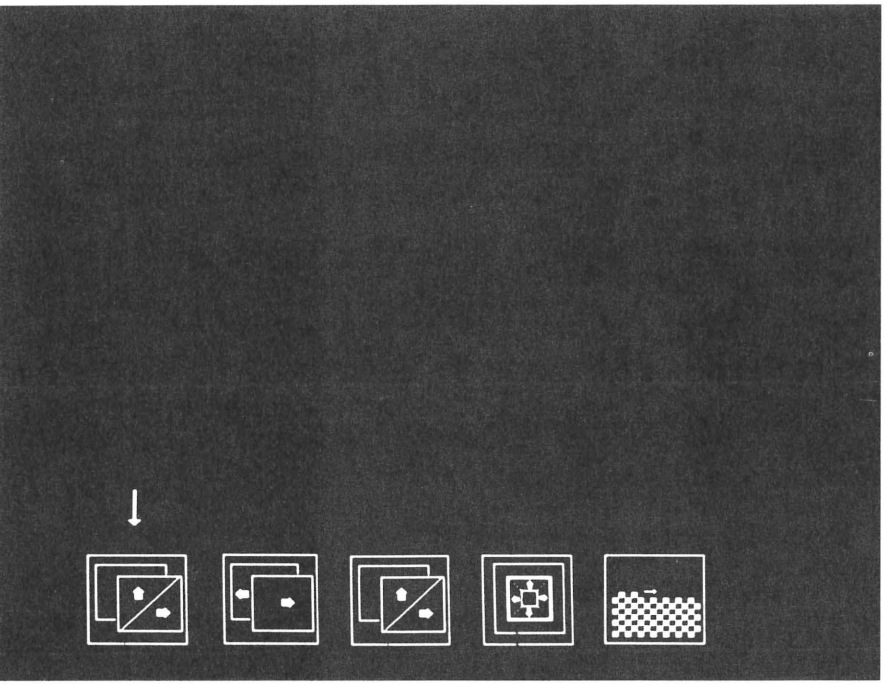
2b
Place this one (the highlighted box)...



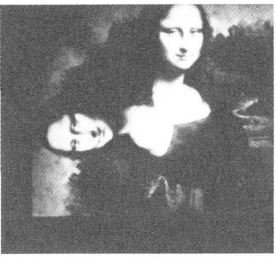
2c
in front of this one (the highlighted box).

2d
The new sequence redisplayed.

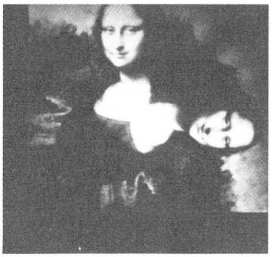




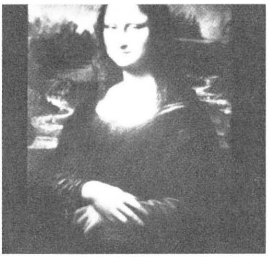
3a
A final script being played.



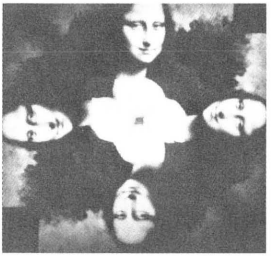
3b Original image



3c Diagonal reflect



3d Flop



3e Diagonal reflect

Each icon carries with it vital information about itself, such as its number in a sequence, its position on the screen, and what procedure it points to. A script can be defined by icons, saved, *played*, and re-edited any number of times (**Figure 3**).

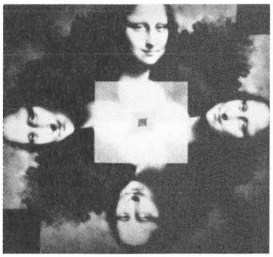
Icon is an accepted computer term to describe a graphic representation on a computer of a real person, object or function. The term *icon*, by definition, is an image, portrait, or semblance, especially of a saint. It comes from the Greek word which means *to be like* or *likeness*. The root of *icon* is also related to the Lithuanian word meaning *to occur* or *to come true*. [2]

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 *combine* two icons and they modify each other. To throw something away on the Apple Lisa the user *picks up* the object icon, *drags* it to the trash can icon and *lets go* of it. The object icon then disappears.

