Elements of Visual Language

Ted Selker Larry Koved

Thomas J. Watson Research Center
IBM Research Division
P.O.Box 218
Yorktown Heights, N.Y. 10598

ABSTRACT

Visual language is the systematic use of visual presentation (graphic) techniques to convey meaning. This paper proposes a structural classification and vocabulary for these languages. Visual languages, like verbal languages, are defined by a grammar and semantics. This paper defines the visual grammar, the elements of visual language.

Visual elements are composed of:

visual alphabet	a set of visual primitives used in a visual language
visual syntax	compositions of primitives to form visual statements
interaction	user to system communication
structure	rules combining sublanguages into a language

The classification of the visual elements can be viewed as a linguistic description of visual language.

INTRODUCTION

A programmer writing an application can not avoid being influenced by knowledge of icons, menus, and computer graphics. Should their programs be menu driven? Should all objects be represented by icons? Exposure to design examples is useful to the extent that we know how and when to employ them in user interfaces.

We need a framework for analyzing interactive systems with graphic interfaces. Some questions that can be asked include: How does the user interface reveal the underlying semantics of the program? How does the interface compare to other program interfaces? How does the interface affect the usability of the program? What other kinds of programs could use such an interface? Each of these questions is important. But, first, it is important to consider the graphic elements, or building blocks, from which the visual interface is constructed.

Graphic design provides a set of tools for creating depictive text and graphics on paper. Several books have been written as guides for designing graphic presentations [15, 39]. We should draw from such work to codify and extend graphic design for interactive systems. People in fields ranging from semiotics to sociology, have begun the process by cataloging techniques and describing the value and power of visual languages [3, 7, 8, 13, 16, 20, 21, 24, 34]. From these works, this paper refines the concept of visual elements in interactive computing.

BACKGROUND

Definitions of visual language have taken many approaches. Some have taken a formal mathematical approach to defining visual language [12]. Mackinlay used first order logic to define a formalism for working with presentation graphics [21]. The notable works of Card and Reisner use task action grammars to describe interactive systems [6, 27]. Such formal definitions of visual grammars do not necessarily afford a working definition or feeling for visual language.

Myers defined an operational approach to classifying visual languages directed at describing programming power and expressiveness [24]. The classification segments visual interfaces as supporting:

- programming-by-example or not
- batch or interactive style interaction
- visual programming or not
- visualizing static or dynamic data.

Shu's classification covers much of visual languages along three broad dimensions [34]:

- visual extent
- scope
- language level

Chang's tutorial draws from a broad range of sources including Myers and Shu [8]. Chang segments visual languages into:

- languages that support visual interaction
- visual programming languages
- visual information processing languages
- iconic visual information processing languages

The tutorial describes a number of systems that fit into these categories.

Selker, et al, built a framework for comparing systems with visual interfaces by contrasting [31]:

- the visual elements used in an interface
- their domain of application
- the relationship between the interface and system functionality
- the expressive power of the interface
- the interaction style and mode
- and the visual interface's relationship to other visual interfaces

The framework serves as an umbrella for describing interactive systems with visual interfaces. This paper provides a definition of the elements of visual language. The elements consist of the alphabet, syntax, interaction paradigm, structure and coverage of the language. The visual elements construct a language that the other parts of the framework elaborate.

VISUAL ALPHABET

Visual language is the systematic use of visual presentation techniques to convey meaning. A visual alphabet is the set of spatial, surface and temporal techniques used in a visual language. Iconic systems [8] and visual objects [23] define similar distinctions. Several studies have classified visual output techniques [4, 13, 20]. Techniques for producing visual alphabets include:

- *Images* are derived from photos, digitization or projections. Chang and Lodding describe representational imagery as images derived from photographs, degraded photographs or drawings [8, 20]. Fred Lakin's "visual parser" allows shapes to be interpreted as meaningful language elements in a visual grammar [19].
- *Icons* are symbols with physical world referents [4, 8, 13, 16, 18]. Icons are stylized representations of objects or processes. For example, the fork and knife icon connotes a place to eat.
- *Symbols* are drawings of arbitrary design [20]. The character "a" is an example of a symbol. These are labels with no a priori referent.
- Surface Characteristics include the perceptually separable and salient dimensions of color: hue, saturation, value and texture [22].
  - *Hue, saturation and value* are defining dimensions for color. These are often used to convey information: stop and caution are typically depicted by red and yellow respectively.
  - *Texture* can also convey information. It is often used to differentiate obscured items in a menu or windows on a screen. Textures on computer interfaces, like textures on wall paper, are composed of images icons or symbols (see Figure 5).

