DESIGN EXPLORATION USING A SHAPE GRAMMAR WITH A GENETIC ALGORITHM

Orestes Chouchoulas & Alan Day

Abstract

Although the idea of linking a shape grammar to a genetic algorithm is not new, this paper proposes a novel way of combining these two elements in order to provide a tool that can be used for design exploration. Using a shape grammar for design generation provides a way of creating a range of potential solutions to a design problem which fit with the designer's stylistic agenda. A genetic algorithm can then be used to take these designs and develop them into a much richer set of solutions which can still be recognised as part of the same family. By setting quantifiable targets for design performance, the genetic algorithm can evolve new designs which exhibit the best features of previous generations. The designer is then presented with a wide range of high scoring solutions and can choose which of these to take forward and develop in the conventional manner. The novelty of the proposed approach is in the use of a shape code, which describes the steps that the shape grammar has taken to create each design. The genetic algorithm works on this shape code by applying crossover and mutation in order to create a range of designs that can be tested. The fittest are then selected in order to provide the genetic material for the next generation. A prototype version of such a program, called Shape Evolution, has been developed. In order to test Shape Evolution it has been used to design a range of apartment buildings which are required to meet certain performance criteria.

Keywords: *Shape Grammar, Evolutionary Design, Genetic Algorithm, Design Exploration*

INTRODUCTION

Evolutionary systems, and particularly those using genetic algorithms at their core, are very powerful and versatile. They enable the designer to consider any number of criteria simultaneously without the need to define a priori solutions and provide the means for optimising designs through an extensive examination of alternatives. However, if these tools are to infiltrate architectural design studios they need to be generic. They also need to produce designs that are meaningful to their designers and pertinent to each specific design problem. The key to providing these features seems to lie with the choice of representational system for the designs, as well as the way in which each design is described when using a genetic algorithm (Woodbury, 1993). Shape Evolution addresses this by using a shape grammar to generate the initial designs and then transfers the outputs to a genetic algorithm for design development. The novelty of the approach is in the way in which the linkage between the shape grammar and the genetic algorithm takes place.

SHAPE GRAMMARS

A shape grammar consists of a vocabulary of shapes, a set of shape rules, and an initial shape. The rules result in the transformation of a shape, or collection of shapes, to a new shape. Applied recursively on the initial shape, the rules produce designs that are said to belong to a language (Stiny and Gips, 1972; Stiny, 1975).

Shape grammars have, since their inception, been used for both analysis and synthesis. In terms of analysis, they have been called to serve as descriptors of style and have been successful when applied analytically to the definition of historical styles including Palladian villas (Stiny & Mitchell, 1978), Wren's City churches (Buelinckx, 1993), Frank Lloyd Wright Prairie houses (Koning & Eizenberg, 1981), window designs (ROLLO, 1995), Japanese tearooms (Knight, 1981), Mughul gardens (Stiny & Mitchell, 1980) and Hepplewhite chairs (Knight, 1994).

Another stated aim of the shape grammar formalism is to provide a mechanism for the creation of new design languages. This is achieved by creating a vocabulary and a set of rules from scratch.

The use of shape grammars in a generative, rather than an analytical capacity is of particular interest as a very simple grammar with a limited vocabulary and a few rules can create significantly complex and unanticipated results.

GENETIC ALGORITHMS

Genetic algorithms were developed in the 1970s by John Holland (Holland, 1975) in an effort to formally understand biological adaptation in nature. Much like the organisms they were meant to study, genetic algorithms have taken on a life of their own and have proven robust in tackling optimisation problems and exploring very large search spaces (Goldberg, 1989). This makes them appropriate to the solution of design problems when these are represented as a search through a design space where each point in the space is a potential design. The main strength of genetic algorithms as problem solvers is that they do not require an explicit optimal solution generation method, relying instead on a generate-and-test procedure.

COMBINATION OF GENERATION AND EVOLUTION

Shape grammars are good at providing an aesthetic and organisational specification for the generation of forms. Genetic algorithms are robust search algorithms that can be used very effectively for finding solutions that satisfy a set of defined conditions, such as the quantitative requirements of an architectural design. Combining the two produces a system where the design space is specified using a shape grammar, which is then explored using a genetic algorithm.

The combination of a shape grammar with a genetic algorithm has been proposed by others, including Gero & Kazakov (1996), Rosenman & Gero (1999) and Loomis (undated). Also, Shea's work on shape annealing (Shea & Cagan, 1997) is an example of a similar approach which optimises structural design and produces unconventional but highly functional solutions. The novelty in the current approach lies in way in which the shape grammar is linked to the genetic algorithm, through the use of a shape code.