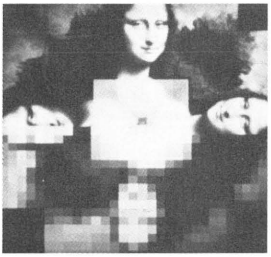
Defining and structuring graphic events on computers with descriptive words instead of pictures, prevents us from using the spatial skills we usually employ in solving visual problems. The sequence of drawings in a sketchpad or a scribbled page of thumbnail sketches are often the methods designers use to develop a visual structure. These methods allow one to see the structure of a piece, keep track of its parts, and visualize the final result.

Computers have eliminated many of these common structural aids or spatial cues because of the limitations of most interface designs. In the past, hardware limitations and high cost presented real restrictions to the use of graphics on computers. These limitations are rapidly disappearing as the hardware becomes more sophisticated and costs continue to drop. Along with these tangible changes, the knowledge and understanding of interface design has improved. Both a cause and result of this has been the inclusion of people from diverse fields, such as educators, psychologists, linguists, and artists, in the design process.

One might say that it has taken interface designers a long time to realize what artists have known all along, that



3f Magnify center



3g Squarify

pictures can tell a story much faster than words. Pictures are often a more direct and universal form of communication; pointing is faster than describing. An understanding of organization and structure is based strongly in spatial cues. "It is surprising how pervasive the underlying notion of spatiality is, even in the symbolic modes of thought." [2]

It is not surprising that images on computers are as important (and popular) as they are in the noncomputer world (eg. signs, logos, photographs). What is surprising is how crucial they are becoming to creating believable illusory environments at the interface. Graphics are becoming the primary tool for creating understandable conceptual environments, with *virtual* objects that function in believable and consistent ways within those environments.

The graphics used are often iconic rather that literal because they need only be *models* of reality. Visually simple, they often carry with them large amounts of information about their behavior in order to act true to their represented reality. Icons are like black box abstractions; as long as the user can read a label, she can work with the icon, as with its reality.

Recent studies have substantiated the use of graphic interfaces by showing that certain interface features evoke a positive response in users. 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) reversibility. [3]

The use of graphics in the interface and visual design tools may seem, at first, to be two separate topics. But the use of more graphics on computers demand an availability of intelligent and fluid design tools to design graphic interfaces. Better visual design tools will, in turn, lead to better, more useable and understandable interfaces.

A Brief History of Graphic Interfaces

Ideas for more accessible interfaces using graphics and pointing instead of verbal descriptions originated a number of years ago. In 1963, Ivan Sutherland created a graphical interface, called Sketch-pad, for his doctoral thesis. In his thesis he states:

> "Heretofore, most interactions between man (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." [4]

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

In 1964, another pointing device, the mouse, was invented by Douglas Englbart of S.R.I. (Stanford Research Institute, now S.R.I. International). It is only within the last couple of years that the mouse is incorporated and utilized in software systems as much or more than the keyboard.

Some of the earliest implementations of graphic interfaces were video games. "Pong" came on the market in the early seventies. It requires about thirty seconds of observation to be-

come a competent novice. Its concept is simple: two rectangles acting as paddles are manipulated with two knobs, and a smaller circle acting as the ping pong ball ricochets off the boundaries of the play area.

What is satisfying about the game is 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.

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 and creates a plausible *virtual* space at the computer interface. It used a joystick to move around that space quickly and easily.

The spatial world of SDMS consists of a single plane called *Dataland*. Dataland is a metaphorical desktop upon which rest, in fixed positions, images of objects: letters, telephone lists, reports, etc. It is presented to the user in two ways: in its entirety in an *aerial*, top-on view displayed on one 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.

