SGtools
A Computer Tool for Exploring Designs with Set Grammars

ROMÃO Luís
School of Architecture and Planning, Massachusetts Institute of Technology, USA

Keywords: shape grammars, human-computer interaction, collaboration

Abstract: A set grammar interpreter is presented in this paper. It differs from previous interpreters in many ways: it accepts any shapes for edition, the user rather than symbols, manipulates shapes, and rules can be stored and retrieved. This tool is intended as a conceptual design tool and not as a tool for full design development. The tool has been developed in the AutoLisp language as a plug-in to AutoCAD, thereby taking advantage of the existent means of visualizations.

1 INTRODUCTION

This paper reports on a part of the research in which the author suggests the need to strengthen the connection between architecture practice and computation. Nowadays the computer is used as a napkin to keep ideas and as a displayer of final designs. However, there is a bridge between both, between architecture practice and computation, which should be further explored. On one hand, the creative mind of the designer should not be afraid of crossing that gap. On the other hand, the graphic component of the designer mind cannot be off to the computer tools.

Designers are always intermediates of the design process. Conversely, to Yona Friedman, what counts is the designer being responsible for the dwellings designs, not someone else (Weinzapfel and Negroponte 1976). Additionally, it is not clear what Bernard Rudofsky presented in his work: *Architecture without Architects*. Without indecisions, architects, as we "know" them, have not design those buildings, but the people with "the" required skills built them. Why cannot we call them designers too?

In association with the work of Nicholas Negroponte (Weinzapfel and Negroponte 1976), the following set of ideas is proposed. First, develop a man-machine interface, especially graphic, which enhances draft and perceptive skills inherent to designers. Second, introduce user-friendly techniques to help designers implement their own designs, and third, create a visual environment to help user and designer dialog comprehensively, specially when the proposed design is not similar with another already built.

In building computational devices, one should attend clearly to the characteristics of the designer. Whatever, we may assess what a designer is, one thing we know, he deals more with shapes and space than with numbers. Consequently, devices should enhance the handling of the forms instead of strings. A good starting point of the development for those devices should take into consideration shape grammars formalism. What is presented here is a tool that pursuits that formalism.

2 CONTEXTS OF THE TOOL

In current early stages of design, initial ideas are depicted by a huge collection of drawings using different viewpoints (plans and elevations, axonometric and perspectives), without concerns of scale, which allow a certain degree of ambiguity as in a drawing by Siza Vieira (IAC 2002). The proposed tool may permit the same ambiguity used in the current practice where the designer assesses results by visual means by allowing different viewpoints. It allows increasing and decreasing levels of ambiguity by introducing rules of adding or/and subtraction, introducing more or less meaning. Therefore, a range of constraints can be used to build a network, which varies accordingly to the designer's intentions. Until reaches a design representation, which is a network of meaning formed by drawings and descriptions, which helps to make unambiguous representations (Stiny 1992).

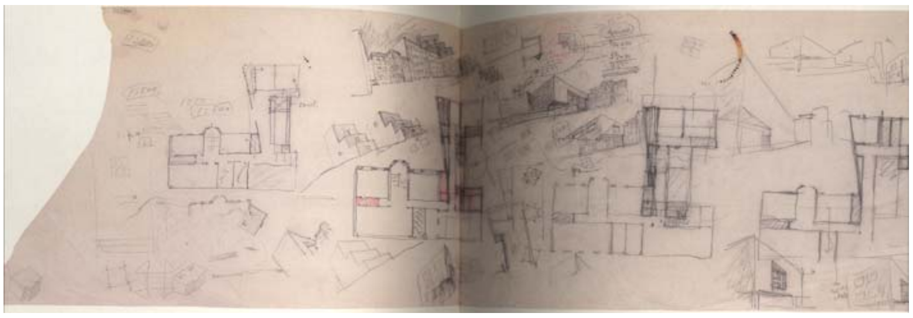

Figure 1 A Siza drawing

2.1 Background

This tool is a computer implementation that proposes mainly an approach to architecture practice using shape grammars. It was influenced by the works of James Gips and other authors. Gips wrote: a "student [an architect] learning to use shape grammars by using an interpreter program is learning both about shape grammars and how to use the program" (Gips 1999). In that work, he listed several computer implementations for shape grammars that have been built for different reasons. To that list can be added the works of José P. Duarte and Y. Wang (2001), *3dshaper*, for education and practice, of Miranda McGill (2001), called *2dshaper*, for education, and the work of Gabriela Celani (Celani 2001), called *Autogrammar*, also for education.

