An Elementary rule interpreter for architectural design

William Fawcett

This paper was published in A Smith (ed) Knowledge Engineering and Computer Modelling in CAD, Proceedings of the Seventh International Conference on the Computer as a Design Tool, Butterworths (London), 1986.


william.fawcett@carltd.com
This paper describes the use of a standard CAD drafting system (AutoCAD) to implement an interpreter for shape generation using shape rules. In recent years the use of shape rules (in the formal structure of shape grammars) has been one of the most interesting and advanced areas of research in design; here we describe a new technique to make these advanced ideas available in an elementary way on present generation CAD systems. An enhanced shape rule interpreter is discussed.

INTRODUCTION

Shape rules and shape grammars have been discussed for over ten years, within a small research community. Some impressive hand-worked examples have confirmed the tremendous potential of the ideas, but little has been done to make shape grammars available for use or experimentation by a wider audience. This paper describes an attempt to do so, by implementing an elementary interpreter for shape rules (ERIS) within an existing CAD environment. AutoCAD was chosen as the host system, but it should be possible to implement the same ideas in most of today's CAD systems.

SHAPE GENERATION USING SHAPE RULES

The shape grammar formalism was pioneered and developed primarily under Dr George Stiny, beginning in the mid 1970's (1,2). It is based on the idea of shape rules. A shape rule has the familiar 'IF ... THEN ...' form, but both the left-hand side ('IF ...') and the right-hand side ('THEN ...') are shapes. To apply a shape rule to a given shape, it is necessary that the LHS of the rule exists as a subshape of the given shape; when the rule is invoked this subshape is replaced with a new subshape corresponding to the RHS of the shape rule (fig. 1). Thus the application of a shape rule involves an erase and an insert operation.

A set of shape rules is the basis of a shape grammar.

Using a shape grammar, designs are generated by a successive applications of shape rules, beginning with an initial shape which is also specified in the shape grammar. By applying rules in different sequences, a large (possibly infinite) number of distinct shapes can be generated by a single shape grammar. This ensemble of shapes is the language of the grammar (fig.2). Each shape grammar has its own language, implicit in the shape rules and initial shape even if it is not enumerated (which of course it cannot be for an infinite language).
Just as a grammar implies an ensemble of shapes, a class of given shapes can often be considered as coming from the language of a hypothetical shape grammar, which would generate shapes with the formal properties of the given shapes. For many classes of designs with distinctive formal characteristics — for example Palladian villa plans (3), decorative Chinese lattices (4), and so on — shape grammars are a way of explicitly defining intuitively recognisable classifications. In this respect shape grammars can be said to have Artificial Intelligence properties, by modelling intuitive judgements.

Furthermore, once a shape grammar has been defined its use is not limited to classification, but more importantly it can generate new shapes with the characteristics implicit in the grammar. This is coming close to human design skills. In general human designers produce designs that are new and unique, but which conform to design conventions, regulations, etc, and which fall within recognisable design classes. So shape grammars can be used as the basis for design expert systems, capable of generating new designs exhibiting definite design characteristics: in the absence of explicit shape rules this is a task requiring human intuition and judgement.

INTERPRETERS FOR SHAPE GRAMMARS

As noted above, the generation of a design shape rules goes through a number of stages; at each stage the shape is transformed by applying a shape rule. Often there may be a number of different rules that can be applied, and the same rule may be applicable in more than one location (fig. 3). These alternatives are identified by pattern-matching the LHS's of shape rules with the shape. The fact that there is choice of rule applications leads to the diversity of shapes in the language of a grammar (fig. 4). Thus the selection of the rule to apply is a key decision in shape generation with shape rules — perhaps one could call it the creative decision. Once the decision has been made rule application is straightforward.

We see that shape generation with shape rules has three components:

1) pattern-matching, to find out where the LHS's of shape rules are present as subshapes

2) rule selection, to determine which rule to apply and where

3) shape transformation, to delete the subshape matching the LHS of the rule and insert the RHS.

The first and last stages require no judgement and can be mechanised with an interpreter: such an interpreter is used interactively with the user performing rule selection. Automating the rule selection stage would require meta-rules to determine which shape rule to select, and that would be automatic design generation, whereas in the interactive mode the shape rules act as expert guidance for the human designer. It is worth noting that shape rules can be used by hand without a computer interpreter at all, but the well-defined and tedious graphic tasks are very naturally suited to CAD.