SHAPE CODE

In Shape Evolution, the generation process involves a chain of shape grammar rule applications starting with an initial shape. The shape grammar is defined with a small number of shapes in the vocabulary along with a small number of rules. Consequently, it is relatively easy to represent the generation process for a design as a simple string. This describes the sequence in which the shape grammar rules were applied. This string, called a shape code, has been employed by Koutamanis (2000) for the purpose of cataloguing and retrieving designs.

Using the shape code as the genotype (the building blocks that are used by the genetic algorithm) offers three very significant benefits. Firstly, it ensures that all the designs created during the evolutionary process are valid in the language. It also means that the genetic algorithm's operations on the genotype, such as crossover and mutation, alter the selection and sequence of shape grammar rules used for the generation of a design, which are potentially very meaningful alterations. In addition, the process of converting the shape code into a three-dimensional form is simply a matter of applying the shape code sequence to the shape grammar, thus producing geometry. The designs can then be evaluated with respect to their physical attributes.

INVALID SHAPE CODES

Using the shape code as the genotype introduces a complication. The spatial overlapping of abstract shapes is usually not a problem, but when these shapes represent physical entities, as happens when a shape grammar is used to generate buildings, clashes can easily result. Invalid shape codes need to be weeded out before they ever make it to the evaluation algorithms. Provided that the initial population fed into the genetic algorithm is comprised entirely of valid individuals, it is only at crossover

and mutation that invalid shape codes can be produced. A checking stage is therefore required at each instance of crossover or mutation. This ensures that new elements are only added to areas that were previously empty.

SYSTEM OVERVIEW

Shape Evolution is a prototype system combining shape grammars and genetic algorithms in the way described, implemented as a computer program. It receives input from the designer and outputs concept design solutions that satisfy a set of requirements and form part of the designer's language of design. Figure 1 shows a general flowchart of the system.

The program starts by receiving user input, which includes a definition of the shape grammar to be used, as well as goal values for quantifiable properties of the design. The genetic algorithm will be using these to assign a score to each design according to how close it is to the goal values. Further user input includes genetic algorithm variables such as population size, mutation rate, and the end condition, which determines when Shape Evolution will terminate its run. This can be satisfied either after a certain number of generations or when designs with scores better than a user-defined threshold value are produced.

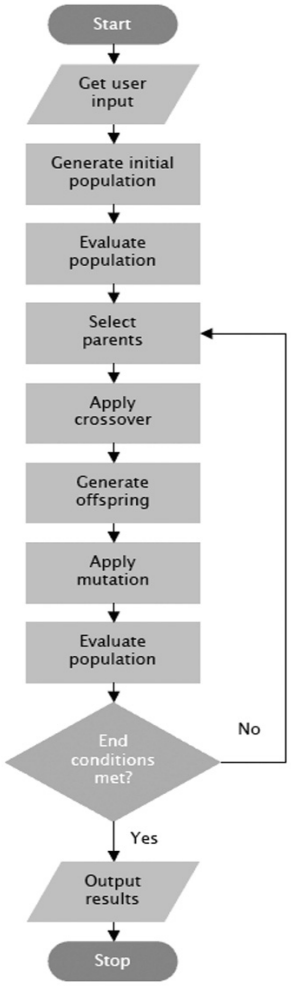

Fig 1. Shape Evolution flowchart

Shape Evolution then proceeds to generate the initial population using the shape grammar. Every design in the population is then evaluated and has a score attached to it. The fittest (highest-scoring) members of the population are then selected. These provide the genetic material from which the next generation is produced. The genotypes of the selected parents are paired up randomly and crossover is employed to generate two new genotypes per mating.

After being mutated probabilistically, these genotypes form the new generation of designs. A further evaluation and scoring of the new population helps determine whether the end conditions have been met. If not, the selection-crossover-mutation cycle starts again and a new population is generated. If the end conditions are satisfied the best designs produced are presented to the designer and Shape Evolution stops. What happens next is up to the designer. Some of the ultimate designs might be usable as they are, others might serve as inspiration, and others may be entirely useless.

THE APARTMENT BLOCK PROBLEM

In order to test the system a simple problem which seeks innovative yet functional design concepts for a multi-storey apartment block has been investigated. For the sake of simplicity, the proposed building consists of a series of similar apartments along with the horizontal and vertical circulation spaces connecting them. A necessary requirement is that all circulation is contiguous and that there is an accessible entrance on the ground floor.

A simple shape grammar producing designs of this sort can be constructed with only two shapes in its vocabulary: a multi-purpose circulation unit and the apartment unit itself. The circulation unit can be

approximated by a 4m x 4m x 4m cube, an envelope that can easily accommodate both vertical and horizontal circulation, as seen in Figure 2.

