DYNAMIC (SHAPE) GRAMMARS

DOUNAS THEODOROS & KOTSIOPOULOS M. ANASTASIOS
Department of Architecture, Aristotetleio University of Thessaloniki, Greece
dounas@gmail.com, kotsiop@arch.auh.gr

Abstract: *The research presented in the paper explores the creation of custom shape grammars with animation tools, either as learning or educational tool or for the purposes of architectural design. Standard shape grammars contain an initial shape or design and one or more transformation rules. In a simple scenario the designer just applies the rules in the initial design or in a complicated scenario has to choose which rule to apply. Dynamic shape grammars on the other hand use animation tools to produce dynamic rules of transformation, or even dynamic – parametric initial shapes on which to apply the rules on. The dynamic state of the rules in our system allows the designer to change the rules during designing without having to abandon a core idea or concept. Furthermore the implementation with an animation tool allows the design system to be form-independent and express the underlying structure of an architectural idea with non-graphical connections like parent and child relationships, or other deformation rules. It can be shown that in a computation context dynamic shape grammars are actually groups of standard shape grammars where the grammars in the group share the classification of the transformation rules they contain. The system that we present allows the designer to change between the grammars in one group in a transparent way without expressing the grammar formally but by only manipulating simple objects inside the animation software package. This transparency focuses the effort of the user in simply design and keeping track of the formal declarations of shape grammars while the multiple dynamic grammars remove the obstacle of conforming to a single set of rules. The benefits of this effort can be especially seen in actual architectural design where the focus is in developing a concept idea and not strictly adhering to the rules.*

1. Applications of shape grammars

Research into shape grammars has managed to produce a more or less feature complete design computation paradigm, even though some of its characteristics remain without explanation or cannot be anticipated. In Architectural Education shape grammars have produced a significant body of work in the teaching of architectural styles and design computation because their basic mechanism of simple shapes and simple rules that apply to the shapes is trivial to understand and applied by the students. In architectural practice, shape grammars still remain affiliated and restrained in academia even though complete mass customisation systems have been implemented (Duarte 2001). Even though the lack of adoption of shape grammars in everyday architectural practise can be attributed to the fact that the architectural professionals still avert their eyes from design computation methods, a common critique directed to the shape grammar paradigm is that it is too rigid to work with in a professional environment. Part of the research presented in this paper is implementing tools that make it easy for the professional architect to adopt the shape grammar paradigm in everyday practise (Gips, 1991), by forging a transparent and intuitive use of shape grammars without abandoning its advantages in form shaping and exploration. (Knight, 1991, 2003)

2. Dynamic Shape Grammars

The process a design professional uses can be characterised rather intuitive compared to the design production that shape grammars embody, because the initial shape-rule-application schema as usually presented relies on an almost religious following of the rules, mainly because it relies on the formulation of the rules before designing.

To take this logistical book-keeping of rule formulation and application out of the equation of design development we present an alternative model of shape grammar implementation. Based on animation tools our model can be called dynamic in the sense that the formulation of rules is dynamic and can be propagated also to the formulation of initial shapes.

The implementation uses the notion of complex grammars (Knight, 1991) and composite grammars (Chase et al., 2005) inside an off-the-self animation program. Like a generic CAD system, animation tools have a simple structure where generic tools combined with one another produce levels of higher complexity, exactly in the way that CAD tools (for example lines, polylines, etc.) when combined together produce complete designs. In effect our system is an "elemental combinatorial system" (Frazer, 1995), where the designer formulates relationships between different parts of each model or architectural artifact while using tools that transform geometry. These two "modes" of the system allow the designer to choose either the top down approach or the bottom up approach or a combination between them where the non-geometric tools and the geometric tools are used simultaneously. An added benefit of using common animation software is the fact that the shape grammars are expressed directly in three dimensions allowing the designer to explore alternatives in complete form.

3. Actual Implementation

The design begins with expressing the structure of the design idea using animation tools (Dounas, 2007) specifically animation key-frames, parent and child relationships between forms and path animations of forms and finally lattice mesh deformations. The structure of the design idea is a set of rules on how the forms behave to one another, what relationships they have and how a form affects another form. For example a rule that make a rectangle move on an axis by two lengths of the rectangle in every step can be expressed by a follow path structure and key framing (Figure 1).

