Architectural Applications of Complex Adaptive Systems

Cory Clarke and Phillip Anzalone
XO (eXtended Office)

Abstract

This paper presents methods and case studies of approaching architectural design and fabrication utilizing Complex Adaptive Systems (CASs). These case studies and observations described here are findings from a continuing body of research investigating applications of computational systems to architectural practice. CASs are computational mechanisms from the computer science field of Artificial Life that provide frameworks for managing large numbers of elements and their inter-relationships. The ability of the CAS to handle complexity at a scale unavailable through non-digital means provides new ways of approaching architectural design, fabrication and practice.

1 INTRODUCTION

This paper documents findings from investigations of the application of Complex Adaptive Systems (CASs) to the practice of architecture. CASs are computational mechanisms from the computer science field of Artificial Life that provide frameworks for managing large numbers of elements and their inter-relationships. Some algorithms that are classified as CASs are Cellular Automata, Lindemayer Systems, Turing Machines and Flocking algorithms. There are many potential applications for these systems in the architectural design process:

- Tools for the automated design of large scale buildings and urban projects;
- Techniques for the design of serialized buildings (e.g. mass housing, franchise buildings, pre-fabricated construction) that are contextually-sensitive and differentiated;
- Platforms for roboticized self-assembling and self-adjusting buildings;
- Methods for designing adaptable buildings with redundant structures with the ability to more successfully withstand damage and catastrophic events.

Critical to our investigation is the marriage of CASs with suitable geometric and structural systems. Complex computational systems have found little application in contemporary architectural practice because their effective realization is often not possible with traditional geometries, such as the Cartesian coordinate system, or

traditional structural methods, such as trabeated construction. In parallel with our investigation of applications of CASs to architectural design has been the development of formal and structural systems with geometries to match the CAS morphology.

This paper describes:

- The concept of CASs and their presentation in architectural terms, addressing ideas of context, site, program and form;
- Development of structural and construction systems that provide a means of constructing results from CAS based design systems;
- Case studies of the application of CASs in architectural design.

2 CAS CONCEPTS IN ARCHITECTURAL DESIGN

CASs are a class of dynamic systems from the field of Artificial Life. These systems typically involve sets of discrete elements that change state (typically visualized by changes in color), based on the iteration of a simple set of deterministic rules (Wuensche, Lesser 1992). The elements' states change as a function of their current state and the rules, producing an unpredictable, complex global behavior pattern.

A CAS changes its state based on the rules of the system. These rules are applied to the current state of every element in the system to calculate each element's new state. Despite the fact that these systems' behaviors are defined by completely deterministic rules, the behaviors of these systems (with sufficient elements in play) are so complex that they cannot be predicted. Since the new state of the system is determined by the current state, different initial states of a CAS can foster entirely different emergent behavior. To change the behavior of the system a CAS can merely be started with new initial state.

Different initial states in CASs will lead to different global behavior and patterns. In this way the interaction of the elements within a CAS can be thought of as a 'calculator' (Neumann, 1966), computing outputs based on inputs. This capacity to derive results from a complex matrix of inputs is an obvious analog to the design process – which must take the complex matrix of requirements – program, site and structure – and translate them into an architectural outcome.

There are several properties shared by CASs upon which we have focused our research. The properties listed below have provided us with a means of linking the processes and

behaviors of CASs to the design and production of architecture. While these properties may not belong to all CASs, they provide a list of traits that can be combined and exploited in the production of architectural design tools. The properties are discrete composition, algorithmic relationships, exogenous control and scalability.

2.1 Discrete Composition

The underlying substructures of CASs are discrete matrices of elements, treated as independent entities, operating in parallel. Despite the discrete nature of these systems at the local scale, they are able to exhibit global behaviors with varying degrees of continuity. The trait of having a discontinuous substructure, but varying continuity at the level of superstructure, is one of the behaviors that make CASs suitable for translation into architecture. This trait provides an analog to the way in which architecture is produced; assembled from discrete elements – either programmatic (rooms and zones) or structural (beams and columns) – that define spaces with varying degrees of continuity (e.g. spatial, programmatic, acoustical, environmental).