Celani uses 2D and 3D shapes controlled by a registration mark that is created by a user after verifying the symmetry conditions of the shape(s). The McGill implementation is displayed in the Internet. By using two shapes, a definition of rules and their application are illustrated for everybody in the world. Both were built for educational purposes. The work of Duarte and Wang was made for practical reasons, in which a connection between designs and rapid-prototyping is shown. It explores and facilitates the control of complex designs generated by shape grammars rules.

In formal terms, this presented tool could derive from the colour shape grammars of Terry Knight: A = {s, P, F}, where *s* stands for a shape, *P* for a label, and *F* for a field colour (Knight 1989). Here a shape is put inside the definition of the shape as this: A= {s1, P, (s2, F)}. In this new formula, *s1* stands for a shape, which is controlled by its symmetry properties. No matter what *s2* shape is, and it can be any shape, the labels assigned to *s1* control it. The stage of development of this tool, *s1* can only be a rectangle or an oblong. At this point, it seems interesting to explore in the future other different shapes that could be taken from Froebel blocks.

2.2 Goals

This tool facilitates the use of shape grammars to those who do not have a programming language background. This will increase the number of possible users and should improve the relationship between computers and designers.

The tool exempts a user from the requirement of using symbolic manipulations to design with grammars. Gips (1999) made a distinction between visual minds and symbolic ones. Designers interact with the tool by manipulating shapes, not symbols, allowing them to do what they know best: compute with shapes and its relationships.

The user introduces shapes and rules, evaluates design results by visual means and changes rules or rule application accordingly. Rules can be stored and retrieved when needed. At the end of this interactive process, he may select a design result and export it to another software application for full design development.

The user who builds the grammar and the one who applies it may be the same. The computer may also function as one joint partner in a group. It can be “taught” and all group members can use it in the exploration of a design world defined by a grammar. The process of developing a full grammar can also be a shared process with one user creating a grammar and others modifying it until an intended design is reached. An example is the Andrew Li proposal (2002): a user creates a grammar and others introduce rules to obtain a meaningful architecture design.

3 THE FRAMEWORK

This application is a parametric tool that works inside AutoCAD. It allows the use of the shape grammar formalism ($A \rightarrow B$) (Stiny and Gips 1978), and it works based on the strategies of set grammars, which means that "consists of subsets of (S, L)* and applies to subsets of labelled shapes to produce other such sets" (Stiny 1982, 113). It works with algebras from U_{00} to U_{03}, in which $_0$ stands for shape dimension and $_3$ for space dimension (Stiny 1999). The script was created in AutoLisp language, a built-in programme language by AutoDesk. Rules are applied in this way:

$$C_{n+1} = (C_n - A) + B,\ n>0,\ n \in N \qquad (1)$$

Descriptions to build parametric set grammars are based on seven parts:

$$Ss = <F,\ V,\ U,\ T\ (\alpha,\ \beta,\ \chi,),\ R\ (\alpha,\ \beta,\ \chi,\ \delta),\ W,\ I> \qquad (2)$$