The Interpreter presented here — called ERIS (for Elementary Rule Interpreter for Shapes) is limited to the third component, shape transformation. It is onerous to record the evolutionary stages in shape generation, even for relatively simple shape grammars, and even if computing drafting is used. By contrast, the first stage, pattern-matching, is a matter of visual inspection and is not laborious, although it is error-prone. However, a more powerful interpreter under development incorporates pattern-matching and other enhancements.
ERIS is not the first shape rule interpreter. A small number of others have been programmed from scratch (Krishnamurti (5), Coyne & Gero (6), Flemming et al (7)); they have been more concerned with research and the theory of shape generation than with the kind of sophisticated user interface and output that we have come to expect in current CAD systems. ERIS takes a different path: it is implemented in an existing CAD environment, and exploits the interface and graphic capabilities of that environment. The benefits and penalties of this approach are discussed later.

THE STRUCTURE OF ERIS

ERIS-A was implemented using AutoCAD version 2.17 on IBM PC-XT and AT microcomputers. It has also been run on the Olivetti M24 and Apricot, and should be portable to any AutoCAD installation. It is probable that ERIS could be implemented in other CAD environments as it makes use of fairly standard CAD capabilities.

(a) Graphic elements

A shape grammar deals with a specific class of shapes, and each shape grammar written in ERIS has its set of graphic components. They are drawn using the usual drafting commands of AutoCAD and stored as named entities, known as 'blocks' in AutoCAD. The LHS and RHS of all shape rules, and the initial shape, are all constructed out of the graphic components and themselves stored as named blocks. All shapes generated by the grammar are also necessarily made up out of the components. The application of a shape rule involves the deletion of a block corresponding to the LHS and the insertion of a block for the RHS of the rule.

In interactive use the current state of the shape is displayed on the screen, together with a menu listing the names of the shape rules. The user decides by visual inspection which rule to apply and where, and uses a pointing device to pick the rule from the menu and then to point to the chosen location. The exact location pointed to is critical - it must unambiguously touch some part of the block to be deleted (the LHS of the rule), and it also marks the precise insertion point for the block to be inserted (the RHS of the rule). In AutoCAD every block is defined with a single insertion point. The block being deleted must be an indivisible entity, and thus identifiable by a single touch of the pointer. However, the block being inserted may contain a number of distinct entities which can later be deleted individually if they correspond to the LHS's of other shape rules. In AutoCAD a block can be inserted as a single entity by specifying its name, or as a collection of parts by adding a star (*) before the block name.

Precision in pointing is achieved by using a 'snap grid' for the pointing device. The pointer is constrained so that it can only point to locations on a regular grid. This has the effect that the shapes generated must be organised on a regular grid.

Applying a shape rule involves block insertion, and in AutoCAD this requires two pieces of information in addition to the insertion point. The first is scale. It is not convenient to change scales within shapes generated by ERIS because ERIS does not know the scale of the blocks it deletes, and so cannot find out the scale at which to insert new blocks; a default scale of 1 is used throughout. The second piece of information is rotation - again ERIS does not know the orientation of blocks erased. Shape rules in ERIS are effectively constrained to orthogonal geometries by the rectangular snap grid, so there are up to four orientations for block insertions. (AutoCAD's iso-grid could also be used to create triangulated shapes.) Not all rotations are relevant for all shape rules, but where alternatives do exist the user is asked to indicate the one required (by picking from a menu) after making his rule selection.
In ERIS there are clearly rather severe constraints on the design of shape rules, but so long as they are met ERIS can handle shapes with unlimited graphical complexity. In this respect ERIS is as powerful as AutoCAD itself, since it uses shape components created with AutoCAD drafting commands. This is in contrast to the research-oriented interpreters mentioned above.

(b) User interface

The user interface of ERIS-A uses a specially written set of AutoCAD screen menus. The ERIS menus can be set as the AutoCAD default, or called during an AutoCAD session. Within ERIS-A all user responses are prompted from menus, so the command window area of the screen is superfluous and AutoCAD can be configured to delete it.

The menus are structured hierarchically, beginning with a header which asks the user which shape grammar he or she wishes to use. Then for each grammar the first menu asks the user to pick a location on the graphic screen for the location of the initial shape; as soon as the position is picked the initial shape is automatically inserted. Then a cycle begins, with a menu offering the names of all the rules in the grammar; after visual inspection of the shape the user selects a rule from the menu. The menu invokes what is in effect an AutoCAD command macro. If necessary a new menu asks the user to select the rotation for the selected rule application. Then a further menu asks the user to pick the location on the screen where the rule is to be applied; as soon as this position is picked the LHS of the rule is automatically erased and the RHS inserted. The transformed shape is immediately seen by the user, and the cycle repeats. This is the process of shape generation using ERIS.