    Halftoning is a process that uses dichromatic texture to recreate gray-scale images. A grid of varying size dots gives the illusion of a gray-scale image (see Figure 1).

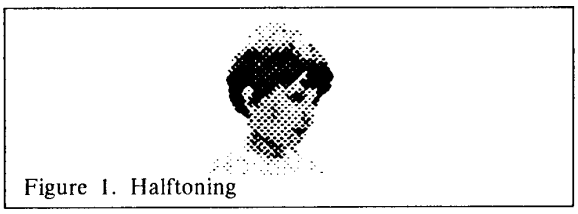
Figure 1. Halftoning

- Rendering is the technology and approach used to present a visual language. Techniques for rendering include drawing with with vectors or polygons. Layout languages such as Press [35], PostScript [2] and ACM/SIGGRAPH "Core System" [1] are languages for rendering. Hooper demonstrates differences in various presentation techniques for similar imagery [14].

VISUAL SYNTAX

The visual techniques described thus far are combined into visual statements by using a syntax. It is evocative to draw parallels between visual syntax and natural language syntax. However, the inherent serial nature of natural language relative to the inherently spatial structure of visual language makes the notion of parsing and statements quite different. Natural language relies on articles, prepositions and punctuation for delineating structure and context. Visual language, on the other hand, quite naturally uses spatial structure to represent the context in which information is interpreted. Linguistic terms are used in this paper solely as points of reference.

Visual parsing distinctions can be separated into syntactical categories:

- positional
- size
- time
- rule

Positional syntax includes situations in which spatial relationships between visual elements confer information to the observer. These situations include:

- **Relative Positional** relationships:
  - *Sequential* - Spatial sequence of objects denote an order in which visual elements should be considered. Written languages use sequential syntax. Words follow each other left to right and top to bottom. Sequential languages can use space differently. Cartoons frames are often arranged on a line or column (see Figure 2).

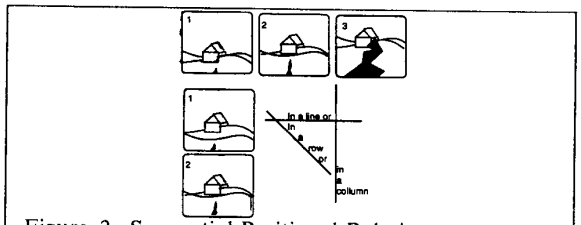

Figure 2. Sequential Positional Relative

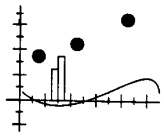
Figure 3. Metrical Positional Relative

- *Metrical* - Measurement is used to denote relative value. Plotting a function uses metrical syntax. Plots using polar, cartesian, projected 3d and logarithmic scales are examples of metric coordinate spaces. A metrical syntax could use position on a B-spline as its metric. Graphs and bar charts use a cartesian coordinate system to relate two dimensions of information (see Figure 3).
- *Orientation* - The relative angular relationship between visual objects. For three dimensional solids, the rotation presents a view of objects or a scene. A rotation of 180 degrees along the horizontal or vertical axis would result in a view of the far side (rear) of the object. For an analog clock, the orientation of the hands with respect to the dial is the indication of the time of day.

- **Interacting Positional** relationships:
  - *Embedded* - spatial enclosure or geometric containment. Examples include text balloons in cartoons (see Figure 4). VennLisp is a programming language using an embedded notation [19]. A program is defined by positioning hollow language key symbols inside of each other. Evaluation in VennLisp proceeds from the outside of the diagram inward.
  - *Intersecting* - partial or total spatial overlap of visual objects. In electrical circuit diagrams, the intersection of straight lines at right angles indicates continuity in the circuit. Venn diagrams use intersecting regions, as well as embedded syntax, to express relationships (see Figure 5). Overlapping window systems often use intersection (occlusion) syntax to define which windows are active.

Figure 4. Embed Positional Interacting Syntax

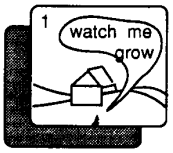

Figure 5. Intersecting Positional Interacting Syntax.

  - *Shape* - Spatial relationships, like those found in puzzles, allow syntax to prescribe how things can be connected together (see Figure 6). Learning Research and Development Center's Bridge system has pieces which fit together like a puzzle to make programs.

- **Positional Denoted** relationships:
  Specifiers, such as lines or labels, indicate relationships between objects.
  - *Connected* - Arcs specify connectivity between visual objects. These connections specify semantic information. Node/arc notations are used in computer flow charts, electrical wiring diagrams, organization charts, etc. Stella is a visual programming language that allows a user to draw several styles of nodes and arcs to specify a program [28].
  - *Labeled* - Spatial relationships can be maintained by symbols. In large circuit diagrams, spatial relationships are not always seen on a single drawing. Labeled wires are used to refer to other drawings (see Figure 7).

