C. M. Herr, N. Gu, S. Roudavski, M. A. Schnabel, *Circuit Bending, Breaking and Mending: Proceedings of the 16th International Conference on Computer-Aided Architectural Design Research in Asia,* 175-184.

TREE FAÇADES

Generative Modelling with an Axial Branch Rewriting System

STUART HANAFIN,[1] SAMBIT DATTA[2] and BERNARD ROLFE[3]
Deakin University, Australia
1. stuart.hanafin@deakin.edu.au, 3. benard.rolfe@deakin.edu.au
2. Curtin University, Perth, Australia, sambit.datta@curtin.edu.au

Abstract. The methods and algorithms of generative modelling can be improved when representing organic structures by the study of computational models of natural processes and their application to architectural design. In this paper, we present a study of the generation of branching structures and their application to the development of façade support systems. We investigate two types of branching structures, a recursive bifurcation model and an axial tree based L-system for the generation of façades. The aim of the paper is to capture not only the form but also the underlying principles of biomimicry found in branching. This is then tested, by their application to develop experimental façade support systems. The developed algorithms implement parametric variations for façade generation based on natural tree-like branching. The benefits of such a model are: ease of structural optimization, variations of support and digital fabrication of façade components.

Keywords. Parametric Modelling; Biomimicry; Lindenmayer Systems; Branching Structures.

1. Introduction

The re-emerging science and philosophy of learning from nature is inspiring architectural design through the concept of biomimicry (Benyus, 1997). Biomimicry uses nature as a model and takes inspiration from natural forms and processes to solve human problems. The methods and algorithms of generative modelling can be improved by the study of computational models of natural processes and their application to architectural design. In this paper, we present a study of the generation of branching structures and their potential application in the development of façade support systems.

Architects use a variety of design approaches in drawing upon nature. For example, Architektonic Architects presented a design concept for the AAC Global Headquarters in China based on the structural logic of tree growth (Figure 1, left), where the building form and structural system replicate the form of trees and branches. Toyo Ito, in his Tod's Omotesando building in Tokyo (Figure 1, centre) draws its inspiration the elm trees that line Omotesando Boulevard. Ito explains the logic as,

> "Trees are organisms that stand by themselves, so their shape has an inherent structural rationality" (Pollack, 2005)

Figure 1. Examples of tree-like façades in architecture. AAC headquarters building (left), Ito's Omotesando building (centre) and A2RC's Brussels meeting centre (left).

In their entry to their Brussels Meeting Centre, A2RC Architects subdivide a glass cube with a treelike structure, gesturing to the adjacent historic garden (Figure 1, right). While these examples of architecture draw inspiration from nature but they do not utilise the formal logic of natural systems to drive the geometry. For example, in the Ito building, the form was developed in a traditional manner in partnership with structural engineer Masato Araya and his firm, OAK Structural Design Office. This paper aims to extend these informal approaches with formal generative methods to façade subdivision, driven by the logic and principles of branching systems.

2. Related work

Becker (2006) demonstrated a novel method for modelling branch connectivity with smooth nodes based on isosurfaces. Vanucci (2008) developed the notion of pluri-potential branching systems based on the interaction between biological processes and computation. Architectural applications of branching structures drawing on biological processes, computation and architectural design are explored in his research. Greenberg (2008) outlined the mathematical logic underlying various types of branching systems found in nature. Serrato-Combe (2005) investigated the application of LindenmayerSystems in architectural design.

3. Branching structures

Branching structures are a dominant occurrence in natural structures, such as trees and rivers. They have been investigated in mathematics and computer science as efficient data structures for search algorithms (Bovill, 1996). In design, novel studies of route networks, structural systems and spatial organisation have also touched upon the concept of branching (Frazer, 1995; Panchuk, 2006). They have been proven to be an optimised form of creating node-edge networks. In this paper, we present the application of two types of branching algorithms to develop façade support systems, a simple recursive bifurcation algorithm (Figure 2) and a rewriting system based on L-systems (Figure 3).