Xerox extended the concept of the metaphorical desktop to a metaphorical office space. In the spring of 1981, Xerox released the 8010 Star Information System. It was a personal computer designed for offices. Xerox devoted about thirty work-years to the design of the Star user interface. Their primary concern was to define a conceptual model of how the user would relate to the system.

A conceptual model is a set of concepts a person learns to explain the behavior of a system. It is a model developed in the mind of a user to enable her to understand and interact with a particular system. The desktop is the principal Star technique for realizing the physical office metaphor. *Icons* are used as the visible, concrete embodiments of the corresponding physical object.

"Star is the first computer system designed for a mass market to employ icons methodically in its interface. We do not claim that Star exploits visual communication to the ultimate extent; we claim that Star's use of imagery is a significant improvement over the traditional human-machine interfaces" [5]

Incorporating many of the interface ideas developed at Xerox, Apple released the Lisa on the market at the end of 1982, and the Macintosh in early 1984. They are personal computers for both office and home use.

Greg Williams, senior editor of *Byte Magazine*, in describing his first encounter with Lisa gives us some insight into its appeal:

"With a few movements of the mouse... I *tear off* a sheet of Lisa Graph *paper*... and give it a heading *Annual Sales*... 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." [6]

Metaphors of the tangible objects are used, being physically moved with actions described as *cutting*, *pulling*, and

dragging. *Picking* with the mouse button has been extended to more sensual and demanding tasks. These are all illusions which expand the real world model of the computer interface.

Another application of icons has been in CAI (Computor Aided Instruction) systems, which often depend on the creation of the real world model as a learning tool. Icons act as models of component *parts* that can experimentally be put together to create models of systems.

A project, called *Steamer*, done at Bolt Baranek and Newman Inc., Cambridge, MA, is an instructional tool for U.S. Naval steam propulsion systems. A library of icons was created to represent component parts. Each icon carries with it information about what it represents and can interpret user input to affect the state of the modeled component, and in turn, the entire system. For instance, touching a gauge causes it to produce a value corresponding to the location of the touch on the gauge face; increasing one gauge reading might effect other gauge readings in other parts of the model.[7]

Overview of Scripting Software

The goal was to implement a system that would enable a user to interactively and fluidly create a visual script. The scripts need to be easy to edit with direct manipulation as the major mode of interaction. Interaction is with a puck and tablet and all procedures are accessed through a main, tree-structured menu.

The scripts, in that particular case, control a series of graphic transformation procedures, which are programs that currently reside on the computer at the Visible Language Workshop at M.I.T.[8] They are image manipulations such as reflecting part of an image across itself or flipping the image upside down.

To implement Icon-Scripting, I first needed to develop a system to create icons.

Icon Editing

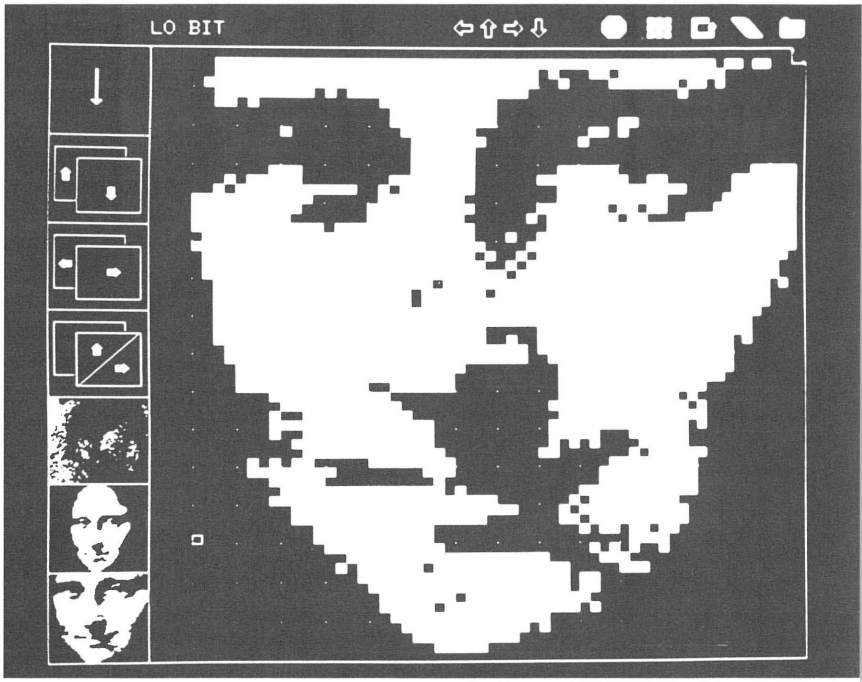
I developed an editor to create, edit, and save icons. Icons can be produced by two methods: (1) making bit-maps from full color images (these have been digitized with a video camera or created with other graphic software) and (2) drawing by grid-filling.