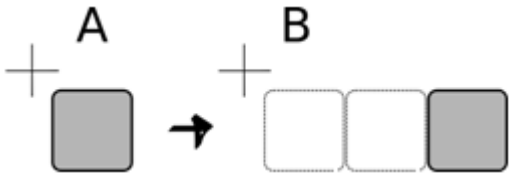

Figure 1: *follow path structure*

A second rule applied just to the initial shape locally (without accounting for the rule that transforms it along the axis) can provide variations, of the initial shape thus producing a group of shape grammars based on the same global rule of transformation along an axis. In Figure 2 the same rule is applied to the initial rectangle that substitutes the rectangle with a circle thus arriving to a different shape grammar

Using classic algebraic notation one can represent these grammars in a form where global rules are classic shape grammar rules informing on the production of the whole system and local rules are rules confined to the transformation of only specific shapes (this means that they transform parts of the shape while the shape retains its individuality as an object). The local rules altering the parts of the shape inform and change the global rules in terms of shape recognition. So when we have a local and a global rule running in parallel we have N shape

grammars running in parallel where N is the number of steps until we reach the termination rule. (Table 1)

Table 1: Local and global rules in parallel

"local"	initial shape A, Rule A-> A^d	A->A^d_1- >A^d_2....-> A^d_N
"global"	initial shape B, Rule B -> C+B ,where B = A^d	B -> C+B -> C+B+C ->...

This example implemented inside an animation program can look like this: A cube is transformed into a parallelepiped by moving one of its sides, while an array modifier produces six consecutive copies of the cube in transformation and rotation (Figure 2).

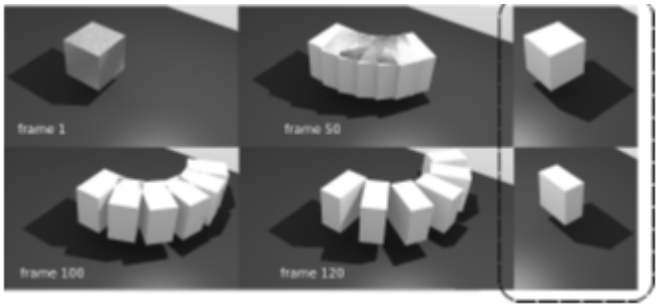

Figure 2: *Parallel implementation of a local and a global rule*

In the framed images on the right in figure 3 one can observe the transformation from a cube to a parallelepiped. In each step (or animation frame) of the example an instance of a different grammar is produced since the initial shape changes in each frame. All these grammars share the same global rule so in they appear to be the same grammar where the first initial shape or shape configuration "A" changes dynamically or is substituted in every step.

4. Implementation through animation tools

This implementation can actually be translated from complex grammars (Knight. 1991) or composite grammars (Chase. 2005) or discursive grammars (Duarte. 2001) and is based on those ideas. The expansion of these ideas though in our implementation lie in the fact that an off-the-self software program is used (specifically blender 3d, an open source modeling, animation, rendering, etc suite), the designer formulates a design idea early on in terms of design structure and expresses his ideas in a direct manner into the software. The implementation relies on animation tools to provide the expression of structure of the design idea and the logistical book keeping of when a rule is applied. They can be categorized in two groups: expression of structure and expression of form

Tools that express structure are keyframes (when is the rule or rules applied), parent -child relationships (handling of the propagation of rule application from the parent to the child), skeleton armatures (they direct the modification of the model in a way that resembles bones and muscles of a body), follow path (to separate the steps of the computation along an axis or curve) and lattice deformation (to deform an object locally inside a parent and child relationship). Tools that express modification of form are the modifier stack (arrays, Booleans, wave, curve, etc) and the duplicates mechanism. One can suppose that the application of these or similar tools are also implemented inside a common CAD software, but the advantage animation tools provide is that of temporal distance in the application of the rules. (table2)

Table 2: Temporal Distance in application of the rules

Rule a	Rule b	Rule e	Rule f
Rule c	Rule d	Rule g	Rule h
Rule j			

The manipulation of “When” (i.e. in which step of the computation) a rule is applied is accessible through the Interpolation and extrapolation (when we need a rule to be applied recursively or constantly) Curves of Animation software (Figure 3).

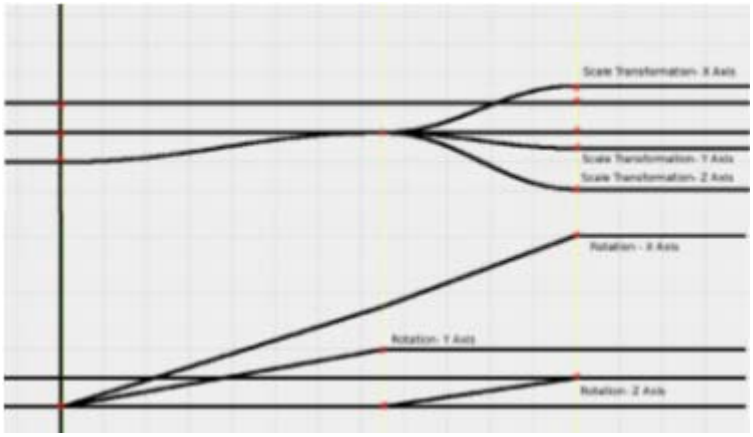

Figure 3: Interpolation curves showing the application of transformation rules on an object

The designer then can manipulate the duration of the rule, move the curve unmodified and consequently put the rule before or after another in the X axis or modify the difference of the curve in the Y axis thus increasing or decreasing the influence of the rule. Of course the designer can also do the same manipulations in a visual manner directly in the animated model, since the IPO curves are usually used when very exact manipulations are needed.

