7
Designing with Diagrams: A Role for Computing in Design Education and Exploration

Stephen M. Ervin

Department of Architecture
School of Architecture and Urban Planning
Massachusetts Institute of Technology

Introduction

Environmental designers, design educators and design students using computers are a constituency with a set of requirements for database structure and flexibility, for knowledge representation and inference mechanisms, and for both graphical and non-graphical operations, that are now articulatable [23] and to-date largely unmet. This is especially so in the area called preliminary' or schematic design, where our requirements are related to, but different from, those of our colleagues in mechanical and electrical engineering, whose needs have dominated the notable developments in this area. One manifestation of these needs is in the peculiar form of graphics called diagrams , and the ways in which environmental designers (architects, landscape architects, urban designers) use them. Our diagrams are both similar to and different from structural, circuit, or logical diagrams in important ways. These similarities and differences yield basic insights into designing and design knowledge, and provide guidance for some necessary steps in the development of the next generation of CAD systems. Diagrams as a form of knowledge representation have received little scrutiny in the literature of graphic representation and computer graphics (but see [1, 3, 5, 8, 10, 11, 13, 17, 18, 22, 25, 27, 28, 29]) In the following sections I present an overview of the theoretical basis for distinguishing and using diagrams; examine some of the computational requirements for a system of computer-aided diagramming; describe a prototype implementation called CBD (Constraint Based Diagrammer) and illustrate one example of its use; and speculate on the implications and potential applications of these ideas in computer-aided design education.

A Model of Designing with Diagrams

Real-world knowledge can be represented by many kinds of graphical constructs-photographs, sketches, maps, plans, diagrams, charts, graphs among them. The question of which kind of graphics to use in which circumstances is the usual Al question of choosing an appropriate knowledge representation formalism, guided by both the intended application and the inherent structure of the knowledge. Pictorial graphics (images like photographs, sketches and some maps) are powerful data that exploit our visual ability for parallel processing, but make no commitment to use or structure -- they are literal(ly) data. Propositional graphics (some maps, plans and diagrams, e.g.), by contrast, constitute information (or even knowledge): they embody some media-independent abstraction(s), typically are associated with some particular inference-making use(s), and require a commitment to some model(s) of the structure of the knowledge being conveyed. Categorizing graphics, however, is not a Linnaean enterprise and no hard and fast distinction can be made (see discussions in [7,10,17]).

Two criteria that are useful for distinguishing pictorial representations from propositional ones are their respective inferential purposes and the levels and types of abstraction they contain. I propose a simple assertion: pictorial graphics are concerned with shape, shape-like and detail attributes: color, curvature, texture, balance, proportion, e.g. while propositional graphics are concerned with form, and attributes that are abstract and topological (meta-shape, if you will): existence, number, magnitude, closure, connection and others. The former are appreciated visually, judged 'holistically', and generally defy symbolic translation; the latter may be represented symbolically, judged 'logically', and are designed for visual inference rather than appreciation per se.[1].

Definitions of design and models of designing have been many (for example, [2, 6, 20, 33]), and I'm not proposing any new one here. I take the common and uncontroversial position that designing is a process of preparing for the production of an artifact (at whatever scale, in whatever medium, at whatever level of abstraction or aggregation) subject to some set of goals and constraints. In final 'working drawings', neither goals nor constraints may be explicit-they will have been transformed into specifications on material properties. In preliminary 'design drawings, including diagrams, goals and constraints are more likely to be represented directly. In this model of designing, then, goals and constraints must be made explicit, represented graphically, and turned into specifications on material properties. The possible paths from concepts to drawings are several, and need not include diagrams, but often do.