A bit-map (**figure 4**) is an image containing 1-bit of information for every pixel, or a one color image. They are made by reading the color value of every pixel of a full color image to evaluate whether it falls within a predefined range, and depending on its value the pixel is either turned *on* or *off*.

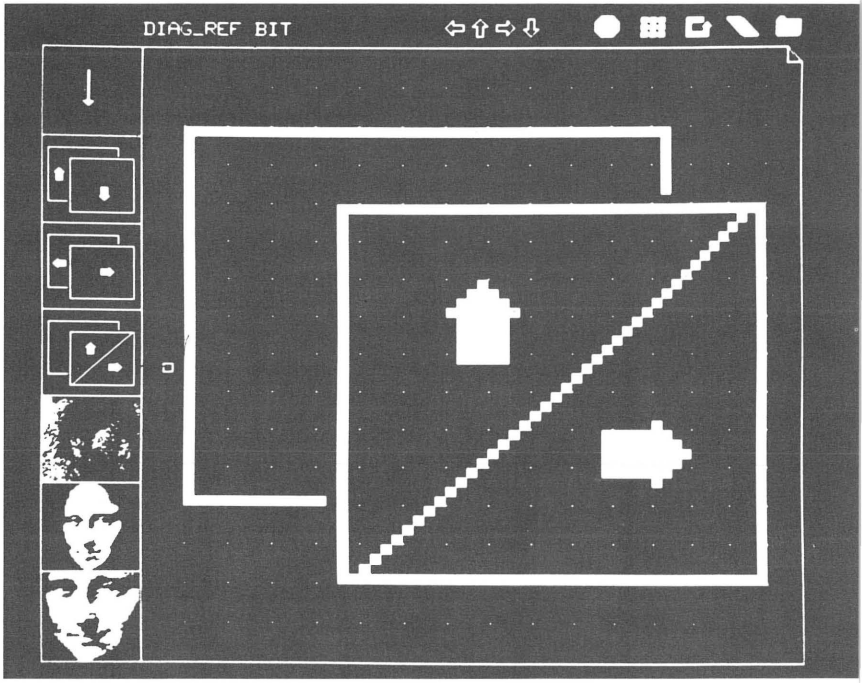
Grid-filling (**figure 5**) is simply filling in or erasing squares on a grid. New icons can be created with this method. It is also the method used to edit previously made icons.

