Characterizing Visual Languages

Darrell R. Raymond

Department of Computer Science, University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Abstract

A better understanding of the visual character of languages is important in developing our ability to exploit the human visual system. I briefly outline Goodman's distinction between notational and analog languages, and describe its use in developing the notion of syntactic and semantic density as the defining characteristic of visual languages. Several languages are evaluated for their use of density. I conclude that practical languages are most visually effective when their layout is constrained by an important semantic domain.

1. Introduction.

Visual programming has a credibility problem. As in user interface design, hypertext, and artificial intelligence, solid work in visual programming is often drowned in the ballyhoo of extravagant claims. And as is also true in the other areas, workers in visual programming are not completely without fault. We are generally quite ready to claim, as does Shu:

> Pictures are more powerful than words as a means of communication. They can convey more meaning in a more concise unit of expression.
>
> Pictures aid understanding and remembering.[1]

We are much less eager to publicize Shu's warning:

> Not every picture is a good picture.

The trouble comes when we try to identify *why* pictures should be an aid to meaning or understanding, *how* pictures could be more powerful than words, and *what* constitutes a good picture rather than a bad one. Here is where the claims are put to the test, and too often they have been found wanting. Green,[2] for example, conducted an extensive and thoughtful cognitive analysis of various visual diagramming tools intended to replace flowcharts, including Nassi-Shneiderman, Rothon, Jackson, and Bowles diagrams. His somewhat reluctant recommendation:

> For a reasonably sophisticated readership, try using Bowles's notation and hope that they can learn to read it. For a less sophisticated readership, stick to flowcharts but make heavy use of layout.

In other words, cross your fingers and rely on something other than the formal characteristics of the notation. Other observers have not even been this kind. In 1987 it was possible for Fred Brooks to say:

> A favorite subject for PhD dissertations in software engineering is graphical, or visual, programming... Nothing even convincing, much less exciting, has yet emerged from such efforts. I am persuaded that nothing will.[3]

In addition to adverse opinion, empirical work is appearing that questions the basic assumptions of many visual languages. Data flow languages, for instance, exhibit lower comprehensibility than textual languages for some tasks,[4] and graphic display of data can lead to overconfidence in judgements of accuracy[5] or poor estimates of correlation.[6] Thus asking the why, the what, and the how questions about visual languages becomes a matter of practical import, not just theoretical interest.

We should not hope to develop a complete answer to these questions anytime soon. The dichotomy of picture and word is an enduring theme of philosophy, art, literature, and psychology. Even though we are not obligated to solve a difficult multi-disciplinary problem simply so that we can create better software, something can be gained by considering such questions. Attempting to define more precisely what we mean by visual programming and visual languages can lead us to consider alternatives to current implementations which might not otherwise have been ventured. Better characterization of visual languages also has important pedagogical value in highlighting essential ideas for students and practitioners.

I call for better characterization of visual languages,

and not their formalization. The ideas discussed in this paper conflict with current attempts at formalization, such as iconic sentences and picture grammars. More importantly, I think formalization is best left to a time when we have gained a better informal grasp of visual languages. It is perhaps too adventurous to formalize something that we still find difficult to discuss informally. Since our confidence in the power of visual languages is largely based on our own empirical experiences of the value of informal diagrams and pictures, we should already be accustomed to the notion that informal arguments have value. My call for characterization, then, is a call for better ''visualization'' of visualization itself.

2. Notations and analog representations.

A thoroughly intriguing view of languages and notations is found in Nelson Goodman's *Languages of Art*.[7] Goodman considers a variety of philosophical problems connected to notation. It is possible to make a forgery of a painting, for example, but what of poems? What is the connection between denotation and representation? What difficulties are there in developing a notation for dance? Problems of this sort often turn on the characteristics of the language used to express the artifact. Goodman highlights certain essential characteristics with his distinction between *notational* and *analog* languages.

We begin with some definitions. A *character* is an equivalence class of inscriptions, utterances, or marks which are interchangeable with one another. Our usual notion of character is a specialization of this definition, since Goodman permits compound characters, such as an entire novel or painting. A *compliant* is an equivalence class of objects or ideas whose members are denoted by some character. A compliant is what we are intended to understand when we encounter the character. A *language* is a set of characters and their associated compliance classes. A language is a *notation* if it exhibits the following five properties:

Character indifference. For any inscription X that is deemed to be a member of the language, X must belong to at most one character class. The property of character indifference induces a partition on the set of possible inscriptions.

