13
Saying What It Is by What It Is *Like*
Describing Shapes Using Line Relationships

Milton Tan

Harvard University
Graduate School of Arts & Sciences / Graduate School of Design

Shapes - taken as well-defined collections of lines - are fundamental building blocks in architectural drawings. From doodles to shop drawings, shapes are used to denote ideas and represent elements of design, many of which ultimately translate into actual objects. But because designs evolve, the shapes representing a design are seldom static - instead, they are perpetually open to transformations. And since transformations involve relationships, conventional methods of describing shapes as sets of discrete endpoints may not provide anappropriate foundation for schematic design.

This paper begins with a review of the perception of shapes and its significance in design. In particular, it argues that juxtapositions and inter-relationships of shapes are important seedbeds for creative development of designs. It is clear that conventional representation of shapes as sets of discrete lines does not cope with these 'emergent' subshapes; the most basic of which arise out of intersecting and colinear lines. Attempts to redress this by using 'reduction rules' based on traditional point-and-line data structures are encumbered by computational problems of precision and shape specification. Basically, this means that some 'close' cases of sub-shapes may escape detection and their specifications are difficult to use in substitution operations.

The paper presents the findings of a computer project- Emergence 11 which explored a 'relational' description of shapes based on the concept of construction lines. It builds on the notion that architectural shapes are constructed in a graphic context and that, at a basic compositional level, the context can be set by construction lines. Accordingly, the interface enables the delineation of line segments with reference to pre-established construction lines. This results in a simple data structure where the knowledge of shapes is centralized in a lookup table of all its construction lines rather than dispersed in the specifications of line segments. Taking this approach, the prototype software shows the ease and efficiency of applying 'reduction rules' for intersection and colinear conditions, and for finding emergent sub-shapes by simply tracking the construction lines delimiting the ends of line segments.

Introduction

There are 'external facts', and we can say what they are. What we cannot say -because it makes no sense - is what the facts are independent of all conceptual choices.'

It goes almost without saying that the representation of architectural ideas on drawing - all the way from tentative doodles to precise 'working' drawings -depends considerably on the use of shapes. Ordinarily, shapes define edges or boundaries by using an integrated collection of lines and curves. They can be used to denote space- such as a room- or building elements-such as columns and beams -or, indeed, more abstract concepts like circulation patterns or spatial relationships.

If a designer produces shapes in a process analogous to a musician playing a musical composition, then shapes are little more than passive end-products. But shapes have a far more significant role in the design process than simply being 'static' building blocks for drawings; they are active ingredients in design transformation.

The basic idea behind design transformation is a simple one - designs evolve. Although the concept is simple, its implications are profound. The most significant, though subtle, is that design evolution requires not only that shapes be definable but that they be retrievable (in whole or part) for subsequent transformations.

This paper begins with an investigation into the demands of shape definition and recognition. It then deals with shape transformation, specifically in terms of parametric variation and shape substitution; leading to the problem - or opportunity - of emergent subshapes. Taking a relational approach, it presents a project to manage shape definition and recognition based on line relationships rather than the conventional end-point specifications of discrete lines. In this sense it attempts to describe shapes by what they are like, by-passing the encumbrance of specifying what they are.

Shapes - why we need to be able to define them

Shapes can be constructed using a variety of instruments and on different media; this does not exclude freehand, of course. But, unless there is access to their defining parameters, they will, in effect, remain fixed templates - actual or imaginary - and their reusability for shape transformation limited. The seriousness of this limitation is probably best contrasted by the richness of 'parametric variation'.

An example of parametric variation is the computer program *Miesing About*[2]. It encodes the parameters of the wall alignments of Mies's Brick Country House as variable line segments along the common edges of the 'spiraling' rectangles that characterize many Cubist work[3]. Figure 1 shows the wall configuration of the original house plan whilst Figures 2a shows a variation using the program.

