Generation of alternative designs in architectural problems using Shape Grammars defined with animation tools

A computer implementation of shape grammars using modelling and animation software

Theodoros Dounas, Anastasios M. Kotsiopoulos
Aristoteleio University of Thessaloniki, Greece
arlav.blogspot.com, arch.auth.gr

We present a model of generation of alternative designs to selected architectural and spatial configurations of small complexity. Specifically we present a production pipeline of architectural / spatial configurations using the context of animation and time based design tools. Our model consists of time and space design constraints of boundaries / objects affecting a given architectural design, thus producing an alternative solution for every timeframe of the animation cycle. The alternative designs vary from the original according to their temporal and/or spatial distance from the original object on the animation time-line.

The constraints placed upon the objects , used as actuators of Shape Grammars, are defined informally by the user/designer while their influence can vary according to time, speed, location, configuration of the object and/or the constraint itself. However the constraints further function as formal rules for the Shape Grammar creation so that our model tries to predict ahead of time the emergence of alternate designs.
The employment of animation tools [shape driven curves, speed and time-line functions, parent child relationships] in the shape generation of our model empowers the user/designer to configure whole sets of shapes and designs interactively and without the need to define every solution independently. Simultaneously, a different, time-focused view of our model describes its use on designs that develop different configurations over time. Thus a duality of our model is established: either the animated schema may be a sum or family of various designs or the animated time-line represents a single design which changes over time.

Finally the possibility of an automated analysis of every design is discussed, using Space Syntax diagrams so the designer can quickly evaluate the various spatial configurations produced by a single original.

Keywords: *shape computation; shape grammar computer implementation; alternative designs; animation software techniques*

Shape Computation and its relation to Animation pipelines

In the field of shape computation, a variety of computer programs have been developed: interpreters – generators of shapes from shape grammars, parsers – programs that analyse a given shape and determine if it belongs in a given shape grammar, inference programs – programs that analyse a given family of similar shape configurations and produce a shape grammar that incorporates the designs. Despite the fact that there are many programs dealing with the computation of shape grammars, writing such a program still remains an ability reserved for those people that manage to learn the purely symbolic thinking of computer languages (Gips, 1999). Moreover even if the program gets written despite the barriers that symbolic thinking imposes on visual thinkers, there exists the issue of the interface and steep learning curve since the user splits her attention between learning the program and learning to use shape grammars.

Comparatively, in similar fields of visual thinking, CAD, photo editing and animation programs are commonly available with a standard set of features and expected functionality (Gips, 1999). A user that has mastered one computer program in a category can learn another one with minimal effort, something that in the field of shape grammars and design computation is difficult to happen due to lack of "industry standard programs", common interfaces and functionality. Alternatively, a user could implement a shape computation program as a plugin inside a generic CAD program using scripting languages, implement his shape grammar manually or use visual automation tools provided by the CAD program manufacturer to build the computation. Unfortunately the first case is exactly the problem a visual thinker tries to avoid, the second one is actually a manual implementation employing a computerized shell that just handles drafting and not the computation itself and the third case is purely theoretical since no off-the-self CAD program has any shape computation visual tools.

Animation programs on the other hand do provide a visual mechanism capable of building shape computations disguised as simple or complicated animations (Roosendaal T & Selleri S,2005) A typical task in an animation cycle is that of transforming a shape according to one or more rules, so that in every frame-step of the animation, smaller or bigger changes will create and propagate the effect of movement (Figure 1).

In a complete analogy animation uses shapes and rules of transformation just like computation (Stiny, 2001) but the steps of shape grammars are called frames of a time line in an animation. The shapes change from frame to frame in an animation whether that is an animation of a character or an object such as a building. It is important to emphasize that even though one uses the phrase computer animation one does not imply the use of animation tools as a means of presentation, but the actual mechanism of animation computer programs.

To test this analogy and its effectiveness in architectural design with the use of shape computation, an open source program was chosen, Blender.