Size - Relative size of objects influences how and when component items are interpreted (parsing order). Catalogs often contain large pictures of expensive objects, smaller pictures for less expensive. Ancient petroglyph languages sometimes use size as the organizing feature. Importance of people and things are expressed by relative size on cave walls.

Temporal - Many visual languages use time as an organizing dimension. Computer drawing programs use temporal information to affect meaning in several ways. Recency can be used to determine the order of spatial overlapping. Blinking draws the attention of the reader. Most mouse oriented scenarios present some functionality through temporal actions, such as pressing the mouse button twice to start an application.

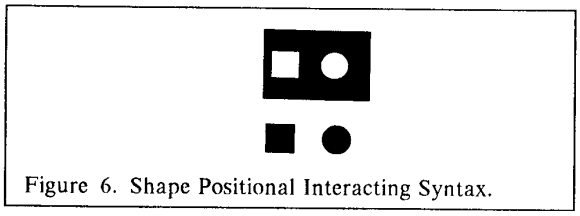

Figure 6. Shape Positional Interacting Syntax.

Rule - An algorithmically described visual relationship. Cursors in text editing environments are often designed to stop at the end of a line, wrap around to the next line or to the top of the screen.

Visual syntaxes can be combined in relationships using constraints JUNO [26] and ThingLab [5] use "constraint" languages to define the spatial relationships between displayed objects. Things can "attract" nearby objects.

The Gauges system uses rules that are not two way constraints [36]. The system updates a gauge when a variable changes, moving a gauge needle does not change the variable it represents.

Peridot depends heavily on rules [25]. Peridot defines these as two one way relationships rather than expecting them to be symmetric constraints.

INTERACTION

To talk about interactive systems, we must describe the input techniques and their relationship to output techniques. By gaining insight into the interaction process, we can learn how the input, application (semantics) and visual language (output) are interdependent.

- *Feedback* - A system can be open loop or depend on feedback to support interaction. An open loop system might simply delete a file in response to a user issuing the erase command without indication of success. At the other extreme of the continuum is the immediate response to mouse movements or a text editor that responds to each keystroke.
- *Interactivity abstraction* - A mapping exists from the user's action to the system's response. One end of this continuum is a transparent mapping between user action and system response. A relatively direct correspondence between action and response exists when a person is using a screen based editor. The computer echoes the symbols the person types and moves the cursor around the screen when the arrow keys are pressed. The other end of the continuum is described by a discontinuity between user actions and system responses. This is typified by batch and command line interfaces. An indirect correspondence exists in a text editor when a person types on a command line and the computer responds with a message elsewhere on the screen.

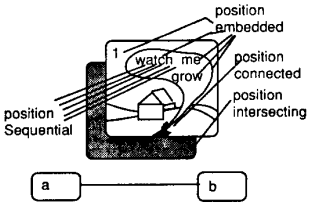

Figure 7. Labeled Positional Denoted

- *Response time* defines how quickly the system responds to the user. Traditional operating system literature distinguishes Real-time Interactive-time and Batch-time as definitions for computer response. Response time can vary from being temporally indistinguishable to temporally distant from an action. Fast response in a time-shared computing environment makes a large difference on productivity of individuals [9].

Prior discussions of interaction have ranged from interactive vs. batch [24] to the use of direct manipulation [32]. Our classification defines "direct manipulation" as *closed-loop* feedback, *transparent mapping* between action and consequence, and *temporally indistinguishable* response time. Rouse discusses issues of appropriate feedback and response in some detail [29].

Input paradigm is the sequence of actions for communication from a user to the system. The visual output techniques described above need to be matched by some set of input techniques. These techniques include selection, gesture and shape interpretation.

- *Keyboard Entry* includes text entry and menu selection. The keyboard entry device may be similar to a typewriter and includes variants such as chord keyboards [11].
- *Point and Pick* is a menu selection paradigm. Selection can be made with a mouse, joy stick, digitizing tablet, touch screen or other direct manipulation technique. Menus allow a user to select items from a list, labeled buttons, a set of pictures, or structured spatial display [33].
- *Point and Move* is a paradigm where an object is selected and moved. Many windowing systems allow the user to point to an object and drag it to a new location on the display. This technique was first demonstrated in the Sketchpad system [37]. Many systems (video games, CAD systems, etc.) use gesture, shape and temporal movement to dictate the "parse" of an image.
- *Point and Draw* is the general drawing technique which leaves a mark wherever the cursor has been. MacPaint relies heavily on this technique [17]. Grail was the first system to demonstrate the use of drawing in an interface [10].

