2

Conf-9110268--1

UCRL-JC-108350
PREPRINT

Visual Tools and Languages: Directions for the '90s

Ephraim P. Glinert
Rensselaer Polytechnic Institute
Troy, NY 12180

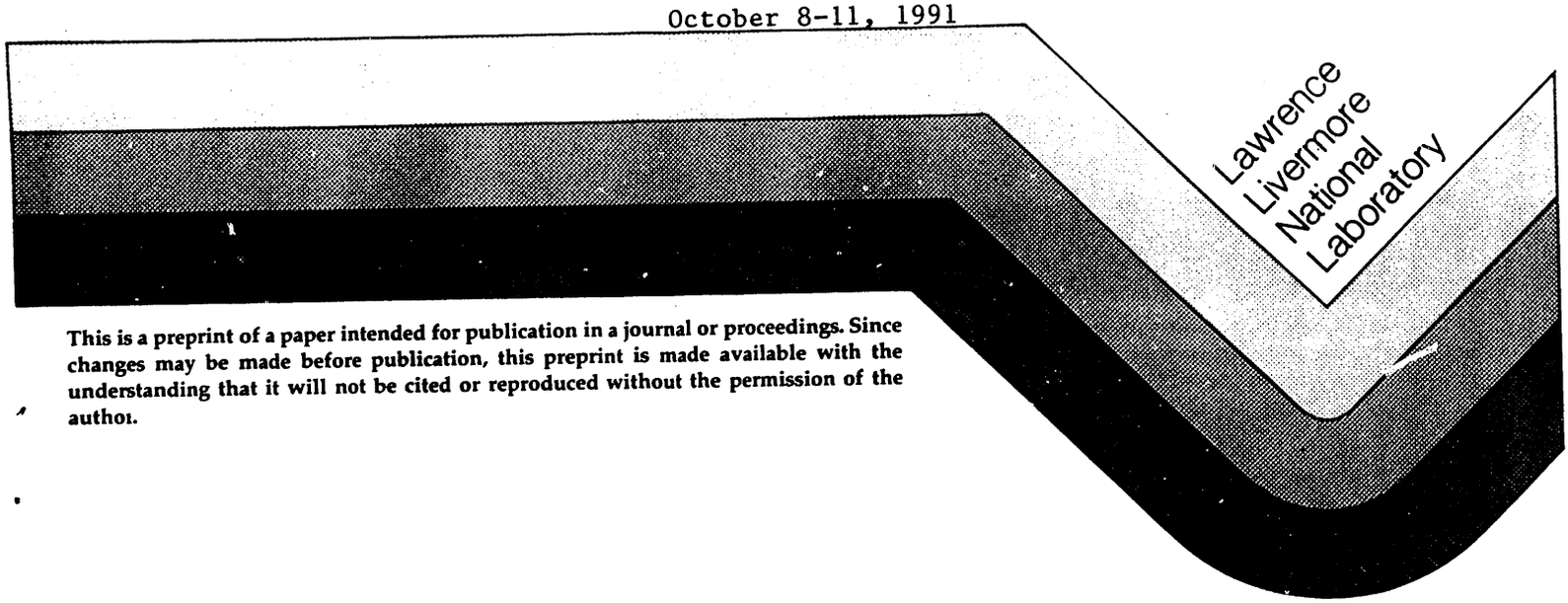
Meera M. Blattner
Lawrence Livermore National Laboratory

Christopher J. Frerking
University of California, Davis,CA

To appear in 1991 IEEE Workshop
on Visual Languages

Kobe, Japan

October 8-11, 1991



UCRL-JC--108350

DE92 001820

Visual Tools and Languages: Directions for the '90s*

Ephraim P. Glinert
Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180

Meera M. Blattner†
Christopher J. Frerking
Department of Applied Science, University of California, Davis, &
Lawrence Livermore National Laboratory, Livermore, CA 94550

Abstract

We identify and discuss three domains where we believe that innovative application of visual programming languages is likely to make a significant impact in the near term: concurrent computing, computer–based assistance for people with disabilities, and the multimedia/multimodal environments of tomorrow in which it will be possible to hear and physically interact with information as well as see it.

