Mathematics 9 pages

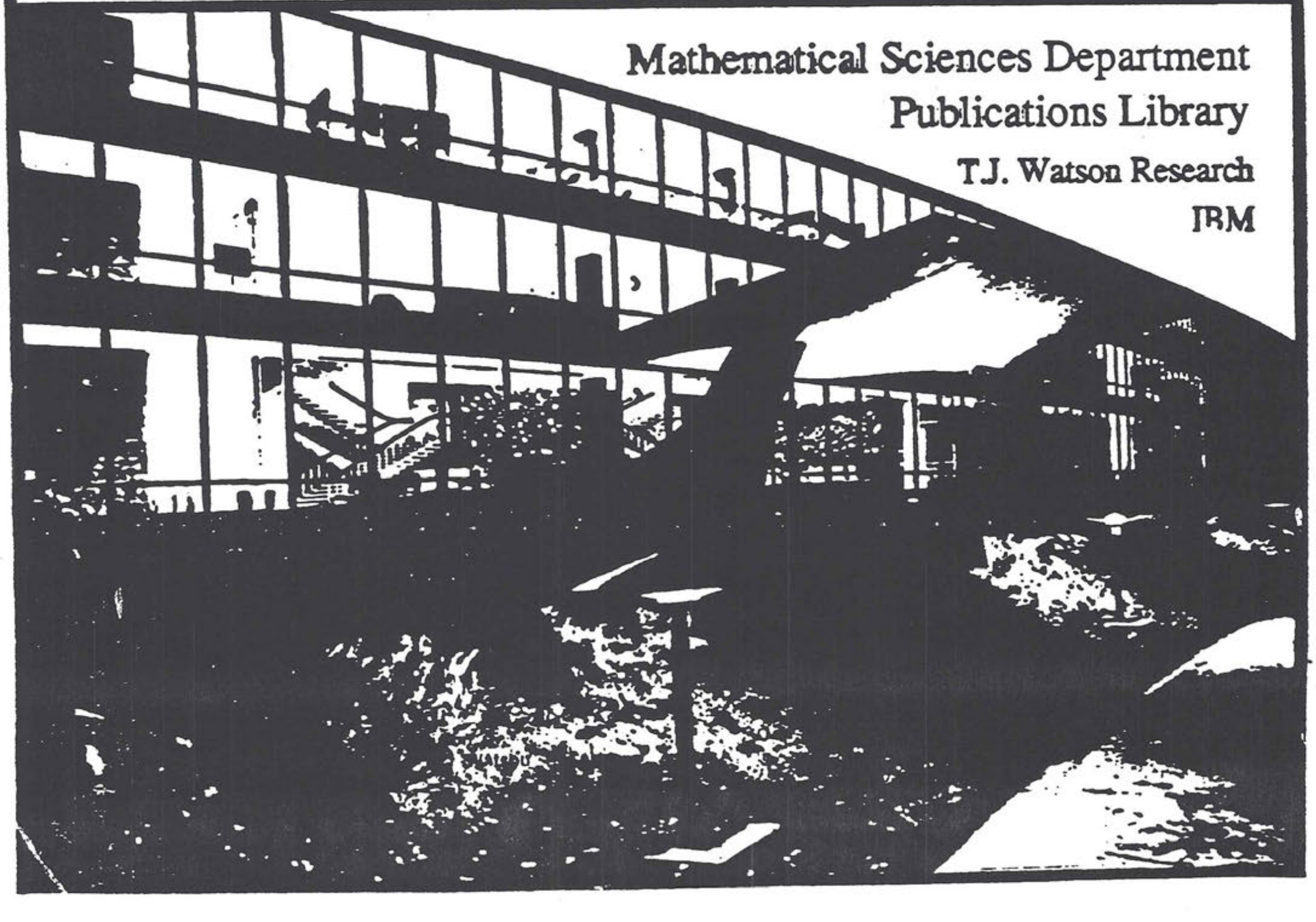
Published

Research Report

The Visual Elements Workspace (VIEW) Scenario

Ted Selker

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598



The VIsual Elements Workspace (VIEW) Scenario

Ted Selker

IBM Research
T. J. Watson Research Center
Yorktown Heights, NY 10598

VIsual Elements Workspace (VIEW) can be used as Hypertext for learning about Visual Language and a construction tool for building them. This paper describes the scenario by which a VIEW user interactively builds a working visual language and parser.

A formal definition of a visual language guides a user in building a working visual language interface. A user drives VIEW using a mouse and keyboard to define a visual language.