Parametric variation is only a small aspect of shape transformation. As demonstrated in the program *TopDown*,[4] it also includes standard Euclidean operations

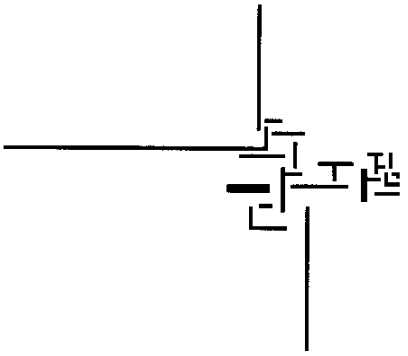

Figure 1 Reconstruction of the plan of the "Project for a Brick Country House (1923)" by Mies van der Rohe.

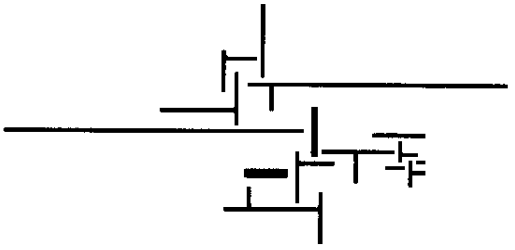

Figure 2a: Variation on Mies's Brick Country House produced using Miesing About.

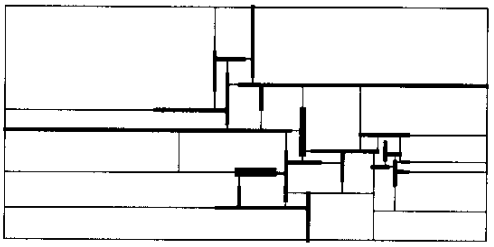

Figure 2b: The underlying rectangle 'alignments' for fig. 2a in Miesing About,

translation (moving of the whole shape), reflection, rotation and scaling - resulting in a wide repertoire of transformation methods to create instances of types.

But, parametric variation, particularly when localized to a part of a shape, can change its character considerably. Although this is precisely what one often desires, it can reach a point where, at best, the shape is better defined in some other terms or, worse, the shape looses its principal characteristic altogether. The first is illustrated by a triangle with a 'sliding' vertex[5]:

Although figures 3a and 3e are 'right-angled triangles', 3b and 3d are better classified as 'isosceles triangles' and 3c as another 'isosceles triangle' (or an equilat-

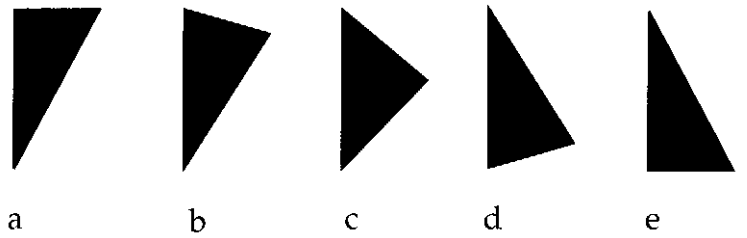

Figure 3: Triangle with a sliding vertex.

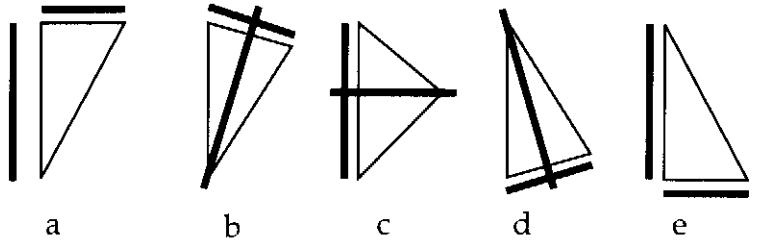

Figure 4: Dominant axes of the five triangles from figure 3.

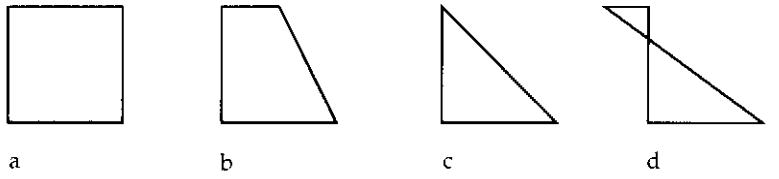

Figure 5: Parametric variation on a length of a side of a square.

era! triangle' if its three angles or sides are equal) as indicated in figure 4.

Taken further, parametric variation can be even more destructive. For example, figure 5 shows the effect of progressively reducing the length of a side of a square.