2.2 Algorithmic Relationships

The behavior of a CAS is prescribed by sets of rules and algorithms that dictate the relationships between elements within the system. The relationships between elements in a CAS are iteratively re-evaluated based on these rules, providing the means for the system to adapt to internal and external changes. The clear definition of relationships between elements, and the reiteration of those relational rules over time, finds an analog in the design process – where design intentions are described in terms of relationships (privacy, lighting, proximity), and the final product emerges from the continual reiteration and refinement of these relationships.

A specific example of this can be seen in the computational algorithm described as flocking. Basic flocking systems (Figure 1) define a set of desired relationships between elements in a flock (such as ideal distance between elements and preferred position relative to other elements) and attempt to maintain these relationships by continually readjusting the positions of each element (Reynolds, 1987). When the rules of proximity and position are applied to a group of elements they exhibit complex adaptive behavior similar to a flock of birds. This simple set of rules of proximity and position can find application in architecture, for example, as rules for positioning of programmatic or structural elements.

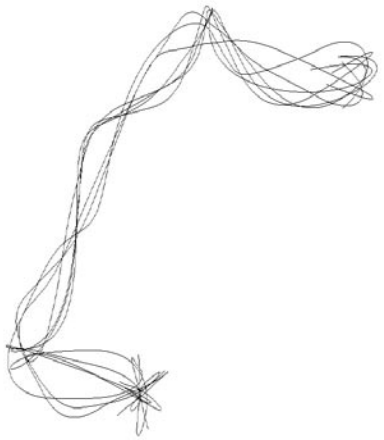


Figure 1 - Paths described by the motion of flocking particles .

2.3 Exogenous Control (Contextual Awareness)

Along with the ideas of internal constraints, many CASs have the capacity to react to exogenous control, which in architectural terms can provide mechanisms for contextual

and environmental awareness, and influencing the system to incorporate additional design intentions. CASs have two primary mechanisms for reacting to environment and context – initial state definition and avoidance behavior.

Since all CASs operate on a deterministic set of rules, the initial state of the system ultimately determines its outcome. Thought of in this way, the outcome of a CAS can be seen as a registration of the rules upon a particular initial state. With a simple CAS, such as Cellular Automata, the entire range of possible outcomes for every rule from a single initial state can be mapped (Wuench, Lesser 1992). In an architectural application the initial state can be defined in terms of the environment, and the behavior of the CAS would be a direct expression of the relational rules of the system within that environment.

Along with the ability to register the initial state of their environment, CASs can react to external change via mechanisms such as avoidance behavior. This behavior is comprised of basic mechanisms that monitor the incremental growth of a CAS for collisions with objects or fields in the environment and provide for additional reactions. If elements in the system are currently on a collision course the monitoring mechanism signals for new behavioral rules to be invoked that react to the potential collision.

The obvious architectural implication of the collision detection and avoidance logic is to install the system with an awareness of surrounding buildings and landscape. However, the behavior can also be applied to more abstractly zoned volumes within the systems surrounding context as well. The growth of a system could respond to zoning envelopes, acoustical envelopes, sun and shade volumes, view corridors, program areas, and any other abstract constraints that can be represented volumetrically.

Collision detection can be further extended as a mechanism for control by inverting the logic of avoidance behavior so that the collision detection mechanism can be used as a tool for searching out attractive areas of an architectural context. If the avoidance behavior were to be changed to tend towards potential collision instead of away, the morphological tendencies of the CAS will grow towards specific areas in a site that are designated as desirable. Furthermore the volumes of desirability, and undesirability, can be weighted in influence to produce a multivalent map of context.