The apartment unit is a generously sized single bedroom apartment at 16m x 4m x 4m. A possible internal layout for the apartment is suggested in Figure 3. To allow more flexibility in the arrangement of the apartments, we can accept that the large window in the living room can be placed on any of the three walls. Also, the entrance to the apartment can be moved a few metres along the wall without any problems.

Simplified abstract representations for these units will be used, as shown in Figure 4. The circulation block is represented by an open framework and the apartment by a grey box, with the darker end denoting the location of the living room.

In order to eliminate the possibility of the entrance being surrounded by built form and therefore rendered inaccessible, this building will not have any ground floor apartments. Consequently, the initial shape for this grammar becomes a stack of two circulation blocks.

In order to build upon the initial shape, there need to be rules that add shapes to the circulation block at the top of the initial stack. Contiguous circulation and accessibility for every apartment can be assured at the level of the shape grammar definition by using the last placed circulation block on the left hand side of every rule. Apartment blocks can be added to circulation in two different ways and a third rule can attach another circulation block to an already existing one. This creates a chain of circulation spaces which becomes the circulation core of the building. Figure 5 shows these three basic rules.

Explicitly defining the usable transformations of these rules produces a total of 22 rules, as seen in Figure 6. Rules 1 to 8 are rotations and reflections of the first basic rule on the horizontal plane, and rules 9 to 16 are rotations and reflections of the second basic rule. Finally, rules 17 to 22 extend existing circulation in all six directions. Although all these rules are not strictly necessary, as the same result could be achieved by adding labels to the first two rules, the variations are made explicit so that they can be used in the shape code.

Concept designs for apartment buildings can be

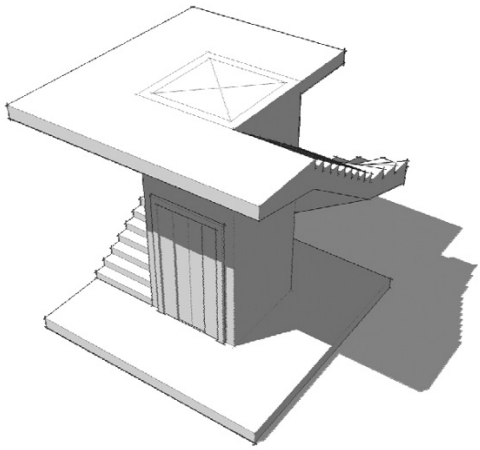

Fig 2. Example use of the circulation block

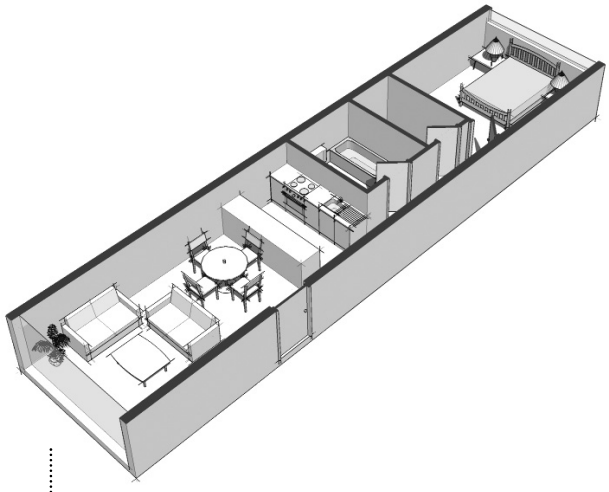

Fig 3. Possible internal layout for the apartment unit

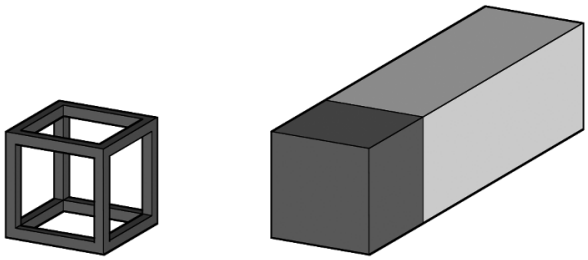

Fig 4. The two shapes in the apartment block shape grammar vocabulary

produced using this shape grammar by starting with the initial shape and then applying rules sequentially. An example, showing the generation of a simple apartment building is shown in Figure 7. Each new stage in the sequence is the result of applying the rule designated above each arrow on the last placed circulation block.

