GRAMATICA

A general 3D shape grammar interpreter targeting the mass customization of housing

Rodrigo Correia[1], José Duarte[2], António Leitão[3]
[1,3]IST - UTL Portugal, [2]FA - UTL Portugal.
[1]rodrigo.correia@ist.utl.pt, [2]jduarte@fa.utl.pt, [3]antonio.menezes.leitao@ist.utl.pt

Abstract. *This paper presents a general 3D shape grammar interpreter named GRAMATICA and illustrates its use for the implementation of several shape grammars, including the one used in the design module of a specific shape grammar for mass-customized housing, called DESIGNA. The underlying shape representation, generation and control are discussed. The resulting shape grammar interpreter tries to support designers' ways of thinking and working by acting as a bridge between shape grammars, the formalism that captures a design process, and a CAD application, for post-processing the computed design. This bridge is implemented by Rosetta, which ensures portability among different CAD applications.*
Keywords. *Grammar interpreters; mass customization; housing; Malagueira; Siza.*

INTRODUCTION

Shape grammars are generative systems based on rules that allow capturing, creating, and understanding designs. They are based on the production systems of Emil Post (1943) and the generative grammars of Noam Chomsky (1957). Shape grammars work directly with shape computations rather than through symbolic computations (Knight 2000), where a shape is conceived as a finite collection of maximal lines (Stiny 1980). Designs are created by recursively applying a set of rules to an initial shape until a design is completed or no more rules can be applied. In general, several rules can be applied to any given shape, thus producing many different designs.

Parametric shape grammars can generate an even greater variety of designs by allowing the shapes to which rules are applied to have parameters. Even though this increases flexibility, it entails a more complex implementation mainly because the number of design solutions that a system can produce becomes extremely large, if not infinite.

Emergence is the ability to recognize and, more importantly, to operate on shapes that are not predefined in a grammar but emerge, or are formed, from any parts of shapes generated through rule applications (Knight 2000).

Shape grammars were developed by Stiny and Gips (1972). From this original work that illustrates the application of shape grammars for interpreting and evaluating works of art (Knight 2000) spawned a broader range of shape grammar theory and applications. Parametric shape grammars (Stiny 1980), color grammars (Knight 1989), grammars with weights (Stiny 1992), description grammars (Stiny 1981), structure grammars (Carlson et al. 1991), attributed grammars (Brown et al. 1994), and parallel grammars (Stiny 1991) are examples of research studies in the field of shape grammars. Shape grammars have been applied in areas, such as, Architecture, Engineering, and Product Design. Examples of applications include Queen Anne houses (Flemming 1987), Marrakech Medina urban form (Duarte et al. 2007) and coffee makers (Agarwal and Cagan 1998).

To automate the application of shape rules, researchers have concentrated their efforts on developing shape grammar interpreters. Previous summaries (Gips 1999; Chau et al. 2004) show that these researchers have focused on representations of shapes and algorithms for subshape detection and emergence, user interaction, and integration into the design process. Examples of shape grammar interpreters include the works of Gips (1975), Krishnamurti (1982), Flemming (1987), Chase (1989), Heisserman (1991), Tapia (1999), and Jowers and Earl (2010), among others.

GRAMATICA

Because current computer systems are implicitly symbolic, a shape grammar interpreter that is implemented in these systems needs to represent a shape symbolically. This section describes (1) the underlying data structures which GRAMATICA uses to represent shapes and labels, (2) the mechanisms used to control the application of rules to shapes, (3) and how the system generates and decides what rules to apply at a given time. Finally, GRAMATICA is evaluated as a means to implement several shape grammars and we show how a designer can use GRAMATICA in his creative process.

Shape representation

In order to represent a shape, GRAMATICA explores the work of Heisserman (1991) regarding logical reasoning about solids using first-order logic. Heisserman uses a split-edge data structure to represent solids which is a graph-based boundary representation. This representation allows the specification of clauses for matching on conditions of solid models and the generation of modifications to those solids.

