Dynamic Algebras and Grammars

Dounas Theodoros  
Aristoteleio University of Thessaloniki  
http://arlav.blogspot.com  
dounas@gmail.com

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. The designer just applies the rules in the initial design or has to chose 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 structural 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.

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 produce 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 practise 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
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. This situation does not follow with design as an intuitive practise, even though it can be shown that architects use shape grammars in their practise, by formulating, applying and testing rules concurrently with the development of their design.

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 chose 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 an off-the-self animation system is the fact that the shape grammars are expressed directly in three dimensions allowing the designer to explore alternatives in complete form.

Actual Implementation

The design begins with expressing the structure of the design idea using animation tools (Dounas 2007) specifically ‘animation keyframes’, ‘parent and child relationships’ between forms and ‘path animations’ of forms and finally lattice mesh deformations. The structure of the design idea is of course 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](image)

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(figure 2)
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)

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 3).

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 that they appear to be the same grammar where the first initial shape or shape configuration ‘A’ changes dynamically or is substituted in every step.

**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 modelling, animation, rendering, etc suite), the designer formulates a design idea early on in terms of design structure and does not manipulate rules directly but 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. The animation tools can be categorised in two groups one that expresses structure and one that modifies 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. (table 2)

The manipulation of ‘When’ (ie 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 4). 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 tools 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. Animation tools 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. 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 an 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.

**Implementation Example**

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. We tested our system in a design project in an architectural competition. The competition asked for an innovative mixed use development, in a site in Walsall UK. The site is a triangular wedge bounded by water and facing a wide street, which provides the basic lines for the development of the architectural concept presented here. The initial grammar we chose to implement was that of a box rotating from the street towards the tip of the wedge while simultaneously elevating towards the sky. (figure 5)

<table>
<thead>
<tr>
<th>Structure of Architectural Idea</th>
<th>Animation Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear development</td>
<td>follow path</td>
</tr>
<tr>
<td>scale / proportions</td>
<td>parent child relationship</td>
</tr>
<tr>
<td>connected shapes</td>
<td>armatures</td>
</tr>
</tbody>
</table>

This simple grammar produces the single envelope for the building where the volumes of the building are translated and rotated according to the grammar. The final envelope consists of four volumes housing the parking spaces, shopping centre, a conference and entertainment centre and a hotel. The conference and hotel volumes are cantilevered at +9 and +18 meters above ground respectively. This posed the problem of how to formulate the structural
system to support the cantilevered volumes. This was solved by creating a system of lattices that would create a monolithic structural system supporting the building. The lattice a very simple shape grammar that was next transformed using four array modifiers, rotation, translation and a parent and child relationship applied to a single structural beam. (figure 6) The final result of the design showing the instances of the two animated grammars is shown in figure 6.

The example with the mixed use building in Walsall 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 are 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.

**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 questions 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 an 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 an algebra are still continuing, hoping to extend our system even more in the future.

**References**


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

Chase, S.: 1996, Using logic to specify shapes and spatial relations in design grammars, Workshop Notes, Grammatical Design, 4th International Conference on Artificial Intelligence in Design, 22 June 1996, Stanford University, USA, Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-0001.


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