"The limits of my language mean the limits of my world"
Ludwig Wittgenstein [1, Prop. 5.6]

1 Introduction

Philosophers have long speculated that natural language may to some degree circumscribe our thoughts, that it may be difficult or even impossible for us to think about that for which we lack words. The Shakespearean scholar and amateur mathematician Edwin Abbott explored a hypothetical two–dimensional "Flatland" [2] whose inhabitants were unaware that their world was in fact embedded in a three–dimensional space which, if they could but perceive it, would afford startlingly novel views of their universe.

Those whose research falls within the area known as *visual programming* believe that textual programming languages constitute just such a flatland. They hold that computer graphics can expand the programmer's horizons and be used to advantage for defining or helping to define programs. For the near term there are two goals:

- To find ways to empower people who would otherwise be unable to program to do at least simple things without having to rely solely on canned software.
- To find ways to enable highly skilled professional programmers to successfully tackle domains for which traditional (textual) languages are inadequate.

For the long term, the hope is to develop "general purpose" *visual languages* which combine textual and graphical elements in so powerful and compelling a manner that tomorrow's programmers would prefer them to purely textual languages for their every–day tasks.‡ Such languages might also provide a medium for expressing algorithms in ways which we at present cannot even imagine.

Are such goals achievable? What role computer graphics should play in human–machine communica-

*The authors may be reached via e–mail as follows: *glinert@cs.rpi.edu*, *blattner@lll–crg.llnl.gov* and *frerking@lll–crg.llnl.gov*. The authors were supported, in part, by the United States Department of Energy through Lawrence Livermore National Laboratory under contract W–7405–Eng–48 while this research was being carried out. The first author was also supported, in part, by the National Science Foundation under contracts CDA–8805910 and CDA–9015249.

†Meera M. Blattner is also with the Department of Biomathematics, M.D. Anderson Cancer Research Center, University of Texas Medical Center, Houston.

‡Note that if we assume that, as traditionally used with respect to programming, the word "language" connotes a means for specifying a broad and varied spectrum of algorithms in a form amenable to mechanical processing, then many of the best–known systems developed to date for the visual domain are in fact *visual tools* of limited (albeit important) applicability.

tion is far from settled. Colin Ware has tried to explain some of the skepticism as follows [3]: "It is generally accepted that written language has evolved from the pictographic and iconic to the abstract and syntactic. Of all languages, the languages of mathematics and programming tend to be the most abstract of all. ... Thus, in a sense, the attempt to use graphical languages to design precise formal systems is a regressive step. If a complex, rich syntax is attempted then the graphical notation will become as obscure and unintuitive as a purely symbolic representation. ... This is not to say that we should ignore visual programming techniques, it is just that they will have only limited and specialized utility." While such arguments are hard to refute, it is interesting that the early developers of higher–level textual programming languages encountered similar resistance from devotees of assembler! As late as 1966, articles were being published with titles such as "*Assemble or Compile?*" [4].

Admittedly, many fundamental issues relating to visual languages are not yet well understood. What is the essence of visual as opposed to textual language? What is the role of the language as opposed to that of the interface? The power of natural language for conveying complex abstract ideas is undeniable. A picture that represented some word and had no other associations would be no advantage over the word itself. The strength of visual languages lies in their potential ability to simultaneously show multiple relationships.

Consider, for example, an undirected graph $\mathcal{G}$ with nodes p, q, r and s, and edges $\overline{pq}$, $\overline{pr}$, $\overline{ps}$, $\overline{qr}$, $\overline{qs}$ and $\overline{rs}$. To understand even this simple graph, most readers probably will be inclined to draw a picture. But which picture? Is $\mathcal{G}$ a square with two diagonals? Or is it a triangle whose vertices are connected to a common interior point? Perhaps $\mathcal{G}$ is a tetrahedron? Because in the absence of an appropriate context these and many other interpretations are all equally valid, a textual representation such as the above is most likely the best choice; any visual representation may imply features which $\mathcal{G}$ doesn't possess. Within a given context, however, a drawing will be far superior for imparting a desired interpretation to viewers, as well as for helping them to later recall it [5].

