Environment and Planning B: Planning and Design, 1986, pages 391 - 404

Towards a shape editor: the implementation of a shape generation system

R Krishnamurti
EdCAAD, University of Edinburgh, 20 Chambers Street, Edinburgh EH1 1JZ
C Giraud
Institut International de Robotique, et d'Intelligence Artificielle de Marseille, 2 rue Henri-Barbusse-CMCI, 13241 Marseille
Received 4 July 1985; in revised form 22 March 1986

Abstract. A fundamental problem in editing shapes is the recognition of partial shapes in a drawing to which changes are to be made. In this paper the possibility of using shape rules as a mechanism for effecting such changes is explored. Shape rules represent spatial relationships between two shapes α and β with the interpretation that any instance of α in a shape can be replaced by a 'similar' instance of β. A shape generation system implemented in PROLOG is described.

Introduction

As the title suggests, this paper is concerned with the implementation of a shape generation system. However, the implementation details in themselves are of relatively little importance. The motivation for this paper stems from an interest in the problematic issues associated with 'shape editing', and how they may be approached within a shape generation system.

Shape editing is difficult to define precisely and no definition will be proffered in this paper. This is partly because of our present limited understanding of shapes. However, it is worthwhile briefly to examine editors in general. There are two main aspects to an editor. First, the subject of a typical editor in any domain—namely, some sort of 'document', be it a manuscript, a programme, a table, a memo, or a drawing etc—may be regarded as satisfying certain rules of form and meaning that we can loosely classify as the particular 'document rules'. In other words, we may state as a belief that document construction—spatial or otherwise—can be set within an environment that is based on some rules.

Second, the tasks associated with an editor include the creation, deletion, change, arrangement, and copy of documents (Meyrowitz and van Dam, 1982). These tasks involve some process of recognition and some process of alteration. Consider interactive text editors as implemented in computers. Alteration to text is basically some form of text replacement. Text recognition, in its simplest form, corresponds to some form of pattern matching. However, in principle, recognition of text can be driven by other considerations that rely on knowledge about either the form or the content of the document. In other words, recognition may be carried out either syntactically or semantically. Of the two, semantic recognition is much harder. As an extreme example, imagine trying to scan for a piece of text that has the same 'meaning' as the given text!

By analogy, a shape editor is feasible only when the problems of shape recognition and shape replacement are resolved—albeit partially. A shape editor deals with drawings. Drawings are a medium through which people communicate. Drawings reflect, in part, a person's perception of a spatial problem and, in part, his or her conception of a spatial solution. Drawings represent pictorial descriptions of abstract (and perhaps nonspatial) relationships. The relationships between the

shapes in a drawing and the nonspatial descriptions to which they refer constitute the semantics of the drawing. Drawings often materialise through a trial and error sequence with each successive drawing arising out of changes made to the preceding drawing. These changes generally correspond to spatial alterations—for example, the introduction of a wall into a plan, or a rearrangement of room spaces within a plan. These changes involve some sort of shape replacements.

Let us clarify this. Imagine a person sitting in front of a drawing board or a graphics terminal. Let us suppose that the person initially produces an outline drawing. Then, finer details are gradually introduced into the drawing. The details may be replicated in various parts of the drawing. The details may be subjected to further spatial editing either globally or locally. The details may of course be associated with partial semantic information, often illustrated through text in the drawing.

Although conventional graphics systems perform these and other tasks reasonably, they do have their drawbacks. Such systems typically employ representations for objects that distinguish between a segment[1], an object made up of segments, assemblies made up of sets of objects, and so on. Composite objects are named. Composite objects are essential if they are replicated in the various parts of the drawing. Two principal problems, when one attempts to edit drawings, can be identified. First, changes made to composite objects tend to have a global effect. That is, the changes are manifested in every occurrence of the object in the drawing. Second, it is difficult to recognise and modify just parts of objects. The problem is important in situations where semantic links are attached to those parts of the shape. For instance, 'the corner of a room' may have to be identified in order to carry out a spatial consistency check.

The concept of spatial relations (Stiny, 1980b; Earl and Krishnamurti, 1984) promises a mechanism whereby such shape editing tasks may be carried out. In its simplest form a spatial relationship can be expressed as a shape rule relating two shapes, α and β. Figure 1 shows examples of edit rules for tiling the plane.

A shape rule is interpreted as follows. Any occurrence of α in a specified shape under some transformation can be replaced by an occurrence of β under the same transformation. Shape rules work in much the same way as string replacement rules in text processing, except that they have also to take into account the possible geometric transformations associated with α and β. Figure 2 illustrates shape rule application.

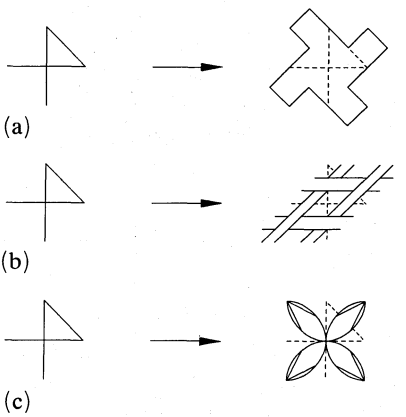
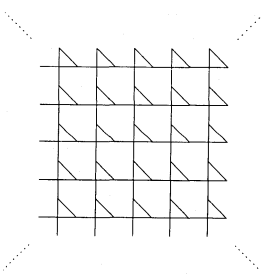


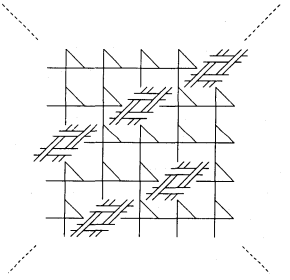
Figure 1. Examples of edit rules for tiling the plane.