Finite syntactic differentiation. There must be some finite difference between inscriptions that count as different characters. That is, there is some minimum level of discrimination which can be applied to decide whether a given mark belongs to one character or another.

Non-ambiguous. Any and all inscriptions of a character denote the same compliance class.

Disjoint compliance classes. The set of compliance classes induces a partition on the set of objects or things that can be described by the notation. As a result of this partition, no two characters may refer to the same compliance-class (for if they did, by the property of character indifference they would be the same character).

Finite semantic differentiation. There must be some finite difference between objects or ideas that count as members of different compliance classes. That is, there is some minimum level of discrimination which can be applied to decide whether a given object belongs to one compliance class or another.

The above discussion presents only the essential aspects of the five properties, and many other interesting aspects can be found by consulting Goodman's book. Goodman's properties are necessary but not sufficient for notation. Languages with no characters or only one character, for instance, satisfy them trivially, though they are virtually useless notations. The properties also do not address many other important aspects of symbol schemes, such as restriction to a reasonable number of characters, provision of concise rules for character combination, choice of graphically distinctive characters, the appropriateness of the partitions, or the ease of character duplication and presentation. Each of these aspects and many others are important and essential in producing *good* notations, but the five properties are required for *any* notation.

Non-notational systems are just those systems which fail to satisfy one or more of the five properties. Among non-notational systems are a specific class known as *analog* systems. An analog system violates the second and fifth conditions; neither characters nor compliance classes are finitely distinguishable, but instead are *dense*. Syntactic density implies that between any two characters it is (at least theoretically) possible to find another character; semantic density implies that between any two compliance classes it is (at least theoretically) possible to find another compliance class. Density characterizes analog systems, while discreteness characterizes notations.

We can find several ready instances of analog and notational languages. Any bounded set of integers, for instance, is notational; there is a functional mapping between the character classes (i.e, the numerals) and the compliance classes (i.e., the numbers). Bounded sets of rational numbers, on the other hand, are analog. While there is a mapping between characters and compliance classes, there is no minimum level of differentiation that can be used to partition the compliance classes. No matter which pair of rational numbers we choose, we can always find another rational in between. Similarly, there is no minimum level of differentiation that can be used to partition the characters. A minimum level of differentiation implies a bounded number of digits, but we can always find an infinite number of rationals which differ only beyond that bound.

The line between notational and analog representation

can be crossed in a more subtle fashion. For example, a clock face with markings may be either a notational or an analog device, depending on whether the spaces between the markings are considered to be syntactically and semantically dense. If in reading the clock face one is constrained to report only the individual markings, then the system is notational; if on the other hand the reader interpolates between the markings to report intermediate values, then the system is analog. The example of clock faces indicates how we should treat diagrams and maps. Purely topological diagrams, such as schematics, are (largely) notational. Maps or charts which permit interpolation of distances or other dense values are analog.

Most practical languages are composed of both notational and analog characteristics. Musical scores, for instance, contain notes, time signatures, and accidentals, each notational — each of these symbols is finitely distinguishable from its fellows and refers to a specific member of a finitely distinguishable domain. Written instructions for tempo such as *allegro*, *adagio*, *minuetto*, on the other hand, are analog, because the indicated tempos are not finitely distinguishable. A more important and subtle analog aspect can be found in the layout of musical notation. The vertical dimension of a single staff is one of pitch and the horizontal dimension is one of time. Though the notation breaks these two dimensions into the discrete classes known as notes, the layout of the notation presents the notes in roughly the position they would occupy if the two dimensions were presented in an analog fashion. Thus the layout of musical notation presents implicit analog spacing information. Any beginning piano player knows, for instance, that it is best not to try pieces that have lots of black crammed closely together. The density of the notation communicates the density of the playing that will be required, and density is one of the key elements in facilitating sight-reading of music.[8] Good analog spacing in musical notation, though not notationally necessary, is a *de facto* requirement for a creditable score.[9]

3. Using density to evaluate languages.

How can Goodman's theory of languages help us characterize visual languages? We cannot prove that a formal notion of visual expressiveness is the same as our intuitive notion of visual expressiveness, for the same reason that we cannot prove that formal notions of computability (such as Turing machines) are equivalent to our intuitive notion of computability. At best we can conjecture an equivalence and then look for contrary examples. My conjecture is that the visual aspect of any language is rooted in its analog element; *a visual language is one which maps dense syntax to a dense domain*.