Despite all of the good work which has been done [6], the study of visual languages and programming is still in its infancy. Many avenues remain relatively or even completely uncharted. Pronouncements regarding the imminent demise or limited future role of this approach are therefore premature. In continuation of his remarks quoted above, Colin Ware goes on to proclaim: "Right now [the study of visual languages] is an exciting area for the researcher, because there are many discoveries waiting to be made." Where may such discoveries be expected in the coming decade? In this paper we explore possible answers to this question.

2 Visual Tools and Languages for Concurrent Computing

Designing, implementing and maintaining reliable programs which exploit novel parallel and distributed architectures is extremely difficult. Since so much effort has gone into exploring these domains, which most people see as the central focus of computer science for the foreseeable future, it is reasonable to ask why currently available tools are only marginally useful.

The conventional approach has been to incrementally extend the (textual) techniques which worked so well for serial algorithms. In the domain of parallel and distributed architecture, however, there is no doubt that graphical depictions can convey information regarding processor connectivity, computational dependencies, physical resources and the like more readily than either recursive expressions or pseudocode. As Kahn and Saraswat have recently put it [7]: "[Concurrent programs are] inherently about connections between computational agents and as such seem especially well suited for visualization." Even a cursory glance at the papers published by researchers in this field (e.g., in journals such as *IEEE Transactions on Machine Intelligence and Pattern Recognition* or *IEEE Transactions on Parallel and Distributed Systems*) confirms, that words routinely fail to adequately describe the architecture(s) under consideration. It then becomes necessary to resort to a profusion of diagrams of 2–D and even 3–D processor meshes, etc., to get the desired point across.

Several systems have been implemented which show that the visual approach can be successfully applied to the parallel and distributed domain. The trick is to abandon all vestiges of the serial programming approach, and instead adopt one in which the environment allows users to work at a higher level of abstraction. For example, Norton's NOVIS environment [8] supports interactive, visual design and animated execution simulation of complex parallel networks of communicating processes. Computations are specified as data flow graphs (DFGs) whose nodes represent processing elements (PEs). The DFG's edges dictate connectivity among the PEs, and represent the communication paths. Dependencies are specified by these edges, while synchronization is enforced by the execution schedule. Color and iconic image are employed as significant means of communicating information.

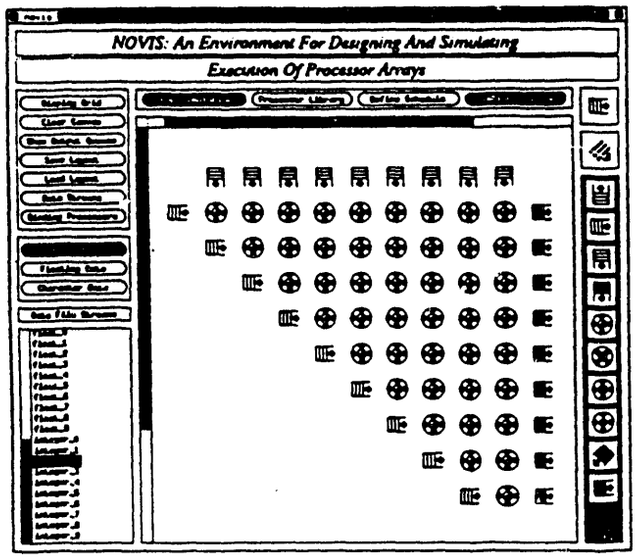


Figure 1: Norton's NOVIS System: A Processor Array for Sorting.

NOVIS's main contribution is that it provides a "laboratory" wherein users can experiment and explore their creativity in a natural manner. Information hiding encourages the design of parallel arrays from an abstract perspective and their modification "on the fly" followed by immediate resimulation. A variety of standard architectures such as SISD, SIMD, MISD and MIMD may be simulated with no need for intimate familiarity with the details of the internal workings of the system on the part of the user. An example NOVIS applications (sorting) is shown in Fig. 1.