In the group of shapes there are: *F* is a finite set of shapes, rectangles and oblongs; *U*, is a finite set of shapes; *V*, is a finite set of labels; *W*, is a finite set of colours. In the transformation group, there are: $T\alpha$; translation, reflection, scale and rotation, or a finite composition of these, $T\beta$; scale and rotation, or a finite composition of these $T\chi$; translation. For the rules, $R\alpha$; a set of generic rules, $R\beta$; a set of substitution rules, $R\chi$; a set of parallel rules, $R\delta$; a set of rules made from existent rules. Finally, *I* is a initial shape. The condition to apply the rule is $\nu \rightarrow \psi$, $\{\nu,\psi\}\ \varepsilon\ R^+$. Result: T [L ($\nu$)] [ϖ; where T^+ is a possible transformation. Mode of production application is serial and parallel (rewrite rules). Addition operator is the insertion of a shape. The subtract operator is a substitution shape rule. The rule definition is [$T\chi$ ($T\beta$ ($T\alpha$ (F, V, U, W)))].

3.1 Shapes

There are two types of vocabularies for shapes and one for colour. There are "phantom" grids where transformations will occur in algebra up to U_{03}, that are called *frames*, *F*. The other type of shape is any shape in algebra up to U_{03}, that it is called *shapes*, *U*. In addition, the *W* element introduces colour to shapes.

The next picture shows the vocabulary of frames that have two shapes *F*, a rectangle and an oblong, and the vocabulary for shapes *U,* that can be any shape represented in two and three dimensions (Figure 2). The oblong or the rectangle shape works as a hidden skeleton or a "phantom" grid. Euclidean and affinity operations are used to transform shapes and to define spatial relations using its symmetry. Shapes in *U* will be subordinated to the transformations of *F*.

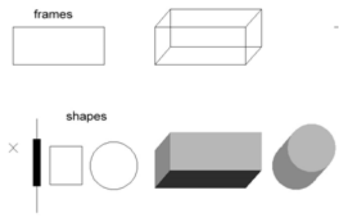

Figure 2 Frames and Shapes are not the same thing

Initial shape can be an empty shape, a “dummy”, or an existent part of U.

3.2 Transformations

In Figure 3, can be seen how spatial relations can be applied to frames and how symmetry properties help control transformations for rules. Frame shapes in both algebras U_{02} and U_{03} define grids in space where transformations will occur. These grids can vary in size. In the end, they will be omitted or erased.

Euclidian transformations and scale can be applied up to three levels of transformations and in combination for each rule. A first transformation is mandatory.

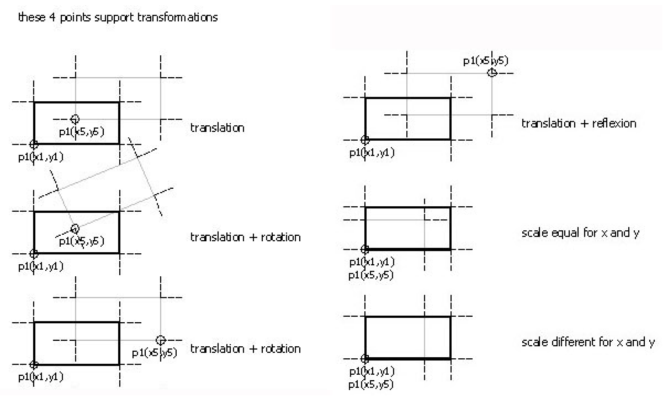

Figure 3 The symmetry properties of the shape and its transformations

3.3 Rules

There are rules of adding and subtraction. We can define them by visual means, and by introducing numeric values. To simplify the following notation the rule definition ($T\chi$ ($T\beta$ ($T\alpha$ (F, V, U, W)))) will be omitted. The following types of rule can be performed:

Adding rules: Type 1, $\& \rightarrow B$: ($b+T\alpha(b)$); Type 2, $\& \rightarrow B$: ($b+(T\beta$ ($T\alpha(b)$))); Type 3, $\& \rightarrow B$: ($T\chi(b+(T\beta$ ($T\alpha(b)$)))); where “b” stands for a shape.