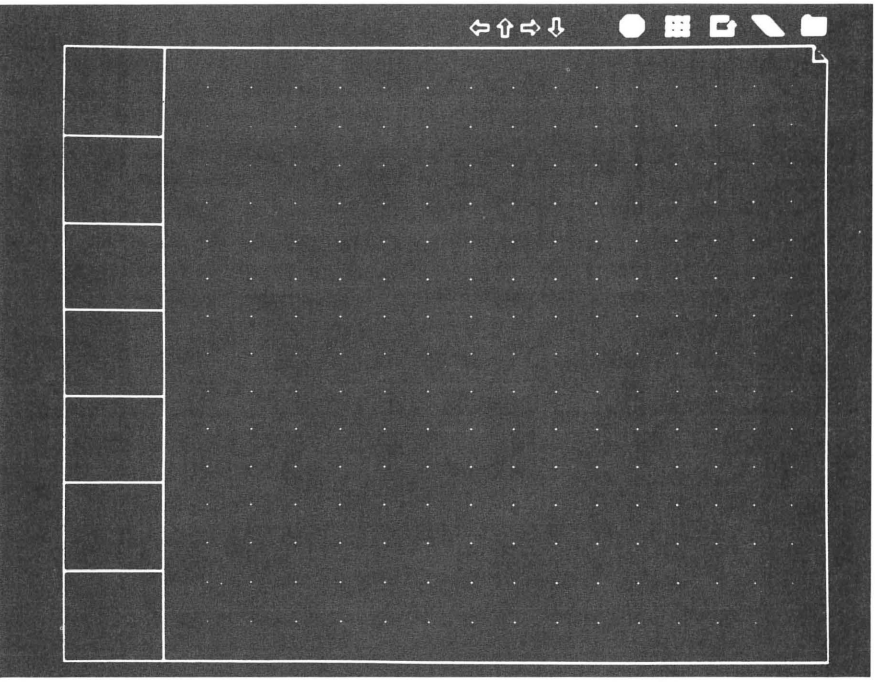
All activity in the Icon-Editor occurs on the overlay planes, leaving the frame buffer free for reference images. Copying an image on the overlay plane from an image in the frame buffer is a useful technique for creating an iconic image from a more complicated, full color image.