Systems such as NOVIS are but a first, tentative step in applying the visual approach to the domain of concurrent computing. A central problem is that of mapping an algorithm onto a given parallel architecture. The diversity of these architectures often make it necessary to "massage" the algorithm so that it will fit a particular scale (e.g., number of processors) or model (e.g., message passing vs. shared memory) of parallel computation.

Most currently available tools support just a single parallel architecture or computational model. This leads to a lack of portability and limited scope of expression. As Browne has pointed out [9, pg. 76]: "Expressing a wide variety of algorithms in a single model of parallel computation is sometimes awkward, or leads to complex code." Carriero and Gelernter [10] detail three simple methods of writing parallel programs using message passing, distributed data structures, and "live" data structures. They also outline how a program written using one method can be transformed into a program based on a different method. Williams also discusses such models, along with methods for transforming the number of processes and for translating between models. She points out [11, pg. 158] that: "In an ideal world translations [between models] would be fully automatic (like a compiler) but at the current state of the art such translations are much more likely to be done with human interaction."

A compiler capable of automatically extracting a maximum degree of parallelism from a program, while resolving issues such as processor allocation and scheduling, does not seem realizable in the immediate future. We must therefore develop tools which:

- Assist in the design of parallel algorithms.
- Support multiple architectures or models.
- Assist in transforming programs from one scale or model to another.

This will, in part, involve devising mapping techniques which, given an algorithm, find the best architecture in terms of scheduling, processor allocation and communication, and scalability. Ideally, each part of a program would be described in terms of the model to which it most naturally fits, and then interactively translated to fit the architecture on which it must run.

> *WE PROPOSE that members of the visual programming community focus their efforts towards development of tools and languages to facilitate interactive coding of parallel programs and their transformation among architectures and models.*

Visual techniques could be used to good effect throughout such systems. In initially describing the program, restrictions on symbol placement, color changes, simple animation, and other visual cues could be used to suggest program layout to the user. Different computational models could be kept distinct through use of families of visual representations. Animation could be employed, when programming, to show restructuring processes (e.g., by moving the pieces of a program slowly from one model/graphics set to another, stopping occasionally when hints from the user are needed) and, at run time, to make visible algorithm performance. Multiple views could be used whenever choices must be made, to allow the user to compare the results of each of several possibilities before deciding how to proceed. All of these features, when taken together, should allow the user to quickly see how well his/her

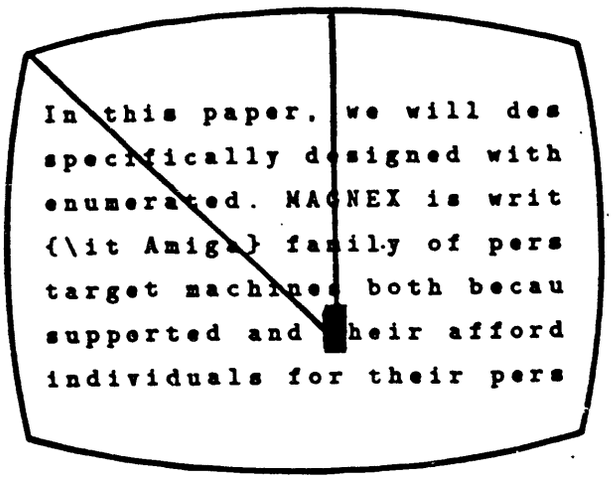


Figure 2: Schematic Drawing of the "Cross–Hair" Cursor in Vener's MAGNEX Editor.

code fits into various architectures, and thereby help him/her to gain (over time) a degree of intuition about the layout of parallel code.

3 Visual Tools and Languages for People with Disabilities

One aspect of the visual approach which has received far too little attention to date is that it may be both a blessing and a curse to people who are physically handicapped, and in particular to the large number who are visually impaired (not necessarily blind). In part, this is because there are so many different kinds of (visual) handicaps. More to the point, however, is the fact that most systems software has been, and continues to be, designed by and for people who are not disabled.