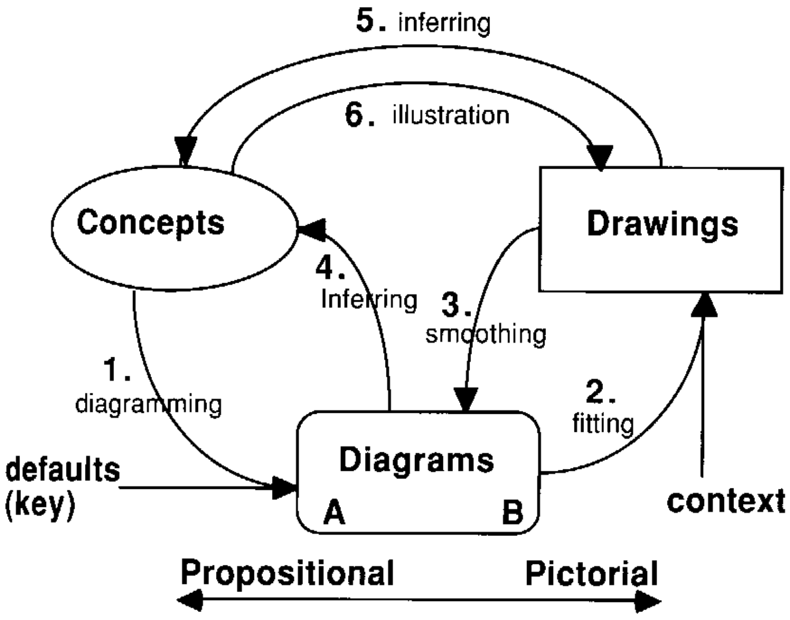


Figure 1. Concept - Diagram - Drawing Transformations.

A model of the relation between concepts and drawings-with diagrams in between-is shown in figure 1. Between the three nodes that describe forms of representation (concepts, diagrams, drawings) are six arrows, representing transformations between the representations.

This model of designing doesn't specify a starting point or any sequence of operations; nor does it require diagrams at all. There is no claim that designing always starts from concepts and proceeds to drawings, or in any other direction; the poles and the paths between them are caricatures, but they are useful. The decision about which route to take from concepts to drawings-a question about the necessity and utility of diagrams-is not resolvable by any theory. It is surely possible for designers to live without diagrams, though it might not he comfortable. This comfort derives from the place of diagrams between graphics and concepts, and their role in visual reasoning and inference.

The above assertions about graphical distinctions, design knowledge and design operations are arguable, but the question should not he "Are they right?" Rather, the question here is "Are they useful? Implementable? Testable?" I believe the answer is "Yes' to all three, which is where computing comes in

Constraint Based Diagrammer

The diagram machine described here is the CBD (Constraint Based Diagrammer), designed to explore the ideas about designing with diagrams, and using constraints to do so. The realm I've chosen is in urban design. The foundation of this machine, though, is not urban design knowledge, but rather geometric and topological. Computing with geometry has received a lot of attention in CAD and solid modelling applications; less attention has been given to topological descriptors [15, 36]. The assumption here is that the abstraction inherent in diagrams calls for less precise descriptions than shape, size and position; that relational descriptors are the essence of the knowledge, and that particular details are generated by default only when required for display.

Propositional and Graphical Database

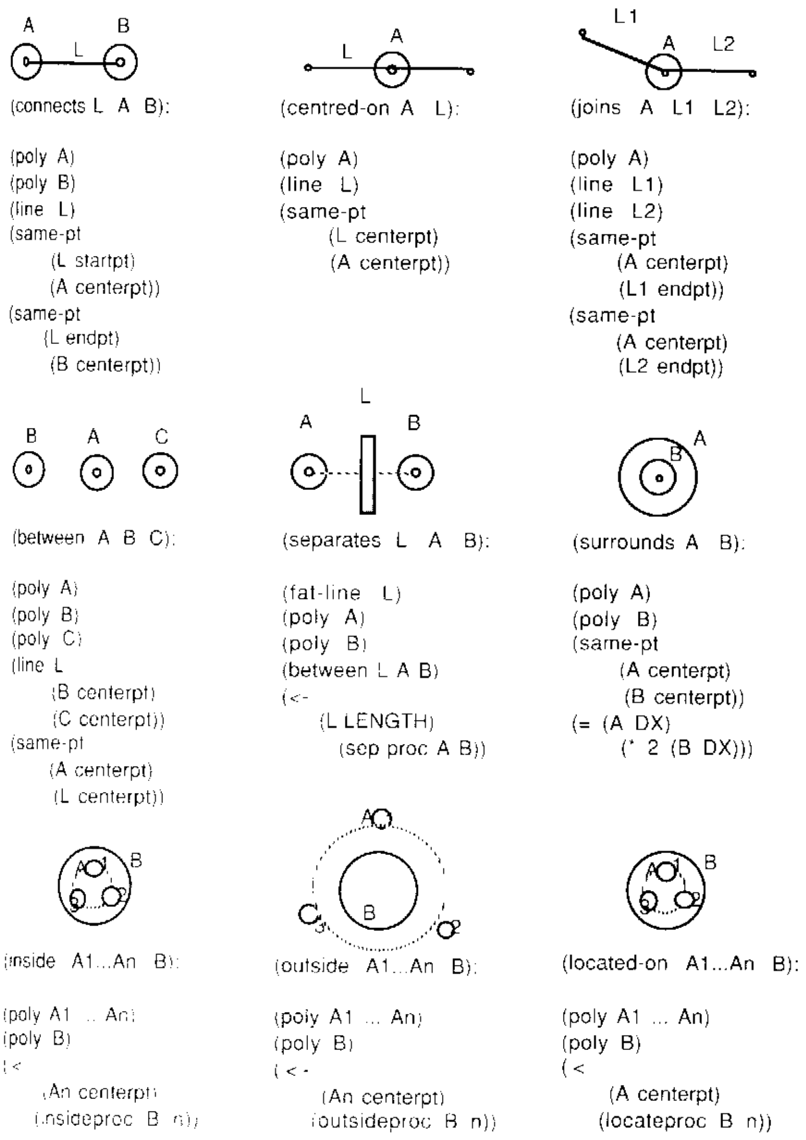
Clearly a diagramming machine will have a graphical component, with display and editing operations, and as such will require a commitment to a graphical database and data structures. An additional database is required that contains the propositional knowledge being conveyed by the diagram. The particular choice in the machine described below is an object-oriented approach, in which objects and the relations between them are the principal data structures.

