MoSS: Morphogenetic Surface Structure A Software Tool for Design Exploration

Peter Testa, Department of Architecture, MIT
Una-May O'Reilly, Artificial Intelligence Laboratory, MIT
Markus Kangas, Department of Electrical Engineering, Computer Science, MIT
Axel Kilian, Department of Architecture, MIT

ABSTRACT

We report our progress in designing a modeling software tool we have called MoSS: Morphogenetic Surface Structure. Moss is based on Lindenmayer systems in three dimensions which "grow" surfaces by applying re-write rules to an axiom with an accompanying interpretation of movement and drawing in space. It is written in C++ as a plug-in to Alias|Wavefront Studio. The tool, still rudimentary, allows an environment to be defined to influence surface growth by means of boundaries and objects that attract or repel direction of growth. It optionally applies different joining filters to surfaces that become disjoint as a result of encountering environmental factors.

1.0 INTRODUCTION

Modern CAD tools such as AutoCAD or Alias|Wavefront Studio allow architects to hand-draw three dimensional surfaces and volumes. However, this capability falls short of software that actively enables designs with deliberate precision or that demonstrates adaptiveness to site conditions. We have also noted that architects frequently take inspiration from natural forms and the process of organic growth. As a response to these initial issues, we have designed a tool called MoSS: Morphogenetic Surface Structure. The initial goal of the MoSS tool project is to provide architects with a model of surface growth in three dimensions that can be simply, yet precisely, defined and understood. In addition, our goal is that as growth is influenced by tunable factors an architect can use MoSS to model surface geometry within a three dimensional shaping environment. With a rudimentary version of such a tool in place, we have two further intentions that we can report have been fulfilled. First, we have encouraged architecture students to use MoSS so we can learn how its users engage it and what they find useful or limiting. We report on this activity in this paper. Second, we have always foreseen the need for MoSS to fit into a larger design process; one that includes the construction of physical models and structural elements using computer aided manufacturing processes (CAD/CAM). In this report we will present examples of such steps in the process. The culminations are descriptions and images of physical surface constructions that were first generated using MoSS, unfolded using CAD/CAM software into two dimensional format for laser cutting, and assembled to form three dimensional counterpart physical models.

The paper proceeds as follows: Immediately, in Section 2.0, we will briefly discuss work related to MoSS. In Section 3.0 we will describe the computational basis of MoSS. That is, we will explain how Lindenmayer systems (L-systems) work and how we used the principles of L-systems to implement a surface modeler inside a CAD tool. We then describe features of MoSS that are particular to its context of use. In Section 4.0 we present a number of examples of MoSS. Each example can be simply defined in terms of its grammar and axiom plus the state of the environment in terms of boundary, attractors or repellors. However, a complex design results from re-writing the axiom and interpreting it as drawing instructions. The MoSS derived design will be shown as a computer generated image. For some examples, we will also show the physical model resulting from the MoSS design. Our current appraisal of the project will comprise Section 5.0. We include observations about MoSS in its current prototypical state and our plans on how to extend it in the future.

2.0 RELATED WORK

A significant motivation with MoSS is to model the development of pattern, form, and surface structures in the domain of architecture. In this context a number of computational approaches to modeling morphogenesis are compelling to study. One approach is cellular automata (CA) models [1]. CAs can be uni-dimensional [2], two dimensional or even higher dimensional. A CA is a spatial system consisting of units (cells) that have state. As time in the model steps forward, one or more cells update their states based on the states of neighboring cells and, optionally, random probability. From typically simple and local rules of interaction, the global state of the entire cell array emerges. Analysis of the local rules typically does not yield a description of the global state. When CAs are visualized states are represented by colors. This highlights patterns representing emergent interactions. CAs have been defined that are accurate models of actual physical systems which grow into clusters or distinct aggregations [3]. They also can model physical processes [4]. The argument against using CAs for surface modeling is their need for a definition of spatial extent and the way in which growth is defined (i.e. local rules of interaction).

Another approach is reaction-diffusion models [5]. Such a model is defined by coupled equations that incorporate concentrations of chemicals, non-linear reaction terms and diffusion constants. Under the basic principles of local activation and long range inhibition, concentrations of the chemicals vary spatially and result in patterns. The mathematical and chemistry oriented definitions of reaction-diffusions models imply it is difficult to generate a desired pattern (or character of pattern). Thus, while potentially interesting as a means of directing architectural surface morphology, reaction-diffusion models lack a means of straight-forward, clear interpretation that make them readily applicable to our domain of surface modeling.