Consider, for example, the window–based interfaces which come with most modern personal computers and workstations. Menus, resource files or other hooks are typically provided to allow users to customize their working environment to suit their personal tastes. Windows can be repositioned and resized; foreground and background colors can be changed. The size and style of the font used to display text can often be adjusted. Sometimes, even keyboard functionality (bindings) can be reconfigured

Options like these are no less (and perhaps more) valuable for handicapped users than they are for users at large. But they are not enough. What is required are mechanisms for fundamentally modifying the visual aspects of the display in conjunction with the manner in which information is conveyed.

One solution is to design innovative visual software tools. An excellent example is provided by Vener's MAGNEX editor for the Commodore Amiga personal computer [12]. In this system, a novel "cross–hair" cursor makes location of the current typing position a simple task even for those who suffer from severe tunnel vision; cf. Fig. 2. While one of the cross–hair's two converging lines is vertical as we might expect, the other is anchored at one end to the top left–hand corner of the screen. This diagonal component provides a constant initial reference point for those users who may experience difficulty finding a conventional cursor on a (large) screen. From this initial reference point, the eye is drawn naturally down to the point where the diagonal component impinges on the horizontal in the central visual field; this acts as a feedback mechanism to signal proximity of the desired location, and permits the eye to rapidly home in on the appropriate character by following the convergence.

Carefully crafted visual tools such as MAGNEX tell only part of the story, however. Blissymbolics [13] is a semantically–based visual *language* which has proven remarkably effective as the basis of alternative communication systems for people who are severely disabled. The language contains a core vocabulary of over 2,400 symbols. Although some of these are pictographs which depict the outline or shape of the object they represent, most are ideographs which derive from a fixed set of so–called key graphic forms, each of which has a well–defined root meaning associated with it; cf. Fig. 3. Users who need to expand the system can do so by modifying existing symbols and/or by generating new ones by applying logical strategies and simple syntactic rules. In the HYPERBLISS system developed by Shalit *et al.* for the Apple Macintosh personal computer [14], the user applies his/her acquired knowledge of the structure of Blissymbolics ideographs to synthesize or approximate (in his/her mind) the graphical representation of the object he/she wishes to "talk" about. The user then retrieves the desired symbol by selecting one of its component parts from the keyboard.

> *WE PROPOSE that members of the visual programming community should play a central and leading role in developing languages and systems which are accessible to people with disabilities.*

As Alistair Edwards has recently written [15], in the past research related to human–computer interfaces "has largely been directed at taking account of quite subtle variations (such as novices versus experts). More significant variations in users' abilities—such as whether a user can see a screen or type on

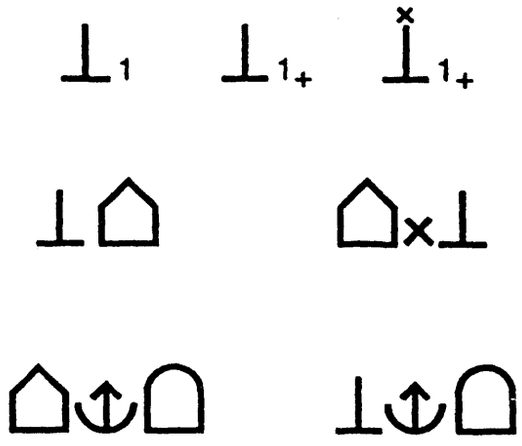


Figure 3: Examples of Blissymbolics Ideographs (After Shalit *et al.*). Top Row (Left to Right): *I*, *My*, *Our*. Middle Row: *Resident*, *Residential Institution*. Bottom Row: *School*, *Teacher*.

a keyboard—are described as 'disabilities' and accommodating them has been seen as a separate topic [lying] outside the mainstream ... [The study of] human–computer interaction has [now] reached a stage of maturity when it should be embracing the broader community." Legislation promoting computer accessibility for the handicapped recently enacted in the United States assures that interest in this field will grow markedly in the near future.