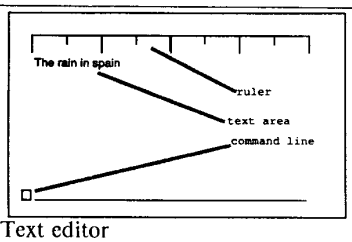

Figure 8. Text editor

- *Gesture Interpretation* is the use of motion through space and time. Some mouse drivers (i.e., X-windows [30]) use the speed with which the mouse is moved to accelerate cursor movement on the screen. Handwriting recognition systems can use gesture recognition to interpret characters and editing commands [38, 40, 41].[1]

STRUCTURE AND COVERAGE OF VISUAL LANGUAGES

We tend to think of a visual interfaces as homogenous. In reality, these interfaces are composed of heterogeneous interacting visual languages. Menus are often augmented with text entry or other direct manipulation techniques. Recognizing how languages are used for different purposes in a visual interface is part of visual language design. Use of separate visual languages in a system can segment function, system structure, type of manipulation or perspective on information.

Structure pertains to the segmentation of activities and commands. A typical text editor has several visual languages in its interface (see Figure 8). The text area uses direct manipulation for positioning characters, words and blocks of words with a cursor. A ruler at the top of the window (borrowed from the typewriter) changes its appearance as a user types in the text area. A set of commands can be accessed by a menu or via a command line.

The highly moded structure of the editor can be held in contrast to the program interpreter for the language BASIC. BASIC has only one visual language, set of commands and functionality in its interface.

Coverage of the language is described by Shu as one of three dimensions for classifying visual languages [34]. Coverage ranges from special to general purpose. A special language might be a notation for describing organic chemical bonds and only describes a subset of a system. Conversely, a calculator usually shows all of its functionality through a single language and interface.

In contrast are the instruments in the LOOPS system. Its structured icons show all of the functionality of the system. By choosing various parts of the icon, a user accesses various subsets of its capabilities (see Figure 9).

[1] Handwriting can also be analyzed through image interpretation techniques.

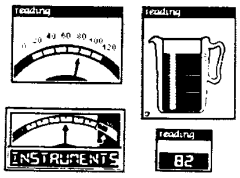

Figure 9. Instruments

An instrument is chosen to view and change a variable in a program. When an instrument, such as the pitcher, is chosen, the coverage of the instrument is one variable, not the whole program.

SUMMARY

This paper is a piece of the framework of issues that we believe should be considered when analyzing or designing a visual language [31]. By separating out elements of visual language, this paper gives a framework for defining a visual language. We have made the following segmentation of visual language:

Visual Alphabet	- building blocks of a visual language
Images	- photos
Icons	- caricatures
Symbols	- designs
Surface	- **Color, Texture**
Rendering	- presentation technique
Visual Syntax	- creating graphic statements
Positional	- spatial graphic relationships
Relative	- **Sequential, Metrical, Orientation**
Interacting	- **Embedded, Intersecting, Shape**
Denoted	- **Connected, Labeled**
Size	- size graphic relationships
Temporal	- time graphic relationships
Rule	- defined graphic relationships
Interaction	- relationship of user input to system's output
Feedback	- system response
Abstraction	- transparency of action-response mapping
Response time	
Input paradigm	- **keyboard, point & pick, point & move, point & draw, gesture**
Structure	- graphic sublanguages
Coverage	- specificness of graphic language

These lend structure to techniques available for creating graphic computer interfaces. We believe that our delineation of visual elements can help in the process of creating and evaluating visual languages. Decomposition of parts of visual language is essential for building a visual linguistics. We hope that our structure of visual elements will help in that process.

REFERENCES

1. ACM, editor. "Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH," in ACM, editor, *Computer Graphics*, vol. 11, no. 3, Fall 1977.

2. Adobe, *PostScript Language Reference Manual* Addison-Wesley Publishing Company, Inc., 1985.

3. Rudolf Arnheim, *Visual Thinking* University of California Press, 1969.

4. Charles Bigelow and Donald Day, "Digital Typography," *Scientific American*, vol. 249, no. 2, pp. 106-119, August 1983.

5. Alan Borning, *Thinglab -- A Constraint-Oriented Simulation Laboratory*, PhD thesis, Stanford University, 1979.

6. Stuart K. Card, Thomas P. Moran, and Allen Newell, *Psychology of Human Computer Interaction* Lawrence Erlbaum Associates, 1983.