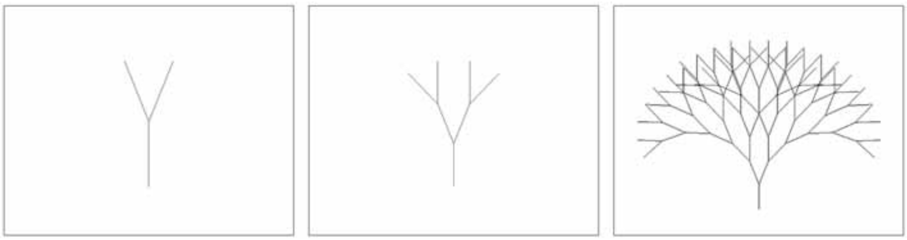

Figure 2. Bifurcation with recursive subdivision with depth d =1, 2 and 6.

3.1. RECURSIVE BIFURCATION

Bifurcation is the process of division of a main body into two parts or branches. This, when replicated, allows for the development of a simple branching system. This system is based off a single starting line that recursively generates new lines at the endpoint of the parent line. This allows for infinite branching and therefore requires careful control of the recursion depth. The control of the angle of division allows for the exploration of the geometric generation.

3.2. REWRITING SYSTEMS

Rewriting systems are a powerful method of generative modelling in design. Using a set of rules or productions, they allow for the parts of a simple initial object to be successively replaced resulting in a more complex final object.

String re-writing systems were first formally defined at the beginning of the 20th century by Thue, a Norwegian mathematician. String rewriting gained wide interest in the late 1950's due to Chomsky's work on formal grammars. At the centre of attention were sets of strings, called formal languages, and the methods for generating, recognizing and transforming them. These systems became critical to the development of computer languages.

In 1968 Aristid Lindenmayer, a biologist, introduced a new type of string-rewriting mechanism as a theoretical frame work for studying the development of simple multi-cellular organisms, and subsequently was applied to higher order plants. This type of string re-writing mechanism is termed an L-System and is based on the central concept of rewriting (Przemyslaw et al, 1990). L-systems provide a mathematical description of tree-like shapes and methods of generating them.

The key difference between Chomsky's grammars and L-systems lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, while in L-systems they are applied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems, with the productions intended to capture cell divisions (Prusinkiewicz and Lindenmayer, 1990).

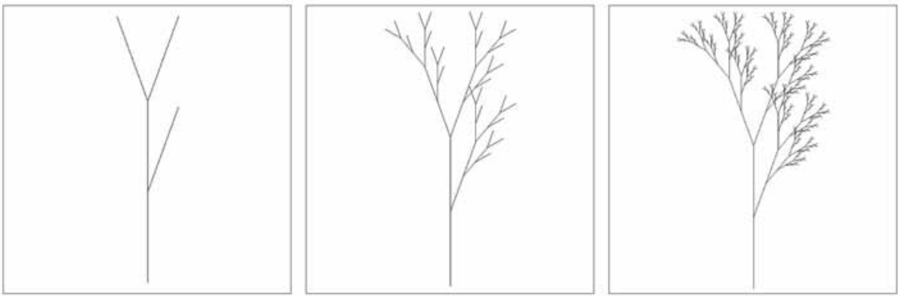

Figure 3. L-system based rewriting system with depth d =2, 4 and 6

4. Recursive bifurcation model

The standard model of recursive bifurcation, explained in Section 3.1, takes an initial line and places two copies of it at its end point. These copies are rotated by a rotation parameter with a positive and a negative phrasing to allow for a left and right branch generation. This process is then repeated on each of the new branches until the depth of the recursion is met. Recursive bifurcation was implemented within Generative Components with two extensions, the rotation and scaling variables were parameterised and randomised.

The rotation angle was modified by replacing the branching angle input with a random generator. Within this random angle generator, limits were set to ensure that the differences between branches were not too extreme (Figure 4, Left). As a result the script generates a completely different tree form each time the script is run. The second modification was to introduce a variable scaling input, which adjusts the length of the copied line by a variable scale factor. (Figure 4, Right).

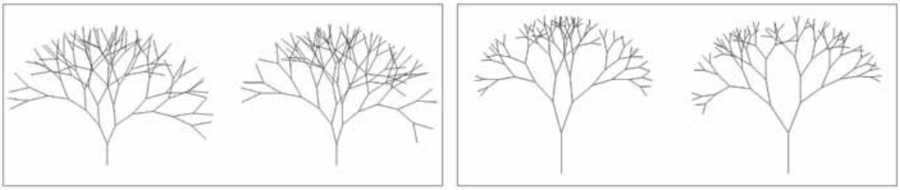

Figure 4. Bifurcation system with random angle (Left) and variable scale factor (Right).