The conjecture needs a much more thorough defence than can be given here, but roughly it depends on two ideas. The first is the assumption that our perceptual systems are non-propositional.[10] The second is the observation that symbolic abstraction in digital computers tends to involve discarding nonessential elements of language, which are typically the non-notational or analog elements. Thus our interest in visual languages is sparked by a desire to leverage human perceptual capabilities and to restore some element of manual languages that we feel has been lost through computerization.

Figure 1. Miró security specification.

Let us consider two existing languages to see whether they employ dense syntax to represent dense domains. The first example is the Miró system,[11] which uses Venn-like diagrams to present security constraints. An example is shown in Figure 1. Each set in the diagram is indicated by a rectangular region. Unlike normal Venn diagrams, overlapped regions in a Miró diagram are not meaningful unless a set is explicitly contained within the overlap. Arcs between regions are directed hyperedges between members of the sets. Miró's arcs represent permissions or constraints on the members of the regions. In Figure 1, for example, the arc without an ''X'' represents permission to read a file i.e., Alice can read her mail, while the arc with an ''X'' represents a constraint that prohibits permission to read a file i.e., no members of the World can read Alice's mail. Since Alice is a member of the World and so two conflicting constraints apply to her, we further adopt the convention that the constraint deriving from the lowest level of the nested set is the overriding one.

Let us consider the visual character of the Miró specification. First, as the Miró developers themselves have been careful to note, it is not difficult to create pictures

Figure 2. Contradictory specification.

that express contradictory security constraints, such as the one in Figure 2. The Miró developers call these pictures *ambiguous*, which is something of a misnomer. An ambiguous picture is one which admits of more than one interpretation, but Figure 2 really shows a picture with a single, impossible interpretation. Logically, the picture is equivalent to *p & not-p*. That we can depict such a situation is a warning that the language being presented is more notational than visual. We cannot make a picture that *resembles* a logical impossibility, because such things do not exist in perceptual domains. We can make something that *represents* a logical impossibility, but what we make is not a picture, but a notational mark that is *defined* to stand for the impossibility.

A second aspect of Miró is the problem of overlaps. Miró pictures are topological Venn diagrams; the overlaps are not semantically meaningful.[12] We are not able to turn off our perceptual senses quite this arbitrarily, however; we normally think that an overlap means something. Indeed, in the simple example of Figure 1, the Miró developers have chosen a very specific layout, one that minimizes the number of superfluous overlaps. We could imagine many other semantically equivalent Miró pictures in which all rectangles were much larger and overlapped with all other rectangles in the diagram. Though such a diagram would be semantically equivalent to the diagram in Figure 1, it would be less desirable. Miró's developers chose the particular layout shown in Figure 1 because overlaps do communicate information, even though that information is not a part of the Miró specification.

There is dense syntactic information in a Miró picture. The sizes and position of the regions, the sizes and position of the overlaps, and the density of the arc pattern are all analog characters which are susceptible to an infinite variety of changes. However, none of these aspects map into the semantics of a Miró specification. The layout is visually expressive, but layout is not formally part of the notation.

Is there dense semantic information in a Miró picture? Miró's developers suggest that security lends itself naturally to visualization because the domains of interest are sets, and sets can be easily visualized with Venn diagrams. The sets in question, however, are not dense; thus by the conjecture we would tend to reject the notion that visual techniques can be usefully applied to describe the sets. Even if security constraints were a dense domain, however, it is not the case that sets and their intersections are always easily depicted by Venn diagrams. If the relationships are not hierarchical in nature and if users tend to belong to more than one group, layout becomes very problematic. On a sample of machines at the University of Waterloo, for example, the mean number of groups per userid ranges from 1.7 to 2.6; worse, as many as 63% of users belong to 3 or more groups (one user belongs to 15 groups). We can conceive of Miró diagrams even for these situations (consider for instance a matrix of long horizontal rectangles for users and long vertical ones for groups), but the correspondence between the visual impact of the diagram and its semantic content is especially low. In such situations it seems that Miró diagrams are more notational than visual.