Concept designs produced using the shape

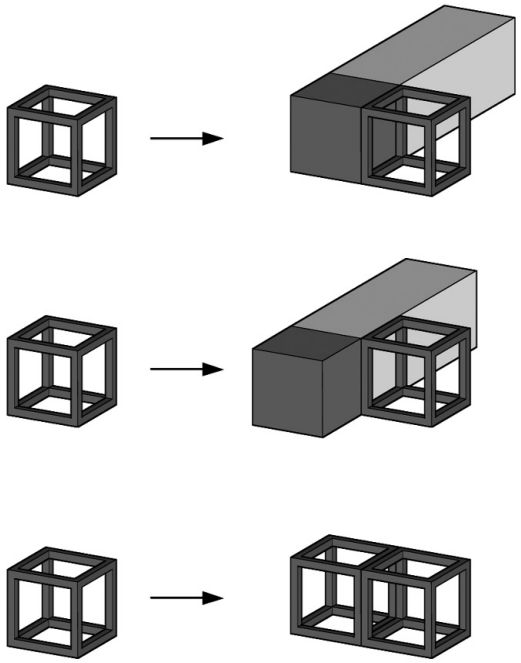

Fig 5. The three basic rules of the apartment block shape grammar

grammar described above display characteristics that can be described quantitatively and can be used to define a set of desirable characteristics for the optimisation of designs by Shape Evolution. Each design can then be scored by comparing its measured characteristics with the goal quantities. The graphical user-interface developed for inputting the goal values in shown in Figure 8.

The system uses three ways to represent each design. The first is the shape code string that describes the rule sequence that generated the design. This can be translated into a second representation, where a three-dimensional array is populated with integers that represent different kinds of cubic modules (0 represents a void, 1 represents a circulation block, and each room of the apartment is represented by numbers from 2 to 5). For the purposes of presenting a design to the user in an intelligible format, Shape Evolution converts the three-dimensional array into a VRML file which can be viewed interactively in an appropriate browser, as shown in Figure 9.

GENERATION OF THE INITIAL POPULATION

The genetic algorithm provides a mechanism to search the problem space but a random initial population must be provided. In order to generate this population, a user-defined number of random rules are selected and applied. As invalid rules are possible, the initial population generator must be discerning about how the random rule sequences are produced. Instead of trying to resolve this issue by

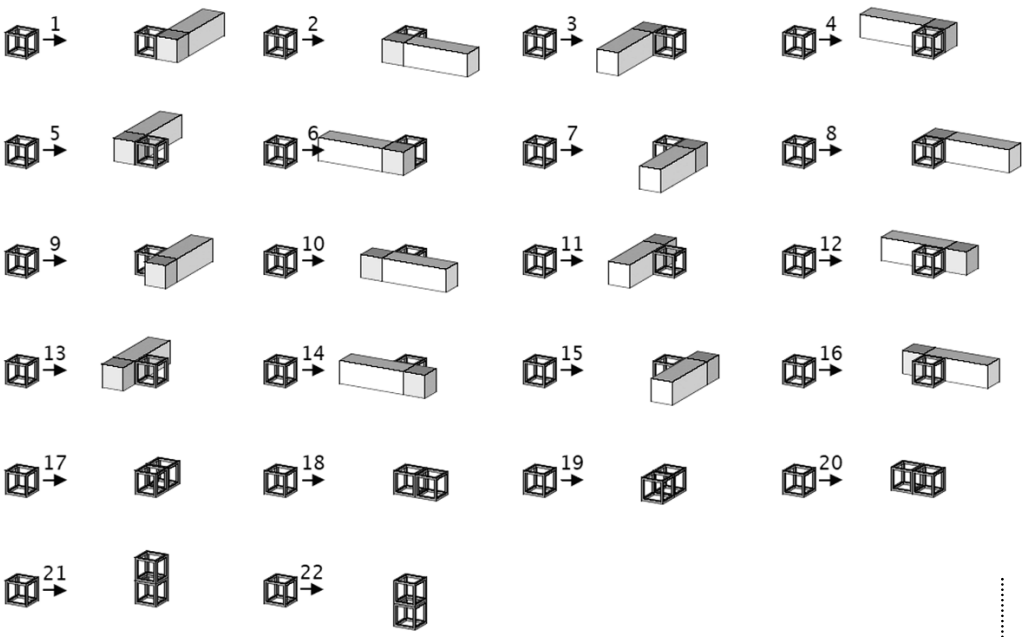

Fig 6. The 22 explicit rules of the apartment block shape grammar

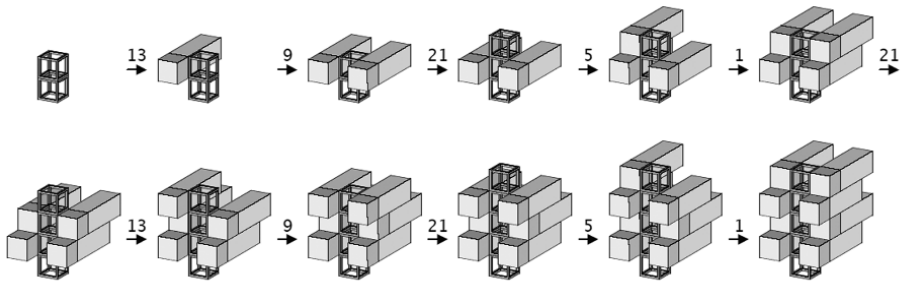

Fig 7. Example sequence of generation of an apartment building using the shape code 13 9 21 5 1 21 13 9 21 5 1