4 Visual Languages for Building Multimodal Interfaces

The introduction of graphics in the human–computer interface was hailed at the time as a means for making sense out of the mountains of unintelligible numbers which computers could spew out faster than people could cope with them. The availability of graphics in the interface led, in turn, to exploration of the possible role of visual languages in computing.

Technology doesn't stand still, however. We can postulate a not–too–distant future in which many of our senses will play a role in the interface. In such a world, one will be able to touch and hear information as well as see it; one will be able not only to experience data, but also to interact with and manipulate objects using one's hands, voice, etc.

That visual languages can and should play a prominent role in building the new multimedia and multimodal environments[5] cannot be disputed. After all, a visual language has been widely and successfully used for centuries by composers for recording musical scores. Visual languages are also commonly employed to prescribe choreography for the ballet.

To develop visual languages for defining and controlling the human–computer interfaces of tomorrow, we must first learn how these will differ from today's interfaces with respect to each of the two primary sensory modalities.

4.1 Increased Retinal Dimensionality

Jacques Bertin has enumerated the components of the "graphical vocabulary" [17] as being *size*, *saturation*, *texture*, *orientation*, *shape* and *color*, where the first two properties are useful primarily for conveying quantitative information, the last two for conveying qualitative information, and the middle two for both purposes. It has been established [18] that these six constituents are not of equal importance. For example, the relative position of two objects is more immediately discernible than which has the greater volume. They are also not independent (e.g., the ability to distinguish among shapes varies inversely with size).

Features which tomorrow's visual languages will need to be able to talk about include color, animation and 3–D. Conversely, these features might be used as language elements themselves.

It has been shown [19] that color can be incorporated into a display to aid in the location of objects, and that it is one of the best ways to cue in a discrimination task. This is because color is, as a rule, identified more accurately than size, saturation or shape. Nevertheless, color must be used with care, lest the screen become too busy and distracting [20].

The potential advantages of incorporating animation in the displays associated with certain tasks is also well documented. Systems have been implemented [21, 22] which demonstrate conclusively that displays of time varying data may be effectively used for representing abstractions such as algorithms.

Additional ways in which color and animation might be used other than for the visualization of data include *animated icons* [23] (e.g., in an effort to help users understand their meanings), and *animated assistants* [24] (e.g., "guides" with user selectable faces and personalities, who become more and more agitated as they vie for the user's attention).

With regard to the potential use of 3–D elements in visual languages, it is noteworthy that the literature

[5] For a discussion of the difference between the terms "multimedia" and "multimodality" see [16].

is replete with examples (e.g., random walk theory, knot theory and graph theory) where increasing spatial dimensionality from two to three (or changing its parity from even to odd) results in qualitative differences in results obtained. It is also known that extending the domain of discourse often leads to the solution of seemingly intractable problems. Would 3–D visual languages bear similar fruit? Only time will tell.

4.2 Increased Modal Dimensionality

Modern workstations come equipped with sound generation facilities. Because auditory perception spans a domain orthogonal to the visual, it is tempting to try to exploit nonspeech sound to allow users to efficiently cope with data of higher dimensionality than can be accommodated by visual means alone. Sound in the interface might make it possible to simultaneously use the display for more than one task. Alternatively, it could reduce the need for scrolling in large workspaces; if the user needs information not currently on display, he/she could listen to rather than look for it.

Incorporation of sound in the interface would naturally promote computer accessibility for blind and visually impaired users. Its potential role in languages for concurrent computing is less clear at this time, although it could surely be useful in environments for visualizing concurrency (e.g., for conveying information regarding algorithm and system performance). So, effective visual languages for parallel and distributed systems might well need to specify audio output.

For all these reasons, tomorrow's visual languages should be able to drive the use of sound.