Default Translation rules

The most distinctive component of the diagramming machine described below is a system for translating symbolic propositions into graphics. In this implementation I'm concerned with the process of going from propositions to a diagram (the arrow labelled 1-diagramming in figure 1 above). This translation require a set of default rules for creating objects and representing class distinctions between them, a vocabulary of diagrammatic relations between objects, and a set of default rules for generating graphics from these objects and relations. A partial vocabulary is illustrated below, and defaults are expanded upon, but first I describe the mechanism for managing objects and relations between them.

Constraint Management

The constraint management paradigm [19, 21] provides a concise, flexible and expressive means for expressing relations (constraints) between objects and among classes of objects. The forms of the relations may include equations, inequalities, and one-way assignments (inferences of the IF-THEN' form.) These relational descriptions form the basis for the computational mechanics behind the constraint model of designing, including satisfaction, propagation, conflict resolution and block-structuring. The approach of designing with constraints has been explored in various fields, and has been applied to both preliminary and detailed design [32J.



```
(connects L  A  B):

(poly  A)
(poly  B)
(line  L)
(same-pt
      (L startpt)
      (A centerpt))
(same-pt
     (L endpt)
      (B centerpt))
```

```
(centred-on  A   L):

(poly  A)
(line  L)
(same-pt
      (L centerpt)
      (A centerpt))
```

```
(joins   A  L1  L2):

(poly  A)
(line  L1)
(line  L2)
(same-pt
      (A centerpt)
      (L1 endpt))
(same-pt
      (A centerpt)
      (L2 endpt))
```

```
(between  A  B  C):

(poly  A)
(poly  B)
(poly  C)
(line  L
      (B centerpt)
      (C centerpt))
(same-pt
      (A centerpt)
      (L centerpt))
```

```
(separates  L   A   B):

(fat-line   L)
(poly   A)
(poly   B)
(between  L  A  B)
(<-
      (L LENGTH)
        (sep proc  A  B))
```

```
(surrounds  A   B):

(poly  A)
(poly   B)
(same-pt
      (A centerpt)
       (B centerpt))
(=  (A  DX)
      (*  2  (B  DX)))
```

```
(inside   A1...An   B):

(poly A1  ...  An)
(poly  B)
(<
     (An centerpt)
    (insideproc  B  n))
```

```
(outside   A1...An   B):

(poly A1  ...  An)
(poly  B)
( < -
     (An centerpt)
    (outsideproc  B  n))
```

```
(located-on   A1...An   B):

(poly A1  ...  An)
(poly  B)
( <
     (A centerpt)
    (locateproc  B  n))
```

Figure 2. Lexicon of Relations in CBD.

CM2 [14] is an implementation of a constraint manager[2], based on the principles outlined above. It has been used in the development of several design laboratories, micro-worlds constructed for the exploration of designing by constraints. It provides an object-oriented constraint manager with a graphic interface and basic mechanism for constraint compilation, satisfaction, propagation and conflict resolution through dependency directed backtracking. It *serves as* a substrate for the Constraint Based Diagrammer (CBD) described in figure 2.