While our bifurcation script allows for the generation of a tree pattern, it lacks any boundary definitions that can relate it to a building. If the tree is maintained within the façade boundaries, problem areas arise were the tree falls short of the boundaries. This is solved by ensuring that the tree is larger than the surface area required and the designer can then manually determine which section of the tree pattern will be used (Figure 5).

The bifurcation model successfully subdivides a façade while providing a tree like pattern. While this model offers advantages over manually drawing the subdivision, both in the variation within and the speed at which different options can be generated, it is still limited to the simple use of bifurcation. To implement deeper logic within tree branching, we have developed a rewriting script based on L-Systems.

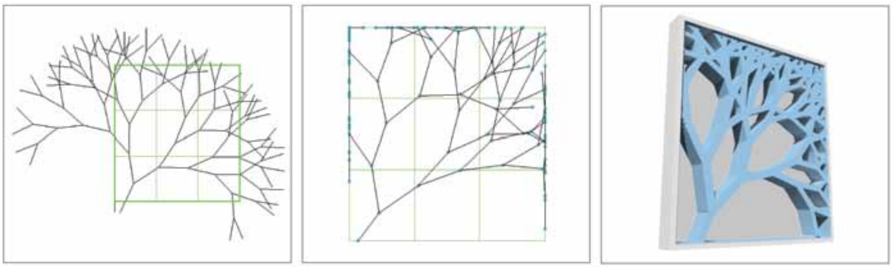

Figure 5. Boundary area (Left) and projection (Middle), final Façade model (Right).

5. Rewriting system model

In order to address the limitations of the recursive model outlined above, a second method of branch generation was investigated. In this scheme, a façade model based on L-systems was developed.

5.1. DOL SYSTEM

The simplest class of L-systems are those that are deterministic and context-free, called DOL-systems. A DOL-system is defined as the ordered triple $G = (V,w,P)$, where:

- V = {s1,s2,....,sn} is an alphabet composed of a set of distinct symbols.
- w, known as the axiom
- P, known as the finite set of productions

In order to understand how they work consider a string built of two letters *a* and *b*. Each letter is associated with a rewriting rule, in this example *a* → *ab* and *b* → *a*. This means that the letter *a* is replaced by the string *ab* and the letter *b* is replaced by *a*.

The rewriting process begins from a distinguished string called the axiom in this case *a*. In the first derivation step the axiom *a* is replaced by *ab*. The word *ab* consists of two letters, so in the second step both are simultaneously replaced, *a* with *ab* and *b* with *a* resulting in the string *aba*. In a similar manner the string *aba* results in *abaab*, which results in *abaababa*, and so on (Figure 6, Left).

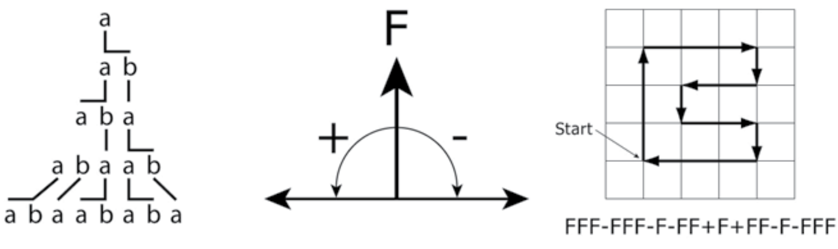

Figure 6. D-OL System (Left), Turtle Interpretation of string symbols (Middle) and Interpretation of a string (Right).

5.2. TURTLE INTERPRETATION

An L-system in its own right only provides a string and in order to develop the string into geometry we require a turtle. A turtle is a historical term that comes from the early days of computer graphics, where a mechanical robot turtle, capable of simple movement and carrying a pen, would respond to instructions (McCormack, 2008). The string is the instructions that are interpreted by the turtle as movement, orientation and geometry building actions. Reading the string from left to right the turtle interprets specific symbols responding accordingly (Figure 6, Middle), and given a step size and angle increment the turtle responds to commands represented by the following symbols:

- F Move forward a step of length d.
- + Turn left by angle δ. (Orientation of angles is counter clockwise)
- − Turn right by angle δ.