Agent-based models of pattern formation are yet another approach [6]. Such a model typically consists of a bounded environment with resources and behavioral entities or processes, termed agents. Within the environment, time steps forward and agents navigate the space under behavioral specifications that take into account things such as: the need to collect resources, the need to stay away from malevolent environmental features and how to respond to other agents. Agents can both be "born" and "die". The system can be studied dynamically or at an end-point in time (e.g. when all resources are gone). Agents and resources are color coded in visualizations and their dynamics result in patterns. We have used Artificial Life inspired agent-based computation in another project where we focus on the identification of elements in a design and the need to understand how they interact. This seems a strength of agent-based models that is orthogonal to the goals of surface modeling.

Perhaps the approach that is most successful at achieving morphogenetic outcomes that are faithful to biological reality is that of Lindenmayer systems [7,8]. L-Systems are capable of simulating the development of specific plants such as *Mycelis muralis* or *Hieracium umbellatum*, the hawkweed flower [9]. They originally specified a topology of branching structures but subsequently were updated with geometric interpretations that produce visualizations of developmental outcome. As shall be explained in greater detail in Section 3.0, growth in an L-system is controlled by a context-free or context sensitive grammar that essentially is a set of re-write rules that are serially applied to an initial axiom. The re-write rules can be interpreted geometrically. In three dimensional Cartesian space this geometry can be thought of as "agent" based. That is, an agent moves through space with a position vector and three orthogonal axis vectors. Its momentary orientation is determined on the basis of the re-write rule that might, for example, direct it to move ahead, upwards or to the side by a certain degree.

There are elegant implementations of L-systems in both 2D and 3D available at this time, for example, Jacob [10]. One that is particularly relevant and related to MoSS is that of Coates et al [11]. Rather than manipulate grammars to develop surfaces, the Coates et al system develops branched structures in an isospatial grid by means of filling traversed points with spheres. This emphasizes a branching structure in which branches have 3D extent. A goal of the Coates et al system is to study how function can direct form. The choice of an isospatial grid (in contrast to the 3D Cartesian grid of

MoSS) wherein a point has 12 neighbors defined by a dodecahedron with equal point to point distances thus is deliberate since its lack of orthogonal bias and homogeneity allow for complete freedom in choosing form. Coates et al's system searches for functional form using genetic programming [12]. At present, MoSS does not have a search component. It relies on its user to direct form by changing its inputs. Following the Coates et al lead, we intend to outfit MoSS with a genetic programming search component. The choice of objective function criteria for MoSS will differ, however, in a manner suitable to its potential intended purposes of aesthetic or load carrying satisfaction. Coates et al's system has a 'design world' that has a directional supply of nutrients or pathogens. The role of these elements is to be either avoided or caught by the 3D structures. Thus, the structures can be evaluated in terms of how well they do this. Clearly, the distinctly different goals of MoSS and Coates et al have resulted in systems that yield distinctly different designs.

3.0 A DESCRIPTION OF MoSS

MoSS is a plug-in to Alias|Wavefront Studio written in C++. It incorporates a specialized implementation of 3D L-systems and interfaces with A|W to draw its forms. The plug-in aspect implies MoSS is integrated with an existing CAD tool. This is compelling to its users as it provides them with the ability to use Alias|Wavefront capabilities on a completed MoSS design such as analysis, rendering, exporting or further modification. It is a feature for MoSS from an implementation point of view as it obviated the need to develop software that handles displaying and drawing. It also eliminated any necessity to add features not focused on the concept of L-systems.

The software that implements the 3D L-systems consists of three components:

1. Re-write rule application with its accompanying geometric drawing interpretation.
2. The MoSS growth environment (i.e. boundaries, attractors and repellors)
3. Filters for repairing surfaces that become disjoint as a result of attractor or repellor interaction.