a knowledge-based system that defines which rules are applicable at a particular point in the sequence, the generator uses the three-dimensional information provided by the array representation to determine whether a rule application would create an invalid design. Adding this checking step ensures that all the array positions to be changed had originally a value of 0, i.e. that the was empty.

EVALUATION ALGORITHMS

The next step in the process is to evaluate the performance of designs in the initial population by comparing them with the user-provided optimisation goals. This task can be reduced to teasing information about a design's physical attributes from the already known data about the design: the three-dimensional array and the shape code. From these sources it is possible to extract the number of apartments, the area and volume of the completed building, along with its height and footprint. It is also possible to establish which apartments have unobstructed view from the living room and whether the stacking of the apartments provides an area outside the living room that could be used as a balcony.

The next phase in the genetic algorithm is to select, from the evaluated and scored population, the parents that will contribute genetic material to the next generation. In order to do this Shape Evolution implements a simple binary tournament selection scheme. The shape codes of the two parental genotypes that have been selected are bisected at a random position and the halves are swapped and rejoined to create the genotypes for the offspring. The parental genotypes are paired sequentially: the first individual is paired with the second, the third with the fourth, and so on. The crossover process fills the new population with valid individuals that are composed of genetic material

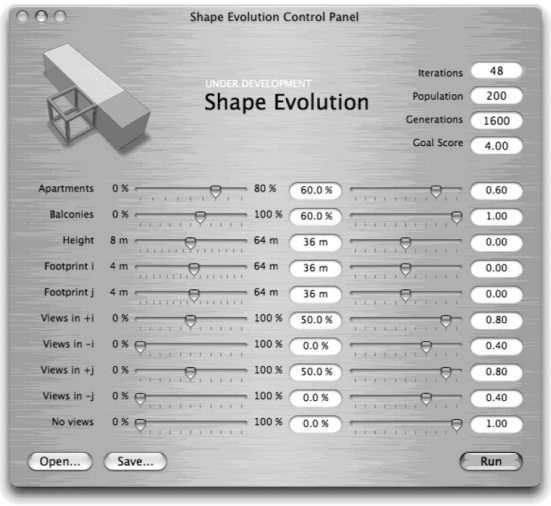

Fig 8. Unit interior design examples: Left. Luan Truong's design; Middle. Travis Louie's design; Right, Liko Dowling's design