Rules of adding and subtraction are: Type 4, $A \rightarrow B$: $(a + T\alpha(a))$; Type 5, $A \rightarrow B$: $(a + (T\beta\ (T\alpha(a)))$; Type 6, $A \rightarrow B$: $(T\chi(a+(T\beta\ (T\alpha(a))))$; Type 7, $\& \rightarrow B$; Type 8, $A \rightarrow \&$; Type 9, A: $(a, l^1) \rightarrow$ B: $(T\alpha(a, l^2))$; Type 10, A: $(c + d)\ \{c, d\}\ \varepsilon\ R\alpha\ V\ R\beta \rightarrow$ B: $(a + T\alpha(a))$; where "a" stands for a shape.

The first three types are common and they add shapes to design recursively. A shape can have three levels of transformations. The first is obtained from the spatial relation defined by visual means. The second is defined from the previous plus a rotation or scale transformations, which are defined by numeric values. The third is a previous rule definition plus a translation transformation, which is applied when the rule is introduced.

The substitution rule is applied to an existent shape. We can apply rules of this sort: $(A \rightarrow \&)$ or the opposite $(\& \rightarrow B)$. This means that the initial shape can be a set that includes &. Rules can be used in a parallel way at the time we apply the rule. Stiny defined this parallel concept in an early work (Stiny 1975). Rules can be also created from subordinate shape rules: $(A \rightarrow B{:}(TA+A))$. Or: $(A \rightarrow B{:}(T\ (TA))+A))$ and $(A \rightarrow B{:}(T\ (T(TA))+A)))$. We can have a simple rule where $(A \rightarrow B)$ and/or when B is a new shape that the only variation is the label: A:$(a, V^1) \rightarrow$ B:(a, V^2). All rules except this last one can be applied recursively.

The rule type 10 is particularly important for an automatic context. They can be picked shapes on design to create new rules. This is a concatenation rule that ties dissimilar types of rules and shapes.

4 APPLYING THE TOOL

Here are two examples that show how the tool works. Rules are used and presented in a graphic way allowing a better understanding.

4.1 A Simple Example

Let us choose a group of shapes and a frame. Shapes are defined in 3D but the rule will run in a plane, within a 2D frame, using rectangle symmetry. Then, we choose a spatial relation and define rules, for example, a location for patio houses. A first schema is drawn and it may be accepted or not depending on evaluation. We can change labels and shapes anytime, but not frames. We can go back and decide a new spatial relation, and/or a new rule. We can compare different solutions by visual means.

The benefits given by the graphic environment are that shapes can be seen through different viewpoints: linear perspective, axonometric and orthogonal projections (Figures 4 to 6). By using the computation, in U_{03} algebra, we can see architecture descriptions as the Siza's example. We are not constrained by the paper limits and we have to use our vision skills. In addition, we do not need to work with scale, only with proportions.

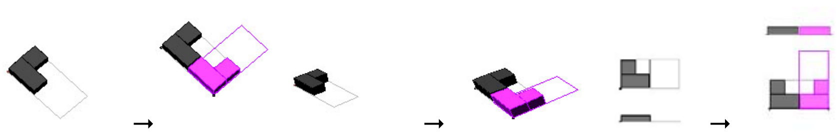

Figure 4 The left and the right hand side of the rule in three views

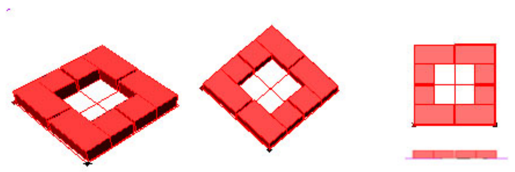

Figure 5 A design is shown after three serial rule applications

The design represents one-storey building, later substituted by a two-storey one. Therefore, the spatial relation is the same but with different shapes. A substitution rule was used. A new shape of the right side of the rule will substitute shapes that correspond to the left side of the rule (Figures 7 and 8).