Lexicon of Constraints

The vocabulary of elements and relations used by the CBD is simple. Elements are points, lines, and simple polygons (rectangles and ovals). Relations in CBD are defined as LISP procedures, in the form

(RELATION ELEMENTI ELEMENT2 ELEMENT3 ... ELEMENT n)

that behave as follows when entered: I.) Check the types of ELEMENTs to make sure that they are appropriate for the relations, and signal an error if not; 2.) If the elements do not already exist, create them according to an appropriate type determined by default; 3.) Install a set of constraints between the elements, according to the rules of the constraint manager, and turn over processing to the constraint manager to check and propagate if possible. Figure 2 shows a selected set diagrammatic relationships in CBD, that have a constraint implementation.

Graphical Defaults

The CBD uses the above vocabulary of relations between elements, relying on the constraint manager (CM2) to enforce the relations numerically as required. Numerical values are required for the purposes of producing graphics, but in CBD these numeric values are secondary to the propositional relations such as those listed above. In fact, these values are instantiated as defaults-sensible values that make for a legible diagram in the chosen graphic environment, but that can be over-ridden by hand by the designer, or by calculation of the constraint manager.

CBD has both general and diagram- or context-specific defaults for graphics production. The defaults are in general values to be installed when graphics attributes with no value are specified. The choice of defaults is dependent upon a hierarchy of rules that determine the level and appropriate default decision. These defaults, and the rules that govern them, can be divided into three layers. At the top-level, are domain-specific defaults like, for example: "Where a screen is desired, use a linear element This is an example of domain expertise, in urban design-the choice to use a single nonlinear object, for example, might be more appropriate in a sculpture garden. The next level of defaults governs details of graphics appearance: for example, that screens are to be represented by a single wide line in a gray tone, or that the line should be vertical instead of horizontal if there is a choice. These defaults too are modifiable by the designer at will, but are decisions based more on immediate graphics feedback and fine-tuning than on domain knowledge. Finally, there are defaults that are generally stable, not usually changed by the designer, for example that a single line will be one- or two-pixels wide. This level of default is a graphic attribute that depends on the current graphics environment (size of window, extent of diagram etc.)

Defaults enable drawing of incompletely specified objects. Some example defaults in CBD are: "All objects are circles", "All dimensions are 20", "All colors are black." These defaults can be overridden at any time, by the designer or by a constraint in the system.

A library of common defaults is an important part of the domain-specific knowledge contained in environmental designers and their diagrams; although some common diagrammatic approaches exist, they are not nearly so well-defined as to be considered conventions. A computer-aided diagramming program must allow for easy user-definition of default graphical symbols and interpretations of propositions into graphics. This ability is supported by the object-oriented, interactive nature of the program, in which prototypes are easily developed, and modifications and alternatives easily explored. The following example shows some of these features at work.

Urban Design Scenario

Diagrams play an important role in all kinds of design-electrical, software, architectural, e.g.-but are especially important in urban design. Often urban designers are in the position of proposing conceptual master plans that others will fill out in more detail. The complexity of design at the urban scale particularly calls for the benefits of simplification gained through abstraction (with the proviso and warning that final decisions in the urban context must be made in cognizance of the full complexity, and not based solely on diagrammatic abstractions.)

A number of approaches have been taken to formalizing design knowledge at the site and urban scale, some proposing systems for computer aids (for example, [4,5,12,16,24, 34,371). The domain of urban design provides an area in which some clear rules may be formulated for preliminary design, but the chances are very small that algorithmic or production-system solutions can be found to any but the most trivial problems. For this reason, the approach taken is an interactive one, in which computer aids are devised to support human designers at several levels: rule-based diagramming tools for conceptual design; drawing, calculating and modelling tools for design development and production. The following example illustrates the former only, and the interaction between the propositional and graphical databases in the Constraint Based Diagrammer.

New Town Example
This example describes a design process for a proposed new town, using a theory of urban design [9] based on the fundamental concepts of 'containers' and connectors', and including terms like 'magnet, 'axis, 'link, 'ring, etc. These terms are used without formal definition, but with conventional meaning. The design anticipates districts of different uses (commercial, residential, industrial), and several types of connectors (pedestrian, automotive, transit). The sequence of propositions below describes a structure and form of the new town, without specifying a shape or details:

1 "the new town has a commercial center, located on a main axis"
2. "the main axis connects two external magnets"
3. a main ring road surrounds the commercial center
4. "several residential centers surround the commercial center, inside the ring road"
5. "a minor transit ring connects the residential centers"
6. "automotive links connect the residential center and the ring road"
7. "industrial centers are located outside the ring road'
8. "automotive links connect the ring road and the industrial centers"
9. "pedestrian paths connect between the residential centers"
10. "pedestrian paths connect the pedestrian path network to the industrial centers"

From these ten propositions the CBD can construct a diagram giving preliminary shape to the proposal. The diagram is a plausible, default graphical representation of the elements and their relationships.