[1] A segment is either a line or a curve between two finite points.

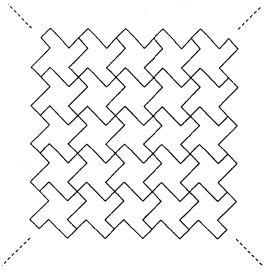
In this paper we illustrate the application of shape rules using the above interpretation in the editing of shapes. For ease of explanation we cast shape rules in a shape grammar formalism and describe an experimental implementation of a shape generation system.

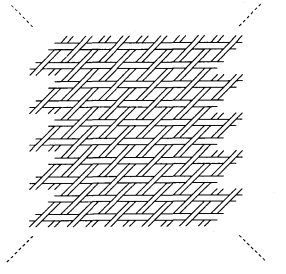


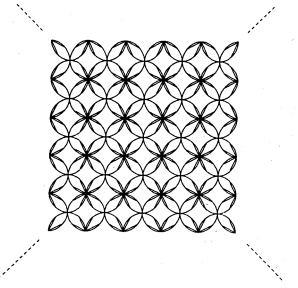
a fragment of a marked tiling of the plane



shape rule [figure 1(b)] applied to specified tiles







shape rules applied globally to all tiles

Figure 2. Application of tiling rules.

Shape and shape representation

Following Stiny (1980a) a *shape* is made up of a set of line segments and a set of labelled points, the elements of which each correspond to a point associated with labels chosen from a specified vocabulary. The labels have no geometric import. A shape is defined with respect to a cartesian coordinate system, which allows us to 'fix' the shape in some Euclidean space. The empty shape contains no line segments or labelled points. There are no spatial constraints imposed on the line segments or labelled points in a shape. The line segments do not have to enclose a region. Nor do they have to be connected in any manner. This definition of a shape does not preclude the existence of a vocabulary of shapes, since any composition from the vocabulary of shapes can be represented by a set of line segments. We do not, however, consider parametric shapes, since these pose some technical problems a discussion of which is beyond the scope of this paper.

A *line segment* l, $l = \langle t(l), h(l) \rangle$, is given by its end points $t(l) = [x(t), y(t), ...]$, $h(l) = [x(h), y(h), ...]$, that are respectively referred to as the *tail* and *head* of l. The end points of a line segment can be rearranged so that $t(l) < h(l)$, where $<$ denotes the order relation defined as follows. Let a, $a = (a_1, a_2, ..., a_n)$, and b, $b = (b_1, b_2, ..., b_n)$, be two n-tuples of numbers. Then, $a < b$ whenever there is a k such that $a_k < b_k$, and $a_j = b_j$ for all $j < k$. A line segment l *contains* a line segment m whenever the end points of m coincide with points on l. A line segment in a shape is *maximal* whenever no other line segment in the shape contains it. Without loss in generality we may assume that a shape is described by its maximal line segments. It is straightforward to convert any set of line segments into its unique set of maximal line segments.

A line segment can be associated with a *descriptor* that describes the (infinite) line on which it is defined. The line descriptor partitions the set of line segments in a shape into equivalence classes each of whose elements are made up of *collinear* line segments, as illustrated in figure 3. The choice of the descriptor is immaterial provided that two noncollinear lines have distinct descriptors. Typically, a line