4
Bitmap

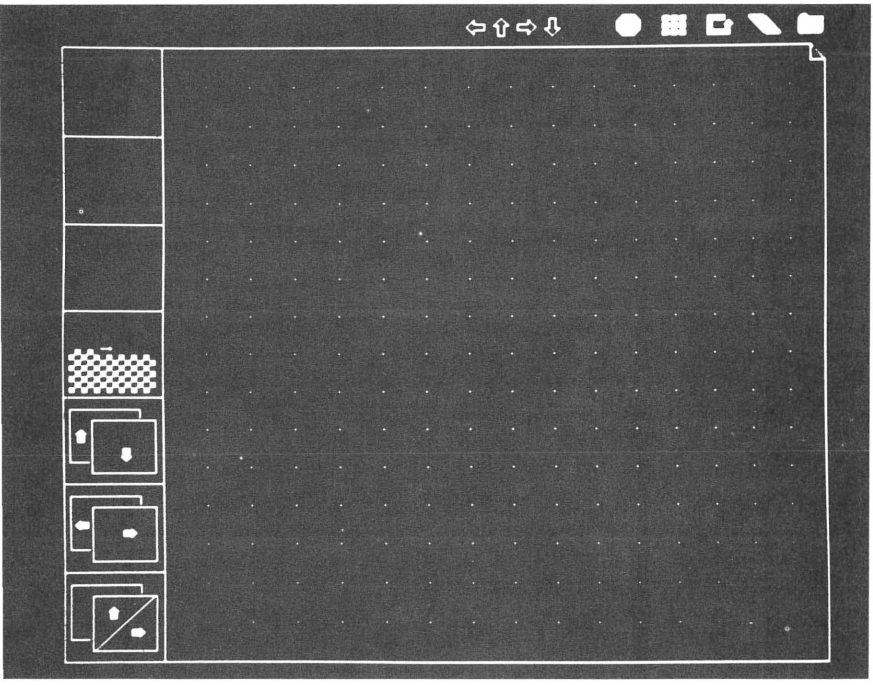


5
Gridfilling





6
Icon-Editing



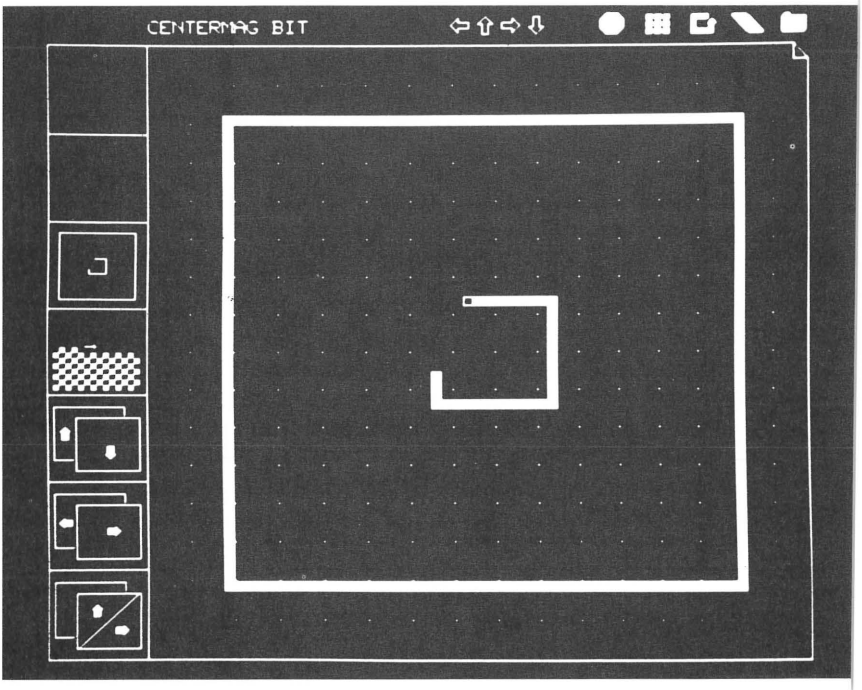
7
Choosing
icons to edit.

The screen (**figure 6**) in the Icon-Editor is divided into three areas: (1) drawing or work area, (2) display area of up to seven *current* icons (left side of the screen), and (3) the function icon menu (top edge of screen). The available functions are drawing, erasing, storing, moving, starting over, and displaying the contents of the icon directory. The name of the current icon is also displayed in the top left-hand corner of the screen.

The process of editing begins with a display of an icon directory on the upper monitor. The user chooses, using the tablet, puck, and cursor, the images to be modified (**figure 7**). Seven or less icons can be chosen. As they are chosen, each icon appears in a slot on the lower monitor. After the selection process is complete, the attention is directed solely to the lower monitor.

To modify an icon the user chooses it from the display area on the left side of the screen, and it is drawn, enlarged, in the drawing area. To draw a new icon the user picks a blank square from the display area (**figure 8**). The user can

8
Starting a new icon.



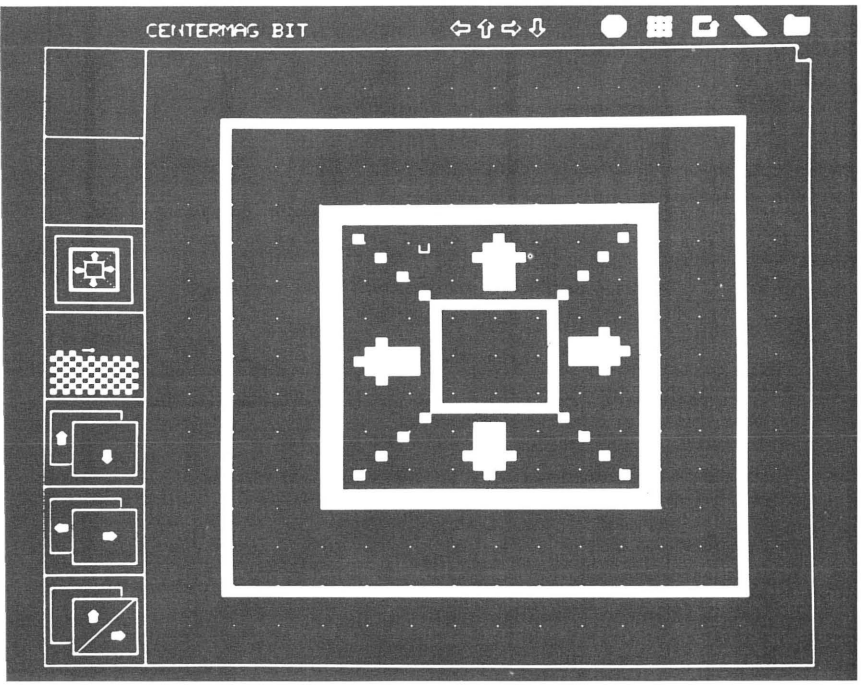
draw or erase squares at three module sizes. Drawing at the smallest module size (a 7-pixel square), 1-pixel changes are made at the icon size. These changes are seen simultaneously in the enlarged version and in the original icon (**figure 9**).