Each shape rule in a grammar has its own menu, but the menus are highly repetitive. It is possible to edit a set of non-specific dummy menus with a word processor to create the menus for a new shape grammar.

From the initial menu for each grammar and from the cycle menus the user can also select other options:

- view a specimen shape generated with the rules
- view the shape rules themselves
- invoke a menu of housekeeping and other actions.

The first two are held as AutoCAD 'slides', that is screen images that can be thrown up immediately without the delay of regeneration. The housekeeping tasks include zooming in and out, presenting or hiding the snap grid, and ending a session. Also, when AutoCAD Extension 3 is available, its limited 3-D viewing capability can be used.

ERIS-A eliminates keyboard input; with custom menus and a pointer the user is able to create complex shapes in the formal vocabulary of the shape rules he has selected.

SHAPE RULES AND DESIGNS

ERIS is independent of any particular set of shape rules. In this section we illustrate its operation with one example, called BASILICA as it generates shapes which are immediately recognisable as the plans of romanesque basilica churches. The shape rules were originally written and used before ERIS, working entirely by hand. Experience with ERIS quickly suggested improvements which were simple to incorporate.

BASILICA uses eight shape components, drawn using the conventional AutoCAD drafting commands (fig. 5). There are seven shape rules, each with an LHS and
RHS; all of the LHS's and some of the RHS's consist of a single component, but four of the RHS's are made up of more than one component (fig. 6). There is no reason why LHS's should not also be composite. Notice that many components and rules contain 'markers'. These are integral parts of the relevant components, and have to be considered when pattern-matching, but they disappear when 'terminating' shape rules are used and do not appear in finished shapes. Some shape rules are labelled 'parallel'; this means that when the rule is selected, all matching subshapes should be transformed simultaneously, or in parallel. However ERIS-A requires the user to indicate each location (and rotation) in turn, using an abbreviated command sequence. The initial shape contains four shape components (fig. 7).

Eight stages in a typical generation are shown, starting with the initial shape and ending with a shape that is complete in the sense that no more rules can be applied to it, as it contains no rule LHS's as subshapes (fig. 8). Shapes can be viewed with AutoCAD's 3-D visualisation facility (fig. 9).

When BASILICA was used by students and teachers at a school of architecture, two enhancements were suggested: first, it should be possible to generate shapes that extend up and down in addition to the left-and-right growth axis implied in the initial seven rules; and secondly, it should be possible to create cloisters, which are characteristic of romanesque church complexes. At the same time a doorway was added. These enhancements were achieved by adding two shape components, and three new shape rules using them, all with AutoCAD drafting commands (fig. 10). The BASILICA menus were modified, and new menus for the new rules added, with a word processor.

A shape generated with the enhanced rule set is shown (fig. 11).

The language of BASILICA contains an infinite number of shapes, but they share very strong formal characteristics. They form a very restricted subset of the shapes that can be drawn with AutoCAD. The restrictions of the language, and its formal character, are complementary phenomena - without restrictions there would be no formal character. Some users, however, have felt frustrated by the restrictions, because they envisage shapes that the rules do not allow them to generate. One response is to augment the rules, as described above. A more negative response is the rejection of shape rules altogether, throwing the user back onto his intuitive assessment of formal properties, and losing the 'artificial expertise' built into shape rules. The frustrations were felt by trained architects and students, whose intuitive abilities in handling floor plans were well developed and for whom the BASILICA language was straightforward. The situation would be different for architecturally less skilled users, or more complex formal languages.

Simple as it is, BASILICA indicates how shape rules can express formal properties, and guide users in the design of shapes with those formal properties. And shape rules are not unalterable, but retain the possibility of modification and development.

ENHANCEMENTS

Two main problems have been experienced when working with ERIS-A. First, it is not possible for users to correct errors: once a subshape is inadvertently erased, or inserted in the wrong position, shape generation has to be aborted and restarted from the initial shape. Secondly, users do not always make legal rule applications, and therefore they arrive at shapes that do not fall within the language of the rules - although they use its graphic components.
Both these shortcomings can be overcome by adding memory and intelligence to a rule interpreter, and other benefits follow too. The memory records the graphic elements that exist in a shape, and also the sequence of rule applications leading to it. The memory might be the CAD system's own database, or a separate memory recording the names of the 'blocks' in the shape with their insertion points, scales, and rotations. The intelligence must be able to interrogate the memory, pattern-match on it, and instruct the CAD system to perform erase and insert operations when a shape rule is invoked.