Blender is open source software for 3D modeling, animation, rendering, post-production, interactive creation and playback used worldwide with great success (www.blender3d.org). Blender was

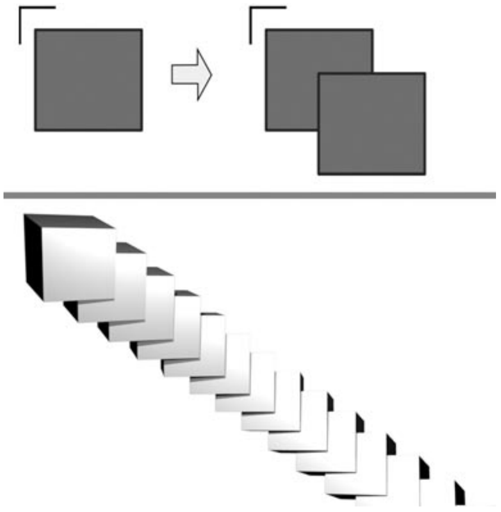

Figure 1
Comparison of shape grammars (top) and animations of shapes (bottom)

chosen not only because of its current capabilities and codebase but also because its code is open, has a very small footprint, is available for many operating systems, from Windows and virtually every flavour of Unix/Linux. Also on top of its modelling and animation capabilities, it can be extended with scripting via the Python scripting language and has a versatile variety of scripts at its disposal.

Simple generic example of an animated shape computation.

A house of small complexity in the traditional area of Ano Poli in Thessaloniki (Figure 2) was chosen as a typical case study. For simplification focus was given only on the external envelope of the house. A simple model was made according to the original design of the house , with simple primitives representing the volumes of the external shell. The main envelope of the house was represented with a cube, from which the voids forming the hayat were subtracted using standard boolean operations. Any protruding elements like balconies were formed by slabs or by parallelepipeds Then a simple animation was setup, consisting of 30 steps or frames. Key frames were marked at the beginning, frame 01, and at every 10 frames after that, frame 10, 20, 30. At every key frame the void-hayat, slab and parallelepipeds move from side to side, changing sides on the shell at every key frame (Figure 3).

The program (Blender) then produces the in-between frames. The result is the development of a simple shape grammar from an original design in 30 steps. Although the designer knows before hand the result on frame 30, the in between steps of the shape grammar cannot be guessed and are in fact emergent (Knight 2003). The constraints imposed on the simple elements of the model are in fact known before the actual animation, so there is no question of prediction in emergence of shapes but a tighter control on part of the designer on the formulation of the "animated" shape grammar. The mechanism of modelling shape grammars using animation techniques enables the designer the desired interactivity with the computer program and the ability to stage some steps of the computation with tighter than usual control over the outcome. Moreover every step of the computation can be made "real" and not just a part of an animation or computation so that the designer can apply new rules in a step of the computation that has the desired traits (Figure 4).

Using animation software , Blender in this case, does not restrict one only in simple shape grammars but also in parametric computations or genetic algorithms computations . The mechanisms of animation that have suit-

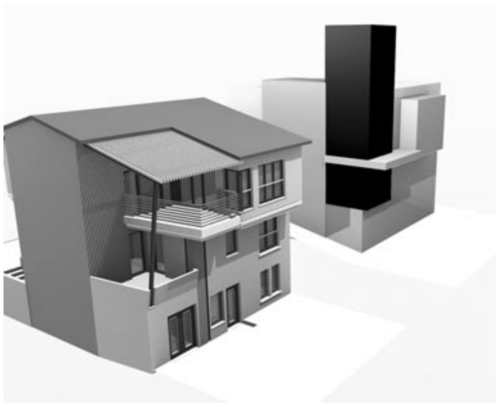

Figure 2
Residence in Ano Poli Thessaloniki, and simplified animation model.

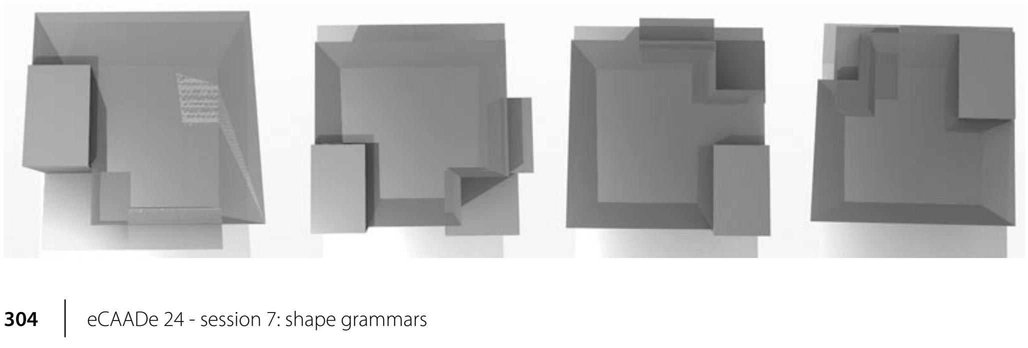

Figure 3
Key frames 1,10,20,30 from left to right.