Using animation tolls does only benefit the combination of rules during a simulation but also helps the designer clearly structure her ideas employing the tools inherent in animation tools. Armatures for example are used in animation to simulate how bones and muscles work encapsulating constraints, functions and possibilities of how a shape can be de-formed, translated etc. This encapsulation of constraints and functions means that the designer can express a basic deformation structure inside his basic design idea: for example how tall or big can a cube/room be in relationship to another cube/room. Tools that carry constraints and functions of shapes related to one another are parent and child, follow path relationships and lattice deformations, mentioned earlier. Depending on the nature of the “design problem” and the conceptual idea the designer wants to develop the corresponding “structure” tool is used. Of course no single tool can represent a real world design problem or idea so the designer practically combines various tools used globally, locally or partially in a model to represent her idea in animation terms. This process allows the designer to think in architectural terms and not in a computation – specific terminology and process. Basic structural terms like “scale”, “axis” etc can be represented by the structural animation tools: scale relationship by a parent child-relationship, axis by a follow path relationship etc. After modelling the structural relationships between shapes in her conceptual idea the designer can start animating essentially producing steps in the computation, without really needing to define computation rules, termination rules, substitution rules etc.

Emerging from this mapping of “architectural” rules to animation tools is a classification of shape grammar rules (Table 3). The simplest structure of an architectural idea can be

mapped to a specific animation or transformation tool and thus every structural decision can be expressed as a stack of one or more animation tools that cooperate on shape and form in a given order. Of course changing the order that the rules are applied or deleting one rule and inserting another create a different series of designs and shape grammars. Taking this idea a bit further one can suppose the creation of a substitution and classification algebra governing the mapping, stacking and substitution of rules and/or animation tools inside the implementation software. The algebra can codify the structure of the rules, and then the structure of the architectural idea thus capturing the structure of the underlying shape grammar or grammars.

This mechanism of representing architectural ideas with shape grammar rules leads to a possible classification of shape grammars and of architectural configurations. A classification based on shape grammar rule is not of course new but the mechanism of mapping the rules to a stack of animation tools provides a practical implementation of capturing and storing explicitly the classification schemes inside the algebra. In a situation where the architect desires to retrieve an earlier idea she can do it by querying about the algebraic schemas that contain a certain rule/animation tool.

5. Implementation Example in a specific Building.

The scope of our system is the production of a system that can actually be used in architectural practice without forcing the designer to learn the exact formal mechanisms of shape grammars. With this in mind we tested the system in a real architectural commission for a multi unit housing building in Thessaloniki Greece. The building (Figure 4) houses 5 family apartments

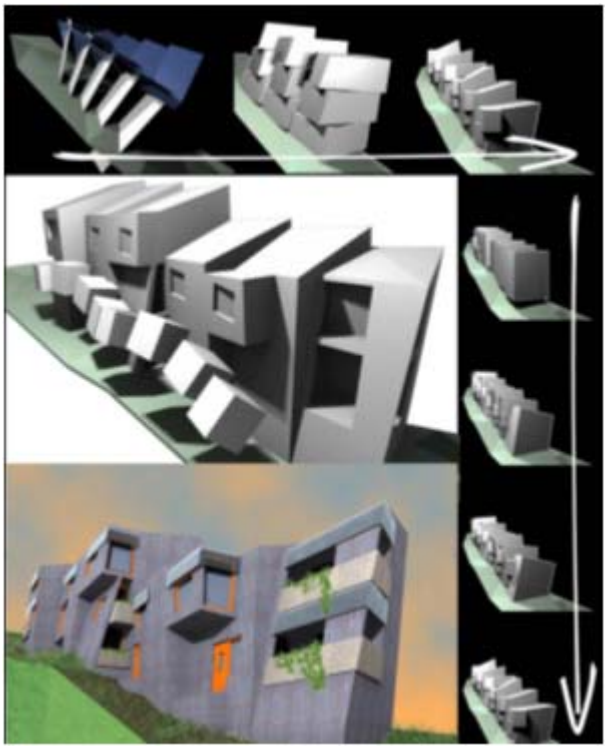

Figure 4: Building designed with dynamic Shape grammars

From 70 to 120 m2. The main axis of the brief for the building was the encapsulation of environment friendly and sustainable technologies like photovoltaic panels on the roofs, so as to take advantage of the luminous climate of Greece. Installing Photovoltaic panels on the roofs and/or walls of the building meant that some of these building elements had to be slopped. We decided to develop the building around this core idea of a slopped building

element, whether that is a roof or an external wall. In figure 4 the initial shape (top left corner) and consequential shapes developed are shown.

