Generative Systems Based on Animation Tools: Shaping Alternatives to Structure and Form in Architectural Design

Dounas Theodore
Aristotelio University of Thessaloniki, Greece
http://arlav.blogspot.com
dounas@gmail.com

Our system is based on computer animation tools, employed in the context of architectural synthesis. The pipeline of the tool consists of “time” and space design constraints of boundaries / objects affecting a given architectural design, with an output of alternative designs. The alternative designs vary from the original according to their temporal and/or spatial distance from the original object on the animation time-line. Specific tools [shape driven curves, speed and time-line functions, parent child relationships, boolean operations, arrays, lattices etc.] empower the designer with the ability to transform his original idea in almost every way the designer likes in a completely visual, interactive and straightforward manner. The solutions sets change according to time, speed, location, configuration of the objects and/or the constraints and rules the designer/user configures.

Previous work on our tool stemmed from the rule based structure of shape grammars, but has evolved to a tool where the designer does not have to define a rule set beforehand but does so as his idea evolves during design. The animation tools used are form independent (that is the designer can use any two- or three-dimensional form) and can represent structural decisions in a given design. The independence from form makes the tool suitable for almost every kind of architectural design, be it experimental, real or conventional while the ability to represent structural decisions and relations of parts in a design implicitly help the designer/user acquire a clear picture of each design and solution set.

Finally the possibility of a structured graph representing each solution is discussed, where the designer can evaluate the merit of an individual solution in terms of conforming to the initial core idea or where alternative spatial configurations evolve in a different structure from the original design.

Keywords: Generative system; animation; geometry.
1. Generative systems in architectural design, programming code versus visual setup

Although commercial computer software has maximised the representation potential of architectural design and has increased the quality of architectural representation, graphic or other, architectural design software still has a basic limitation: It still confined in the logistics of handling representation and 1 to 1 scale modelling for the architect. It does not have a generic systemic approach parallel to the way architects think about space, or manipulate form and variance thereof. Even though academic paradigms of computing in architecture have produced many implementations and prototypes of computer software capable of extending the mental toolset of the architect, there are very few that can encompass the infinite possibilities of design.

In effect the basis for this research is the notion that generative systems in architecture should not be hard to manipulate and interface with [Gibs 1999]. Most of generation systems presented at related literature [Caldas,Rocha 2001] are based on hard coded functions or similar: usually the designer-user of the system has to have a basic understanding in programming code when the most usual training of a designer is visual. The present paper proposes a visual system for generating alternative forms from initial core ideas or design in architecture, where the designer does not write any code, but interfaces with the system in a visual way. The proposed system is based on animation tools for generating alternative designs from a given initial design “solution” which is provided by the designer [Dounas 2006]. Also it should be noted that the system described in the paper has a rule based approach that is based on shape grammars but deviates from them on the level of grammar implementation.

2. Animation as a generative system

Animation tools in form of computer programs are now a standard in the architecture industry, providing with presentation capabilities which were unheard of in the previous decades. Their basic mechanism is the production of the illusion of “motion” between specific frames. For example when animating the walk cycle of a man the user of the program sets a couple of frames as a start and an end to the walk cycle and the animation program produces the in-between frames. The walk cycle of a human has specific rules to follow in animation so that the illusion of motion is sustained. Animation programs are thus structured in a way to follow rules that are constructed by the designer during the setup of an animation.

A generic tool, capable of producing complexity

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.

3. The animation program Blender

The program chosen for our generative system is Blender. Blender is an open source 3d modelling, rendering and animation program with similar or better capabilities to commercial programs. Blender was chosen instead of any other commercial program because it is open source and cross platform, has a fast pace of development and extensive scripting capability and finally has a dual mode of object/edit handling where the user edits the geometry of
an object in a differently from object transformations [Roosendaal T, Selleri S 2005]. The main blender tools used in the generative system are presented bellow. First the tools that are used in structuring the architectural idea are presented where the designer formulates relationships between objects in a model and then the tools that are used in geometric transformations are analyzed.

**Key frames**

For the system to work the user makes extensive use of key frames. Before setting up any rule the user sets the first frame as a key frame. Each consecutive change of the rules or the animation model is marked with a key frame up on a timeline.

**Parenting – parent/child relationships.**

The user can setup some objects as childs of other objects called parents. Every child object receives the transformations of its parent but the transformations of the child object do not elevate to the parent object.

**Paths**

The most generic use of a path is that of an object following a path. In our system a “follow path” modifier commonly helps distinguish individual solutions by forcing a design solution to move along a path during an animation. At the end of the process the designer can materialize his alternative designs as single instances of the animated objects on the path. Thus all of the alternative solutions are presented side by side for comparison. Another use of a path modifier is that where the path object is used as an actuator either for an array or a parent-child object assembly like beads on a string. Since the path can be a curve or a straight line, it can be used for the production of alternative designs of curved tiled roofs or series of openings on a building.