2.4 Scalability

Many CASs exhibit a fractal behavior - the behavior of a small number of elements is congruent to the same system with exponentially more components. This scalability of behavior has two benefits to the study and implementation within architectural practice. The first is one of study and testing; since the behavior of a large system

is similar to that of a small system, methods for design can be modeled in a limited capacity and still translate to the larger scale of a building or urban project. The second benefit of the scalability of CASs is their potential to be nested fractally. Their scalability makes them essentially scale-less; a particular set of rules that functions well as a global organization system could be used at the scale of an urban project, while each 'cell' could in turn be filled with a smaller version of a CAS with a detailed behavior at the scale of building or dwelling unit.

3 STRUCTURAL SYSTEMS

The characteristics of the organizational strategy outlined above, and the diverse range of possible complex forms that may result, require a structural solution with similar geometric and algorithmic traits. Our research focused upon the three-dimensional differential space-truss.

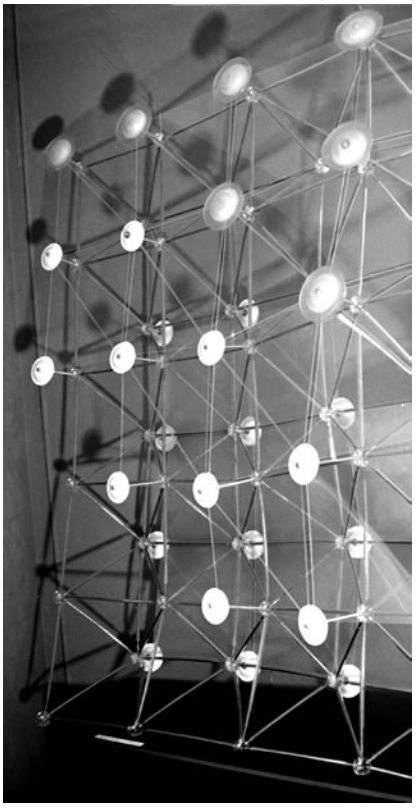


Figure 2 - Fabricated prototype of differential space-truss.

The traditional space-truss is a lattice structure of standard elements, typically leading to architecture of regular geometrical forms, as in the geodetic domes of Buckminster Fuller and projects such as I.M. Pei's Javits Convention Center in New York. The traditional space-truss employs standard elements because of constraints of design, analysis and fabrication – constraints now surmountable through computer-based techniques. The differential space-truss uses non-standard elements; by allowing each element to be unique it can take on complex three-dimensional curvilinear form as well as basic linear geometry.

During our research we built and tested several scale models of three-dimensional differential space-trusses with non-uniform elements (Figure 2). The manufacturing of the structural elements requires the use of Computer Numerically Controlled (CNC) fabrication techniques. The specific biases and limitations of these construction methods can be built into the relational rules of the CASs as further means of control. The differential space-truss has three major traits that match many of the behaviors and properties of CASs: discrete composition, lattice geometry and scalability.

3.1 Discrete Elements

Space-truss systems are comprised of two basic components, linear struts and connecting nodes; through the manipulation of these two types of elements the system can yield a diverse range of complex three-dimensional forms. The fluid nature in which a space-truss can transition between curvilinear form and linear geometry is similar to that of monolithic structural systems such as cast concrete, yet it is constructed from repetitive discrete elements. Its range of formal expression, coupled

with its discrete composition, makes the space-truss an ideal structural expression for the organizational strategies of CASs.

3.2 Lattice Configuration

The lattice configuration of a space-truss, a continuous network of nodes and struts, allows it to effectively behave as a monolithic system; load forces passing through the system on a global scale as if it were a uniform structure. The overall structural capacity emerges from the configuration of the discrete elements and their inter-connections, mimicking almost identically the emergent behavior of the coordinated elements in the CAS.

3.3 Scalability / Micro-scale Components

Structurally, the space-truss has no predetermined scale, capable of functioning equally with struts the length of 10 meters as with struts only 10 centimeters. The economies of scale behind traditional construction have led to the implementation of space-trusses only at a large scale, a single strut commonly measuring more than a meter in size. The use of CASs, which are capable of managing innumerable discrete elements, especially if they are coupled with CNC fabrication techniques and roboticized assembly methods, can allow the space-truss to be implemented at a finer scale, giving more adaptability and greater range of formal expression.