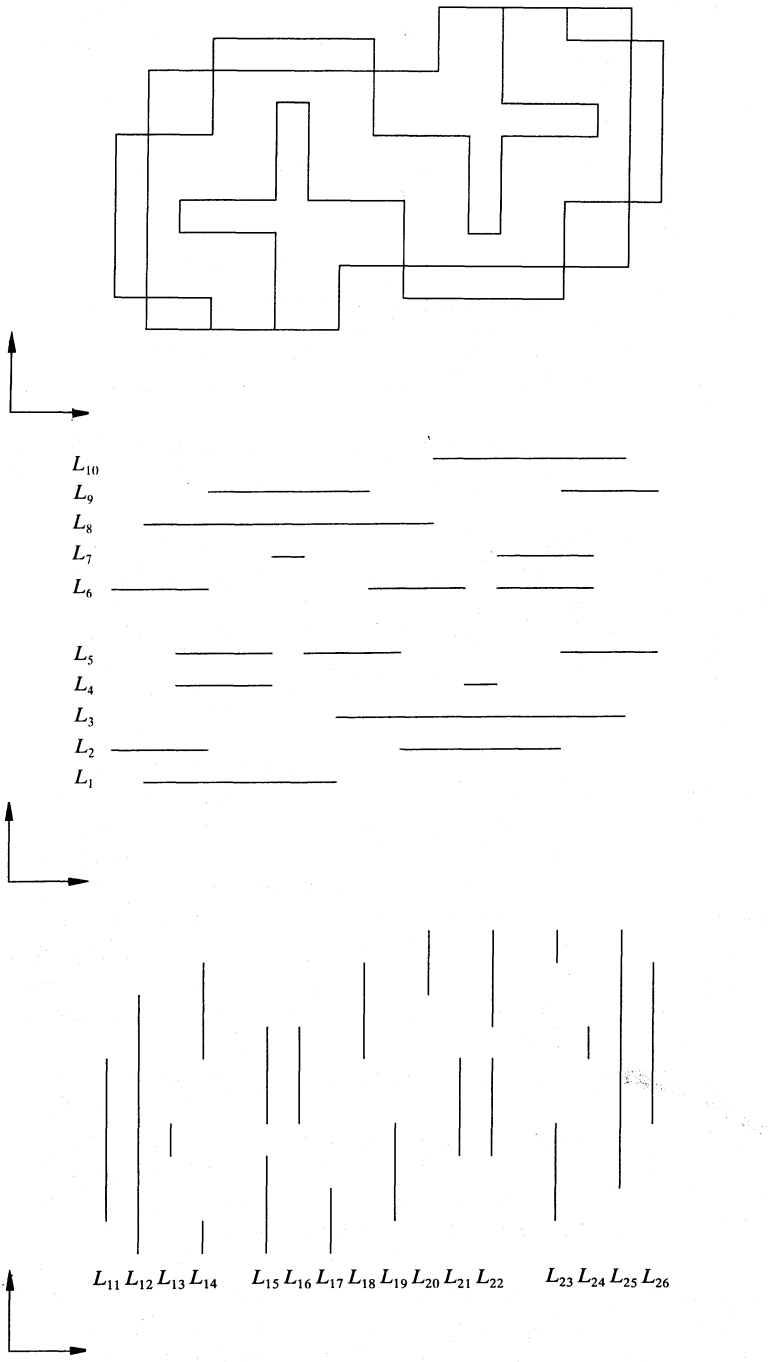


Figure 3. Decomposition of a shape into disjoint sets of collinear maximal lines (L_i are equivalence classes of line segments).

descriptor is made up of two parts, one that gives the direction of the line and the other the moment of the line.

A labelled point (p,A) is made up of a point p and a set of labels A. No two labelled points share the same point; otherwise, the labelled points (p,A) and (p,B) can be combined into a single labelled point $(p,A+B)$, where $+$ denotes set union.

In addition to the lines and labelled points, a shape may be tied to a state that associates the shape with certain aspects or properties. A state may be a collection of statements about the shape that hold. The statement may be a label, a predicate, or, in general, a description. Labels as state designators were introduced by Knight (1983).

Thus, a shape $\sigma, \sigma = \langle S,P \rangle$, can be represented as the pair of sets S,P;

$$S = \{(d_1,L_1), \ldots (d_m,L_m)\},$$

where d denotes a line descriptor and L a set of collinear line segments;

$$P = \{(p_1,A_1), \ldots (p_n,A_n)\},$$

where p denotes a point and A a set of labels. A state Q consisting of a set of statements can be associated with a shape σ. When Q is nonempty, σ is said to be *in* state Q, denoted $Q(\sigma)$.

The representation of a shape given above can be further organised so that the descriptors satisfy $d_1 < d_2 < \ldots$, and $p_1 < p_2 < \ldots$. Each set of collinear line segments $\{l_1,l_2,\ldots\}$ can be arranged to satisfy $h_1 < t_2, h_2 < t_3, \ldots$.

Rearranging a shape in this fashion does not affect the presentation of the shape. On the other hand, it provides a simple 'pattern matching' criterion by which equality of shapes can be easily determined. Indeed, two shapes are (pictorially) *equal* whenever their representations are identical. Moreover, this method for representing shapes provides for efficient shape algorithms that have time bounds linear in the number of maximal line segments in the shape.

Rational shapes

We now introduce a restriction on the class of shapes that are dealt with in this paper. This restriction—albeit a practical one—is necessitated by the fact that algorithms are defined with respect to a computing machine. In a random access machine a real number is represented by a finite approximation determined by the word size of memory. This makes for inexact arithmetic. Consequently, exact transformations for general shapes cannot be determined. Since correct algorithms require exact arithmetic it is necessary to consider just those shapes that can be so described. Therefore, our attention is directed to shapes that are, in the mathematical sense, rational.

There is a theoretical justification for restricting the class of shapes that are considered. Stiny (1975) has demonstrated the equivalence of rule-based shape generation systems to general computation (Turing) machines. Consequently, for any shape procedure guaranteed to 'halt', the corresponding computation program must be shown to halt. Since the halting problem for comparing any pair of real numbers is undecidable it follows that determining equality of general shapes is undecidable.

Moreover, most graphics devices use a limited number of pixels each of which is addressed by a pair of integral coordinates. Inevitably, all shapes displayed by these devices are rational. More recently, Earl (1985) has provided additional arguments about the value of rational shapes in a design context.

The following definition makes the notion of a 'rational shape' precise.

A point p is rational whenever each of its coordinates $x(p)$, $y(p)$, ... can be expressed as a ratio of two integers. A labelled point (p,A) is rational whenever p is rational. A line segment is rational whenever both its end points are rational. A shape is rational whenever its maximal line segments and labelled points are rational.

Any ratio n/m of two integers, expressed as the ordered pair $\langle n,m\rangle$, may be uniquely described by its primitive form. A pair of integers $\langle n,m\rangle$ is *primitive* whenever the following conditions are satisfied:

(a) n and m are integers,

(b) $m \geqslant 0$,

(c) n and m are relatively prime—that is, there is no integer a greater than 1 such that a divides both n and m.

When $n < 0$, the primitive is said to be negative.

Thus an integer n is described by its primitive $\langle n,1\rangle$; infinity by the primitive $\langle 1,0\rangle$; and zero by the primitive $\langle 0,1\rangle$. Any other rational number, n/m can always be reduced to its primitive form by applying Euclid's greatest common denominator (gcd) algorithm. The following procedure outlines the steps involved in determining the primitive form for the ratio n/m of any two integers. Let $a = |\gcd(n,m)|$, where $|q|$ denotes the absolute value of the number q. Let $s = |m|/m$. Then, the primitive form of the ratio of the two integers, $r = \langle n,m\rangle$, is given by $\text{prim}(r) = \langle sn/a, |m|/a\rangle$.