**Arrays**

Arrays are n-number copies of an initial object where the transformations of the object apply recursively n times on the copies. The transformations can be manipulated either by using transforming the initial object, or by transforming a second controller object, an actuator. With actuators the problem of managing the complexity of n-arrays becomes a simple task confined to handling only a small number of objects compared to handling the full array (figure 1).

**Lattices**

Lattices are actuator objects which are used to deform or transform another object locally, in a parent-child relationship. An example of this is when the designer modifies the envelope of a building locally: A roof that gains more height while its base remains unmodified etc. Again the transformation through a lattice can be animated, producing various alternatives to the current design (figure 2).

**Dupliverts / Dupliframes**

Dupliverts and Dupliframes are a tool unique to Blender. Dupliverts produce copies of an object for every vertice of another object while dupliframes produce copies of an object for every frame of animation that the object participates in. For example
in a walk cycle of a simple human figure the designer using dupliframes would instantly see all the steps of the animation simultaneously on the program’s 3d screen. Used in architectural design the user using dupliframes would be able to see all of his alternative designs simultaneously (figure 3).

**Boolean operations**

Boolean operations are known to architects from generic 3d cad systems. They allow the union, subtraction and intersection of forms. Combined with animation, the designer can produce alternatives based on the relative position of each object participating in the Boolean operation. For example a window can be produced by subtracting a cube from the shell of a building. By animating its size and position the user can evaluate better alternatives interactively.

4. **Case Studies**

Although our tool is a general and generic one where the designer decides what techniques to use depending on the nature of the design task, case studies have been chosen as examples. The criteria for choosing the following examples are the clear core ideas behind each design and the simplicity of the idea relatively to the rich complex design. The examples serve more as guide for the usage of the tools than as a specific paradigm.

**Tods Gallery at Omotesando, Tokyo, by Toyo Ito**

This building for Tods’ gallery in Omotesando Tokyo (figure 4) was designed by Toyo Ito of Japan. It has a generic L shape plan with a characteristic lattice facade resembling tree branches. In fact Toyo Ito has used the image of interlocking tree branches to design the building’s façade. A strip made out of the same tree repeated endlessly is wrapped around the L shape of the plan and thus the building is produced. Although Toyo Ito does not claim anything about implementing his idea in an algorithmic method, the core idea that produces the building is in fact an algorithm.

Using Arrays, we first produced a simple generic Tree branch (figure 5).
With another array we produced the multiple copy strip that will become the envelope of the building. Then using actuator objects and key frames we produced alternative designs to the initial facade: in a first instance we used a symmetrical array where each frame consisted of an alternative symmetrical façade (figure 6).

In the second instance we removed the symmetry so that in each frame/alternative design we produced asymmetrical facades, closer to the original design by Toyo Ito. The next step was to wrap every alternative in a L-shaped path by using a follow-path modifier, thus implementing a parametric model of all the alternative instances of the building.

**The box at Culver City, California, by Eric Owen Moss**

The cube is a meeting room in an old factory building redesigned by Eric Owen Moss of California (figure 7).

The core idea of Moss is a cube rolling on a sphere. In the design the sphere is actually a void where only two of its meridians are kept as structural support for the box/cube. Our implementation of Moss’ idea is literally a cube rolling on a sphere: since is no tool to choose paths on spherical objects, the sphere is replaced by to circular paths positioned either perpendicular or at an angle (figure 8).

This construction of a spherical path from two circular ones creates an infinite amount of positions for the cube on top of the sphere. By changing the angle between the paths and the speed of the cube the designer interactively produces and evaluates alternative designs of Eric Owen Moss initial core idea.
5. Coupling with space syntax as an evaluation tool

The technical modelling of shape computations analysed here can quickly overwhelm the designer with the amount of alternative designs. 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 evaluate a large amount of computation data of multiple animation grammars happening simultaneously, with the added benefit of an analysis tool at hand. We believe that although 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 possibility of producing alternate designs that conform to specifications set by the designer. The techniques that are now being developed are an automatic graph generator which will produce simple three dimensional graphs for each alternative design, and also a visibility analysis tool. These techniques are borrowed from the framework of space syntax analysis and although space syntax has been formulated initially an urban configuration tool its use for analysing building designs and configurations has a proven record. [Duarte 2001]. Our theory is that it is important for the designer to be able to understand which part of the design in a specific frame is the integration core of his building [Duarte 2001]. As the integration core is the most important part of a building in its spatial structure, the designer would be able to modify the rules of his animation so that specific criteria can be met.

Acknowledgements

Theodore Dounas would like to acknowledge the invaluable contribution of professor Anastasios M. Kotsiopoulos in the research presented in this paper. His ideas and deep knowledge of architectural design and theory have acted as a guiding light through every difficult stage.

References