Initial reduction of the side transforms the original square into a trapezium but reduces to a triangle when the length reaches zero; the shape becoming two similar triangles when the side takes on a negative value.

So, what started out as an innocent move to create variations of a shape can end up in 'total' transformation from a shape to another. But, from a creative design point of view, this is by no means an undesirable thing! The practical implications of taking this seriously is, however, more sobering. Before going on to examine them, the complementary aspect of shape recognition needs to be addressed.

Shapes - why we need to be able to recognize them

Parametric variation is generally confined within a scheme - defined here as a collection of shapes. Although a shape's characteristics can drastically change, as noted above, the overall 'density' of the scheme remains largely the same, ie parametric variation does not normally change the number of constituents in a

shape or scheme. So, although parametric variation affords degrees of freedom for shapes to flex their profiles, it does not handle the more drastic transformations needed to bring schemes to different levels of resolution. These require shape substitutions.

Shape substitution is the mechanism to raise (or lower, as the case may be) the level of detail or elaboration of a design. It basically takes the context of a shape and introduces a new shape with or without the removal of the original. For a design to remain coherent, the substitute shape should, of course, enhance its new context. Figure 6 shows the substitution of the abstract capital of a classical column by a Tuscan capital using the program *Topdown*.

Shape substitution presupposes that the shapes to be substituted can be explicitly located in the first place. For a system which maintains a list of all the discrete shapes that make up the scheme, this is a simple task of searching the list for the required shape; that is, if all that you ever need to look for are shapes which have been explicitly defined previously. Unfortunately, that is not all the shapes that there are, and definitely not all the interesting ones!

Emergent Shapes

The human imagination is capable of finding recondite or unlikely relationships between seemingly disconnected things, then rallying support within itself to stabilize and defend them, often against all odds. There is a curious satisfaction in being able to seek out novel shapes from a familiar scene; indeed, if shapes are synonymous with problems, what characterize creativity more is not 'shapesolving' but shape-finding[6], in the simple case of two overlapping triangles - figure 7- the initial set of 'emergent shapes' are the contiguous closed sub-shapes of the scheme (figures 8a - Se); these can be combined in groups of 2 to 5 of the contiguous closed sub-shapes as shown in figures 8f - 8p.7 (There are, of course, another range of sub-shapes comprising 'open' figures with varying number of line segments; and yet another with disconnected juxtapositions of varying numbers of lines or closed shapes.)

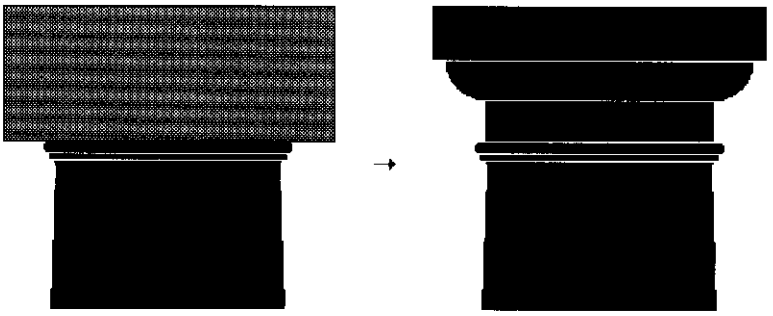

Figure 6: Substituting a Tuscan capital using *Topdown*.

Figure 7 Overlying two equilateral triangles

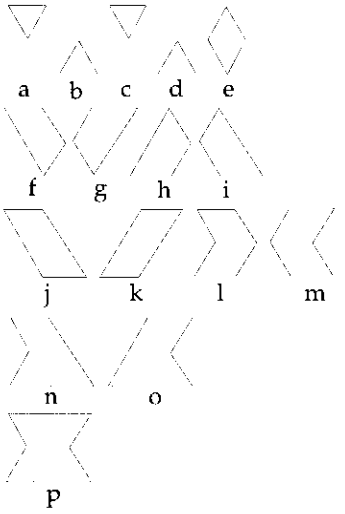

Figure 8 Closed sub-shapes from figure 7

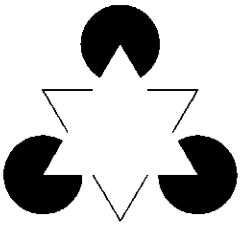

Figure 8b: 'Phantom Figure' by Bradley & Petry.