The user scenario for defining a language starts by defining a set of visual primitives or visual alphabet. Building this alphabet consists of selecting from pre-built primitives or creating new images, icons, symbols or gestures. These primitives will later be put together into visual utterances. The user selects an input syntax to describe how alphabetic items can be placed or moved in the visual language. This input syntax can consist of pointing, gesture or keyboard actions. A parsing syntax is chosen to define legal utterances. Parsing syntax depends on position, size, time, or other spatial and temporal properties. The definition of semantics is beyond the scope of the visual language definition; VIEW allows a user to associate a predefined semantics with the language or write a script of their own.

1. Introduction

Modern computer systems use visual interface techniques to enhance or replace text. Efforts to demonstrate the utility of visual interface techniques have more often taken the form of systems rather than scientific analysis. GRAIL [1] was built in the early 1960's to demonstrate the possibility of using interactive graphics to work with a computer. Systems such as the popular Apple Macintosh [2], modeled after the Xerox Star [3], use spatially organized screens with icons to represent objects and actions. The success of electronic spreadsheet programs [4] has been attributed in part to their spatial presentation of information.

However, spatial presentation of information by itself doesn't solve all problems. The appearance of icons, graphics and other visual language techniques on computer screens does not automatically improve interfaces any more than the printing press assured that written material would be worthy of reading. One behavioral experiment [5] found that users' performance and preference differences among seven interfaces were not determined by whether the interface style was iconic, menu-driven or verbal command language. The authors concluded that careful design is more important than interface style.

A prerequisite for careful design is to understand the range of techniques available and the reasons for choosing one technique over another. Efforts have been made to apply the principles of graphic design to visual user interfaces. Tutorials, such as ones given by Aaron Marcus at the ACM CHI conferences, have put forward these design principles [6]. Mackinlay's thesis demonstrates automated choice of visual presentation based on the structure and kind of information being viewed [7]. Another recent paper describes the set of different design constraints involved in designing visual interfaces [8].

It would be nice to have a translator which would interpret utterances in visual languages in an interface as we do for a programming language. Fred Lakin's VMACS visual parser does just that [9] for several special cases. It allows shapes to be interpreted as meaningful language elements in a visual grammar. VMACS has special parsers for at least 7 different visual languages.

Defining such visual languages and their parsers might be simplified with tools. The elements of visual language can be decomposed structurally [10]: visual languages are made up of an alphabet with input syntax, a parsing syntax and semantics.

This paper describes a system which operationalizes our formal definition of the elements of visual language [10]. We use the elements of visual language as the basis for a system for designing working visual languages.

2. USING VIEW

The VIEW system uses a direct manipulation [11] scenario for building visual a language with an associated interpreter. As well, the system allows a user to assign semantics to the language. Using a mouse to draw and make menu selections, a user defines input syntax, alphabet, parsing syntax and semantics for a language. The keyboard is used to type symbols and define new semantics for languages.

This section sketches such a path through the VIEW system. Part of a language definition session is described with the goal of giving a feel for the scenario.

The VIEW scenario begins with a circular menu (Figure 1). This menu directs a user to try out pre-built visual languages, learn about elements of visual language or build one themselves.

2.2 Build a Visual Language.

To *build a visual language*, a user follows menus to work on the imagery used in the alphabetic items (Figure 3, Figure 4)