To produce the diagram, the propositions above must be stated in the formal vocabulary and syntax of the CBD. (This 'translation' must be done by the user / designer-this implementation doesn't address questions of 'natural language' interface or automatic translation from text to propositions.) The relationships expressed in LISP syntax, follow:

```
(inbetween looprd washdc balto)
(centred-in townctr Iooprd)
(ringbetween transloop townctr looprd)
(located-on resareal transloop)
(located-on resarea3 transloop)
(located-on resarea2 transloop)
```

(located-on resarea4 transloop)
(links 11 resareal Iooprd)
(connected resareal resarea2)
(links 12 resarea2 Iooprd)
(connected resarea1 resarea3)
(links 13 resarea3 looprd)
(connected resarea1 resarea4)
(links 14 resarea4 Iooprd)
(connected resarea2 resarea3)
(connected resarea2 resarea4)
(connected resarea3 resarea4)
(links 15 townctr transloop)
(outside c1 Iooprd)
(links cl1 c1 Iooprd)

These propositions, in this order, result in a default diagram, which is shown in figure 3, and after some hand-manipulation to adjust shape and other graphic attributes for visual clarity in figure 4.

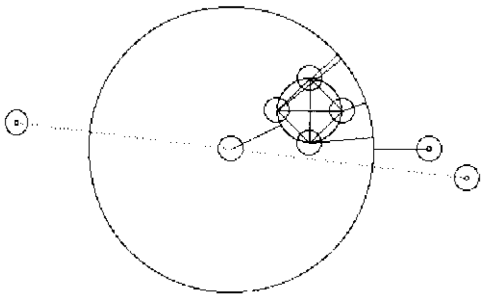


Figure 3 Default Diagram

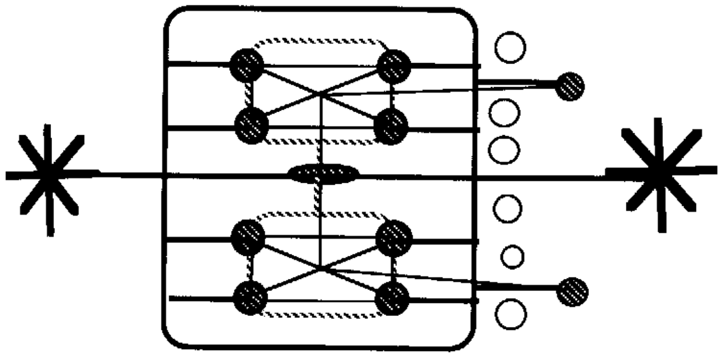


Figure 4 Hand Adjusted Graphics

The initial diagram produced by CBD, shown in figure 3, is topologically equivalent, though visually quite dissimilar to, the original source diagram. Figure 5 shows a facsimile of that diagram taken from the study mentioned above, a hand-drawn diagram illustrating the schematic organization of the new town of Columbia, Maryland. The changes required to produce figure 4 are not changes in the list of propositions themselves, but in graphic attributes of the diagrammatic elements produced by the CBD.

The reasons for these changes can be traced to two general sources. One reason is to increase the analogical and metaphorical qualities of the elements and relations in the diagram: making external magnets into asterisks, conventional symbols that contain an element of literal analogy, and making residential centers and town centers filled-in solids, for example. Another important reason for these changes is less clear, but is concerned with 'visual comfort--making the principal axis horizontal, and the diagram generally symmetrical, and the lineweights generally heavier, for example, are all moves that result in simplicity and clarity in the diagram, thus increasing its visual appeal and inferential potential. Detailing, describing, and capturing these kinds of rules, in order to embed them into the knowledge bases and defaults procedures of the diagrammer, are all tasks for continued research, that will improve the power and utility of the CBD.