The following benefits would also result:

- pattern-matching can be performed by the interpreter rather than the user
- the user does not need to specify rotation for rule applications
- rules which change scale can be handled
- shape generation can backtrack
- user-selected locations for rule applications can be checked for legality.

Note that the snap grid for pointing, and therefore the regular organisation of shapes, is still required.

This is the agenda for IRIS - an Intelligent Rule Interpreter for Shapes. It is intended to implement IRIS within, or communicating with, existing CAD environments. IRIS, however, needs a true programming language, unlike the simple command strings of ERIS, and its implementation is highly dependent on interfacing a suitable programming language. The use of an interpreter within a host CAD environment retains the benefits of using the CAD system's interface and graphic capabilities, as well as making the interpreter more readily accessible to users at CAD-equipped sites.

An important consequence of using a host CAD system, which applies to both ERIS and IRIS, is that pattern-matching relies on recognising the names of predefined graphic blocks, rather than searching the actual graphical primitives in a shape, as in the case with a generalised shape rule interpreter. This is a significant theoretical simplification making pattern-matching much easier. The penalties include, first, the fact that interpreter cannot recognise shapes unless they are described in terms of the named graphic blocks, even if they contain matching graphic primitives; and secondly, the interpreter cannot recognise 'emergent' subshapes, that is subshapes matching the LHS of a rule that emerge from the juxtaposition of other shape elements. The simplest example is the emergent square that is created when two squares are overlaid. It is not yet clear whether these problems will be significant in practice, but the supposition underlying this project is that the needs of design will benefit from the simplicity and accessibility of the approach we have described.

CONCLUSIONS

ERIS is a first step, but it already offers a working tool to explore decisive new developments in CAD and design research. Indeed users with no previous experience of CAD or shape grammars were able to generate shapes very easily with ERIS, and proposed extensions to the BASILICA shape grammar; this would be inconceivable had BASILICA only been accessible as a paper exercise.
It is worth emphasising that design systems using shape rules are not intended to compete with CAD drafting systems. Any shape whatever can be drawn with a drafting system, but shape rules define a limited language of possible designs. The 'artificial expertise' of shape rules may often be useful, but there will always be times when open-ended drafting is desired, not least when creating graphic components and shapes rules themselves. And since ERIS exists within a CAD system, CAD drafting commands are actually available at all times when using ERIS, and ERIS-generated shapes are simply normal shapes to the CAD system. Thus rule-generated shapes can be freely mixed with other shapes, or rule-generated shapes can themselves be edited in arbitrary ways.

ERIS demonstrates that today's CAD systems can be used in conjunction with advanced ideas from design research, to open new fields for CAD applications (8). We suggest that the leverage obtainable by the development of intelligence - Artificial Intelligence - in CAD may be greater than can be expected from further refining what are already highly developed graphic/drafting capabilities.

REFERENCES


5. KRISHNAMURTI, R. 'SGI: a shape grammar interpreter'. Centre for Configurational Studies (The Open University) and E&CAAD (University of Edinburgh) (1982 et seq).


Fig. 1  A shape rule, and its application to a shape

Fig. 2  Shapes in the language of a grammar
rule 1 \[ \] \rightarrow \[
\]
rule 2 \[ \] \rightarrow \[
\]
rule 3 \[ \] \rightarrow \[
\]
rule 4 \[ \] \rightarrow \[
\]

wall \mid apse \]
end \mid niche \]
side \mid cols \]
temp \mid bay \]

Fig. 3 Alternative rules applicable to a shape

Fig. 4 Alternative sequences of rule applications

Fig. 5 BASILICA shape components
extend 1 \[\vec{p}\] \rightarrow \[\bar{p}\]
extend 2 \[\vec{p}\] \rightarrow \[\bar{p}

\vec{p}

extend 3 \[\vec{p}\] \rightarrow \[\bar{p}\]
\[\bar{p}\]

apse \[\vec{p}\] \rightarrow \[p]\]

niche parallel \[\rightarrow \]

cols \[\vec{p}\] \rightarrow \[c\]

bay parallel \[\rightarrow \]

Fig. 6  BASILICA shape rules
Fig. 7  BASILICA initial shape

Fig. 8  Stages in the generation of a BASILICA design
Fig. 9  3-D visualisation of BASILICA design

extend 4 \[ \rightarrow \]

cloister \[ \rightarrow \]

doors \[ \rightarrow \]

Fig. 10  New shape rules for BASILICA

Fig. 11  Design generated with enhanced BASILICA shape rules