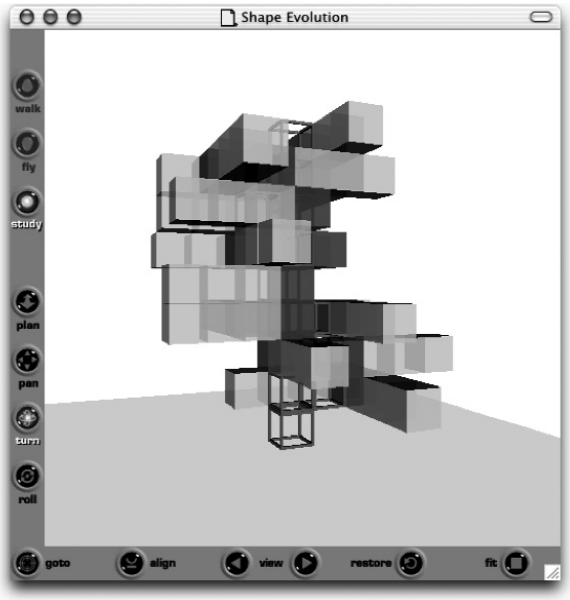

Fig 9. A typical apartment block design as viewed in a VRML browser

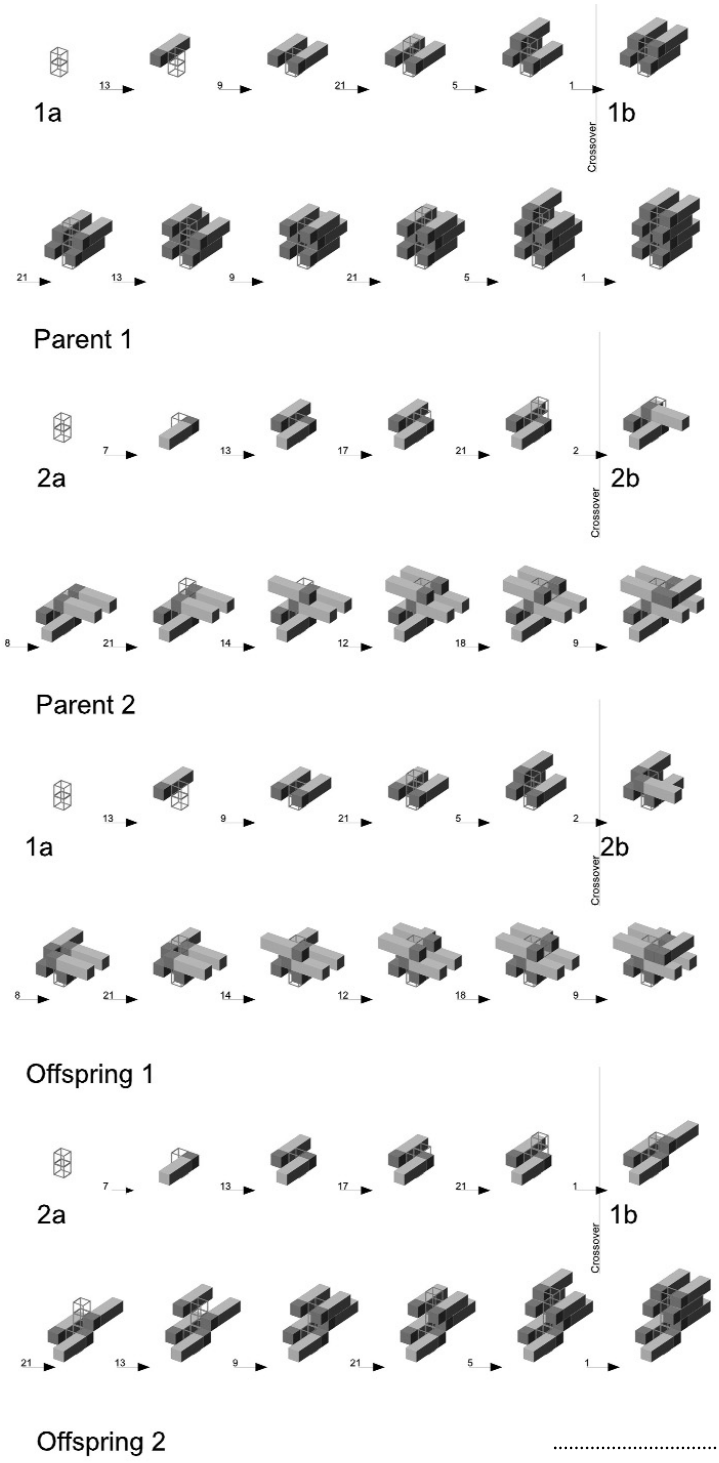

Fig 10.
Generating offspring from two parent genotypes

provided by the parents from the previous generation. Figure 10 illustrates how two parents are paired along with the resulting offspring.

The initial crossover point may result in invalid designs and, if this happens, a second crossover point is selected at random. This is repeated until either successful crossover is achieved or all possible crossover points have been tried, at which point the parents are returned to the population pool.

To introduce diversity into the new population and allow the exploration of new areas of the problem space, mutation is applied probabilistically. Every bit of every genotype in the population is capable of being mutated according to a user-defined mutation rate. Mutation actually involves changing the value of a particular bit to a random rule number from 1 to 22 and the mutation rate can be set by the user at any value between 0 and 1.

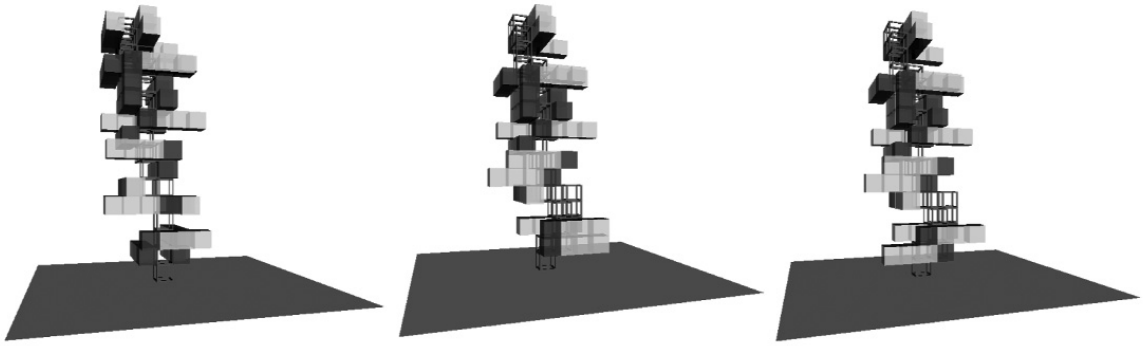

Fig 11.Some of the top designs produced for the tower block test

Each design that has been developed through the application of either crossover or mutation is converted into a three-dimensional array in order for it to be evaluated. As this process could also result in an invalid genotype the validity of each resulting genotype is checked to ensure that no clashes have occurred.