3.1 Re-Write Application and Surface Drawing:

L-systems use re-write rules. These rules are used to construct a complex object starting from a simple one. The definition of the simple object, such as a text string, is progressively refined by replacing its parts, or substrings, with strings defined by the re-write rules. Often the re-write rules and simple object are referred to as 'production rules and an axiom" or "generators and an initiator'. The set of text strings or productions that result from applying all re-write rules are termed a language. An example is:

W = a	The axiom
P1: a –> ab	Production 1
P2: b –> ba	Production 2

Language:
Generation 0: a
Generation 1: ab
Generation 2: abba
Generation 3- abbabaab
Generation 4: abbabaabbaababba

In L-systems, one starts with a text string axiom and re-write rules. Designated symbols in the axiom as it is progressively re-written are interpreted as geometric movements to move an "agent" in 2 or 3 dimensional space. In 2D space, one symbol, typically F, means move forward plus draw a line segment and 2 other symbols, typically R or L mean rotate on the 2D plane left or right by alpha degrees where alpha is a parameter of the system.

Assuming alpha is 90 degrees, with the grammar:

W –>L
L –> LLR
R –> R

In Generation 4 the rewritten axiom is: LLRLLRRLLRLLRRRLLRLLRRLLRLLRRRR. Replacing every L with F- and every R with F+, the left hand image in Figure 1 is obtained. When the axiom is rewritten for 10 generations the well known dragon curve is recognizable (see right hand image of Figure 1).

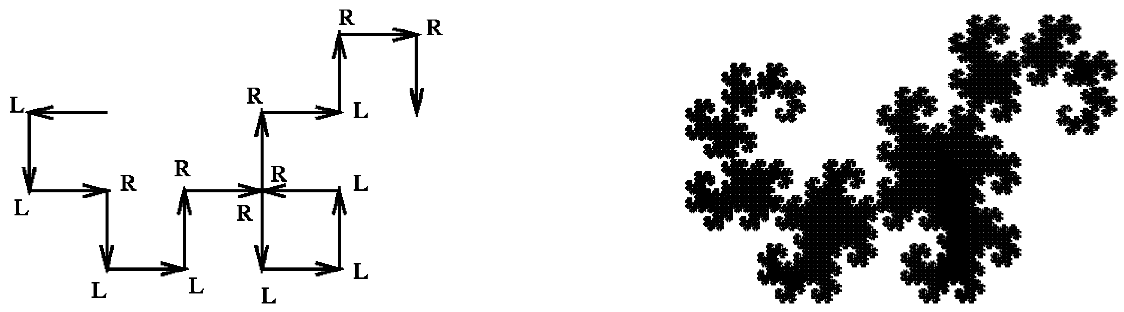


Figure 1: Generations 4 (left) and 10 (right) of the dragon curve grammar.

In MoSS, which works in 3D space, we define alpha, beta and gamma as three angular degrees for specifying movement in the roll, pitch and yaw (X, Y, Z axes) directions. The symbols + and - indicate the agent is to roll the alpha specified degrees in either the positive or negative direction along the axis of movement. Likewise the symbols ^ and & indicate positive and negative pitch rotation and the symbols < and > indicate positive and negative yaw rotation. Movement forward in MoSS *without drawing a line segment,* is denoted by the symbol f in addition to the typical use of the symbol F for drawing a line segment. As in all L-systems, MoSS interprets the symbols [and]. These symbols [and] represent the action of pushing and popping the agent state in space. Thus the agent's position can be 'memorized', it can perform movements from that point and then revert back to that point to perform other movements.

In MoSS we focus on grammars that define surfaces. For example, given alpha=beta= gamma=90^{o}, a square is defined by:

S–> F>F>F>F

And, a cube is defined by making six calls to S to form each face followed by a command to return the agent to its original position, then a bracketed section to form the top of the box and, finally, placement of the box bottom:

C–> Sf^Sf^Sf^Sf^[<^S]f>f^S

A 2X2X2 block of cubes is created with two additional productions that, first, create 4 cubes in a 2X2 square, then, second, stack two 2X2 squares on top of each other:

B2–> CfC>fCf>fC
B–> B2^f&B2

Replication or infinite sequences are defined by recursive productions or recursively coupled productions. For example, an infinite series of cubes and blocks can be created by:

W–> CfV
V–> BffW

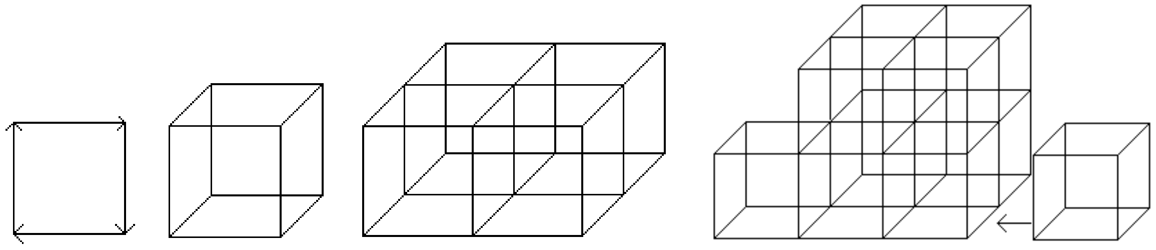
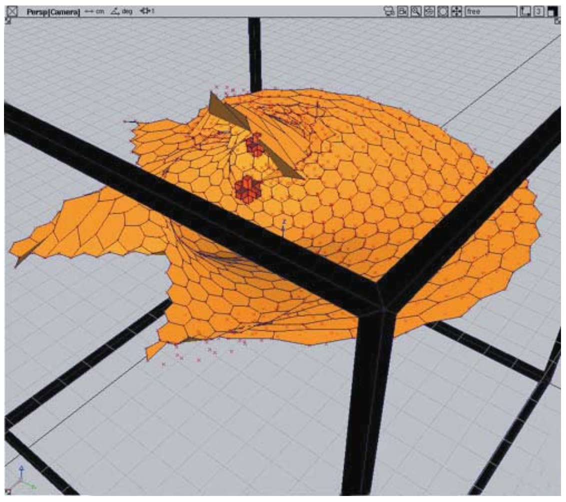


Figure 2: Square surface, cube, block of cubes and alternating set of cubes and blocks.

3.2 The Environment

In MoSS the environment dictates the final geometry of a surface or structure. Two parameters: resource level or max-generations, determine how large the outcome will be by either limiting the number of line segments drawn or the number of times the axiom is re-written. The resource level is a simple way to model the cost and complexity of material in the real world. The user can potentially explore the limits of resources as they relate to constraining design while still experimenting with different designs via production rule changes. Setting the maximum number of generations limits growth from the starting point radially. This proves useful when spatial distance from the origin is important.

The environment includes a rectangular bounding volume to force growth (i.e. movement and drawing) within its limits. The line segment that would intersect a plane of the boundary non-orthogonally is shifted to a legal position that allows its length to remain unchanged. An orthogonal intersection results in a random new direction being chosen.



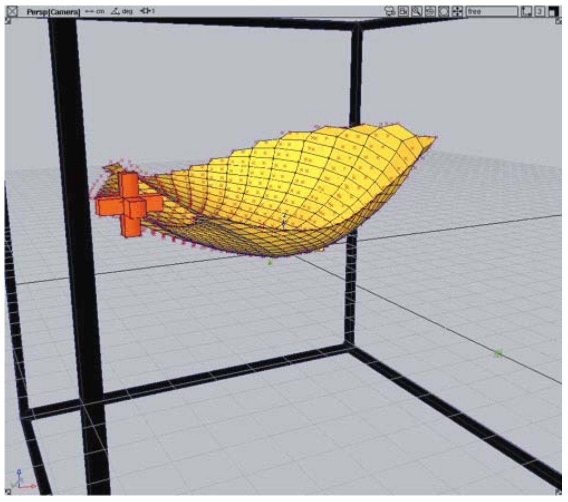


Figure 3: MoSS running inside Alias|Wavefront API. The black lines are the bounding box.

An attractor or repellor is defined as a point in 3D space. Around this point movement and drawing is warped to force the movement closer to the attractor or farther from the repellor. The amount of warping depends on distance from the attractor or repellor and the force of the attractor or repellor. This force is parameterized for the user to set. Attractors and repellors significantly alter the generated surface from its specification in axiom and production rule terms. They can be used to represent various factors in the 'real' environment, for example, topographic features.