Primitives allow us to compare two numbers for equality. Two rational numbers r_1 and r_2 are *equivalent* whenever their primitives $\text{prim}(r_1)$ and $\text{prim}(r_2)$ are identical. Thus, equality of numbers is reduced to 'pattern matching'.

The above formulation for primitive can be extended to k-tuples, $k > 1$, of ratios of integers. It is easy to demonstrate that a k-tuple of rationals can always be uniquely represented by a $(k+1)$-tuple of integers.

Boolean operations on shapes

The *union* of two shapes is the shape consisting of the union of the sets of line segments in both shapes and the union of the sets of labelled points in the two shapes. The *intersection* of two shapes is the shape consisting of line segments in common with both shapes and the labels that share the same points in both shapes. The *difference* between two shapes is the shape consisting of just those line segments in the first shape not occurring in the second shape, and the labelled points in the first shape not in the second shape, together with the set difference of them. The *subshape* relationship between two shapes holds whenever each line segment in the first shape is contained in a line segment in the second shape and for each label in the first shape there is a corresponding label in the second shape and both labels share the same coordinates. Implicit in the above definitions is the fact that every segment contained in a maximal line segment of the shape is a line segment in the shape.

It is possible to extend the definitions of the Boolean operations to representations of shapes to yield the representation of the resulting shape. To do this we must define Boolean operations on line segments. The first step is to notice that to perform any of the operations we need consider only collinear line segments, that is, line segments that share the same descriptor. So, whenever we compare two collinear maximal line segments $l_1 = \langle t_1,h_1\rangle$ and $l_2 = \langle t_2,h_2\rangle$ from the two shapes respectively, we have one of three situations to consider:

(a) $h_2 < t_1$,

(b) $h_1 < t_2$,

(c) $h_1 \geqslant t_2$ and $h_2 \geqslant t_1$.

The three cases are illustrated in figure 4.

Cases (a) and (b) correspond to the situation when the line segments are disjoint. The last and more interesting case (c) corresponds to the situation when the two line segments share either a common point (when $h_1 = t_2$ or $h_2 = t_1$) or a common line segment (when $h_1 > t_2$ and $h_2 > t_1$). It is case (c) we must consider when performing the Boolean operations on line segments. Suppose this is the case.

The union of two line segments is the line segment $l = \langle t, h \rangle$, where $t = \min(t_1, t_2)$ and $h = \max(h_1, h_2)$. min and max respectively refer to the minimum and maximum value of two given values.

The intersection of two line segments exists only when the two line segments overlap. That is, when $h_1 > t_2$ and $h_2 > t_1$. Then, the common line segment is given by $l = \langle t, h \rangle$, where $t = \max(t_1, t_2)$ and $h = \min(h_1, h_2)$.

The difference of two line segments is a bit more tricky since difference may yield zero, one, or two line segments. Again, difference of line segments need be performed only when the segments overlap. For ease of explanation we introduce the notion of an 'empty' line $l = \langle p, p \rangle$, for any point p, and the function 'nonempty' that produces the nonempty elements in a list. Then, the difference of two line segments is the list given by nonempty($\langle t_1, t \rangle$, $\langle h, h_1 \rangle$), where $t = \max(t_1, t_2)$ and $h = \min(h_1, h_2)$.

The above operations can be incorporated into procedures to perform the appropriate Boolean operation on two ordered lists L_1 and L_2 of collinear line segments. Consider the problem of showing L_1 is a subshape of L_2. It goes without saying that lists L_1 and L_2 share the same line descriptor. Let us select the first segment in L_1, say l_1, and successively compare it with segments in L_2. Consider one in particular, l_2. When we compare two line segments l_1 and l_2, we have as before the three cases given above.

If case (a) holds, the line segments are disjoint and we select the next line segment l_2 from L_2 if it exists; otherwise the subshape relationship does not hold.

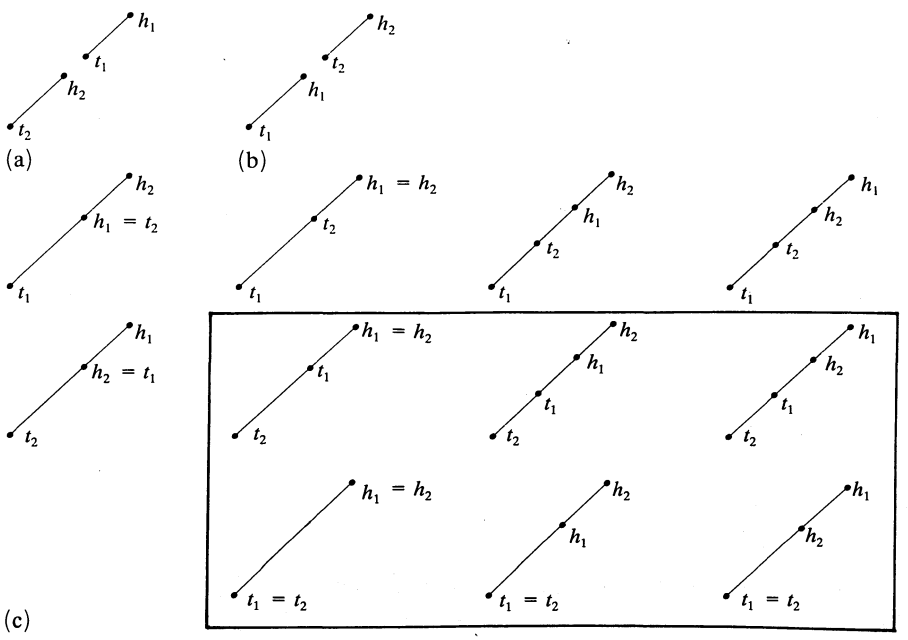


Figure 4. End-point conditions for a pair of collinear line segments $l_1 = (t_1, h_1)$ and $l_2 = (t_2, h_2)$ with (a) $h_2 < t_1$, (b) $h_1 < t_2$, (c) $h_1 \geqslant t_2$ and $h_2 \geqslant t_1$. The cases within the box correspond to those for the subshape relation to hold.