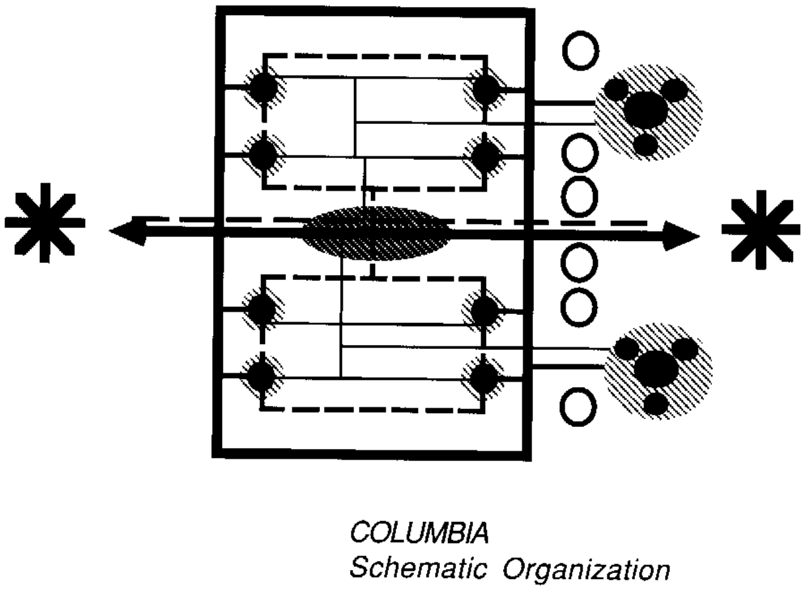


Figure 5 Original Diagram of Columbia, after Chermayeff, et al

Discussion

What are the advantages of having constructed the diagram this way? There are three principal ones. 1.) The knowledge that describes the town being designed resides in a database of propositions; from these propositions, more can be inferred, and questions can be asked. For example, without visual inspection, a geometric reasoner could determine that at some point, pedestrian paths must cross the main ring road. This in turn triggers a set of conventional responses (overpasses, underpasses, traffic lights, e.g.), and also brings the potential conflict to the attention of the designer. 2.) The graphical representation is directly manipulable, but is also tied to the database. Constraints may be observed, propagated, or modified by direct manipulation of the graphics. The conflict resolution procedures in the constraint manger make the act of graphical editing an interactive, rather than a one-way activity. 3.) The graphics can be changed without changing the propositions, just the defaults

The sum of these advantages is that all of the usual graphical and knowledge-based examinations of the proposed design can be made, with a tight interconnection between them. Further development, into a specific design proposal, is guided by the diagram in explicit ways.

Implications for Design Exploration and Education

Claiming to aid designing is a bold claim; and even bolder in the context of design education. It may be immodest to claim that these ideas about diagramming or the program described here constitute design aids, but it is not entirely without reason. Field experience using constraint management and diagrams in design education is limited, but my expectations are not. Projecting future developments in computatational reasoning and rendering systems, we can predict the appearance of powerful and fertile systems for articulating design knowledge and manipulating that knowledge in several different representations, including diagrammatic.

Constraint Models for Design Exploration

The use of constraint models for design development and exploration puts an emphasis on conceptual understanding and reduces the impact of graphical skills on novice designers ability to communicate. The ability to recognize and name design elements, and to articulate chains of design reasoning, is a real test of design ability. The traditional system of presentation drawings judged by jury leaves all such knowledge implicit and untested, and depends strongly on graphics to evoke discussion of design ideas. The possibility of explicitly articulating bits and pieces of design knowledge, and tying them together in defensible ways, represents a departure from traditional atelier-based design education, but one that is in tune with the opportunities presented by knowledge-based computing.

Front-End to CAD and Modelling Packages
The above is not to deny the value of "pictorial" drawings, and their role in visual inference for design development and communication, Computer graphics already offers the possibility of "levelling the playing field" between design students of varying graphical capabilities; consistent lineweights, harmonious color combinations and perspective projections are equally available to all. CAD programs are already part of the palette of modern designers and students. What corresponding advances exist in knowledge acquisition, representation and exploration? Diagrams provide an ideal front-end for developing graphical representations of formal concepts for further refinement in sophisticated 2-d and 3-d graphics modeling packages.