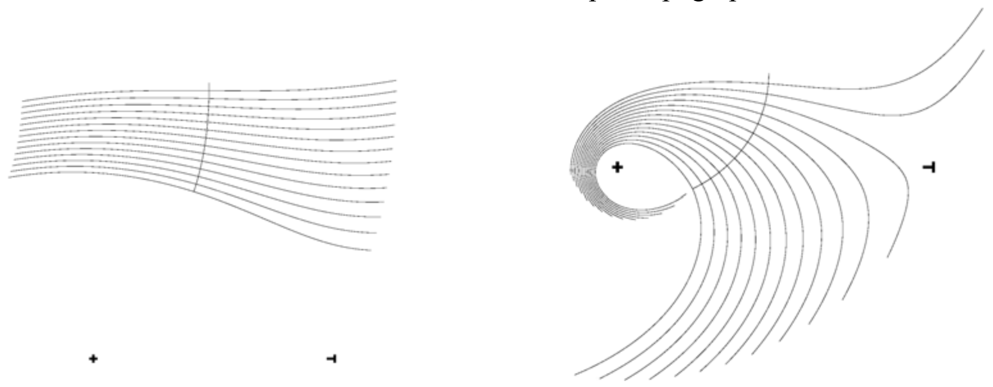


Figure 4: Effect of attractors and repellors on direction of growth. Repellors are indicated by sideways T and attractors by plus sign.

However, one important issue arises regarding the attractors and repellors. They disrupt the joints of surface plates and bend them away from each other leaving gaps. We have given our users two choices. The first (and simplest) is to accept the resultant gaps because they meaningfully relate to some modeled feature where the discontinuity is desirable or interesting. Second, we provide filters to repair the gaps. These are now described.

3.3 Filters

MoSS currently has three filters that address, in various ways, the distortion of surface geometry due to the effects of attractors and repellors. All filters attempt to move vertices on the surface so that they match up to their neighbors and form a closed surface.

The first filter, Filter 1, parses the set of surface vertices after the initial grammar generation and joins surface vertices within a certain radius of each other. This radius is specified as a Filter 1 threshold parameter. If there are more than three vertices, Filter 1 tends to leave gaps. The advantage of Filter 1 is that it works with all grammars since it is not dependent on number of surface vertices. It also joins any surfaces that are close together regardless of lines of growth. That is surfaces generated in different generations will be united even though, timewise, they are unrelated.

Filter 2 is an optimization of the principle that drives Filter 1 and is specifically used when the grammar defines hexagonal surfaces. It locates three vertices that are closest to one another and then converts them to a single vertex by averaging. This effectively closes any gap under a certain limit of discontinuity. This limit is parameterized numerically in terms of distance. Vertex merging is limited to vertices within a maximum distance from each other

Filter 3 is based on the idea of using the links which derive from the non-deformed grammar as the pattern for joining the surfaces after the attractor and repellers have asserted their effect. Using this filter guarantees a grammar that forms a continuous surface in an environment without attractors and repellors will form a continuous surface in an environment with them as well. It advantageously makes the outcome more predictable. Each filter can be used exclusively or not at all.

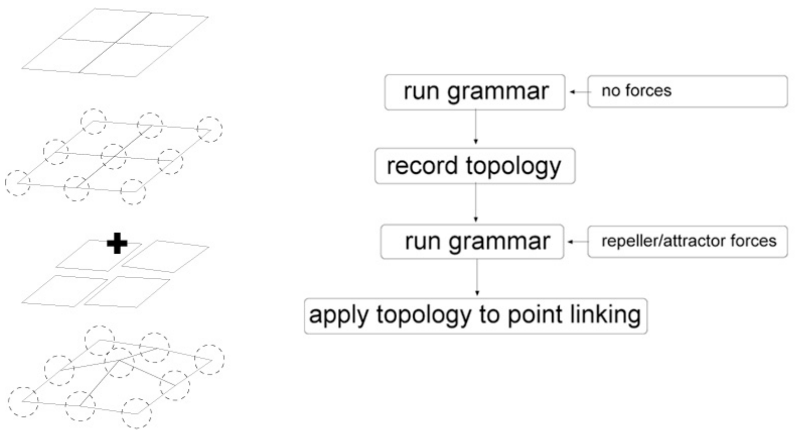


Figure 5: Illustration and Flow Diagram of Filter 3

4.0 CAD/CAM OUTPUT

In this section we describe the process of producing counterpart three-dimensional physical models. Three dimensional MoSS surfaces are unfolded using CAD/CAM software (Autodessys FormZ) into two dimensional format. These patterns are cut using a flat bed laser cutter.