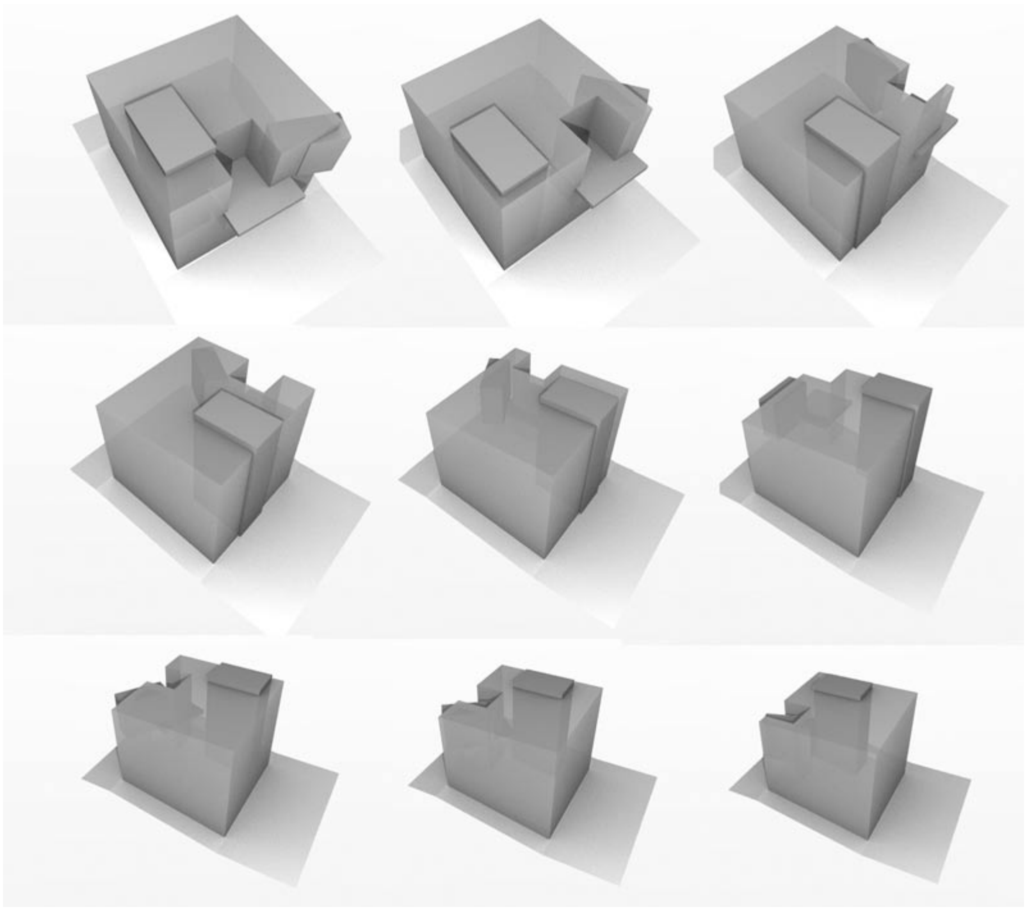

Figure 4
In between steps/frames of computation

ability for use in design computation are CAD-like modifiers, Boolean Operations, Shape constraints and drivers, Curve and Object driven animations (Figure 5).

Coupling of Shape Grammars in "Shape Animation" with analytical tools: Space Syntax

The technical modelling of shape computations analysed here can quickly overwhelm the designer with the amount of alternative designs, because the frames of an animation can quickly escalate in high numbers, without providing for a mechanism to distinguish between frame with small changes between them. To overcome this problem we intend to extend the capabilities of Blender with scripting in Python using Space Syntax techniques. These techniques will allow the designer to investigate a large amount of computation data of multiple shape grammars happening simultaneously , with the added benefit of an analysis tool at hand. We believe that although Shape grammars are a versatile tool for the creation of alternative designs, they could be coupled with an analysis tool. The combination of a shape grammar with an analysis tool like space syntax would provide the