Can complex information be presented usefully in an auditory pattern? Two basic paradigms for incorporating nonspeech sound in an environment have been distinguished in the literature:

- *The translation of data points into sound.* Sound may be used in the interpretation of abstraction with no physical realization, as in Bly's work [25] where data points have spatial and sonic parameters associated with them. Wrap–around sound can be used to augment sight and facilitate depth and direction discrimination when interpreting data in virtual reality [26]. Work at the University of Lowell exploits the "sonification" of textured objects to impart additional perceptual characteristics [27].
- *Sound messages.* These may be *auditory icons* [28], (that is, everyday sounds such as waves breaking on a beach or machines humming) which serve as messages rather than interpretations of data. Alternatively, they may be *earcons* [29] which are hierarchically constructed from building blocks called motives; in studies of turbulence, earcons have proven valuable for communicating information such as changes of state, temperature, speed and energy [30].

Can the auditory and visual modalities be combined in computer–based systems to facilitate the processing of information? Again, the answer appears positive [31, 32]. Nevertheless, our understanding of the use of audio in the interface is still far from complete; cf. [33].

> *WE PROPOSE that efforts be directed by members of the visual programming community towards development of visual languages for defining and controlling tomorrow's multimedia and multimodal environments, which will be characterized by innovative and coordinated use of color, animation, 3–D and sound.*

5 Concluding Remarks

We have discussed three areas where we believe visual languages may make major contributions by the year 2000. To achieve these breakthroughs, complicated software systems will have to be designed and implemented. Unfortunately, each such undertaking must at present essentially begin from scratch, due to a lack of appropriate visual metatools. It is essential that this situation change!

> *IT IS URGENT that members of the visual programming community pool their forces to set standards and develop high–level visual metatools which will facilitate creation and maintenance of tomorrow's visual tools and programming environments.*

References

[1] L. Wittgenstein. *Tractatus Logico–Philosophicus (English translation by F. P. Ramsey).* Routledge and Kegan Paul, Ltd., London, 1922.

[2] E. A. Abbott. *Flatland: A Romance of Many Dimensions.* Dover, New York, NY, 1952.

[3] C. Ware. Visual Information Display. Unpublished panel position paper prepared for INTERACT'90, the 3rd IFIP Conference on Human–Computer Interaction, Cambridge (UK), August 27-31, 1990.

[4] C. J. Shaw. Assemble or Compile? *Datamation*, 10(6):59–62, September 1966.

[5] J. H. Larkin and H. A. Simon. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1):65–99, 1987.

[6] E. P. Glinert, editor. *Tutorial—Visual Programming Environments; Volume 1: Paradigms and Systems; Volume 2: Applications and Issues*. IEEE Computer Society Press, Washington, DC, 1990.

[7] K. M. Kahn and V. A. Saraswat. Complete Visualizations of Concurrent Programs and their Executions. In *Proc. 1990 IEEE Worksho, on Visual Languages (VL'90)*, Skokie, pages 7–15, October 4-6, 1990.

[8] C. D. Norton and E. P. Glinert. A Visual Environment for Designing and Simulating Execution of Processor Arrays. In *Proc. 1990 IEEE Workshop on Visual Languages (VL'90)*, Skokie, pages 227–232, October 4-6, 1990.

[9] J. C. Browne. *Software Engineering of Parallel Programs in a Computationally Oriented Display Environment*. The MIT Press, Cambridge, MA, 1990.

[10] N. Carriero and D. Gelernter. How to Write Parallel Programs: A Guide to the Perplexed. *ACM Computing Surveys*, 21(3):323–257, September 1988.

[11] S. A. Williams. *Programming Models for Parallel Systems*. John Wiley & Sons, Chichester (UK), 1990.

[12] A. R. Vener and E. P. Glinert. MAGNEX: A Text Editor for the Visually Impaired. In *Proc. 16th Annual ACM Computer Science Conference*, Atlanta, pages 402–407, February 23-25, 1988.

[13] E. S. Helfman. *Blissymbolics: Speaking Without Speech*. Elsevier Nelson, New York, NY, 1981.

[14] A. Shalit, D. A. Boonzaier, L. G. Underhill and R. Williams-Short. Multidimensional Scaling of Blissymbolics Data. Interim Research Report, Dept. of Biomedical Engineering, Univ. of Cape Town, 1988.