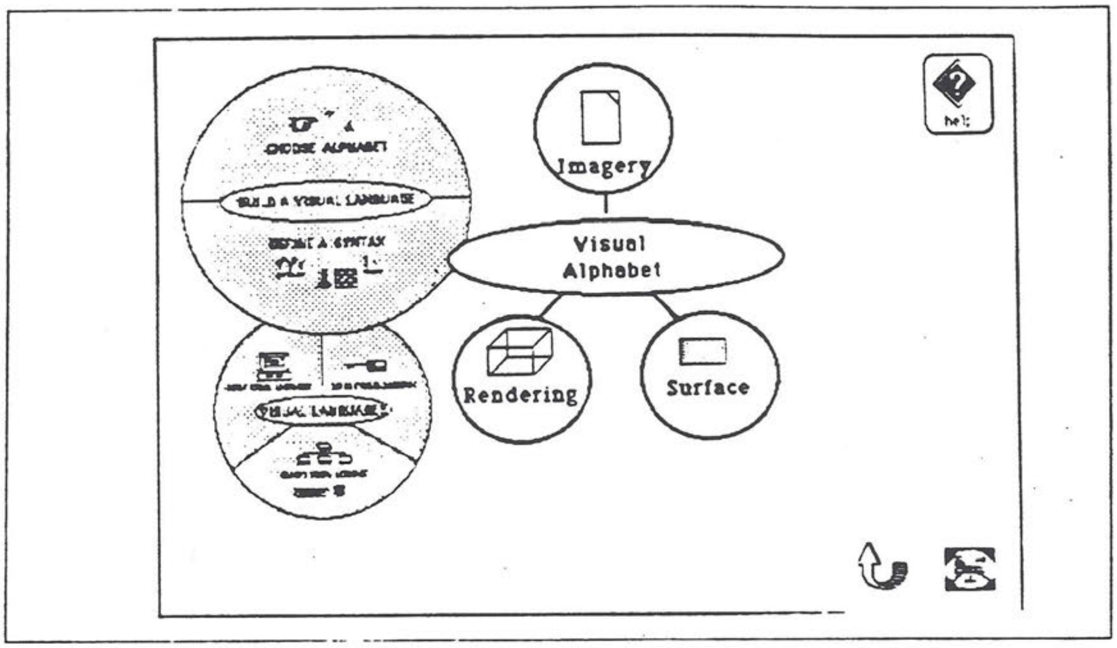


Figure 3. Visual Alphabet.

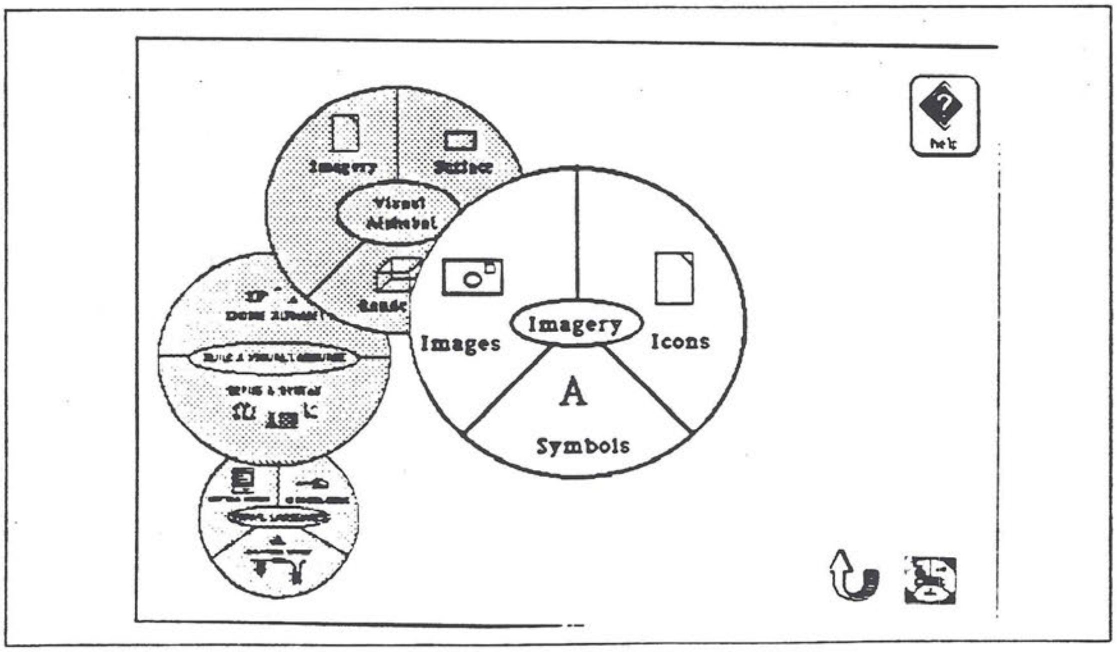


Figure 4. Alphabet Imagery.

The user may decide to make the language using icons chosen from an icon library. Alternatively, a drawing subsystem allows a user to draw their own icons. A command line allows a user to type in symbols, images and gestures can be included in alphabets as well.

