Animation as a Framework for Generative Design

Theodoros Dounas
Aristoteleio University of Thessaloniki
www.chorodomi.gr
dounas@gmail.com

Abstract: The paper presents a framework for parametric and generative design based on shape grammars, implemented inside a 3d animation tool. A simple description is given on how animation works, along with parity features between shape grammars and animation tools. Work covered in previous papers by the writer, namely how the designer constructs individual tools from simple animation mechanisms is here expanded in a framework of algebras that not only function in geometric dimensions but also in time.

Keywords: Shape grammars; animation; shape algebras; design generation.

Introduction

The paper presents a cohesive and complete framework for the use of animation in parametric and generative design. Previous work based on animating shape grammars has shown that off-the-shelf animation software can be used to develop shape grammars of a design using animation transformation. Animation has been used by the writer and Earl Mark as a generator of alternative designs in practice and in an academic context respectively. The present paper discusses a framework of constructing “animated” models in design that belong to a developed grammar, as a means to capture and represent design intent and decisions. Furthermore the potential of animation as a generative system is established where input is not only geometric but also symbolic in terms of relationships between parts of the design.

Rules of animation and design

Animation as a discipline relates transformation of form over time. This basic premise has driven the internals of contemporary animation software: they provide tools to control change of form, or transformation over time. The element of control is crucial to an animator since it provides the ability to refine their animation to the point where it appears to simulate life, in the eyes of the beholder. Thus animation software, either 2d or 3d, incorporates three basic elements: a time keeping tool, transformation tools and control tools. Control tools usually consist of means to structure transformation and couple it with change of time. This way an object appears to move, a character appears to talk, walk, etc (Figure 1)

These control tools can in effect be used to build structured models of architectural designs. The parametric structure in the designs emerges from the semantic information, which the designer uses to build the models. In the same way BIM software maintains relationships between parts in a model, animation software can capture relationships between parts and can produce alternatives of a given design essentially by “animating” the model, where the time keeping tool records one alternative design for every frame of the animation (Dounas, 2006). The control mechanism presented in the paper allows for a complete control of the design hierarchy in terms of bottom up or top down approaches. The designer/user
of the system builds a structured geometric model of his design, then employs transformation rules on the design by manipulating the structure of the design generating alternatives to the initial “position” of the design. These alternatives can then be evaluated according to specific criteria set by the designer.

**Software used**

We based our framework on Blender, an open source modeler and renderer, which incorporates a robust scripting engine based on the python language. As Blender is open source, access to the source code allows for various tools to be constructed specific to architectural parametric and generative design. Although for most operations described in the animation framework, almost any 3d animation software can be used; Blender has the advantage of allowing direct access to geometry structure and configuration compared to proprietary software like 3d Studio MAX™.

**Animation = shape grammars**

Shape grammar and animation are equal/equivalent in terms of generative power and process because of the way they deal with spatial relations and rules. What animation has in addition to shape grammars is the use of time as a transformation tool. But what it lacks is very formal declaration of the rules, although note-taking of the rules can be established by examining the structured, “rigged” model and its semantics. The context of comparing shape grammars and animation lets us consider the following example of a shape grammar (Figure 2):

In Stiny’s example the designer uses two rules: an addition rule and an erasure rule. In animation software, deleting the shape can happen in a lot of ways: we can actually erase the shape, or we can just hide it from the view of the designer in another layer. In Stiny’s example, the level of ambiguity rises because parts of the shape have been erased and changed the spatial relation of the other parts. From one point of view this ambiguity does not exist in animation when we implement the rule of erasure (subtraction) by hiding the shape, since the designer can retrieve it by making the layer visible. Let’s take a look at another example form Stiny’s book “Shape” (Figure 3).

In the second example, the designer has these choices on how to reproduce the exact shape grammars that Stiny gives as examples in his book (Figure 3). He can do it by addition and subtraction or he can do it by rotation and subtraction. In fact in animation
circumstances. This is nothing new in shape grammars. Animation on the other hand can reveal and capture the in-between states of these two rules and thus create distinct spatial relations, and along with them new shape grammars can be derived from an initial one (Figure 5).