SHAPE EVOLUTION OUTPUT

Once the first new generation has been evaluated, its designs are scored, new parents are selected, crossover and mutation are applied, and another generation of designs is produced. This is repeated for either a user-defined number of generations (the end condition used in the prototype) or until a generation is produced with individuals whose score exceeds a user-defined threshold value. The results of the Shape Evolution run are then presented to the user. For the purposes of analysis, the prototype outputs an HTML document that collects and displays a significant amount of information about the last Shape Evolution run. Part of this information is the user-input variables: target values and weights for all criteria are presented, as well the population size, the number of generations, and the length of the genotype, i.e. the number of shape grammar rules used in each design. The user-defined value for the mutation rate is displayed alongside the actual mutation rate over all generations. The maximum score (a function of the various optimisation weights) is also shown.

A record is kept of all the champion designs. Shape Evolution does not produce ultimate, fully optimised solutions, so it would be deceptive to offer a single design at the end of its run. The purpose of the program is to inspire the designer by suggesting good solutions. Offering a number of alternatives does this, and also allows the designer a choice that can be influenced by criteria that were not dealt with by the Shape Evolution process.

TESTING SHAPE EVOLUTION

A number of design situations were used to test the Shape Evolution prototype with a selection of the evaluation parameters. As a basic goal, all designs attempted to minimize the number of apartments with blocked views. In each test Shape Evolution was run fifteen times, using a range of different settings for population size and mutation rate. For the purposes of this paper only one of the design situations will be considered: a tower block where the intention was to generate tall and thin buildings.

This was encoded by setting the goal value for

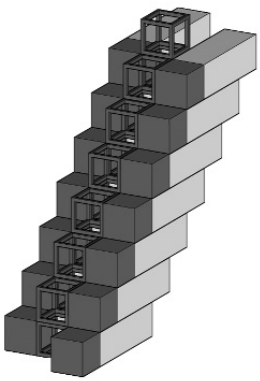

Fig 12. A design produced using Shape Evolution output as inspiration

the building's height to the maximum value, 64 metres. Both dimensions for the footprint were set to 24 metres, equivalent to six cubic modules. The weights for the height and the two footprint criteria were set to 1 and the genotype length was set to 48, meaning that 48 shape rules were applied to the initial shape. A series of runs were carried out with mutation rates set at values between 0.005 and 0.5.

In general, the building design concepts produced by the prototype were aligned with the design intentions. Furthermore, they presented novel solutions within the shape grammar defined for this class of designs. In this respect, the tool has been entirely successful: the designer has used a shape grammar and a set of design criteria as inputs and the program has generated a range of possible designs, which both meet the criteria and are interesting. For the purposes of experimentation with the Shape Evolution prototype, aspects of building functionality such as structural concerns, or the vertical continuity of elevator shafts were not considered. Still, for the most part, the requirements that were encoded were resolved successfully. Furthermore, the results display another positive trait, diversity. In most of the runs, the champion designs, i.e. the highest-scoring designs of all generations, were very different. Figure 11 shows some of the tower designs produced by Shape Evolution and Figure 12 shows one of the final designs.

In many of the test runs the genetic algorithm displayed a bias towards the exploration of new areas of the search versus the exploitation of high-scoring designs already existing in the population. This is not a desirable characteristic as it can lead to a fall-off in performance and changes to the genetic algorithm operators could be implemented in order to control this bias. Furthermore, the genetic algorithm performed more efficiently in the test cases where the design goals were not contradictory, as was the case with the tower block.

CONCLUSIONS

Having performed tests with the prototype, general comments can be made with respect to how Shape Evolution has addressed the original requirements. The goal was to produce a method and a tool that would be usable, i.e. fast and uncomplicated to its intended users, able to suggest functional solutions, and inspiring (by suggesting innovative design ideas). Also, by using shape grammars to generate the initial designs the tool satisfies another stated goal, the ability to produce designs consistent with the designer's stylistic and aesthetic requirements.

The test runs of Shape Evolution took several hours to complete. While the performance of the prototype may not be ideal, and a far cry from the envisioned real-time system, the reasons for this lie in the inefficient genetic algorithm code. The coding process was result-oriented, only aiming to produce working code, which inadvertently sacrificed efficiency.