7. Shi-Kuo Chang, *Visual Languages* Plenum Press, 1987.

8. Shi-Kuo Chang, "Visual Languages: A Tutorial and Survey," *IEEE Software*, pp. 29-39, 1987.

9. Walter J. Doherty and Bucky Pope, Computing as a Tool for Human Augmentaion, IBM Research, Yorktown Heights, NY, 11622, Jan 1986.

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

11. W. K. English and Doug C. Engelbart, "A Research Center for Augmenting Human Intellect," *Proceedings of the National Computer Conference*, IFIPS, 1968.

12. K. S. Fu, "Languages for Visual Information Description," *1984 IEEE Computer Society Workshop on Visual Languages*, pp. 222-231, December 1984.

13. David Gittins, "Icon-based human-computer interaction," *International Journal Man-Machine Studies*, vol. 24, pp. 519-543, Academic Press, London, 1986.

14. Kristinal Hooper, Experimental Mapping The Perceptual Representation of Environments, University of California, Santa Cruz Technical Report, 1982.

15. Darrell Huff, *How to Lie With Statistics* Norton, 1954.

16. W.H. Huggins and Doris R. Entwisle, *Iconic Communication* Johns Hopkins University Press, 1974.

17. Carol Kaehler, *MacPaint* Apple Computer, Inc., 1983.

18. Robert R. Korfhage and Margaret A. Korfhage, "Criteria for Iconic Languages," in S.K. Chang et al, editor, *Visual Languages*, Plenum Press, 1987.

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

20. K.N. Lodding, "Iconics - A Visual Man-Machine Interface," *Proc. Nat'l Computer Graphics Assoc.*, vol. 1, pp. 221-233, NCGA, Fairfax, VA., 1982.

21. J. Mackinlay, *Automatic Design of Graphical Presentations*, PhD thesis, Stanford University, 1986.

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

23. Fanya S. Montalvo, Diagram Understanding: The Intersection Between Computer Vision and Graphics, MIT AI Lab., Memo 873, November 1985.

24. B. A. Myers, "Visual Programming, Programming by Example, and Program Visualization: A Taxonomy," *CHI '86 Proceedings*, pp. 59-66, 1986.

25. B. A. Myers, "Creating Dynamic Interaction Techniques by Demonstration," *CHI '87 Proceedings*, pp. 271-284, 1987.

26. Greg Nelson, "Juno, a Constrain-Based Graphics System," *ACM SigGraph Proceedings*, vol. 24, 1985.

27. Phyllis Reisner, "Human Factors Studies of Database Querey Languages: A Survey and Assessment.," *Computing Surveys*, vol. 13, March 1983.

28. Barry Richmond, Peter Vescuso, Steve Peterson, and Nancy Maville, *A Buisness Users Guide to Stella* High Performance Systems, 1987.

29. William B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems," *Computing Surveys*, vol. 13, no. 1, March 1981.

30. Robert W. Scheifler, *X Window System Protocol, Version 11* MIT, 1987.

31. Ted Selker, Catherine Wolf, and Larry Koved, "A Framework for Comparing Systems with Visual Interfaces," *Interact '87 Proceedings*, North Holland Amsterdam, September 1987.

32. Ben Shneiderman, *Software Psychology* Little, Brown and Company, 1980.

33. Ben Shneiderman, *Designing the User Interface* Addison-Wesley, 1986.

34. Nan C. Shu, "Visual Programming Lanuages: A Dimensional Analysis," *Proceedings of the International Symposium on New Directions in Computing, Trondheim, Norway*, August 1985.

35. Bob Sproull, William Newman, and Joe Maleson, The Press File Format, Xerox PARC, December 1979.

36. Mark Stefik, *The Loops manual* Xerox PARC, 1985.

37. Ivan E. Sutherland, "Sketchpad: A Man-Machine Graphical Communication System," *Spring Joint Computer Conference Proceedings*, vol. 23, 1963.

38. C. C. Tappert, J. M. Kurtzberg, and Paula S. Levy, Elastic Matching for Handwriten Symbol Recognition, IBM Research, Yorktown Heights, NY, 9988, May 1983..

39. Edward R. Tufte, *The Visual Display of Quantatative Information* Graphics Press, 1983.

40. J. R. Ward and B. Blesser, "Interactive Recognition of Handprinted Characters for Computer Input," *IEEE Computer Graphics and Applications*, vol. 5, no. 9, pp. 24-37, Sept. 1985.

41. Catherine G. Wolf and James R. Rhyne, A Taxonomic Approach To Understanding Direct Manipulation, IBM Research, Yorktown Heights, NY, RC13104, July 1987.