Initially we formed a battery of a core element comprised of slopped roofs and walls. This was realized inside blender 3d as an array of the core element following a curved path while the path was following the slope of the site. This array was the initial core shape. By varying the initial core shape (local rule) and the slope, curvature and control points of the two curves (global rule) we developed variations of the initial shape. The next step was to develop a building envelope comprised from the volumes defined in the building brief (the 5 apartments). A simple parallelepiped was used as volume for the apartments. An array inside the software allowed for copying the typical apartment creating floors. Trying to find a slopping wall inside the typical parallelepiped we started twisting its geometry arriving at the top middle shape shown in figure 6. Combining the first and second idea we arrived at a third grammar that incorporated the slopping walls and roofs from the first grammar arranged in a typical apartment module (second idea) spanning two floors. This third grammar was further developed by introducing a second parallel grammar: Small parallelepipeds spanning the length of the building elevated on the first floor were put in an array on the external envelope of the building, creating extrusions that resemble the configurations of the local traditional Hayat houses. With this parallel grammar the top right shape in figure 5 emerged. This combination of grammars was further developed and evolved by graphically varying the arrays and curves that created it (right column, Figure 5).

The example with the multiunit housing building in Thessaloniki shows some of the advantages of our implementation. The designer expresses her idea graphically, both in terms of form and structure, while developing shape grammars. Formal knowledge concerning the exact mechanisms of shape grammars is not required and the emergence that arises in some grammars can be manipulated by the designer in a creative manner by assigning to emergent shapes a structured animation rule.

Another aspect of the system is that it can provide the designer with alternatives to an initial core design idea: since animation incorporates a time element every instance of the shape grammar per time can be considered an alternative to the initial core idea. The difference between alternatives can be quantified by plotting the relationship between the spatial and temporal distance inside the software. Again here the classification algebra mentioned before is needed to facilitate the assessment of the alternatives both as substitutes to the original and/ or as designs classified in their own grammar group.

6. Transparent, informal, efficient.

While not proposing a complete breakup from the academic paradigm of shape grammars our implementation is transparent enough to the practising designer so as to be able to use it even without any knowledge that she is dealing with shape grammars. This transparency coupled with the lack of any formality in the declaration of the rules enable a designer to develop her designs and ideas in the way she prefers, benefiting from the lack of restrictions on how complex rules can be formed and applied. On the other hand a question of efficiency arises. While it can be shown that the system is very efficient with developing various design ideas, it has not been tested in more conventional situations of simple buildings. This question of efficiency touches also the issue of the algebra that governs our implementation. While the system is intuitive and complete without an algebra that classifies the designs, a development of such algebra would allow for exactness in shaping the rules, and their combination, and also would facilitate the expansion of our implementation by other researchers. Our efforts towards shaping such algebra are still continuing, hoping to extend our system even more in the future.

References

Carlos Roberto Barrios Hernandez: 2006, Symmetry, Rules and Recursion, How to design like Santiago Calatrava, Department of Architecture, Massachusetts Institute of Technology. USA.2006 shape grammars - eCAADe 23

Chase C. Scott: 1996, Using logic to specify shapes and spatial relations in design grammars, Workshop Notes, Grammatical Design, Fourth International Conference on Artificial Intelligence in Design, Stanford University, USA, 22 June 1996, Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-0001

Chase Scott and Ahmad Sumbul: 2005, Grammar Transformations: Using Composite

Grammars to Understand Hybridity in Design, With an Example from Medieval Islamic Courtyard Buildings, Department of Architecture, University of Strathclyde, Glasgow, UK, CAAD futures 2005 proceedings publication.

Dounas T, Kotsiopoulos M.A:2007, Generative Systems based on animation tools: structure and form of core ideas in architectural design, CAADRIA 2007 [Proceedings of the 12th International Conference on Computer Aided Architectural Design Research in Asia] Nanjing (China) 19-21 April 2007, pp. 433-440

Duarte J.P:2001,Customizing mass housing: a discursive grammar for Siza's Malagueira houses, PHD dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Mass

Economou A: 2001, Four Algebraic Structures In Design, Georgia Institute of Technology, USA, 2001: ACADIA

Knight, T: 1991, Designing with Grammars, CAAD futures Digital Proceedings.

Knight T: 2003, "Computing with Emergence" *Environment and Planning B: Planning and Design* **30**(1) 125 – 155

Penras D, Duarte J.P and Branco F :2007, A system for providing customised housing Intergrating design and construction in a design tool, A Dong, A Vande Moere & JS Gero (eds), CAADFutures'07, 153-166.