The split-edge data structure is a variation of the winged edge structure to represent polyhedrons (Baumgart 1972). With split-edge, each edge is separated in two edge-half structures. One face and one vertex are associated with each edge-half, and each edge-half is associated with its other half.
In GRAMATICA, shapes are represented using the halfedge data structure provided by CGAL [1], a library for efficient and reliable computational geometry algorithms and data structures used in the academia and industry.

Similarly to Heisserman, GRAMATICA shapes are represented as a set of vertices, edges, facets, and incident relations between them. This means that, in both data structures, topology is represented as a graph where the nodes are topological elements and the arcs represent the adjacencies between elements.

Contrary to the split-edge data structure, where an edge is split in two, halfedge breaks an edge in two opposing halfedges. One halfedge is associated with its incident vertex and facet, and also with its next, previous, and opposite halfedges. All these incident relations allow the shape to be queried efficiently and easily.

Also similarly to Heisserman, GRAMATICA uses euler operators (Baumgart 1975) to ensure a valid topological construction, thus avoiding invalid topologies, for example, a facet with a hanging edge.

However, unlike Heisserman, GRAMATICA uses multiple representations for numbers and geometric calculations, allowing users to choose between different degrees of precision and speed. With the highest numerical precision, GRAMATICA avoids common problems associated with rounding errors. However, GRAMATICA always uses exact queries, meaning that geometric tests are always correct. For example, checking if a point lays on one side or the other of a plane will not be affected by numerical imprecision. This means that numbers are tailored for speed, but all geometric queries sacrifice execution time and storage space over exactness.

Given the graph representation of a shape and its geometry, labels are directly implemented in the elements of a shape by means of a hashtable, associating key-value pairs. In general, labels are used to distinguish elements of a shape by associating non-geometric data with any topological elements of shapes. Labels could also be used to restrict rule applications by imposing specific conditions on the shape generation.

In summary, a shape in GRAMATICA is represented as a stack of layers where (1) the bottom layer contains topological information relating facets, halfedges, and vertices with each other, followed by (2) the layer of geometric information that is associated to each vertex, and, finally, (3) the layer of labels where non-geometric information is associated to the topological elements of a shape. All these layers are managed by Rosetta (Lopes and Leitão 2011), that abstracts the use of CGAL and allows the visualization of the designs in different CAD tools.

Shape generation and rule representation

Using a rule system to implement a shape grammar interpreter poses several issues mainly how to control the order in which rules are applied. For example, a designer who uses the interpreter to implement a given grammar will end up using specific mechanisms of the rule system, such as, labels or the salience property, to tune the application order, even though these mechanisms were not developed to this end.

GRAMATICA takes a different approach by looking at the shape grammar as a state-space: each state encodes a particular design and each shape grammar rule is encoded as a transition operator that moves from one state to another. The application of the shape grammar to a given design can then be seen as a search in the state-space for a path that connects the initial state to some goal state. This path encodes the sequence of shape grammar rules that, starting from that initial design, leads to the final design.

In general, the specification of the initial state is direct, as there is usually an initial design available. However, the specification of the goal state is much more abstract as, in general, we know some of the properties that it must possess, but not its actual shape. This means that, in practice, the goal state is described by a predicate that is true only when all the properties of the intended final design are satisfied.

There are several different strategies for finding a path in the state-space that connects the initial and final states. In order to understand these strategies, it is important to realize that, in most cases, it is impossible to actually generate the entire state-space because the application of shape grammar rules is a combinatorial problem with an enormous number of possibilities. In fact, for the majority of problems, the number of states grows exponentially with the number of transitions and, due to memory limitations, this means that searching the space-state must be done incrementally, by generating the space-state as the search proceeds. In order to do this, the transition operator becomes a generator: its application to a given state generates the "next" state, that is, the state that represents the design that results from the application of the corresponding shape grammar rule. The application of all possible transition operators to a given state is then called the expansion of that state.