4 Case Study 1: one-dimensional Cellular Automata

We have performed several case studies applying CASs to architectural form; two of which are documented in this paper. Both case studies described here employ algorithms that are categorized in a subclass of CAS called Cellular Automata. The two specific examples are one-dimensional Cellular Automata (Wolfram, 2002) and an adaptation of Conway's Game of Life (Gardner, 1970). These studies focused on investigating ways of formally manifesting the behavior of the CASs via appropriate geometric systems and the differential space-truss structure.

The case studies were produced by writing custom software plugins for Maxon's Cinema4D and Alias|Wavefront's Maya. The software plugins used the CAS rules to produce three-dimensional surfaces and lattice structures. The software was developed to generate forms that are able to be easily realized using CNC fabrication techniques.

4.1 Cellular Automata Rules

The structure of a Cellular Automata is a one-dimensional array of elements with variable states – typically represented as a line of black and white squares. The array is manipulated via a set of rules; in the studies done here we used 3-Neighbor Cellular Automata, which have only eight rules. The rules govern how the states of the individual elements change over time. For each discrete moment in time, the state of each element in the system is analyzed and changed based on the rules. An example of a set of rules is show in Figure 3. Typically the behavior of a Cellular Automata is visualized as a time-series plot, showing snapshots of the state of the one-dimensional array of elements at sequential moments in time all together as a two-dimensional grid.

4.2 Outcome

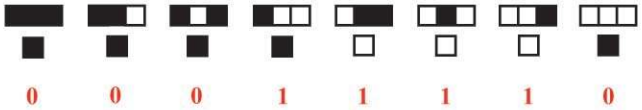
In translating the behavior of the one-dimensional Cellular Automata system to architectural design, the behavior of the system was treated as a series of instructions for growth in three-dimensions. Instead of interpreting the states of the elements within the Cellular Automata as white and black, the states mapped to angular directions for the development of a space-truss lattice.

Forms were generated by incrementally growing a space-truss structure where each new element within the structure was placed relative to the previous element based on the changing state of the Cellular Automata system. We discovered the forms with the most potential for architectural application resulted from the coupling of changes in state within the Cellular Automata and changes in angle within the growth of the space-truss structure. As each element was added to the truss, the angle of the new element relative to the previous element was dictated by the current state of the Cellular Automata system. A Cellular Automata pattern being translated to different surface configurations through the association between Cellular Automata states and various angles is illustrated in Figure 4.

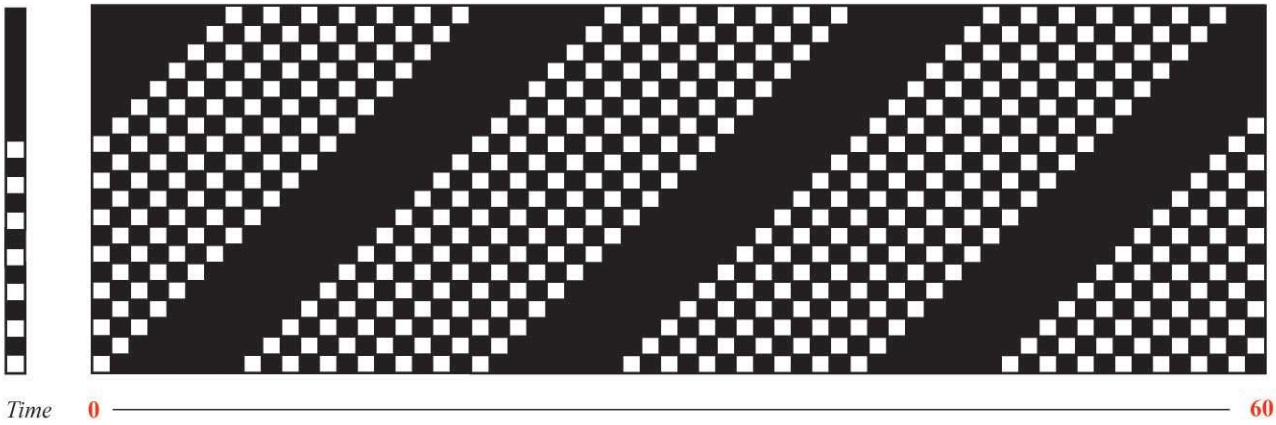
The Cellular Automata system was further studied by augmenting it with rules for exogenous responsiveness and tectonic/structural intelligence. An additional layer of behavior was added to the basic behaviors employed in the previous examples, giving it collision detection so it could respond to boundaries and environmental constraints. Figure 5 illustrates the results of the collision detection in a simple formal study.