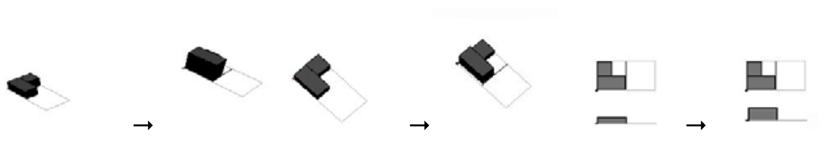

Figure 6 A rule substitution is created

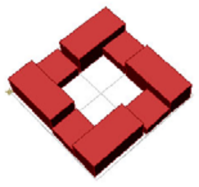

Figure 7 A new design after applying the substitution rule

The next figures illustrate how Euclidian transformations work depending on the relation between shape+frame (A) and shape+frame (B) in a parallel way for rules at different contexts (Figure 8). We have two rules. First, we introduce recursively the *rule1*, the yellow box. Then the second rule is introduced, and an existent shape is picked. It will produce a square building implantation. We can introduce a substitution rule later (Figure 9).

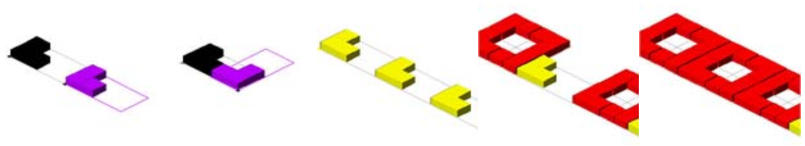

Figure 8 Rule1 and rule2, and rule application

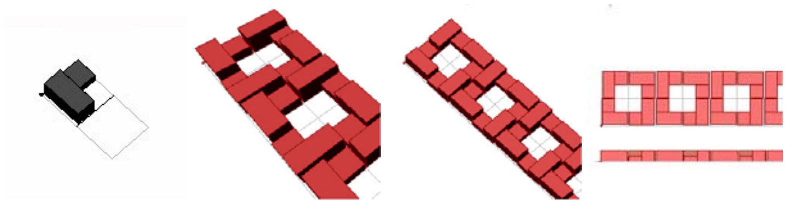

Figure 9 Shapes were changed by the substitution rule

A solution evolves when the "gestures", i.e., when rule applications overlap themselves towards a final design: structural system, floors, windows, doors and so on. The user can play with different grids of shapes, which correspond to different knots of meaning. It could also be introduced an estimate cost for it if a cost per unit is calculated by extrapolation. Other detail rules could be applied and the design could be exported to a full development application.

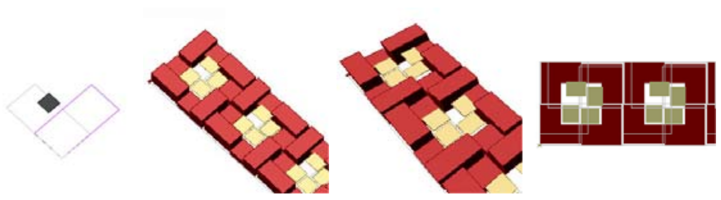

Figure 10 Applying another rule to a previous design

Starting from an ill-defined problem a designer can now try different formal results. A holistic process is assessed; shapes are introduced and evaluated visually in combination. A user can track the entire design. Rules can be kept in a list of commands in a text file for to be used later.

4.2 Another Example

Another example is shown: the Chinese lattice designs, a grammar created by Stiny (1977). It was transposed into a set grammar. Shapes are lightly different. Labels dots were substituted visually by a frame (Figures 11 and 12).