In general, all search strategies are based on the recursive expansion of states, from the initial to the goal state, and the order in which the states are expanded determines the search strategy. However, when the goal state is reached, it might be necessary to know which path was followed. To this end, each state is enriched with additional information, namely, which state and which transition operator were used to generate it. This enriched state is called a node and the exploration of the state-space entails the corresponding enlargement of the graph connecting these nodes. In this graph, the edges correspond to the transition operators (i.e., shape grammar rules) that were applied to a node to produce its descendants. Expanding a node entails expanding the corresponding state followed by the creation of a node for each generated state.

GRAMATICA follows the approach we have just described. In GRAMATICA, each shape grammar rule is encoded as an expansion operator that represents the transformation of a design. Each operator has two parts, known as antecedent and consequent. The antecedent describes the design to which the operator applies, while the consequent describes the design that results from the application of the operator. In practice, to minimize the number of operators that must be written, each operator uses, instead of a consequent, a set of consequents, i. e., a set of possible designs. Given that this set is computed during the search of the state-space, it is possible for this set to be empty, meaning that the specific operator could not make the transition from a given state. Other operators, however, might be able to compute such transitions.

For illustrative purposes consider the following example (see figure 1). The shape grammar rule states: if in a given design there is a shape that resembles a rectangle then this shape can be transformed into a different design where the rectangle is divided in two parts, where one of the parts has a height that is 1, 2, or 3. In GRAMATICA, we encode this rule by defining a transition operator whose antecedent checks that a rectangle is present and whose consequent is a set of three designs, one for each possible height.

Figure 2 presents a different example where the shape grammar rule states: if a facet is found in a given design, transform this facet into a pyramid with the apex at some perpendicular distance from the centroid of the facet.

One possible design from recursively applying this rule to a given facet in a tetrahedron is presented in figure 3.

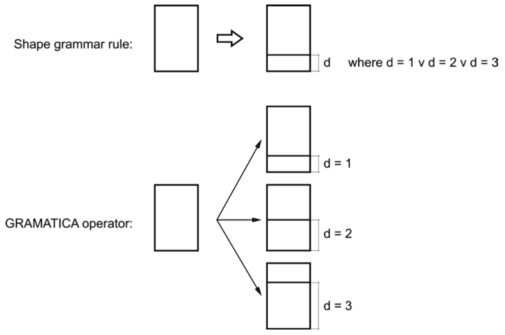

Figure 1
Shape grammar rule vs. GRAMATICA operator.

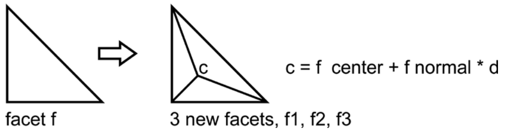

Figure 2
Simple 3D shape grammar rule.

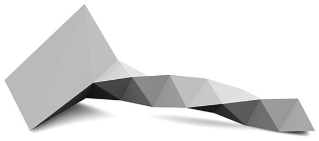

Figure 3
Application of a shape grammar rule.

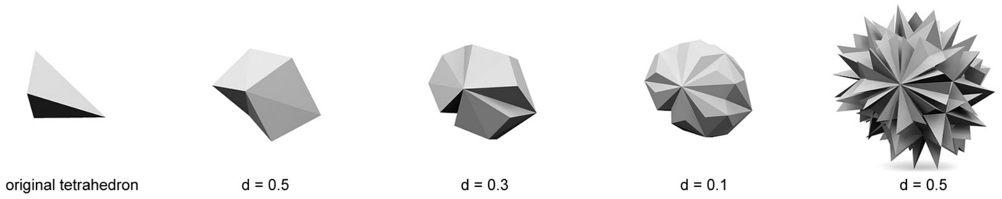

Figure 4
Sequence of shape grammar rule application with parameter variation.

Figure 4 presents the sequence from initial shape, a tetrahedron, to the final design after applying the rule of figure 2 to all the facets of current shape and changing the parameter d at each application of the rule.