Knowledge Based Design Tools and Design Expertise
Diagrams may serve as a useful medium in the enterprise of knowledge engineering, extracting designers' knowledge in a form usable by rule-based and 'expert systems' of various sorts , or in new, as-yet-undeveloped systems for capturing and manipulating design knowledge [31], and so further our understanding of the forms and content of design expertise. I've indicated above that I don't hold much optimism for the enterprise traditionally described as 'expert systems' in urban design, but I do believe in the importance of knowledge-based, and to some extent rule-based, interactive design systems. Diagrams are a concise, expressive, and already-used vehicle for design knowledge and exploration. The questions that remain in this area are many and interesting. These include enumerating the principles of visual clarity and analogy that are so important to the inferential power of diagrams, as described above (and explored in some detail in [25]); methods for dealing with 'emergent form' and the related issue of 'multiple simultaneous representations' of design elements (see the discussion in [35!); as well as fundamental questions about interface and techniques of control that are essential to all designer-friendly computer systems.

Summary

I have argued that understanding diagrams and how they are used by designers is important for developers of computer-aided design tools, especially where these are to be used in design education. Diagrams are distinguished from other graphics by their essentially abstract, propositional nature, and their embodiment of graphic defaults. They are used by designers as a bridge between concepts (propositions) and graphics, especially in preliminary design, and as a medium for visual inference and communication. A system for computer-aided diagramming must maintain both a propositional knowledge base like a traditional production system', a graphical data base like a traditional graphics editor, and a set of rules for default translations between the two representations; these default translations must be easily modifiable by the designer/user. A prototype implementation called CBD (Constraint Based Diagrammer) has been developed on the Macintosh computer in Allegro Object Lisp, and one example of its use was illustrated, showing how the two data bases interact in the production of a diagram of a new town.

These ideas have several implications and potential applications in design education: the use of constraint models for design development and exploration puts an emphasis on conceptual understanding and reduces the impact of graphical skills on novice designers' ability to communicate; diagrams provide an ideal front-end for rapidly developing formal concepts for further refinement in sophisticated 2D and 3D graphics modeling packages; diagrams may serve as a useful medium in the enterprise of knowledge engineering, extracting designers' knowledge in a form usable by rule-based and 'expert' systems of various sorts, and so further our understanding of the forms and content of design expertise.

The development of computational tools and techniques for the manipulation and development of diagrams, based in part on the ideas presented here, will be an important step in the pursuit of designer-friendly computer systems for design education and exploration.

Acknowledgments

The constraint manager CM2 was inspired and in large part coded by my friend and colleague Mark Gross, supported in many ways by Aaron Fleisher, and benefited from the goodwill and good work of the folks at Coral Software (makers of Allegro Common Lisp). Members of the Design Theory and Methods Group and the Design Research Seminar at MIT provided a captive audience and constructive criticism of many of *the ideas presented here. The National Science Foundation and Apple Computer Co. provided partial financial support.*

References

1. Albarn, K and I. Smith, 1979. Diagram; The Instrument of Thought. London: Thames and Hudson.

2. Alexander, C. 1966. Notes on the Synthesis of Form. Cambridge: Harvard University Press.

3. Alexander, C.. and M. Manheim. 1962. 'The Use of Diagrams in Highway Route Location: An Experiment' Research Report R62-3. Civil Engineering Systems Laboratory. MIT.

4. Alexander, C., S. Ishakawa, and M. Silverstein. 1975. A Pattern Language: Towns. Buildings. Construction. New York: Oxford University Press.

5. Appleyard, D., K. Lynch., and J. Meyer. 1961. The View from the Road. Cambridge. Mass: MIT Press.

6. Archer, LB. .1965. 'Systematic Method for Designers". in Cross. N. (Ed.) 1984 Developments in Design Methodology. New York: John Wiley & Sons.

7, Arnheim, R. 1969. Visual Thinking. Berkeley: University of California Press.

8. Bongard, A. 1969. Pattern Recognition. New York: Wiley Interscience.

9. Chermayeff, S. and A. Tzonis. 1967. Advanced Studies in Urban Environments; Toward an Urban Model. Yale University. Institute for Applied Technology.

10. Crowe, N. and P. Laseau 1984. Visual Notes for Architects and Designers. New York: Van Nostrand Reinhold Co.

11. David, RE.. 1971. "Proposal for a Diagrammatic Language for Design". in Kennedy. M. (Ed.) Proceedings of the Kentucky Workshop in Computer Applications to Environmental Design.