If case (b) holds then the subshape relationship does not hold. This is because there is no line segment in the list L_2 preceding l_2 that shares a common line segment with l_1. If that had been the case we would have had either case (a) above or case (c) below. Notice that for every line segment $\langle t(l), h(l)\rangle$ succeeding l_2 in list L_2, $h_1 < t(l)$.

If case (c) holds, the line segments overlap, in which case the subshape relationship can hold only if $t_2 \leqslant t_1$ and $h_1 \leqslant h_2$. In this case, we select a new l_1 and compare it with l_2 and the line segments succeeding l_2 in list L_2.

The Boolean operations for the labelled points correspond to the conventional set operations.

Shape rules

Shape rules as considered in this paper were originally defined by Stiny and Gips (1972) in their seminal paper on shape grammars. A *shape grammar* is an algorithm described in terms of shapes with labels. In it standard form it consists of *shape rules* and an *initial shape*. A shape rule, $\langle \alpha, Q(\alpha)\rangle \rightarrow \langle \beta, Q(\beta)\rangle$, consists of two shapes, α in state $Q(\alpha)$ and β in state $Q(\beta)$. The shape rules are used to change a given shape, namely the current shape γ in state $Q(\gamma)$, into a new shape whenever $Q(\alpha)$ is a subset of $Q(\gamma)$ and there is a transformation τ that makes α a subshape of γ; that is, whenever $Q(\alpha) \subseteq Q(\gamma)$ and $\tau(\alpha) \subseteq \gamma$, where $\subseteq$ denotes a containment relationship—namely, subset or subshape relationship. In this case, the subshape may be replaced by the same transformation of β. That is, the new shape, γ^* is given by the shape expression

$$\gamma^* \leftarrow \gamma - \tau(\alpha) + \tau(\beta) .$$

The state of γ^* is given by

$$Q(\gamma^*) \leftarrow Q(\gamma) - Q[\tau(\alpha)] + Q[\tau(\beta)] .$$

The operators $+$ and $-$ denote union and difference, be it shape, set, or state, and clearly depend on the context in which each operator is used. For instance, if the state is given by a set of labels, the state operations correspond to set operations. However, if the state is given by, say a predicate that corresponds to the sentence 'this rule may only apply under translation and scale and if the current shape has a specific state label', the operations of $+$, $-$, and $\subseteq$ become harder to define precisely though they may not prove difficult to interpret. For our purpose, we may assume that states are defined by sets of labels or symbols.

Shape rules are applied to the initial shape and to shapes produced from it. In this way, a *language* of shapes may be specified by generating its individual members.

It is customary to assume that terminal shapes—that is, shapes in the language—are unlabelled. In other words, labels are nonterminal symbols used to direct the generation process. It is possible to define shape grammars in which the rule shapes have no labels. A good exposition on the role of labels in shape rules is provided by Knight (1983). Briefly, labels in a shape rule supply additional information as to how, where, or when a shape rule may be applied to the current shape. An example of the use of labels to guide shape rule application is shown in figure 5.

Labels may specify the Euclidean transformations under which a rule can be applied to a subshape of γ similar to α. That is, labels can be used to alter the symmetry of the shape in α and thus alter the subshapes of γ similar to α.

Labels may specify to which subshape or subshapes of γ similar to α a rule may be applied. In this case, labels do not alter the symmetries of the shapes in α and of the subshapes of γ similar to α. Thus, they do not in general restrict the Euclidean transformations under which the rule may be applied.

Last, the labels may be used to indicate the state in the generation process. Such state labels typically occur in large grammars when the set of rules may be partitioned into subsets of shape rules and each subset of the rules permits the generation of shapes that serve as initial shapes for another partition of the rules. In other words, they mark the successive stages in the generation procedure. State labels are not necessarily associated with a coordinated point. Sometimes, labels may carry semantic import in that they may name spaces in the drawing created by line segments to which certain shape rules apply and others do not. A good example of the use of such labels can be found in Downing and Flemming's (1981) description of a group of early 1900 bungalow types built in Buffalo. The labels are chosen so that, for example, details that apply to kitchens will never be applied to bedrooms, and vice versa. The details are represented as shapes in shape rules.

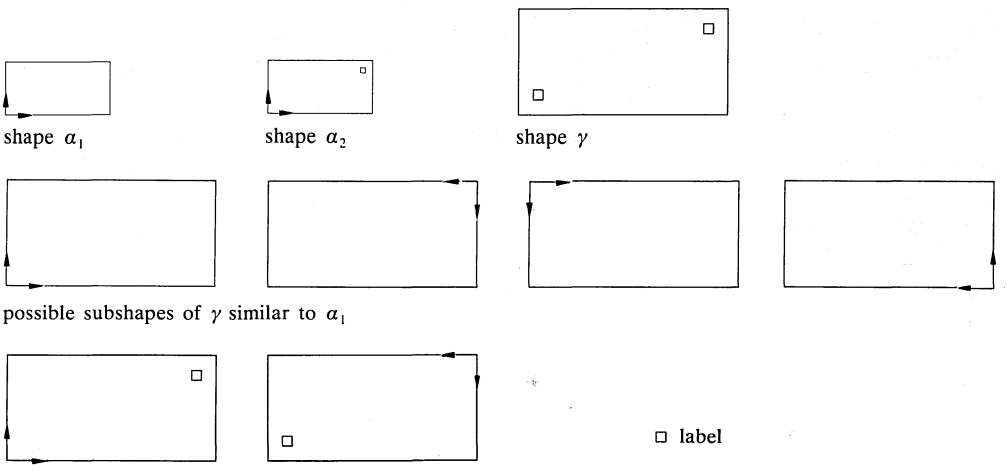


Figure 5. The use of labels to guide shape rule application.