Given that different search strategies can be used and that their success (or failure) depends on the specific problem one is trying to solve, GRAMATICA does not force a particular search strategy and, instead, it provides three different ones: depth-first, breadth-first, and A*. Depth-first search explores a complete path of the state-space graph before exploring another path. Breadth-first search explores all paths with length n before advancing to length n+1. A* search ranks paths according to the cost from the initial state to the current state, and the estimated cost decides the best state to expand based on a function that adds the cost how close a state is to the goal plus the cost of getting from the initial state to the current state.

In fact, GRAMATICA allows designers to define their own search strategies or, alternatively, to specify a particular order of rule application. This is shown in the next section.

EVALUATION

The main focus of this section is the implementation of a shape grammar for mass customization of housing. However, we will start by describing a much simpler grammar that generates ice-rays, as a demonstration of the features of GRAMATICA, namely, maximal lines representation, emergent shapes, and order of applicable rules.

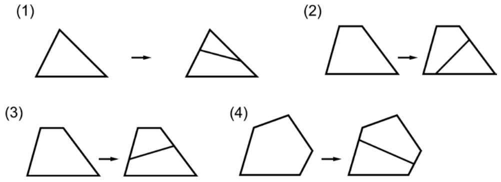

Figure 5
Original (simplified) ice-ray shape grammar rules.

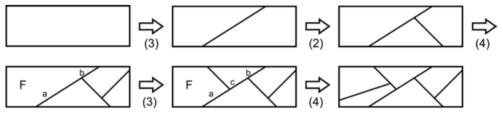

Figure 6
One possible sequence of ice-ray shape grammar rules application in GRAMATICA.

Ice-ray grammar

Ice-ray grammars were first formalized by Stiny (1977) using a shape grammar (see figure 5) as a means to describe the design of Chinese lattice.

Figure 6 shows the result of the above shape grammar for ice-ray, a rectangle as the initial shape, and the sequence 3, 2, 4, 3, 4 for the application of rules. This example shows that GRAMATICA is handling a shape representation as maximal lines. For example, the shape detected as F (see figure 6) the right edge is composed by the segment of line a and b and not only the segment of line a or b. Another interesting feature is the capability of shape recognition as showed again with shape F that was matched with rule 3. For this to happen, the rule needed to encode not only the topological information but also the geometric information, such as, edge length and the angle between edges.

Mass customization of housing

Mass-customization allows high-quality housing at affordable costs by relying on computer-aided design and manufacturing that does not depend on exhaustive repetition. This approach overcomes a common problem faced by designers of dealing with the design of large developments, for example, the difficulty of designing several different houses in a common style and the cost of building them without benefiting from economies of scale.

In this paper, we focus on solving this design problem using a discursive grammar, a rigorous mathematical model for the generation of forms according to a housing brief. From a technical point of view, a discursive grammar (Duarte 2005) is composed of a description grammar, a shape grammar, and a set of heuristics. For each rule in the shape grammar, there is a corresponding rule in the descriptive grammar, so that the shape evolves by rule application and its design description is continuously updated. Heuristics are used to guide the application of rules, for example, to select or limit the rules that can be applied at each step of the generation or to evaluate and select the entities that are closer to some pre-established goal. This allows discursive grammars to generate designs that are not only formally valid but are also semantically correct.

From the operational point of view, the discursive grammar is formed by two independent, but compatible, grammars, linked in sequence. The former is a programming grammar, already discussed in Duarte and Correia (2006) that is encoded as a description grammar, formulating a housing brief constrained by user input and a set of regulations. The housing brief is then used as input to the designing grammar exemplified bellow, which is encoded as a description grammar and a shape grammar tied together. These grammars compute a house design so that its description matches the housing brief. For experimentation and evaluation purposes, we use realistic grammars: the programming grammar follows the rules of PAHP - the Portuguese housing program and evaluation system, while the designing grammar encodes the rules laid out by the architect Álvaro Siza for the design of the Malagueira houses, an award-winning project that is under construction since 1977.

For illustrative purposes, we will only describe two simplified rules layed out by Duarte (2005), which divides the lot into different spaces. Figure 7 shows such rules.