12. Dickey, J.W. and G.G.Roy "C/SI: An Idea Generating System for Urban Planning". pp 125-130. In Alexander. F. (ed.) Proc. 1987 Conference on planning and Design in Urban and Regional Planning

13. Ervin, 5. 1987. "Levels of Abstraction in Environmental Design: A Computational Approach". pp 91-96. Proceedings of the, 1987 Conference on planning and Design in Urban and Regional Planning

14. Ervin, S and M, Gross. 1989. 'CM2 - A Constraint Manager for Design Exploration: User's Manual and Guide". MIT DTM Group Occasional Paper #1-89.

15. Evans, T.G. 1968. "A Program for the Solution of A Class of Geometric-Analogy Intelligence-Test Questions." pp 271-353. In Minsky. M. (ed.) Semantic Information Processing. Cambridge. Ma: MIT Press

16. Fleisher, A.. W. Porter & K. Lloyd. 1969. "DISCOURSE: Computer Assisted City Design in Mi M. (ed.) Computer Graphics in Architecture and Design. New Haven: Yale School of Art and Architecture.

17. Goodman, N. 1976. Languages of Art. Indianapolis: Hacket Publishing.

18. Graf, Douglas. 1985. "Diagrams". Perspecta 22. Yale Architectural journal.

19. Gross, M. . 1985. "Design as the Exploration of Constraints.." PhD Dissertation. MIT Department of Architecture.

20. Gross, M., S. Ervin. J. Anderson, and A. Fleisher. 1987. "Designing With Constraints". in Y. Kalay. (ed). The Computability of Design. New York: Wiley and Sons.

21. Gross. M., S. Ervin, J. Anderson, and A. Fleisher. July 1988. 'Constraints: Knowledge Representation in Design". in Design Studies.

22. Herdeg, W. (ed.) 1981. Graphis / Diagrama: The Graphic Visualization of Abstract Data. 4th ed. Zurich: Graphis Press Corp.

23. Kalay, Y. (ed.) 1987. Computability of Design. New York: Wiley & Sons.

24. Kasmar, J. 1970. 'The Development of a Usable Lexicon of Environmental Descriptors. Environment and Behavior. Vol. 2. No. 2. September.

25. Larkin, J. and H. Simon. 1987 "Why a diagram is (sometimes) worth 10.000 Words". pp. 65-69. Cognitive Science. Vol 11.

26. Laseau, P. 1980. Graphic Thinking for Architects and Designers. New York: Van Nostrand Reinhold Co.

27. M´Pherson, O.K. 'Thinking with Pictures-graphics as aids to complex system design and policy analysis.' pp 26-29. In Langdon. K. and G. Mallen (eds.) Design and Information Technology: Proceedings of an international conference on design. London: The Design Council

28. Maxwell, J.C. 1910. "Diagrams' Encyclopedia Britannica. Vol. 4. 11th Ed. 1910. pp. 146-149.

29. Montalvo, F. 1985. "Diagram Understanding: the Intersection of Computer Vision and Graphics". MIT Al Lab Memo No. 873.

30. Nelischer, M. and D. Hinde 1985. "A Graphic Language for Designers". pp. 60-63. Landscape Architecture. July / August.

31. Schön, D. July 1988. "Designing: Rules. Types and Worlds" Design Studies. Guildford: Butterworth & Co.

32. Serrano, D. and D. Gossard. 1987. "Constraint Management in Conceptual Design" In R.A. Adey and D. Sriram. (ed) Al and Engineering: Planning and Design.

Computational Mechanics Press.

33. Simon, H. 1969. The Sciences of the Artificial. Cambridge: MIT Press.

34. Stiny, G. Fall 1985. "Computing with Form and Meaning in Architecture". Journal of Architectural Education.

35. Stiny, C. 1989 ....What designers do that computers should.". This volume.

36. Woodbury, R. 1988. "The Knowledge Based Representation and Manipulation of Geometry'. PhD Dissertation Carnegie Mellon University Department of Architecture.

37. Yessios, C. 1975. 'Formal Languages for Site Planning". in Eastman. C. (ed.) Spatial Synthesis in Computer Aided Building Design. New York: John Wiley & Sons.

Notes

[1] I have glossed over or done damage to a number of others' thinking about these matters; the subject of graphical representations, their nature and uses, has a long history and a large literature (for example, two notable contributions in 17,171, and an incomplete random sampling of ideas in [10, 25, 26, 291].

[2] Running on a Macintosh computer, in Allegro Object Lisp. Development was partially funded by NET grant #DMC86-11357. [14]

[3] These concepts (and the inspiration for this example) come from an urban design study led by Serge Chermayeff at Yale University in 1967 [9]