Let us turn to a different example of visual expression, Cole's approach to medical statistics.[13] Cole has developed several graphics to help physicians quickly grasp certain problem domains that involve multiple interacting variables or Bayesian reasoning. For instance, most doctors (and most people) answer the following problem incorrectly: given that a certain test shows a positive result for disease X, how likely is the patient to have the disease if 95% of those with the disease test positive, 90% of those without the disease test negative, and if 1 in 1000 persons actually has the disease?

Most people incorrectly rate the odds of the patient having the disease as being somewhere around 95%, because they routinely overemphasize the test's specificity. The diagram in Figure 3 makes estimation of the correct result quite easy. The largest rectangle is the total population; the small rectangle shows the population who test positive, and the smallest region at the bottom shows the number who test positive and who actually have the disease. The likelihood that the patient actually has the disease is 1%.

A different kind of example from Cole is shown in Figure 4. Here the problem is the presentation of the dynamic behaviour of a patient on a respirator. There are several variables involved, including the total volume of each breath, how the volume is distributed between respirator and patient, and the shallowness of the breath. Figure 4 shows a series of graphics that snapshot the breathing activity of the patient. Each snapshot shows one or two rectangles. The leftmost rectangle represents the breathing activity of the mechanical respirator, while the rightmost rectangle (sometimes not present) represents the breathing activity of the patient. The total area of the two rectangles is the total volume of air provided; the relative sizes of the two rectangles gives the ratio of the

Figure 3. Bayesian reasoning.

volume as provided by the respirator and by the patient. The width of the rectangle corresponds to the rate of breathing; the light-coloured area indicates the ‘‘dead space’’ of the patient’s mouth and throat, while the dark area indicates the air within the lungs. The sequence of rectangles shows that the respirator is initially doing most of the work, with the patient’s breathing gradually increasing, passing through a stage where there are mostly shallow breaths, and then finally the patient resumes normal breathing.

Figure 4. Respiration.

Cole’s graphics have an excellent intuitive flair that makes them easily understandable even to lay people. Can we account for this with the conjecture that visualness involves density? First, consider syntactic density. The respirator graphic clearly employs a syntactically dense representation. While the Bayesian graphic is presented on a chart which has discrete divisions, we interpret it not by counting the divisions but by estimating the relative areas of the three different regions. Hence this representation is also syntactically dense. Second, consider semantic density. Both Bayesian reasoning and respiration are suitably dense domains. In the case of Bayesian reasoning, probabilities can take on any value between 0 and 1. In the case of the respirator graphic, we are interested not just in discrete stages of breathing (i.e., breathing or not breathing), but in the process of breathing, how the patient progresses in a continuous fashion from total dependence to independence. Thus Cole’s graphics are good examples of the conjecture of the efficacy of visual languages for analog situations.

4. Visual programming languages.

The examples presented so far serve as a *prima facie* case for the conjecture that visualness involves density. What are the dense semantic aspects of programming, if any? What types of dense syntax could be mapped to these aspects?

At first glance, it is difficult to find density in programming. Digital computers are *ipso facto* discrete machines; their domain of activity is discrete, and any representation made to them must eventually be convertible to a discrete form. However, as noted earlier, density is sometimes a matter of how one reads a device. A quartz wristwatch is a discrete device, but if it drives a pair of hands, then its activity can be represented in an analog fashion.

One example of an analog program on a discrete device is Randy Smith’s technique for doing strip chart recording of damped harmonic motion in the Alternate Reality Kit.[14] The Alternate Reality Kit is an environment in which users can construct and experiment with the dynamics of moving particles. Users can create particles, set them in motion, change the laws of nature, and watch a simulation of the effects of their action. For instance, users could create two particles and instantiate a spring force between them. Adding friction will result in a damped harmonic motion. If the objects are able to draw on a background which is set moving behind them, they will trace out a strip chart of their position over time, and hence of damped harmonic motion. The strip chart recorder is one of a variety of analog devices which can be constructed in the Alternate Reality Kit and other simulation engines. Dewdney[15] describes several real-world analog machines that solve complex problems, including a ‘‘rubber band’’ machine for doing linear regressions, and balance-beam devices for finding the roots of cubic equations; one could imagine simulating these in a computer environment. The analog quality of these devices serves an important role in providing the intuition we need to understand their method of operation, unlike a traditional discrete program, whose operation we must mentally ‘‘execute’’.