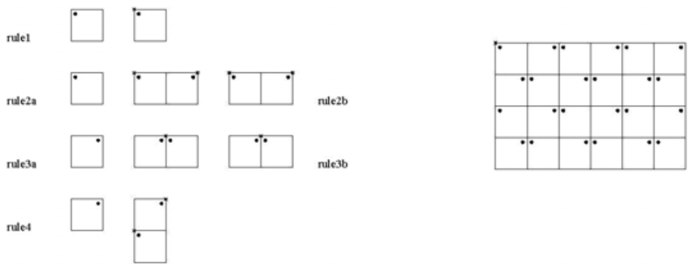

Figure 11 A first example

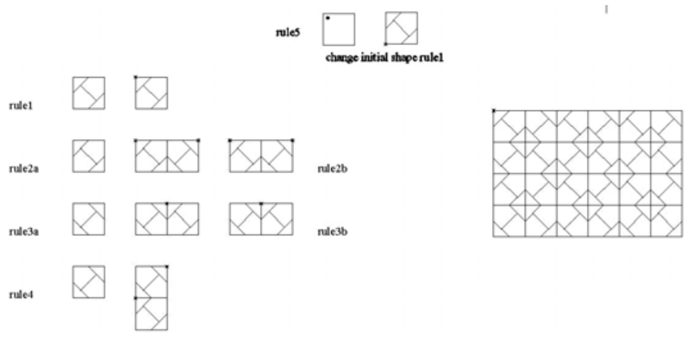

Figure 12 A variation of the previous

Rule1 is embedded in all the other rules. In this case, rules were applied manually. There were created two more rules apart from the original grammar, to define directions: for the right direction there are rules 2a and 3a, and for the left direction, rules 2b and 3b. After introducing structure rules, it was introduced a fifth rule. It changes initial shape of the *rule1*. The fifth rule may be changed at any time.

4.3 Applied to a Classroom

This tool has been tested in classroom. Students use it easily. They do not know any programming language. They work with shapes to create designs, as they could do in design studio classes, evaluating preliminary results and changing rules whenever they want.

5 CONCLUSIONS

This is a work in progress. Although this tool already fulfils some of the proposed goals, it is still behind of what it can and should offer. Meanwhile, the user can create a grammar by using shapes and can also save and retrieve rules. The user can evaluate final designs and change them by changing shape rules instead of parameters in a string. However, there are three critical reasons to keep the work. First, it is necessary to introduce labels describing symbolic or text information related to shapes, giving architectural meaning to them. Second, the interface for automatic generation and evaluation needs to be facilitated. Third, the universe of design exploration would be expanded if the oblong and the rectangular frames took other shapes as well.

ACKNOWLEDGMENTS

The author wants thanking to the reviewers and José Duarte of their comments, and the supports funds given by the Fundação Calouste Gulbenkian and the Fundação Luso-Americana.

REFERENCES

Celani, Gabriela. 2001. MIT-MIYAGI Workshop 2001: An educational experiment with shape grammars and computer applications. *International Journal of Design Computing*, v.3, http://www.arch.usyd.edu.au/kcdc/journal/vol3/index.html; accessed February 2, 2005.

Gips, James. 1999. *Computer Implementation of Shape Grammars* [invited paper], Workshop on Shape Computation, MIT.

Knight, Terry. 1989. Color Grammars: Designing with lines and color. *EPB: Planning and Design* 16: 417-447.

Li, Andrew. 2001. *A shape grammar for teaching the architectural style of the Yingzao fashi.* PhD-Thesis, MIT.

McGill, Miranda. 2001. *A Visual Approach for Exploring Computational Design.* SMArchS thesis, MIT.

Stiny, George. 1975. *Pictorial and formal Aspects of Shape and Shape Grammar.* Basel: Birkhäuser.

Stiny, George. 1977. Ice ray: a note on the generation of Chinese Lattice designs. *EPB: Planning and Design* 4: 89-98.

Stiny, George, and James Gips. 1978. *Algorithmic aesthetics: computer models for criticism and design in the arts.* Berkeley: University of California Press.

Stiny, George. 1982. Spatial relations and grammars, *EPB: Planning and Design*, 9: 113-114.

Stiny, George. 1992. Weights, *EPB: Planning and Design* 19: 413-430.

Stiny, George. 1999. Commentary. *EPB: Planning and Design* 26: 7-14.

Ministério da Cultura/IAC. 2002. *Drawing Design Project,* Lisboa: MC/IAC.

Wang, Yufei, and José P. Duarte. 2002. Automatic Generation and Fabrication of Designs. *Automation in Construction* 11, 291-302.

Weinzapfel, Guy, and Nicholas Negroponte. 1976. Architecture-by-yourself. *ACM SIGGRAPH Computer Graphics* 10: 74-78.