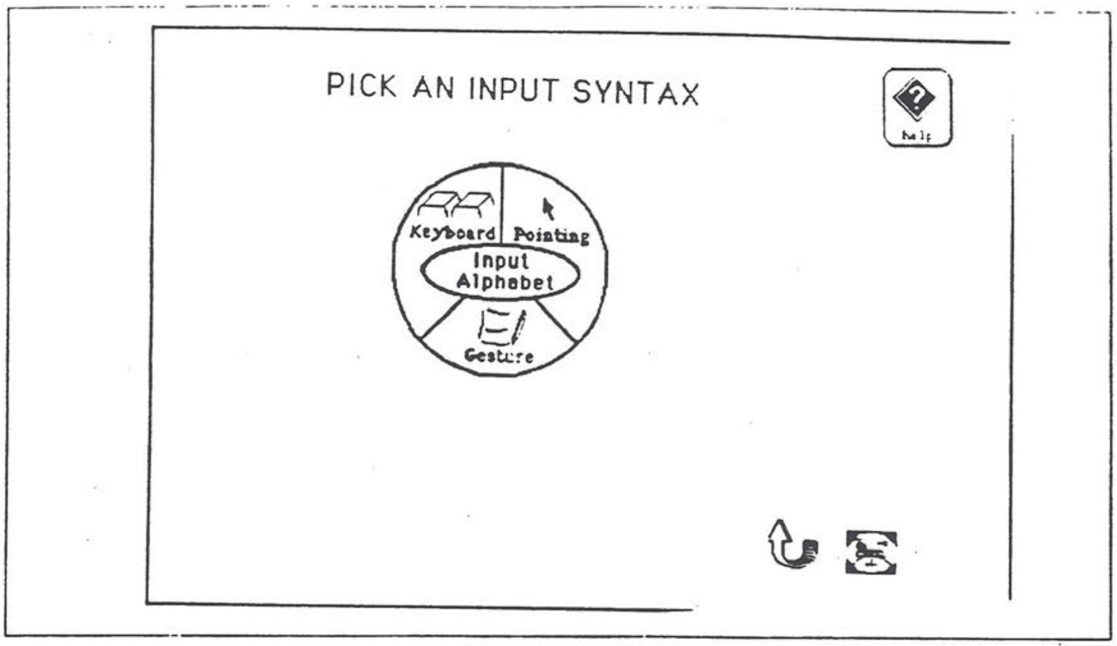


Figure 6. Choosing Input Syntax.

The input syntax circular menus are guides for ways of entering things with a visual language: typing, pointing or gesturing (Figure 7). Choosing the pick sector selects it and returns to the control panel showing the selection as shown in Figure 5.

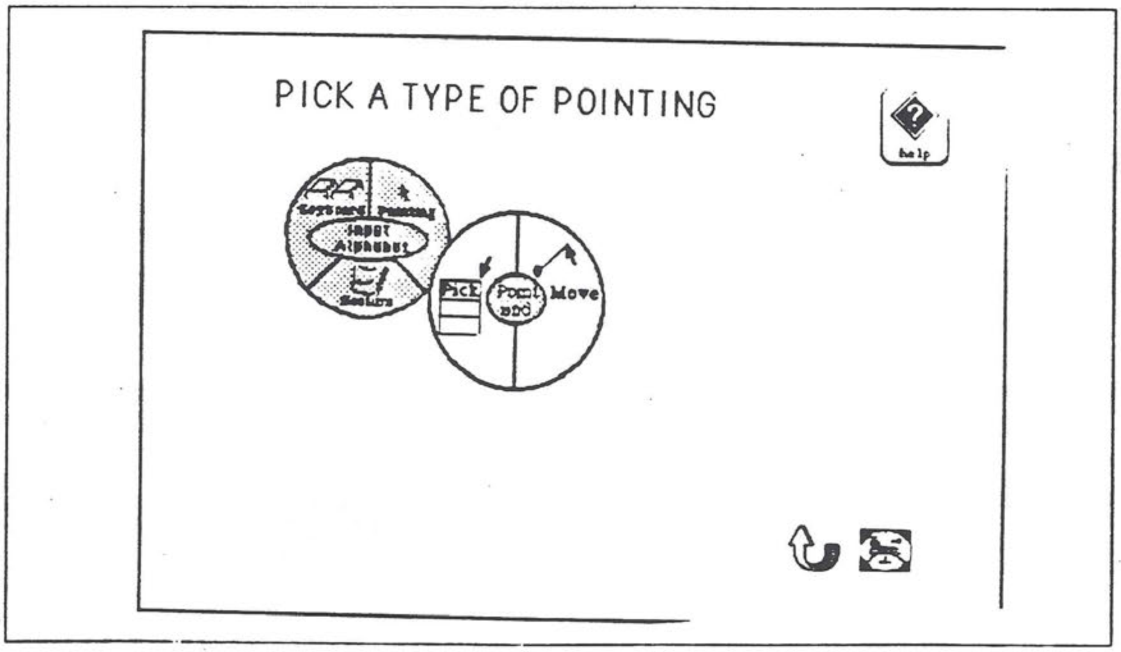


Figure 7. Input Syntax.