The turtle interprets a character string as a sequence of line segments and depending on the segments lengths and the angles between them, the resulting line from the turtle can be self-intersecting and more or less convoluted. The line always remains a single line, but can have some segments drawn many times (Figure 6, Right).

5.3. FAÇADE SUBDIVISION

A DOL-system for façade subdivision was developed with the following parameters:

- V = {F, X, +, -, [,]}
- w : X
- p1 : X → F[-X]F[+X]-X
- p2 : F → FF

This generates the sequence of words, where n is the derivation length = the number of iterations:

- w = n0 = X
- n1 =F[-X]F[+X]-X
- n2 =FF[-F[-X]F[+X]-X]FF[+F[-X]F[+X]-X]- F[X]F[+X]-X
- n3 =FFFF[-FF[-F[-X]F[+X]-X}FF[+F[-X]F[+X]-X]-F[-X]F[+X]-X] FFFF[+FF[-F[-X]F[+X]-X]FF[+F[-X]F[+X]-X]-F[-X]F[+X]-X]-FF[-F[-X] F[+X]-X]FF[+F[-X]F[+X]-X]-F[X]F[+X]-X

This demonstrates how the size of the string increases rapidly. This system was run until n=7 to develop a string comprising of 13,956 characters to input to the turtle. The turtle, described previously, interprets the string as a sequence of line segments. What is slightly different about this particular string is that X is a placeholder and is used to nest other symbols, and has no effect on the turtle. With the string input to the turtle additional inputs are required, the length of the line segment and the angle increment of the turtle when rotating. In this particular case a line segment length of 500mm is used and an angle increment of 22 degrees. This results in an L-system based tree being drawn, which while providing the desired overall geometry present areas for further development.

While the L-system model offers a more natural looking tree, the fixed angle increment results in a sense of regularity within the branching of the tree; similar to the bifurcation model. With this in mind, a new turtle was developed that is able to take a minimum and maximum angle increment input. Using this range the turtle is able to generate a new angle increment value each time that it reads a – or + from that string, resulting in a more varied tree generation due to the variation in the angle increments (Figure 7).

Figure 7. Initial L-system Tree (Left), examples of random angle increment (Middle, Right)

While the random input to the angle increment offers more variety in the tree generation, of more concern is the method in which the standard turtle draws the L-system. Because the turtle interprets the string as individual line segments of a fixed distance the total count of line segments is rather high, in the current example the turtle produces 4118 individual line segments. While this might not be an issue in terms of the visual aspect of the drawing it presents problems if further development and analysis is to be undertaken. In order to simplify the tree the new turtle was developed to simplify the string as it reads it. This is achieved by the turtle looking at the next character as it reads the string:

- If character = F, then advance the count.
- If character = [, then commit new line and start again.

As a result the turtle multiples the F count by the line segment length input and draws the simplified line, when it reaches a [in the string it begins again. This results in the line segment count being drastically reduced, in the current example dropping from 4118 to 2186 (Figure 8).

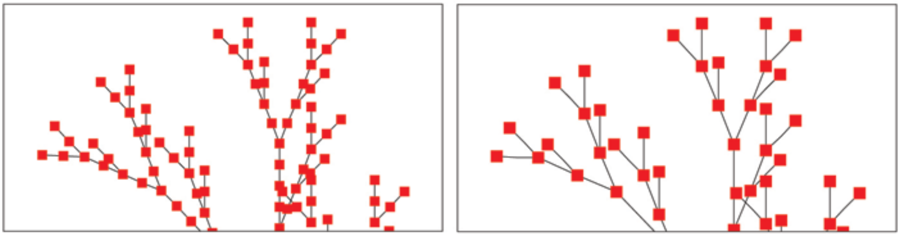

Figure 8. Line simplification before (Left) and after (right).

The required surface area is positioned in the desired section, with the tree geometry then being projected onto the surface (Figure 9). This allows for the surface to be sub-divided into lines of structure and areas of infill panels.