These families of grammars allow for an exploration on a theme or party from the designer’s point of view. This ambiguity of determining the same spatial relations with different rules and/or using the same rules of transformation for determining different spatial relations is easy to exploit in animation tools to produce alternatives to a given design. This ambiguity is the reason, a designer can start with a given grammar and by changing the rules in the same spatial relations can generate more grammars that belong in the same family (Figure 6).

Changing from one grammar to another, in the same family, usually happens in animation tools by reconfiguring the controller objects that structure the animation. The designer can choose to change grammars to explore some of the alternatives of his idea, or structure the grammar in a different way because new, interesting shapes have emerged from the first shape grammar that can answer some of the designs’ requirements.

Specifics of the framework

The animation framework presented here can be analyzed in either a linear process or a feedback loop where the animation as an engine produces alternative designs either at the end of the linear process or at the point the designer chooses during the feedback loop (Figure 7).

The animation tools can be categorized in two groups, one that expresses structure and one that modifies form. Tools that express structure are key-frames (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/transformation 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 common CAD software, but the advantage animation tools provide us temporal distance in the application of the rules (Table 1).

The manipulation of “when” (i.e. in which step of the computation); a rule is accessible through the interpolation and extrapolation curves (depending on whether we need a rule to be applied recursively or constantly).

The designer then can manipulate the duration of the rule, move the curve unmodified and

<table>
<thead>
<tr>
<th>Rule a</th>
<th>Rule b</th>
<th>Rule c</th>
<th>Rule d</th>
<th>Rule e</th>
<th>Rule f</th>
<th>Rule g</th>
<th>Rule h</th>
<th>Rule j</th>
</tr>
</thead>
</table>

---

Figure 6
Families of shape grammars

Figure 7
Function of the framework
the present or future to the past by manipulating the angles of the transformation functions (Figure 8). Essentially, this new algebra $U_{23t}$ would constitute a sub-algebra of the parent algebra $U_{23}$. In this manner shape grammars that are the same in terms of rules but differ in terms of when a rule is applied, can be described with the specific keyframes that record the effect of a rule.

It should be also noted that transformation Rule choice and construction by the designer closely follows the paradigm of shape grammars in the sense that there are no predefined elements used in the computation and also the designer is completely free to build any rule as she sees fit. On the other hand animation as a generative framework diverges from the paradigm of shape grammars in the sense that the designer can structure the design process on non-geometric characteristics of his model. Moreover modern animation software like Blender incorporate physical simulation tools like rigid and soft body collisions, cloth simulators etc. that can greatly expand the design rule vocabulary at the disposal of the designer.

**Evaluation of alternatives**

Previous research has shown that the coupling of a generative or production system (like shape grammars) can be effectively coupled with an evaluation mechanism to enable sorting of the design solution based on specific criteria (Duarte, 2001). Past evaluation mechanisms include space syntax (as an analysis tool), genetic algorithms, or performance based algorithm (irradiation measuring). Our work in the area of evaluation of alternatives is still developing: we are building a generic interface between animation tools and evaluation mechanisms as the above mentioned. As the animation tools can be codified inside algebras that produce consistent results each time a rule is inferred we believe that it is possible to build an almost generic representation schema that can be used as input for evaluation and analysis tools. We are familiar with the difficulties in building

---

*Figure 8*

Insert IPO - function curve – Transformation vs. time function that the designer uses to move one or more transformation rules through time.
such a schema since it is very difficult to construct an adequate design representation that can carry both geometric and non-geometric characteristics, but a forthcoming paper on the matter even with a partial result will benefit the CAAD research community.

For reasons of completeness we present here a fraction of the mechanisms allowing for both the generation of alternatives and their evaluation, based on space syntax. Namely we construct a 3 dimensional model where an animated armature manipulates the transformation of the model in a parent-and-child relationship, according to specific rules. By moving a “bone” in the armature one can transform the model with specific rules, but at the same time the user sees in the form of the armature (which is a form of diagrammatic skeleton) a first image on the integration of spaces inside the model. We are now in the process of evaluating such a mechanism where one can construct the animated grammar, based on the integration or segregation of spaces that the designer chooses.

Possible failure of the framework

Although animation can be a very powerful tool in the hands of a designer that understands it, poor translation of design intent into rules will lead in poor design performance. This can only be resisted through training and education, although we believe that basing our framework on simple and generic animation tool facilitates the use of the system by the designer in a clear and unobtrusive manner.

References