Each of these shape grammar rules was encoded in GRAMATICA and, starting from an initial lot, a search process finds different solutions for its division into spaces. This search process is illustrated in Figure 8 where one can see the different transitions that connect designs and possible designs with only two different rules.

CONCLUSION AND FUTURE WORK

In spite of several decades of research, there is still a lack of shape grammar interpreters, particularly, those that can handle three-dimensional shapes. By using a sophisticated and extensible geometric kernel to represent shapes in combination with customizable precise numeric operations and correct shape predicates, we provide GRAMATICA, a very flexible and generic shape grammar interpreter.

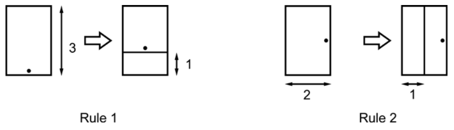

Figure 7
Original (simplified) mass customization shape grammar rule.

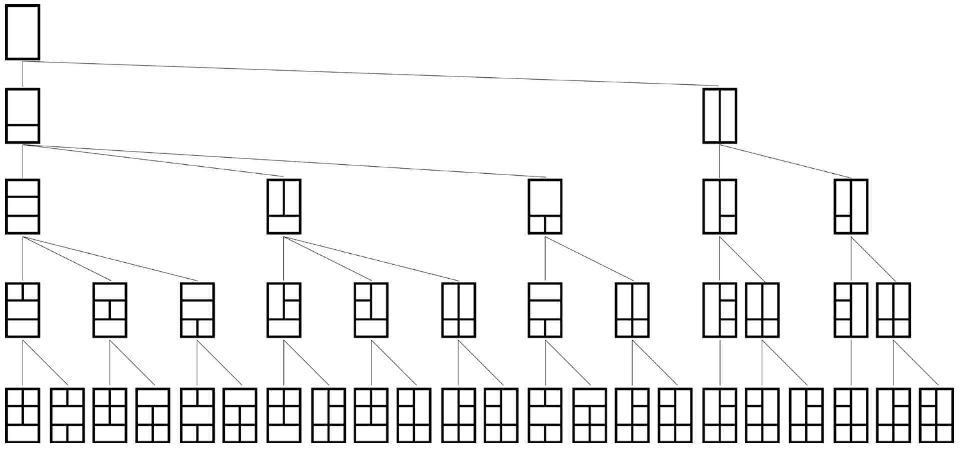

Figure 8
Derivation tree, applying rules of figure 7 to an initial lot.

In this paper, we described the most significant research related to shape grammars, and we presented the fundamental features of GRAMATICA, demonstrating its usefulness in dealing with different problems, both in two dimensions, as well as in three dimensions.

Emergence is still a hot topic of research, particularly, for shape grammars. In general, emergence can be very empirical and it is up to the designer to tell what and how the system recognizes as emergent. In the end, GRAMATICA provides the basic tools and some techniques to implement emergence of shapes. There are plans for linking CGAL to a graph transformation system and thus simplify the description of rules and allow better support to emergence.

In summary, while there's still work to be done, e.g. user interfaces, emergence, GRAMATICA provides the means (shape representation) and the tools (shape generation, search control, and visualization) to develop shape grammars.

ACKNOWLEDGEMENTS

This work was partially supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2011.

REFERENCES

Agarwal, M and Cagan, J 1998, 'A Blend of Different Tastes: The Language of CoffeeMakers', *Environment and Planning B: Planning and Design*, 25(2), pp. 205-226.

Baumgart, B 1972, '*Winged Edge Polyhedron Representation'*, Technical Report, Stanford University.

Baumgart, B 1975, 'A polyhedron representation for computer vision', *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*, ACM, Anaheim, California, pp. 589-596.

Brown, KN, McMahon, CA and Williams, JHS 1994, 'A Formal Language for the Design of Manufacturable Objects', *Proceedings of the IFIP TC5/WG5.2 Workshop on Formal Design Methods for CAD*, Elsevier Science Inc., pp. 135-155.