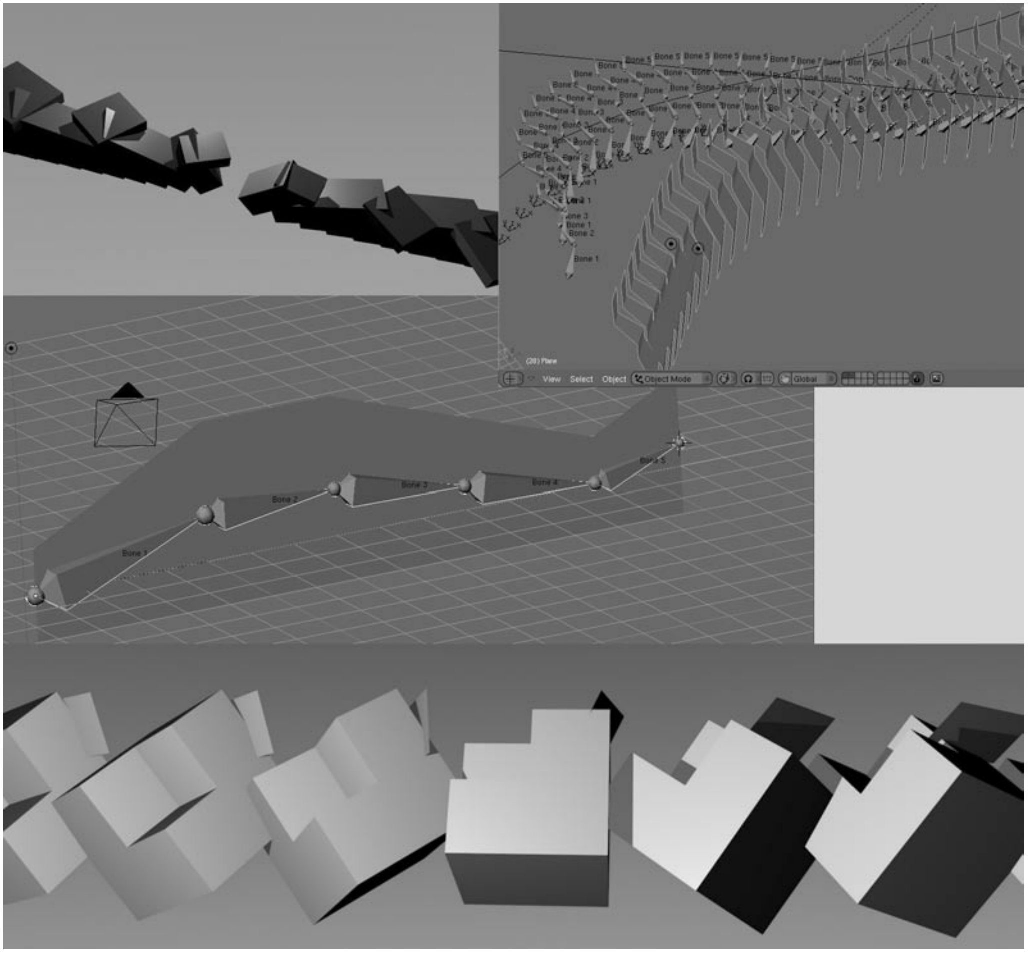

Figure 5
Mechanisms of animation, suitable for design computation.

possibility of producing alternate designs that conform to specifications set by the designer .

Acknowledgements

This research is supported and funded by the General Secretariat of Research and Technology, Ministry of Development, Greece and the Aristoteleio University of Thessaloniki, Greece through the research program "Analysis and Synthesis of complex buildings with the use of Shape Grammars and Space Syntax". PENNED 2003. Theodore Dounas is supported and funded by the General Secretariat of Research and Technology, Ministry of Development, Greece

References

Roosendaal, T. & Selleri S: 2005 ,The Official Blender 2.3 Guide: Free 3D Creation Suite for Modeling, Animation, and Rendering, No Starch Press, 3d edition

Gips, J.: 1999, "Computer implementations of shape grammars" , NSF/MIT workshop on shape computation 1999

Knight, T.: 2003, "Computing with Emergence", Environment and Planning B: Planning and Design 2003, Volume 30, pp. 125-155.

Stiny, G.: 2001 "How to calculate with shapes", Formal Engineering Design Synthesis, Eds E K Antonsson, J Cagan (Cambridge university Press, New York) pp. 20-64.