Simulating analog worlds is intriguing, but is probably a long-term research effort. For existing languages we should investigate the most important (and often only) dense syntactic character, namely layout. In most practical languages, the defined characters are discrete representatives for discrete compliants, and hence are notational rather than visual. Layout, on the other hand, is syntactically dense, since icons, words, or other symbols can be positioned to a theoretically infinite degree of precision.

Layout also seems to map to something of semantic significance, typically some aspect of overall structure, and I suspect that this is a dense domain as well. Layout's importance is empirically observed in both traditional visual tools, such as CAD languages,[16] and in text-based programming languages.[17] Layout is also notoriously difficult to reduce to a discrete set of rules, a good indication of its visual, rather than notational, status.[9]

If the visual character of most practical languages is captured in their layout, then layout ought not to be accidental or arbitrary; it should communicate information about the domain of interest, and the more information it can communicate, the more relevant it is. Curiously enough, the more layout matters the less layout freedom we should provide. For if multiple layouts are possible, then the layout is not maximally constrained by the domain, and hence not maximally representative of the domain. If only one layout can be presented because all aspects of the layout are directly driven by the domain, then the domain and layout are coupled in a maximal fashion. This argument is similar to Popper's epistemological notion that what we desire are maximally *improbable* scientific theories. The more accurate a theory is, the less variance it will tolerate, and hence the more improbable it becomes.[18]

The distinction between notational and analog languages suggests that layout matters, but it does not tell us how to control it. Further work in this direction will depend on our ability to identify the properties of good layout. Larkin and Simon suggest that pictures are more successful than text because (or when) they cluster the information needed for making inferences.[19] Thus we should consider evaluating layouts in terms of how they cluster information needed for inferences. We already make rough judgements of this sort when we claim that textual notations are more difficult if closely related program modules are spread far apart in the code (i.e., are not laid out in proximity).

Larkin and Simon's view of complexity is static. One dynamic measure of complexity is Green's notion of *viscosity*,[20] the amount of effort required to convert a representation from one valid state to another. Highly viscous languages require significant effort to change or update. We should consider comparing viscosity of notations and analog languages. Density seems to imply high connectedness (and hence high viscosity), but this connectedness often comes hand in hand with a compact representation for the connections (and hence low viscosity).

Finally, a word should be said about other dimensions or taxonomies for evaluating visual languages, such as mappability, scope, salience, manipulability, and the ability to represent concurrency.[21] While I have not made a complete survey, many such properties are not exclusively visual; manipulability and scope, for example, are important aspects of non-visual languages as well as visual ones. Thus such characteristics cannot be employed as a line of demarcation between visual and non-visual languages. Practical languages combine both visual and notational elements, however, and we must evaluate a given language for its suitability both visually and as a notation. Thus many dimensions and taxonomies are needed to determine a language's overall effectiveness.

5. Conclusions.

The visual character of languages is rooted in their ability to use dense representations to describe dense domains. Some tasks lend themselves naturally to such representations because they come ready-made with dense domains; in these cases it is relatively easy to find graphics that map directly to the domain. Other tasks, such as computer programming, do not deal directly with dense domains, and thus are not easily represented by visual languages. For this type of problem there are two possible approaches. One is to simulate an analog machine with the appropriate syntactic and semantic density. The second approach is to investigate ways for mapping layout to some useful, dense, semantic domain.

6. Acknowledgements.

The ideas in this paper have benefited greatly from discussions with Frank Wm. Tompa, and also from extensive e-mail exchanges with Paul Andrews, Thomas R.G. Green, Alberto Mendelzon, Bonnie Nardi, and Steve Roberts. Financial support for this work was provided by the IBM Canada Laboratory under grant DC03157, and from the Natural Sciences and Engineering Research Council of Canada under grants A-5692 and A-9292.