Although this work is not the first to suggest that shape grammars and genetic algorithms can be linked, it has suggested a novel way in which this might be done; by using the shape code as the genotype. Shape Evolution allows for the definition of a design space by using a shape grammar, and only searches for solutions inside this space. This offers designers a significant amount of control over particular features of generated designs, namely the aspects of design that can be attributed to a particular style. Using computational power for what it can do best, Shape Evolution carries out a massively parallel exploration of the design space and this can yield high performance solutions.

Were Shape Evolution fully developed to the point where it could be made available to practicing designers, it could provide a powerful tool which would allow layouts of various kinds to be explored interactively. These could be at any level, from furniture layouts in a large open-plan space, through three-dimensional layouts of modular building components (as illustrated here), to the layout of large areas of cities. Its power does not lie in its ability to generate a specific solution, but to help the designer understand the boundaries of the world that he or she is exploring. When designing manually one might generate a few solutions and then choose the one that performs best. Shape Evolution has the ability to generate thousands of solutions and, as a result, the designer can learn how specific spatial characteristics relate to aspects of performance. The designer still has to choose

which of the various possibilities are worth exploring, but using Shape Evolution means that these possibilities are drawn from a much larger pool.

REFERENCES

BUELINCKX, H. 1993, "Wren's Language of City Church Designs: a Formal Generative Classification", *Environment and Planning B: Planning and Design*, 20, pp. 645-676.

GERO, J. S. and KAZAKOV, V. (1996). "Evolving Building Blocks for Design Using Genetic Engineering: a Formal Approach", in *Advances in Formal Design Methods for CAD*, Ed J.S.Gero J.S. (Ed.), Chapman and Hall, London, UK, pp. 31-50.

GOLDBERG, D.E. 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, Reading, MA, USA.

HOLLAND, J.H. 1975, *Adaptation in Natural and Artificial Systems,* University of Michigan Press, Ann Arbor, MI, USA.

KNIGHT, T.W, 1981, "The Forty-one Steps", *Environment and Planning B: Planning and Design,* 8, pp. 97-114.

KNIGHT, T.W. 1994, *Transformations in Design: a Formal Approach to Stylistic Change and Innovation in the Visual Arts,* Cambridge University Press, Cambridge, UK.

KONING, H. & EIZENBERG. J. 1981, "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses", *Environment and Planning B: Planning and Design,* 8, pp. 295-323.

KOUTAMANIS, A. 2000, "Representations From Generative Systems", in *Artificial Intelligence in Design '00*, in J.S.Gero (Ed.), Kluwer Academic, Dordrecht, Holland, pp.225-245.

LOOMIS, B.A. (undated), *SGGA: a user-driven genetic algorithm for evolving non-deterministic shape grammars,* Available: http://architecture.mit.edu/descomp/works/SGGA0502.pdf

ROLLO, J. 1995, "Triangle and T-Square: the Windows of Frank Lloyd Wright", *Environment and Planning B: Planning and Design,* 22, pp. 75-92.

ROSENMAN, M. & GERO, J. 1999, "Evolving Designs by Generating Useful Complex Gene Structures", in P.J.Bentley (Ed.), *Evolutionary Design by Computers,* , Morgan Kaufmann, San Francisco, CA, USA.

SHEA, K. & CAGAN, J. 1997, "Innovative Dome Design: Applying Geodesic Patterns With Shape Annealing", *Artificial Intelligence for Engineering Design,* Analysis and Manufacturing, 11, pp. 379-394.

STINY, G. & GIPS, J. 1972, "Shape Grammars and the Generative Specification of Painting and Sculpture", in C.V.Freiman (Ed.) *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, Holland, pp. 1460-1465.

STINY, G. 1975, *Pictorial and Formal Aspects of Shape and Shape Grammar,* Birkhauser, Stuttgart, Germany.

STINY. G. & MITCHELL, W.J. 1978, "The Palladian Grammar", *Environment and Planning B: Planning and Design,* 5, pp. 5-18.

STINY, G. & MITCHELL, W J. 1980, "The Grammar of Paradise: on the Generation of Mughul Gardens", *Environment and Planning B: Planning and Design,* 7, pp. 209-226.

WOODBURY, R.F. 1993, "A Genetic Approach to Creative Design", in *Modeling Creativity and Knowledge Based Creative Design,* J.S.Gero, & M.L.Maher, Lawrence Erlbaum Associates, New Jersey, NJ, USA pp. 211-232.

Authors' Addresses:
Orestes Chouchoulas & Alan Day,
Centre for Advanced Studies in Architecture,
University of Bath, UK
orestes@fufurasu.org a.k.day@bath.ac.uk