The image can be moved either up, down, left, or right, saved, modified, and resaved any number of times. Erasing is done by toggling the erase icon in the function menu, on or off. It is possible to start an image over by touching the startover icon. The user can return to contents of the icon directory to choose different icons to work on, by touching the return icon.

Icon Scripting

Scripting demands a method to keep track of a sequence of *things*, with the ability to dynamically rearrange that sequence any number of times. I used the conceptual model of a traditional storyboard to develop the Icon-Scripting software. Scripts can be created by two methods: (1) choosing icons from the icon directory, or (2) choosing procedures by name and then later displaying the icon sequence. The sequence of icons can be arranged and icons inserted and deleted. The script can be *played* at any time, and then re-edited.

To initially create a script with icons, the user chooses, using the puck, tablet, and cursor, from the contents of the icon directory displayed on the upper monitor. As the images are chosen, they are displayed on the lower moni-



9
The finished icon.

tor in the order chosen. When the selection is complete, attention is directed solely to the lower monitor. The method used to rearrange icons is to first choose the icon to be moved, then choose the icon to put it in front of. The sequence is immediately updated and displayed in the new order.

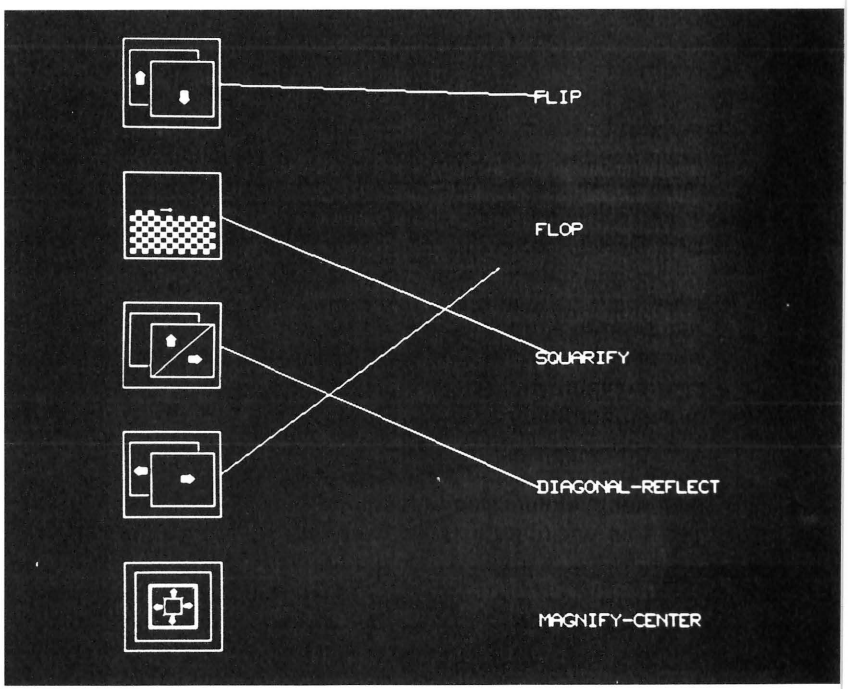
Icons are deleted by choosing *Delete* from the main menu and then pointing to the icon(s) to eliminate. Icons are inserted by choosing *Insert* from the main menu; the contents of the icon directory is displayed on the upper monitor while the user chooses an icon to be inserted and points to the icon in the sequence to insert it in front of. The script can be saved under a script name in the script directory. Scripts can be re-edited by re-storyboarding.