Gestalt psychologists have long maintained that there are 'natural' and more immediate recognition of certain types of shapes over others. But there is no compelling reason why they should be preferred by designers other than perhaps their simplicity; in fact, a creative tendency would be to seek 'non-gestalt' emergent shapes! In any case, emergent subshapes can be important to design and, specifically, to shape transformation and therefore cannot be ruled out of the repertoire of the shape-using designer.

There are many ways in which emergent sub-shapes are formed. In one extreme they may only be implied by some aspects of the geometry of other shapes as in figure 8b.

In another extreme, an emergent shape may be suppressed or enhanced through the use of labels - applied distinctions to categorize them differently from otherwise similar shapes. These labels can take different forms, eg the use of graphic styles to denote material represented, or the attachment of symbols. In this paper, however, the concern is for a more basic but nevertheless important representation of shapes: contiguous line segments connected at end-points or intersections as in a chain. At the heart of this are two fundamental line-relationship conditions:

1 Intersection
2 Colinearity

Emergent Shapes by Intersection

When two lines cross each other-an X-intersection - four emergent line segments are formed with the point of intersection and the end-points of the contributing lines (figure 9), resulting in a total of six lines.

In a T-intersection (a special case of an X-intersection), if an end-point of line B lies on line A, two emergent line segments are formed with the point of intersection and the end-points of A (figure 10), giving a total of four lines.

Emergent Shapes by Colinearity

There are four general conditions of colinearity which produces contiguous line segments (figure 11):

The 'reduction rules' for the intersection conditions produce *emergent shapes* by division whereas for the colinear conditions they are both by division and addition (or union). Either condition can easily be *created by* piecemeal construction of shapes or the manipulation of existing ones - eg by parametric variation and shape substitution.

In practice, the detection of the two conditions are not without problems; particularly in going beyond gross simplifications such as a confinement to orthogonal lines. One of the most serious of *these is* the precision needed to establish cases of line intersection and colinerity.

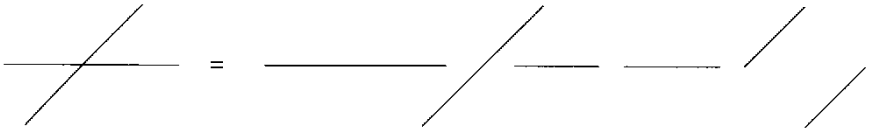

Figure 9: The X-intersection of two lines.

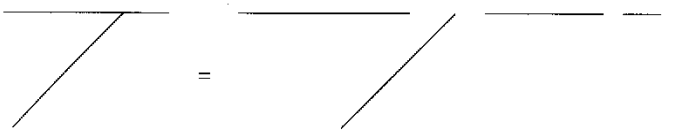

Figure 10: The T-intersection of two lines.

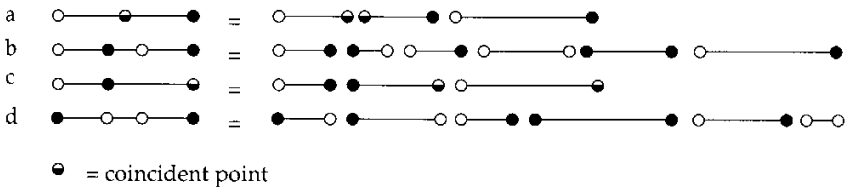

Figure 11: The four colinear conditions producing contiguous line *segments.*

Describing shapes

The popular approach in describing shapes[7] depends on the definition of a line segment, 1, in terms of a pair of end-points, pa and pb:

I = (pa, pb), where
pa = (xa,ya), and
pb = (xb,yb), and where
xa = x-coordinate of point a,
ya = y-coordinate of point a,
xb = x-coordinate of point b, and
yb = y-coordinate of point b.

Since there is a heavy reliance on the coordinates, the accuracy of determining whether an intersection or colinear condition exists in the first place, depends on how tolerances are compounded at the coordinate level - the bane of all who work with serious computer graphics! Often, an intersection should be recognized when it is, informally speaking, "close enough". The problem is illustrated in figures 12 and 13.