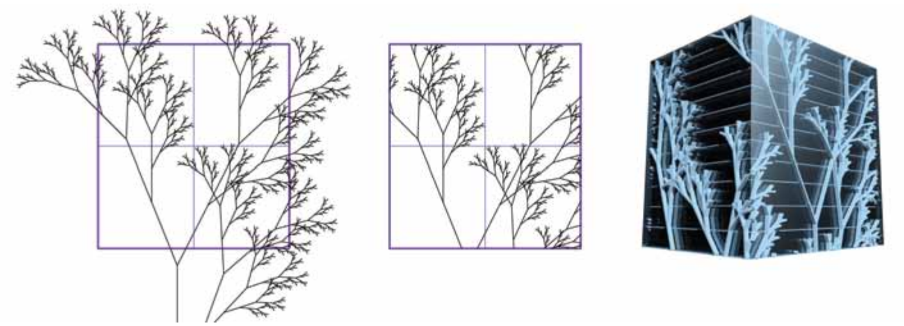

Figure 9. Overall L-system tree (Left) and projection (Middle), final Façade model (Right).

6. Results

This paper demonstrates the generation of façade support systems based on two models of branching, namely recursive bifurcation and L-systems. These two types of branching algorithms are implemented as parametric scripts for façade generation based on natural tree-like branching. We extended the standard procedures for branching variation by employing a randomised rotation operator and a parameterised scaling factor. Further, we introduce a boundary condition operator that permits the mapping of the branching script output to simple façade subdivision. The façade support systems generated by our branching scripts captures not only the form but also the underlying principles of biomimicry found in branching.

While the focus of this paper was to generate plausible branching structures for application to façade support systems, it is limited to two dimensional branching. Our goal is to extend the algorithms to three dimensional branching and to develop efficient and optimised subdivision algorithms that can be scaled to real world design. We intend to extend this research to employ structural optimisation methods to test the feasibility of our generated models in terms of structural support and material constraints. The benefits of such a model will be further tested in future research for ease of structural optimization, variations of support and digital fabrication of façade components.

Acknowledgements

The authors acknowledge the support of Jobrurt Bettadam of Architektonic Architects, Melbourne, Australia, for providing the initial problem of tree-like branching in the context of architectural design.

References

Benyus, J.:1997, *Biomimicry: Innovation* inspired *by nature,* William Morrow, New York.

Becker, M.: 2006, Branches and Bifurcations - Building a framework for modeling with isosurfaces, *in Generative Components in Communicating Space(s), in 24th eCAADe Conference Proceedings*, Greece 6-9 September, 868–873.

Bovill, C.: 1996, *Fractal Geometry,* Birkhauser Boston, Cambridge Mass.

Frazer, J.: 1995, *An Evolutionary Architecture,* Architectural Association, London.

Greenberg, E.: 2008, Observation, Analysis, and Computation of Branching Patterns in Natural Systems source Silicon + Skin: Biological Processes and Computation, *in Proceedings of the 28th Annual Conference of ACADIA,* Minneapolis, 16-19 October, 316–323.

McCormack, J.: 2008, Evolutionary L-systems, *in* P.F. Hingston, L.C. Barone and Z Michalewicz (eds.), *Design by Evolution: Advances in Evolutionary Design,* Springer-Verlag, Berlin, 169–196.

Panchuk, N.: 2006, *An Exploration into Biomimicry and its Application in Digital and Parametric [Architectural] Design*, University of Waterloo, Waterloo.

Pollack, N.: 2005, Toyo Ito fuses structure and wrapper in a network of concrete trees at the new Tod's Omotesanso Building in Tokyo, *in Architectural Record,* **193**(6): 79–85.

Prusinkiewicz, P. and Lindenmayer, A.: 1990, *The Algorithmic Beauty of Plants*, Springer-Verlag, New York.

Prusinkiewicz, P., Shirmohammadi, M. and Samavati, F.: 2010, L-systems in Geometric Modelling, *in Proceedings of the Twelfth Annual Workshop on Descriptional Complexity of Formal Systems*, 3–12.

Ramaswamy, S.: 2007, *Biomimicry: an analysis of contemporary biomimetic approaches,* CEPT University, Ahmedabad.

Serrato-Combe, A.: 2005, Lindenmayer Systems – Experimenting with Software String Rewriting as an Assist to the Study and Generation of Architectural Form source Digital Design: The Quest for New Paradigms, *in 23rd eCAADe Conference Proceedings,* Portugal, 21-24 September, 615–621.

Vanucci, M.: 2008, Pluri-Potential Branching System in Silicon + Skin: Biological Processes and Computation*, in Proceedings of the 28th Annual Conference of ACADIA*, Minneapolis, 16-19 October, 354–361.