Subshape recognition

The crucial step in the application of a shape rule is the recognition problem—that is, determining whether the shape rule applies to the given shape. In general, there may be several subshapes in a given labelled shape to which a given shape rule may be applied. Thus, γ may contain subshapes $\gamma_1, \gamma_2, \ldots$, each of which is similar to α. In other words there may be a list of transformations $\tau_1, \tau_2, \ldots$ such that $\tau_j(\alpha) = \gamma_j \subseteq \gamma$.

We consider the problem of determining one such transformation. The easiest way to look at this problem is to consider the inverse problem. Suppose we assume α is similar to a subshape of γ under a given transformation τ. That is, we are given the correspondence between points and line segments of α and points and segments of γ. We want to find a computation for τ. In other words we want a procedure to determine the coefficients of τ. We now present a simpler adaptation of the procedure given elsewhere (Krishnamurti, 1981) using homogeneous coordinates.

Homogeneous coordinates provide a unified approach to the transformations of translations, rotation, scale, reflection, and finite compositions of these. Homogeneous coordinates have the added advantage that they reduce the arithmetic of rationals to the arithmetic of integers as shown below. For the remainder of this paper, we restrict our attention to two-dimensional shapes and to similarity transformations.

Any point (x,y) can be represented by the homogeneous coordinates (x',y',w), where w is usually 1, $x = x'$, and $y = y'$. Thus, any rational point $(x_n/x_d, y_n/y_d)$ can be described by the homogeneous coordinates (X,Y,W), where $X = x_n y_d$, $Y = y_n x_d$, and $W = x_d y_d$. This homogeneous point can be reduced to its primitive point by dividing each coordinate by $\gcd(x_d, y_d)$.

The general Euclidean transformation, combined with a scale transformation, is described by the 3×3 matrix

$$\begin{bmatrix} ax & bx & cx \\ ay & by & cy \\ 0 & 0 & 1 \end{bmatrix} .$$

Since we are dealing with correspondences between rational quantities, we need consider only transformations with rational coefficients. These coefficients can be replaced by their primitive equivalents.

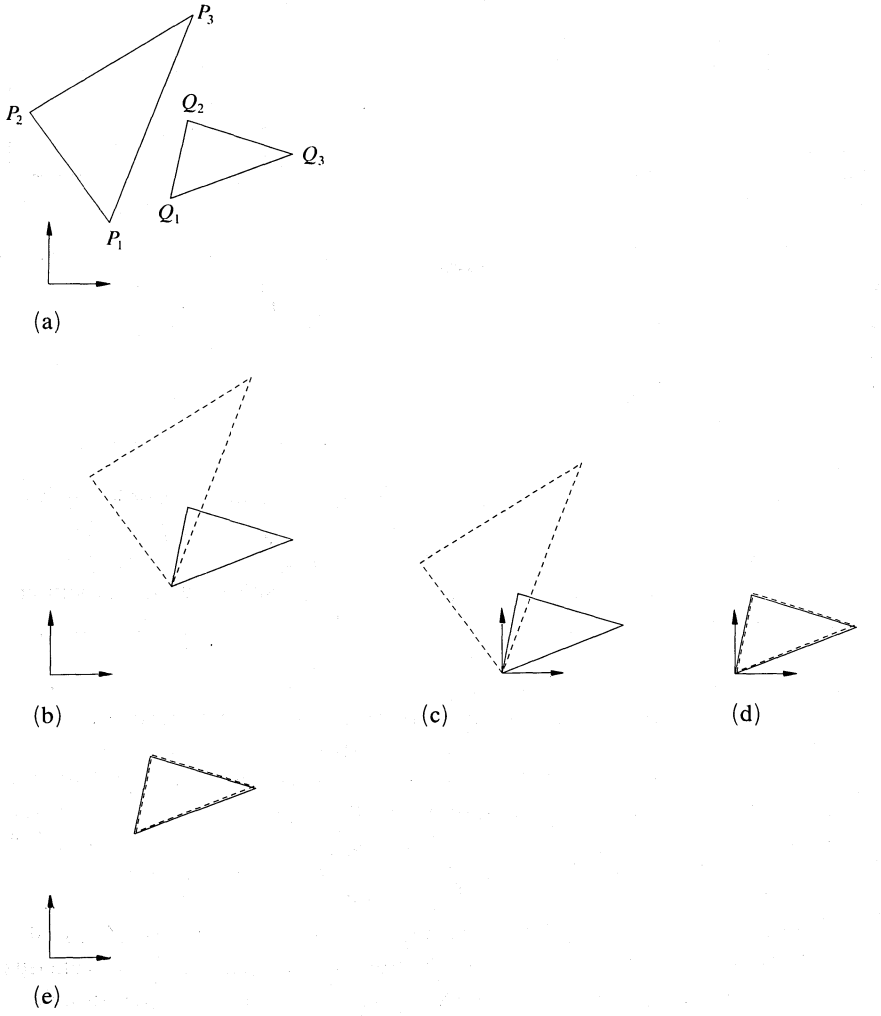


Figure 6. Steps in determining the transformation that takes a triangle to a corresponding similar triangle. (a) The points P_1, P_2, P_3 and Q_1, Q_2, Q_3, (b) Translate P_1 to Q_1, (c) Translate Q_1 to origin, (d) Rotate and scale P_2, P_3 onto Q_2, Q_3, (e) Translate back to Q_1.