The laser cutter alternately cuts and scores the surface allowing for reassembly of the complex curvature of any MoSS generated surface. These models may be produced in a wide range of mediums, including cardboard, plastic, aluminum and wood material. The three dimensional models are an important extension of MoSS as they allow the designer to evaluate the form and study the potential subdivision and joining of surface plates with a view towards manufacturing. The construction of surfaces may not necessarily follow the same scale or geometry of the generated MoSS geometry, as plates may be grouped in larger segments and then joined to form a continuos surface. The production of physical models is the first step towards an integrated system using structural analysis, material properties, finite element analysis (FEA) and CAD/CAM. This orientation is particularly interesting in light of advances in rapid prototyping which allow for diversity in the production process at little extra cost leading to locally optimized solutions.

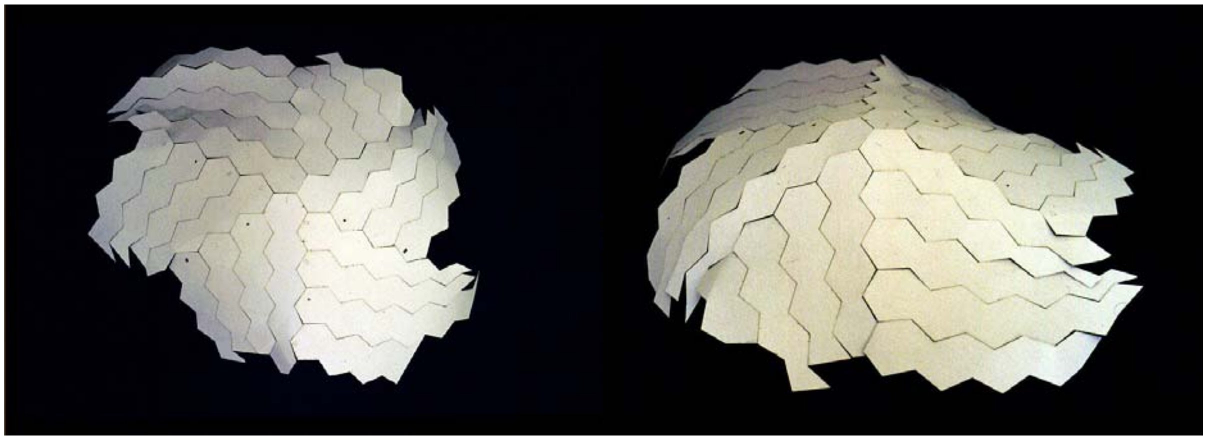


Figure 6: Laser cut three dimensional model of MoSS surface generated with hexagonal grammar.

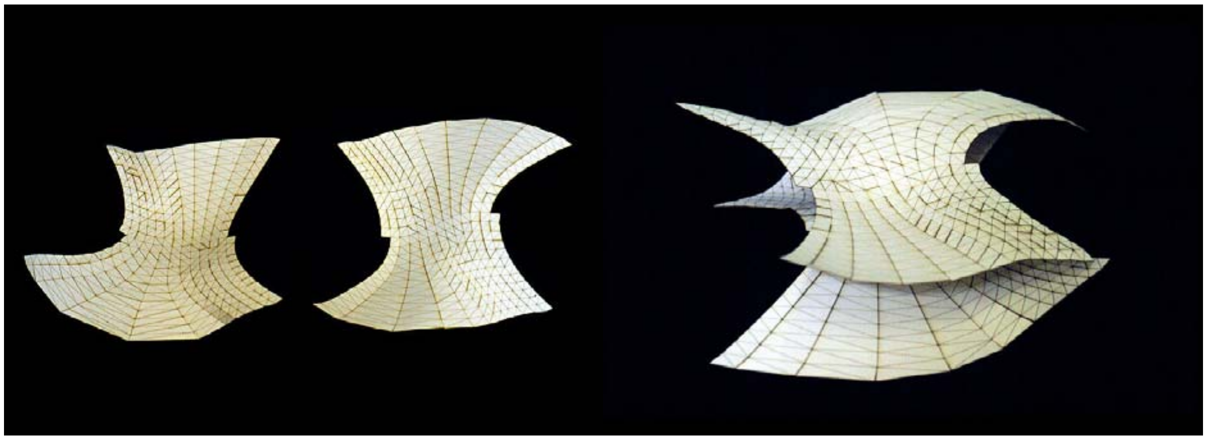


Figure 7: Laser cut three dimensional model of Moss surface generated with square grammar and grown with two offset surfaces. Note the tessellated pattern combines the MoSS grammar with the cut and score lines of the unfold software.