[15] A. D. N. Edwards, S. Brummel, G. Vanderheiden and R. C. Williges. Call for Participation, CHI'91 Workshop on Human–Computer Interaction and Users with Special Needs (e–mail distribution), 1991.

[16] M. M. Blattner and R. Dannenberg. Editors' Foreword to *Interactive Multimedia Computing*. ACM Press, 1992 (to appear).

[17] G. F. McCleary Jr. An Effective Graphic "Vocabulary". *IEEE Computer Graphics and Applications*, 3(2):46–53, March/April 1983.

[18] J. MacKinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Trans. on Graphics*, 5(2):110–141, April 1986.

[19] R. C. Carter. Visual Search with Color. *J. Experimental Psychology: Human Perception and Performance*, 8(1):127–136, 1982.

[20] F. L. van Nes, J. F. Juola and R. J. A. M. Moonen. Attraction and Distraction by Text Colors on Displays. In *Proc. 2nd IFIP Conf. on Human–Computer Interaction (INTERACT'87)*, Stuttgart, pages 625–630, September 1-4, 1987.

[21] M. H. Brown. Exploring Algorithms using BALSA–II. *IEEE Computer*, 21(5):14–36, May 1988.

[22] J. T. Stasko. TANGO: A Framework and System for Algorithm Animation. *IEEE Computer*, 23(9):27–39, September 1990.

[23] R. Baecker, I. Small, and R. Mander. Bringing Icons to Life. In *Conf. Proc., CHI'91: Human Factors in Computing Systems*, New Orleans, pages 1–6, April 27-May 2, 1991.

[24] B. Laurel, T. Oren, and A. Don. Issues in Multimedia Interface Design: Media Integration and Interface Agents. In *Conf. Proc., CHI'90: Human Factors in Computing Systems*, Seattle, pages 133–139, April 1-5, 1990.

[25] S. A. Bly. Presenting Information in Sound. In *Conf. Proc.: Human Factors in Computer Systems*, Gaithersburg, pages 371–375, March 15-17, 1982.

[26] E. M. Wenzel. Three–Dimensional Virtual Acoustic Displays. In *Interactive Multimedia Computing (M.M. Blattner and R. Dannenberg, editors)*, Chapter 15. ACM Press, 1992 (in press).

[27] S. Smith, R. D. Bergeron and G. Grinstein. Stereophonic and Surface Sound Generation for Exploratory Data Analysis. In *Conf. Proc., CHI'90: Human Factors in Computer Systems*, Seattle, pages 125–132, April 1-5, 1990.

[28] W. W. Gaver. The SonicFinder: An Interface That Uses Auditory Icons. *Human–Computer Interaction*, 4(1):67-94, 1989.

[29] M. M. Blattner, D. A. Sumikawa and R. M. Greenberg. Earcons and Icons: Their Structure and Common Design Principles. *Human–Computer Interaction*, 4(1):11–44, 1989.

[30] M. M. Blattner, R. M. Greenberg and M. Kamegai. Listening to Turbulence: An Example of Scientific Audiolization. In *Interactive Multimedia Computing (M.M. Blattner and R. Dannenberg, editors)*, Chapter 11. ACM Press, 1992 (in press).

[31] M. L. Brown, S. L. Newsome and E. P. Glinert. An Experiment into the Use of Auditory Cues to Reduce Visual Workload. In *Conf. Proc., CHI'89: Human Factors in Computing Systems*, Austin, pages 339–346, April 30-May 4, 1989.

[32] W. W. Gaver, R. B. Smith and T. O'Shea. Effective Sounds in Complex Systems: The ARKola Experiment. In *Conf. Proc., CHI'91: Human Factors in Computing Systems*, New Orleans, pages 85–90, April 27-May 2, 1991.

[33] M. M. Blattner, G. Grinstein, E. P. Glinert, W. Hill, C. Levit and S. Smith. Multimedia Environments for Scientists (panel position paper). To appear in *Proc. VISUALIZATION'91*, San Diego, October 23-26, 1991.

END

DATE
FILMED
12/17/91