7. References

1. Nan C. Shu, *Visual Programming*, Van Nostrand Reinhold, New York, N.Y. (1988).
2. T.R.G. Green, ''Pictures of Programs and Other Processes, or How to Do Things With Lines,'' *Behaviour and Information Technology*, **1**(1) pp. 3-36 (1982).
3. Fredrick P. Brooks, Jr., ''No Silver Bullet: Essence and Accidents of Software Engineering,'' *IEEE Computer*, pp. 10-19 (April 1987).
4. T.R.G. Green, M. Petre, and R.K.E. Bellamy, ''Comprehensibility of Visual and Textual Programs: A Test of Superlativism Against the 'Match-Mismatch' Conjecture,'' *submitted to*

Empirical Studies of Programmers (4th workshop), (1992).

5. Tarun Sen and Warren J. Boe, ‘‘Confidence and Accuracy in Judgements Using Computer Displayed Information,’’ *Behaviour and Information Technology*, **10**(1) pp. 53-64 (1991).
6. Thomas W. Lauer and Gerald V. Post, ‘‘Density in Scatterplots and the Estimation of Correlation,’’ *Behaviour and Information Technology*, **8**(3) pp. 235-244 (1989).
7. Nelson Goodman, *Languages of Art*, Hacket Publishing Co. (1976).
8. John Sloboda, ‘‘The Uses of Space in Music Notation,’’ *Visible Language*, **XV**(1) pp. 86-110 (1981).
9. Dorothea Blostein and Lippold Haken, ‘‘Justification of Printed Music,’’ *Communications of the ACM*, **34**(3) pp. 88-99 (March 1991).
10. Susanne K. Langer, *Philosophy in a New Key: A Study in the Symbolism of Reason, Rite, and Art (3rd edition)*, Harvard University Press, Cambridge, Massachusetts (1957).
11. Mark W. Maimone, J.D. Tygar, and Jeannette M. Wing, ‘‘Miró Semantics for Security,’’ *1988 IEEE Workshop on Visual Languages*, pp. 45-51 (October 10-12, 1988).
12. David Harel, ‘‘On Visual Formalisms,’’ CMU-CS-87-126, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania (June 5, 1987).
13. William G. Cole, ‘‘Medical Cognitive Graphics,’’ *Proceedings of the CHI ’86 Conference on Human Factors in Computing Systems*, pp. 91-95 (April 13-17, 1986).
14. Randall B. Smith, ‘‘Experiences with the Alternate Reality Kit: An Example of the Tension Between Literalism and Magic,’’ *Proceedings of the 1987 CHI + GI Conference on Human Factors in Computing Systems and Graphics Interface*, pp. 61-67 (April 5-9, 1987).
15. A.K. Dewdney, ‘‘Analog Gadgets that Solve a Diversity of Problems and Raise an Array of Questions,’’ *Scientific American*, pp. 18-29 (June 1985).
16. M. Petre and T.R.G. Green, ‘‘Where to Draw the Line with Text: Some Claims by Logic Designers About Graphics in Notations,’’ *Proceedings of INTERACT ’90, IFIP Conference on Human-Computer Interaction*, pp. 463-468 (August 27-31, 1990).
17. T.R.G. Green, ‘‘Ifs and Thens: Is Nesting Just for the Birds?,’’ *Software — Practice and Experience*, **10** pp. 373-381 John Wiley & Sons, Ltd., (1980).
18. Karl R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, Harper & Row, New York, N.Y. (1968).
19. Jill H. Larkin and Hebert A. Simon, ‘‘Why a Diagram is (Sometimes) Worth Ten Thousand Words,’’ *Cognitive Science*, **11** pp. 65-100 (1987).
20. Thomas R.G. Green, ‘‘Cognitive Dimensions of Notations,’’ *Proceedings of the HCI ’90 Conference*, (September 1989).
21. Marc Eisenstadt, John Domingue, Tim Rajan, and Enrico Motta, ‘‘Visual Knowledge Engineering,’’ *IEEE Transactions on Software Engineering*, **16**(10) pp. 1164-1177 (October 1990).