5.0 Appraisal and Future Work

How does MoSS meet our goals and expectations given the time invested to date in developing it? We are convinced that the L-system method of generating surfaces is a powerful and advantageous approach in many ways. L-systems allow a wide range of powerful expression and, given experience with students using MoSS, the concepts of grammatical definition and re-writing are simple for users to grasp. While it is not straightforward to use repellors and attractors to influence L-system growth (because they result in disjoint geometry), our filter methods surmount this issue. Through experience using MoSS the compelling reason for L-systems remains their ability to mimic morphogenesis and yield outcomes that are natural in aesthetic, unanticipated to an acceptable degree, and controllable in a simple manner. This will always differentiate MoSS from computer aided designs.

MoSS is still extremely rudimentary. Founding it on the L-system concept is a solid cornerstone for future development. Another positive action in MoSS's development has been that we are quickly handing the system (though prototypical and minimally user interfaced) to architects to use. Thus, we are receiving feedback very early and continually in the tool development process. This has been useful in indicating what we should work on next when, at most points, we find there are a variety of directions we could pursue. For instance, we have delayed adding the search component until we are satisfied that a single design can be grown which meets a user's demands. Only once we have a setup for growing a completely satisfactory design, will we add a component that searches for designs within a design space.

Another lesson from users exercising MoSS concerns the fact that MoSS has been used in a process that culminates in physical models. When the project was initiated, the computer scientists on the project understood the architects' interest in morphogenesis. However, it was difficult for them to grasp how architects would use MoSS surfaces after they were rendered in Alias|Wavefront. This prevented them from thinking of extensions to MoSS. The impressive physical models and the process of moving from MoSS, to 2D cutting layout, to laser cutting, and, finally, to physical models informed us that architects view MoSS as a tool within a series of tools. Another impression from users was that MoSS, in its current state, is primarily a stepping off point for creative design investigation. MoSS appears to play a role for architects in the early, conceptual stages of their design process. Its present value is in stimulation, surprise and experimentation.

Impressions of current ability stated, our users have also helped us prioritize future extensions. They have reported several areas in which they would like MoSS to be strengthened including an improved user interface. While they find the current specification of grammars through file input adequate, they report that being able to manipulate the strength and placement of attractors and repellors from the display and, perhaps, even between generations of re-writing would be beneficial. They also would like to be able to interact with the system's drawing between generations and have their changes impact the next generation's growth. This is consistent with another notion they express: while they can now grow multiple surfaces in one environment, these surfaces, if they intersect, have no impact on each other. In general, the system would be more powerful if growth was responsive, not just to attractors, repellors and the boundaries, but also to other objects in the environment.

In future developments we intend to explore four different interpretations of growth. All four models have a grammar based growth model in common. These developments can be classified into two groups: one directional and two directional growth.

One-directional growth:
a) Lindenmayer systems. As in the current instantiation of MoSS growth is one directional and once a member has been established the growth mechanism does not alter it.
b) Cellular model of growth. Instead of linear growth with one tip, growth could also be implemented using a model of cellular growth where each cell expands constantly after it has been created until it collides with neighboring cells sharing the same space.

Two-directional growth:
c) Adaptive model of growth. In this model generated geometry would adapt with the appearance of new environmental forces. This would require that the grown structure be aware of all its members.
d) Recursive model of growth. A model whereby the growth tips would search the environment for possible joining nodes. Upon detection of a suitable node a connection would be established replacing the search branch with a load bearing one. This process involves backtracking and evaluating the search paths and replacing parts of earlier growth.

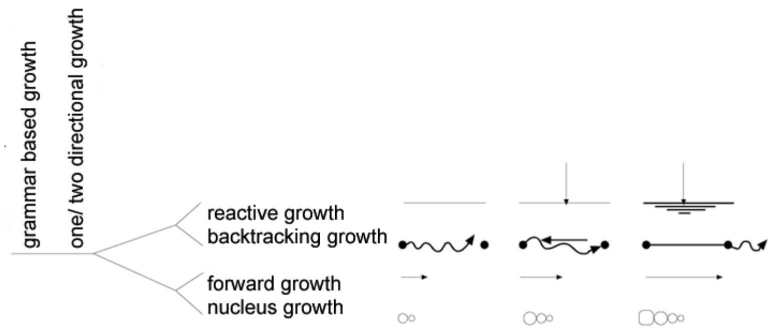


Figure 8: Interpretations of grammar based growth as a basis for possible extensions of MoSS.