The problem diminishes, or even disappears, if manual intervention is permitted - typically by tracing all or part of a shape in question - as in the implementation by Chase (1989) where three points have to be selected on a shape so that a transformation can be mapped onto it. But the challenge, rather, is for a system to find the required shape automatically to satisfy the condition of the transformation rule.

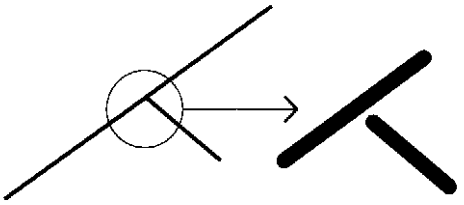

Figure 12: Is this an intersection?

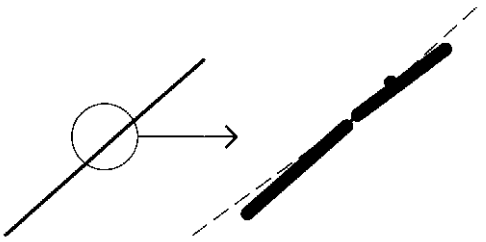

Figure 13: Is this a pair of coincident and colinear lines?

Emergence II

Emergence II was a project set up specifically to circumvent the tolerance issue in automatic shape recognition. Its point of departure from the current trend is to formulate a higher-level descriptor to define line segments (instead of resorting to a lower-level point-coordinate specification). The impetus for this comes from the hypothesis that relationships between lines, not its limits of specification, is the key to handling shape transformation.

The basic concept in Emergence II is that underlying every line segment is a construction line, as in traditional drafting. Moreover, each end of a line segment is marked by an intersection with another construction line. Therefore, a line segment - SLine - is simply defined in terms of three construction lines - CLines:

A line segment, Sa = (C1, C2, C3), where

C1 is the the 'host' construction line and

C2 and C3 are the delimiting construction lines.

The interface enables the creation of 'maximal' construction lines by clicking the 'mouse' at two points in the screen. Line segments are created by using the mouse to pick a 'host' construction line followed by two delimiting construction lines.

Then, a construction line is defined conventionally using a codescriptor, cd, in turn defined as the description of the equation of the CLine:

cd(C1) = (x), if the line Cl is vertical; otherwise

cd(C1)= (s,i), where s and i are the slope and y-intercept, respectively, of the CLine Cl according to the line equation y = sx + i.

To reduce rounding errors, the slope, s, is maintained as a pair of integers, rn and rd-its numerator and denominator (reduced to its primitive form using Euclid's algorithm for the greatest common denominator):

s = (rn, rd)

The comparison of slopes, s(l1) and s(l2), for equality is simply a direct test of equality of the corresponding two integer values, ie (s(l1).rn = s(l2).rn) and (sl1).rd = s(l2).rd).

Only four basic functions are needed to handle emergent shapes

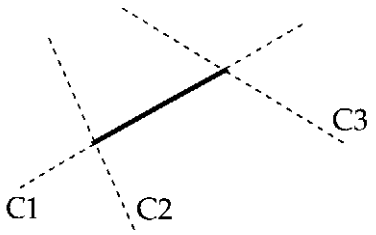

Figure 14: Definition of a linesegment in terms of a reference line (CI) and two delimiting lines (C2 & C3).