Carlson, C, McKelvey, R and Woodbury, R 1991, 'An introduction to structures and structure grammars', *Environment and Planning B: Planning and Design* 18(4), pp. 417-426.

Chase, S 1989, 'Shapes and shape grammars: from mathematical model to computer implementation', *Environment and Planning B: Planning and Design* 16(2), pp. 215-242.

Chau, H, Chen, X, McKay A, de Pennington, A 2004, 'Evaluation of a 3D shape grammar implementation', *Proceedings of the First International Conference on Design Computing and Cognition*, pp. 357-376.

Chomsky, N 1957, '*Syntactic structures*', Mouton, The Hague.

Duarte, JP 2005, 'Towards the Mass Customization of Housing: the grammar of Siza's houses at Malagueira', *Environment and Planning B: Planning and Design*, 32(3), pp. 347-380.

Duarte, JP and Correia, R 2006, 'Implementing a Description Grammar for Generating Housing Programs Online', *Construction Innovation Journal on Information and Knowledge Management in Construction*, volume 6, no. 4, pp. 203-216.

Duarte, JP, Rocha, J and Ducla-Soares, G 2007, 'Unveiling the structure of the Marrakech Medina: A Shape Grammar and an Interpreter for Generating Urban Form', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 21(4), pp. 317-349.

Flemming, U 1987, 'More than the sum of parts: the grammar of Queen Anne houses', *Environment and Planning B: Planning and Design* 14(3), pp. 323-350.

Gips, J 1975, '*Shape Grammars and Their Uses: Artificial Perception, Shape Generation and Computer Aesthetics*', Birkhaüser, Basel, Switzerland.

Gips, J 1999, 'Computer Implementation of Shape Grammars', *Workshop on Shape Computation*, MIT.

Heisserman, J 1991, '*Generative Geometric Design and Boundary Solid Grammars'*, doctoral dissertation, Carnegie Mellon University, Department of Architecture, Pittsburgh.

Jowers, I and Earl, C 2010, 'The construction of curved shapes', *Environment and Planning B: Planning and Design* 37(1), pp. 42-58.

Knight, TW 1989, 'Color grammars: designing with lines and colors' *Environment and Planning B: Planning and Design* 16(4), pp. 417-449.

Knight, TW 2000, '*Shape Grammars in Education and Practice: History and Prospects*', The Department of Architecture School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA.

Krishnamurti, R 1982, '*SGI: a shape grammar interpreter'*, Research report, Centre for Configurational Studies, The Open University, Milton Keynes.

Lopes, J and Leitão, A 2011, 'Portable Generative Design for CAD Applications', *Proceedings of the 31st annual conference of the Association for Computer Aided Design in Architecture*, Banff, Alberta, Canada, pp. 196-203.

Post, E 1943, 'Formal reductions of the general combinatorial decision problem', *American Journal of Mathematics*, 65, pp. 197-215.

Stiny, G 1977, 'Ice-ray: a note on the generation of Chinese lattice designs', *Environment and Planning B* 4(1), pp. 89-98.

Stiny, G and Gips, J 1972, 'Shape Grammars and the Generative Specification of Painting and Sculpture', in C. V. Freiman (eds)., *Information Processing*, 71, (North Holland, Amsterdam, 1972), pp. 1460-1465.

Stiny, G 1980, 'Introduction to shape and shape grammars', *Environment and Planning B* 7(3), pp. 343-351.

Stiny, G 1981, 'A note on the description of designs', *Environment and Planning B* 8(3), pp. 257-267.

Stiny, G 1991, 'The Algebras of Design', *Research in Engineering Design 2* (3), pp. 171-181.

Stiny, G 1992, 'Weights', *Environment and Planning B: Planning and Design* 19(4), pp. 413-430.

Tapia, M 1999, 'A visual implementation of a shape grammar system', *Environment and Planning B: Planning and Design* 26(1), pp. 59-73.

[1] www.cgal.org.