Tying icons to procedures is done in a separate function. Procedure names and icons are displayed together on the screen and the user draws a connecting line between an icon and a procedure name (**figure 10**). Connections between icons and procedures can be made or changed at any time in the printing process.

The second method of creating scripts is to choose procedures by name. The user can *play* the script, display the icon sequence, and proceed from that point to edit the sequence using the methods described above.

This software was written at the Visible Language Workshop at M.I.T., a graduate department and research facility concerned with the application of computers to the visual design fields. The computing environment includes a Perkin-Elmer 3220 32-bit mini-computer and a Grinnell GMR-270 frame buffer for image display with a 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 each with one bit of color information. Each plane can be enabled or

10
Connections to procedures.



disabled individually. When enabled, a plane can 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 one of the planes. 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 four button puck, and a vidicon surveillance camera which feeds non-composite video signals to the frame buffer. The operating system, MagicSix, supports a multi-user environment with a tree structured file and directory system. The language is PL/1, a subset of PL/1. Both MagicSix and PL/1 were developed at the Architecture Machine Group, M.I.T. [8]

Conclusion

This project could be carried farther in several directions. One is to expand it as an interactive visual editing tool. It is a natural computer counterpart to traditional storyboarding, and would be a useful planning tool for film, video, animations, and electronic documents or books. New features could be incorporated, such as automatic icon creation, and animated icons that illustrate the procedure to which they point.

A second direction is toward greater capabilities of information and data display. In the context of an electronic bulletin board or on-line help directories, icons could tell synoptic stories like a picture language.

A third direction is to expand the concept of visual notation into something that approaches a visual language. Language defines the set of conceptions of, and orientations to the world. Through the adoption of and adherence to particular concepts of and orientation to reality, human beings actually create the worlds within which they live, think, speak, and act.

Computers have begun to redefine language. They have created their own dialects and injected our languages with hundreds of new terms and metaphors. Computer interfaces, through the use of language, create a concept of reality. Like language itself, the computer interface becomes a model and teacher of our own reality. People from many disciplines should have the opportunity to contribute their particular expertise to ensure balanced and broad-based interface design.

Notes

[1]
Dr. David Canfield Smith, Charles Irby, Ralph Kimball, and Eric Harslem, *The Star User Interface: An Overview*, PROCEEDINGS OF THE NATIONAL COMPUTER CONFERENCE; 1982, June 7-10, Houston.

[2]
THE OXFORD ENGLISH DICTIONARY, Oxford University Press, Oxford, 1933.

[3]
Dr. Richard A. Bolt, SPATIAL DATA-MANAGEMENT, M.I.T., 1979.

[4]
Ben Shneiderman, *Direct Manipulation: A Step Beyond Programming Languages*, IEEE COMPUTER, August 1983.

[5]
Ivan Edward Sutherland, *SKETCHPAD, A MAN-MACHINE GRAPHICAL COMMUNICATION SYSTEM*, Phd. Thesis, M.I.T., 1963.

[6]
Dr. David Canfield Smith, op. cit.

[7]
Greg Williams, *The Lisa Computer System*, BYTE MAGAZINE, February 1983.

[8]
Albert Stevens, Bruce Roberts, and Larry Stead, *The Use of a Sophisticated Graphics Interface in Computer-Assisted Instruction*, IEEE COMPUTER, March/April 1983.

Special thanks to Prof. Ron MacNeil of The Visible Language Workshop for his help on this project.