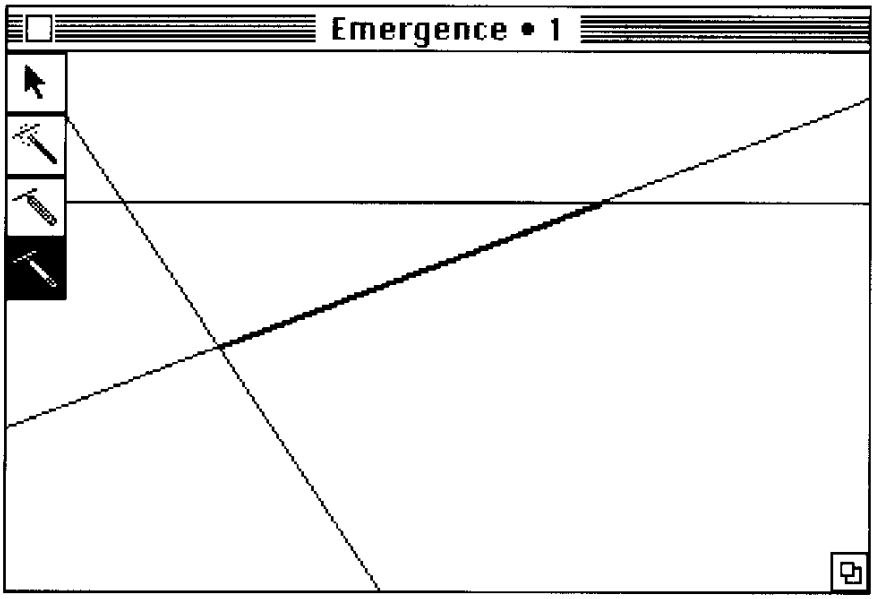

Figure 15: A 'window' in Emergence II, showing the construction of a line segment (thick line) with reference to three construction lines.

Function 1: Point (11,12)
If the coordinates of a point of intersection is needed, it can be determined by solving the equations of the two contributing lines, H and 12.

Function 2: Coincident (p. I)
The test for a point, p. lying on a line segment, 1, returns true if the coordinates of p satisfies the line equation of 1 and that its x-coordinate lies between the x - coordinates of the intersections between the construction line of land its delimiting
construction lines.

Function 3: Intersection (II, 12)
The test for an Intersection (both X and T) between SLines l1 and l2 returns true if the following are true:

a Thehost CLines are not the same:
CLine (11,1) ^ Cline (12,1)
b The point of intersection between H and 12 are coincident with l1 and l2:
(Coincident (Point (l1, l2), l1)) and (Coincident (Point (l1, l2),l2)).

If an intersection exists, a total of 4 possible emergent Shoes are produced:

(CLine (l1, 1), Cline (l2, 1), CLine (l1, 2))
(CLine (l1, 1), CLine (l2, 1), CLine (l1,3))
(CLine (l2, 1), CLine (l1, 1), CLine (l2, 2))
(CLine (l2, 1), CLine (l1, 1), CLine (l2, 3))

For a T-intersection (Ic when a delimiting CLine = a host CLine) two of these are eliminated.

Function 4: Co/incur (11, 12) The Colinear function returns true if the following are true:

a 11 and 12 share the same host CLine:
CLine(l1, 1) = CLine(l2, 1)

b One of the CLines of 11 is the same as one of the CLines of L2:
CLine(l1, 2) = CLine(l2, 2) or
CLine(l1, 2) = CLine(l2, 3) or
CLine(l1, 3) = CLine(l2, 2) or
CLine(l1, 3) = CLine(l2, 3)

This produces the emergent line segment:
(CLine (l1, 1), CLine(l1, 3), CLine (l2,3)) or
(CLine (l1, 1), CLine(l1, 3), CLine (l2,2)) or
(CLine (l1, 1), CLine(l1, 2), CLine (l2,3)) or
(CLine (l1, 1), CLine(l1, 2), CLine (l2,2)), respectively

Functions 3 and 4 show how emergent line segments are specified entirely with reference to the construction lines which define the 'parent' lines. In cases involving more than two lines, recursive applications of the process checks for all combinations of emergent line segments to be enumerated.

So far, only the reduction process is complete. The next phase involves a search for the required shape. The interface enables the shape to be defined as a list of contiguous line segments in the following form (for an x-sided shape)':

(L1.length, LI angle L2.length, L2.angle L3.length, L3.angle
… Lx.length, Lx.angle)

The search procedure then merely attempts to match the definition by testing each item on the list against contiguous line segments in the scheme. Since the definition of every line segment has references to the delimiting lines the next potential line segment will have a host construction line similar to one of them. Because the reduction process has taken into account the intersection and colinear cases, the search includes emergent shapes.

The 'parser' developed to handle this shape definition can accept either an absolute or relative value for either the length or angle entries, as well as arithmetic expressions"', eg

'128,45' specifies a line segment 128 units long and at 45 degrees (clockwise from vertical).