The matrix can be rewritten as a scalar multipler of a matrix with integral coefficients:

$$\frac{1}{D}\begin{bmatrix} AX & BX & CX \\ AY & BY & CY \\ 0 & 0 & D \end{bmatrix},$$

where D is the common divisor of the coefficients of the original matrix. As with the other rational quantities, the matrix can be reduced to its primitive form by dividing throughout by the gcd of all the numbers. We may disregard the resulting scalar multiplier of the matrix since any application of the transformation yields a point which when reduced to its primitive form will cancel out the scalar multiplier. That is, for any transformation matrix **T**, the transformation k**T** for any scalar k has the same effect. To determine the coefficients of the transformation matrix **T** we require the correspondence between three points in α and three points in γ.

These points must satisfy the requirement that the two triangles formed by the two sets of points are similar. Suppose that is the case.

Let the points be P_1, P_2, P_3 and Q_1, Q_2, Q_3, respectively. That is, the matrix **T** maps P_1 to Q_1, P_2 to Q_2, and P_3 to Q_3. We can either solve the resulting sets of three equations in three unknowns directly or use a simplification in which **T** is split into a composition of three translations, possibly a reflection, and a single combined rotation and scale. The steps are illustrated in figure 6.

Once a transformation has been determined all that is required is to test if the subshape relationship holds under this transformation. The set of valid transformations can be obtained by examining all possible correspondences between three 'distinguishable' points in α and mapping them to corresponding points in γ.

It is easy to see that labelled points form good candidates for determining valid transformations. Other candidates include points of intersection of lines. The possible candidates for determining valid transformations for two-dimensional shapes are enumerated elsewhere (Krishnamurti, 1981).

It suffices to consider just two distinguishable points in α and map them to corresponding points in γ. Given two points it is simple to determine a third point such that the two sets of three points form similar triangles from which the transformations can be computed. In this case the mapping between a pair of points in α and the corresponding points in γ yields two possible transformations, one being the mirror image of the other.

The arguments presented above can be extended to higher dimensions and to other types of transformations. Three-dimensional subshape detection is considered elsewhere (Krishnamurti and Earl, 1986).

A simple shape generation system

We have implemented a simple rule-based shape generation system in PROLOG (Clocksin and Mellish, 1981). PROLOG provides a declarative programming environment in which the statement of a problem and, hence, the description of its solution can be given as a collection of Horn clauses. This has the advantage of avoiding much of the attention paid to the nitty-gritty details demanded by conventional programming languages[(2)].

The implementation is, in the most part, a literal transcription into PROLOG of the description of the shape generation process given earlier. The result is a

[(2)] We write from experience of having implemented a shape grammar interpreter in a conventional language (Krishnamurti, 1982).

noticeable saving in the time and effort required for implementation, and has produced quite readable code. As an example, consider table 1 in which we give PROLOG clauses for the shape union of the line segments of two shapes. This procedure can also be utilised to convert a shape given as a set of line segments into its representational form. The clauses are given in the Edinburgh syntax, where variables are given in upper case and atomic constants in lower case.

An unlabelled shape S is represented as a list, the elements of which are structures of the form $co(D,L)$, where D is a triple of integers and represents the descriptor of a list L of collinear line segments. Each line segment is a structure of the form $line(T,H)$, where T and H are triples of integers that correspond to the homogeneous coordinates of the tail and head of the line segment, respectively.

The predicates in table 1 are interpreted as follows. The predicate *shape_union* has three arguments, S_1, S_2, and the resulting shape union S. Statements labelled 1 and 2 are that the shape union of a shape with an empty shape, denoted by the empty list [ ], is the shape itself. Statements 3, 4, and 5 correspond to the cases when the line descriptors D_1 and D_2 satisfy $D_1 = D_2$, $D_1 < D_2$, and $D_1 > D_2$, respectively. The expressions in the form [*First*| *Rest*] correspond to lists whose first element instantiates to the term *First*, and the rest of the list instantiates to the term *Rest*. Notice that the ordering of the sets of collinear line segments is preserved in the resulting shape union. Statement 3 corresponds to the case when the descriptor of the set of collinear line segments is the same for both shapes. Here, the union of the two sets of collinear line segments is performed by invoking the predicate *line_union*.

The predicate *line_union* has three arguments L_1, L_2, and the resulting line union L. Here, L_1 and L_2 represent sets of collinear lines organised as lists according to the ordering on the lines. The first two statements are termination statements for the

Table 1. PROLOG clauses for the shape union of two unlabelled shapes.