Finally, MoSS, at present, assists with the development of a single concept. It does not actively search out designs that meet criteria an architect may have already formulated. We intend to add a genetic programming (GP) search component to MoSS. GP is inspired by neo-Darwinian evolution. A population of designs is tested for fitness, the fitter designs are chosen probabilistically more often to pair and generate 'offspring' designs and, generations later, a design which ultimately meets the fitness criteria adequately is found through this iterative process of selection, inheritance and variation. GP requires an encoding for designs. The encoding of MoSS designs as L-system grammars is a appropriate match for GP because they can be interpreted as tree structures that GP can crossover or mutate. GP also requires a *fitness function*. This is essentially an objective function by which designs can be differentiated and ranked. Determining an effective, user tunable objective function for MoSS remains an open issue. We have ideas for quantitative factors and are considering a subjective ranking interface to account for aesthetic value. The architects who have used MoSS have contributed essential feedback on this issue. At this point, MoSS has no knowledge of support or load. A consideration of these factors may enable us to move MoSS beyond its current contribution of stimulating exploration at an early design concept stage to a more practical stage when materials, weight and bearing are considered. We hope to exploit existing software to evaluate MoSS designs. This will reinforce our approach not to re-write any tool or component already in existence but, instead, to interface seamlessly with them. This extension would also be suitable for expressing a fitness function for the search component.

In summary, this is a report on a rudimentary morphogenetic surface design tool named MoSS. MoSS, written in C++ as a plug-in to Alias|Wavefront. MoSS is based on 3D L-systems and facilitates the growth of surfaces within an environment defined by boundaries, attractors and repellors. MoSS has thus far provided its users with stimulating design exploration capacity. Its foundation on L-systems has proved effective and MoSS is now ready for extensions that will enable it to be even more practical and powerful.

REFERENCES

1

J. Demongeot, E. Goles and M. Tchuente, eds. (1985). Dynamical Systems and Cellular Automata, Academic Press.

2

M. Gardner (1971). The fantastic combinations of John Conway's new solitaire game "Life". Scientific American, 223, 120-123.

3

J.B. L. Bard (1981). A model for generating aspects of zebra and other mammalian coat patterns. Journal of Theoretical Biology, 93, 363-385.

4

U. Frisch, B. Hasslacher and Y. Pomeau (1986). Lattice-gas automata for the navier-stokes equation. Physical Review Letters, 56, 1505-1508.

5

A.M. Turing (1952). The chemical basis of morphogenesis. Philosophical Transactions of the Royal Society of London, Series B, 237, 37-72.

6

Eric Bonabeau (1997). From classical models of morphogenesis to agent-based models of pattern formation. Artificial Life 3, 191-211.

7

Lindenmayer, A. and P. Prusinkiewicz (1989). "Developmental Models of Multicellular Organisms: A Computer Graphics Perspective". In Artificial Life. Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. VI. Reading, MA: Addison-Wesley.

8

Prusinkiewicz, P. and J. Hahn (1989). Lindenmayer Systems, Fractals and Plants. Springer Verlag Lecture Notes in Biomathematics, No. 79.

9

P. Prusinkiewicz and A. Lindenmayer (1990). The Algorithmic Beauty of Plants, Springer Verlag. With J.S. Hanan, F. D. Fracchia, D.R. Fowler, M.J.M. de Boer, and L. Mercer.

10

C. Jacob (1996). Evolving Evolution Programs: Genetic Programs and L-Systems. Proceedings of the First Genetic Programming Conference, MIT Press, 107-115.

11

P. Coates, T. Broughton and Helen Jackson (1999). Exploring three-dimensional design worlds using Lindenmayer systems and genetic programming. Evolutionary Design by Computers, Morgan Kaufmann, 323-341.

12

J. Koza (1992). Genetic Programming, or the Programming of Computers by Means of Natural Selection, MIT Press.