Pressing the "try language out" button takes the user to a workspace where they can experiment with utterances in their new language. If the language is a positional denoted language and has icons with point and move input syntax, the user can place them on the workspace and connect some of them. The computer can tell when tokens are connected. With music semantics the

We have sketched the process of using VIEW. The above tour of VIEW is illustrative, not complete. The system has over 200 screens that a user can visit. The process of selecting, drawing or capturing icons has not been shown, nor have the details of adding rendering and texture to alphabetic items. For a more filled in scenerio, we invite you to see the video tape [12] or a live demo.

3. DISCUSSION

The above scenario works. A user can readily use VIEW to learn about the elements of visual language. A user can try out pre-built visual languages or build and play with a visual language.

One goal of VIEW is to demonstrate structural issues of building a visual language to user interface designers. By constructing a few visual languages using VIEW, a user will learn the set of components of a visual language and (we believe) will be in a better position to make visual language design decisions.

The system has many limitations. Real user interfaces use many visual languages for many purposes. A user often confronts several input syntaxes on one screen. The user might be able to choose from a menu (*pick* input syntax), move something with a cursor (*point* and move input syntax) or select something and change it with keystrokes on the keyboard (*symbol* input syntax). Different visual languages on the screen might have different parsing syntax as well, position of text or data might matter (*metrical* or *positional* syntaxes), whether windows are overlapping might matter (*positional interacting* syntax), whether an item was responded to in 30 seconds might matter (*temporal* syntax), etc.. These many visual languages coexist. VIEW allows users to make visual languages with different parsing syntaxes on different parts of the screen. The relationships between visual languages in most computer interfaces is complex. VIEW is not currently set up to work with relationships between visual languages.

It should be possible to integrate visual languages as alphabetic elements in other visual languages. For example, a slider should be able to be put into a spreadsheet, a menu in a text editor. VIEW demonstrates examples of this but does not yet include natural integration of languages with their associated parsing syntax into other languages.

For a language to be useful it needs semantics. We have provided a set of default actions visual utterances can make happen. A VIEW user can choose from these semantics. We also include a way for a user to interactively write script semantics for a language. Attaching semantics to a visual language is an important part of building a language that does something. A system which allows a user to build a substantial program by visually creating the user interface language and defining semantics, will be an impressive contribution to program development environments.

VIEW is a prototype, built on top of a hypertext system not necessarily designed for such an ambitious exercise [13]. The bugs and edges make it like so many other research systems (some sold as products), at times, throwing the user into the debugger. With use, development, and improvement of the hypertext system it rests on, VIEW will be exportable.

Bibliography

[1] T. O. Ellis and W. L. Sibley, The Grail Project verbal and film presentation, 1966.

[2] Carol Kaehler, *Macintosh* Apple Computer, Inc., 1983.

[3] W. L. Bewley, T. L. Roberts, D. Schroit, and W. L. Verplank, "Human Factors Testing in the Design of Xerox's 8010 "Star" Office Workstation," *CHI '83 Proceedings*, pp. 72-77, 1983.

[4] V. Wolverton, *VisiCalc* Visi Corp., 1983.

[5] John Whiteside, Sandra Jones, Paula S. Levy, and Dennis Wixon, "User Performance with Command, Menu, and Iconic Interfaces," *CHI 85 Proceedings*, ACM NY, 1985..

[6] A. Marcus, *Tutorial 18: User Interface Screen Design and Color* ACM/SIGCHI, 1986.

[7] J. Mackinlay, *Automatic Design of Graphical Presentations*, PhD thesis, Stanford University, 1986

[8] Ted Selker, Catherine Wolf, and Larry Koved, "A Framework for Comparing Systems with Visual Interfaces," *Interact '87 Proceedings*, pp. 683-688, North Holland Amsterdam, September 1987.

[9] Fred Lakin, "Visual Grammars for Visual Languages," *AAAI '87 Proceedings, Vol. 2*, pp. 683-688, 1987.

[10] Ted Selker and Larry Koved, "Elements of Visual Language," *1988 IEEE Workshop On Visual Languages*, Computer Society Press, October 1988.

[11] Ben Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," *IEEE Computer*, no. August, pp. 57-69, 1983.

[12] T. Selker, The Visual Elements Workspace (VIEW) Video Tape, 1990.

[13] Bill Atkinson, *The hypertalk System Macintosh* Apple Computer, Inc., 1987.

[14] S. L. Smith and J. N. Mosier, "The user interface to computer-based information systems: A survey of current software design practice.," *Behavior and Information Technology*, pp. 195-204, July 1984.