Union of two shapes

```
1   shape_union([],S,S).
2   shape_union(S,[],S).
3   shape_union([co(D,L1)|S1],[co(D,L2)|S2],[co(D,L)|S]):-
        line_union(L1,L2,L),
        shape_union(S1,S2,S).
4   shape_union([co(D1,L1)|S1],[co(D2,L2)|S2],[co(D1,L1)|S]):-
        D1 < D2,
        shape_union(S1,[co(D2,L2)|S2],S).
5   shape_union([co(D1,L1)|S1],[co(D2,L2)|S2],[co(D2,L2)|S]):-
        shape_union(S1,S2,S).
```

Union of two sets of collinear line segments

```
1   line_union(L,[],L).
2   line_union([],L,L).
3   line_union([line(T1,H1)|L1],[line(T2,H2)|L2],[line(T1,H1)|L]):-
        H1 < T2,
        line_union(L1,[line(T2,H2)|L2],L).
4   line_union([line(T1,H1)|L1],[line(T2,H2)|L2],[line(T2,H2)|L]):-
        H2 < T1,
        line_union([line(T1,H1)|L1],L2,L).
5   line_union([line(T1,H1)|L1],[line(T2,H2)|L2],L):-
        H1 < H2,
        min(T1,T2,T),
        line_union(L1,[line(T,H2)|L2],L).
6   line_union([line(T1,H1)|L1],[line(T2,H2)|L2],L):-
        min(T1,T2,T),
        line_union([line(T,H1)|L1],L2,L).
```

recursion. The next four correspond to the conditions on the tails and heads of the two line segments under comparison. The last two of these statements correspond to the situation when the line segments overlap. The predicate *min* holds if the minimum of its first two arguments equals that of its third argument.

The shape union of two shapes S_1 and S_2 that results in a shape S is invoked by the following headless Horn clause.

Invoking a shape union
:- *shape_union*(S_1, S_2, S).

Similar predicates can be defined for the other Boolean operations, and relations and the Euclidean transformations of shapes.

The system is implemented as a menu-driven interactive program using a PROLOG graphics software, SeeLog, developed at EdCAAD (Pereira, 1982). Shapes are created using conventional graphics commands. Shapes can be saved for recall at a later time. The system is designed to serve the dual role of an interpreter for two-dimensional shape grammers as well as an experimental shape editing system.

Conclusion

The implementation reported in this paper started out for one of us (RK) as an exercise in learning PROLOG. Since then the work has assumed—at least to us—some relevance particularly from the standpoint of developing good graphics editors with the power equivalent to that found in good text editors. In addition to the usual graphics operations for creating, manipulating, and altering drawings, such systems would need to have recognition capabilities in varying degrees that are necessary for a wide range of applications. An example of such an application is a combined natural language text and graphics dialogue system that EdCAAD is currently involved in as a member of a European collaborative research effort, where semantic links between the drawing and the knowledge base play an important role in shape editing.

A shape editing system that utilises shape rules would of course need to provide greater control over the specification and application of shape rules. For instance, it is important that the part of a drawing to which a shape rule is applied is given more directly, say by 'pointing' at the particular part of the drawing rather than relying on the system to determine the possible shape replacements. Pointing at a drawing may be specified in several ways—for example, by outlining the part of drawing to which a shape edit is applied; using deixis by specifying the semantic links such as the 'rotor arm', 'living room', etc. It is also natural to expect that shape replacements can be carried out globally—that is, the shape rules are applied in parallel to all possible subshape instances, such as 'replace all circular columns in the portico by square columns'. Moreover, it is important to consider shape rule applications under nonisometric transformations, such as independent x- and y-scaling. Parametric spatial relations will certainly enhance the flexibility of a shape editing system.

A shape editor that is based on a 'wireframe' model for shapes may not be suitable for many applications. It is therefore important to develop analogous shape editing facilities for other representational forms for shapes; or, at least, to express the result of a shape edit in these representations.

There are many problems that we have not touched upon in this paper that still must be addressed before we contemplate building a shape editor. Nonetheless, we believe that spatial relations expressed as shape rules or in some other form provide a first step towards such a goal.

Acknowledgement. We would like to thank Aart Bijl for his encouragement and support during the course of this research. We would like to thank the referees for suggestions on improvements to this paper.

References

Clocksin W F, Mellish C S, 1981 *Programming in Prolog* (Springer, Heidelberg)

Downing F, Flemming U, 1981, "The bungalows of Buffalo" *Environment and Planning B: Planning and Design* **8** 269-293

Earl C F, 1986, "Creating design worlds" *Environment and Planning B: Planning and Design* **13** 177-188

Earl C F, Krishnamurti R, 1984, "Spatial relations, kinematics and assembly" in *Proceedings of the International Symposium on Design and Synthesis* Japan Society of Precision Engineers, Tokyo; pp 589-593

Knight T W, 1983, "Transformations of languages of designs: part 2" *Environment and Planning B: Planning and Design* **10** 129-154

Krishnamurti R, 1981, "The construction of shapes" *Environment and Planning B: Planning and Design* **8** 5-40

Krishnamurti R, 1982, "SGI: a shape grammar interpreter" technical report, Design Discipline, The Open University, Walton Hall, Milton Keynes MK7 6AA

Krishnamurti R, Earl C F, 1986, "Shape description and recognition in three dimensions" in preparation; details available from the first author

Meyrowitz N, van Dam A, 1982, "Interactive editing systems: parts I and II" *ACM Computing Surveys* **14** 321-416

Pereira F C N, 1982, "SeeLog—a Prolog graphics interface" EdCAAD working paper, University of Edinburgh

Stiny G, 1975 *Pictorial and Formal Aspects of Shape and Shape Grammars* (Birkhäuser, Basel)

Stiny G, 1980a, "Introduction to shape and shape grammars" *Environment and Planning B: Planning and Design* **7** 343-351

Stiny G, 1980b, "Kindergarten grammars: designing with Froebel's building gifts" *Environment and Planning B: Planning and Design* **7** 409-462

Stiny G, Gips J, 1972, "Shape grammars and the generative specification of painting and sculpture" in *The Best Computer Papers of 1971* Ed. O R Petrocelli (Auerbach, New York) pp 125-135