'a,30' specifies a line of any length, a, at 30 degrees, if 'a' is encountered for the first time in the list, otherwise 'a' takes on a value assigned to it when it was matched to the length of a test line for the first time.

´b*2,x+180´ specifies a line of length 2 x b and at the direction x+180 degrees (ie opposite to direction x degrees).

Future plans to develop the search-shape specification will most likely move away from this rather rigid sequential declaration towards a more hierarchical system capable of responding to priorities. Among other things, it would be better suited to handle disjoint and 'network' shapes. Also, it has been anticipated that the interface should enable a graphic method of specifying a search shape, at least at the outset; the lowest response of which is the 'reverse' of the present drawing and search process.

Conclusion

Describing shapes in terms of how their component lines relate to each other has the advantage that 'b-level' calculations involving coordinates of points can be effectively postponed until they absolutely matter. It also enables a significant amount of searches and tests necessary for shape recognition - particularly for emergent shapes - to be pre-processed. This approach suggests that intermediate representation and formal structures of shape description can be usefully exploited to make the computation of shape transformation and combination more tractable.

Acknowledgements

My thanks to Professor Bill Mitchell (Harvard Graduate School of Design) for his encouragement and constant source of good ideas; also to A/Professor Mark Friedell (Harvard Graduate School of Arts and Sciences - Computer Science) for putting up with this project!

Notes

I Putnam 1987.

2 Tan 1987.

3 Compare the parametized cruciform method for the same project in Mitchell, Liggett & Kvan (1987); p198.

4 Ref. Mitchell, Liggett & Tan 1988, 1989.

5 After Arnheim (1974); pp93 & 94.

6 Csikszentmihalyi (1987) concluded from studies on creativity that "problem-finding" rather than problem-solving is more characteristic of creative lives.

7 For a formal treatment of ambiguous forms' ref. Stiny 1989.

8 In the context of shape grammar, see Krishnamurti 1980 and 1981; for a computer implementation see Chase 1989.

9 cf shape 'script' in Nagakura 1989.

10 For an example of this technique, refer to Winston & Horn (1989); pp 353-356, 455-470.

References

Arnheim R, 1974, Art and Visual Perception. University of California Press, Berkeley & Los Angeles. (Originally published 1954.)

Chase S, 1989, "Shapes and Shape Grammars; from mathematical model to computer implementation." Environment and Planning B; Planning and Design, v16, pp 216-242.

Krishnamurti R, 1980, "The arithmetic of shapes." Environment and Planning B; Planning and Design, v7, pp463-484.

Krishnamurti R, 1981, "The construction of shapes." Environment and Planning B; Planning and Design, v8, pp5-40.

Krishnamurti R, Giraud C, 1986, "Towards a shape editor; the implementation of a shape generation system." Environment and Planning B: Planning and Design, v13, pp391-404.

Mitchell W, Liggett R & Kvan T, 1987, The Art of Computer Graphics Programming. Van Nostrand Reinhold, NY.

Mitchell W, Liggett R & Tan M, 1988, "The Topdown System and its Use in Teaching - an exploration of structured, knowledge-based design." Proceedings of the Association of Computer-aided Design in Architecture Workshop '88.

Mitchell W, Liggett R & Tan M, 1989, 'On Teaching Designers to Program and Programmers to Design." Proceedings of the 1989 CAAD Futures Conference. MIT Press, Cambridge, MA.

Nagakura T, 1989, "Shape Recognition & Transformation - a Script-based Approach". Proceedings of the 1989 CAAD Futures Conference. MIT Press, Cambridge.

Putnam H, 1987, Many Faces of Realism. Open Court, LaSalle, II.

Stiny G, 1989, "What Designers Do that Computers Should". Proceedings of the 1989 CAAD Futures Conference. MIT Press, Cambridge, MA.

Tan M, 1987, "Miesing About - a computer program to investigate parametric variation of the 'Project for a Brick Country House', 1923, by Mies van der Rohe." Unpublished project.

Winston P & Horn, 1989, LISP (3rd. edition). Addison-Wesley, Reading, MA.