Along with the layer of collision detection and responsiveness a third system of behaviors was added to install proximity detection between elements. Each element within the Cellular Automata system was given additional instructions that controlled its distance relative to it neighbors during the generation of the lattice structure (Figure 5). This behavior attempted to keep the spacing between elements at an

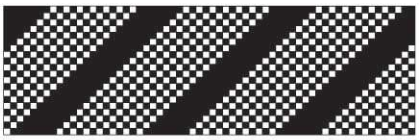
The Cellular Automata (CA) addressed here are simple dynamic systems that evolve through the iteration of 8 rules. These rules operate upon a discreet matrix of 'cells', the results from each iteration are used as input to generate the next. The rules describe how the state of a cell will change in time based upon its current state and the state of its neighbors.



The CA rule above, described by the binary number 00011110, generates the pattern below over a time period of 60 steps. CA are entirely deterministic, their final form resulting directly and predicatbly from the internal rule structure and the initial state of the cells.



The diagrams on the right illustrate the implementation of Relative Positional and Relative Angular interpreters and their formal results. The surfaces are all differing interpretations of the same Cellular Automata system, starting from the same initial state. When the same Cellular Automata system is rendered as a simple 2D representation it appears as below.



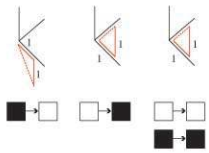
Rule: 00011110

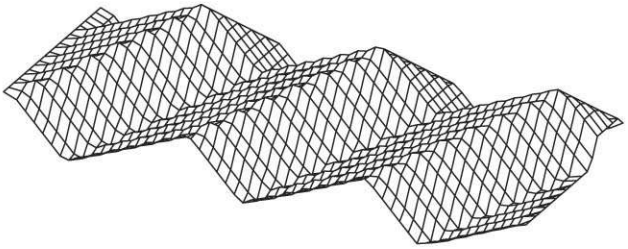
The *Relative Positional Interpreter* renders a surface structure by translating the changes in states as changes in position of points on a surface. The positional changes are incremental, moving up or down relative to the previous point location.

The *Relative Angular Interpreter* operates in a similar manner as the *Relative Positional Interpreter*, translating changes in the states of the Cellular Automata into form. The angular interpreter reads changes in states of the Cellular Automata as turns in a relative drawing mechanism. Each angular change is relative to the previous direction of the mechanism, making the range of possible form (and volume) more diverse than the positional interpreter.

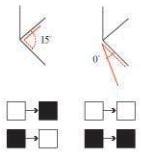
Each interpreted form is accompanied by a key showing the CA state shifts and which angle/distance is used to interpret it into form.

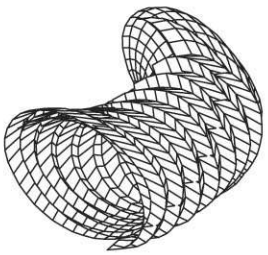
Relative Positional Interpreter

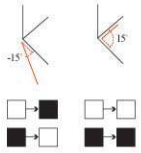


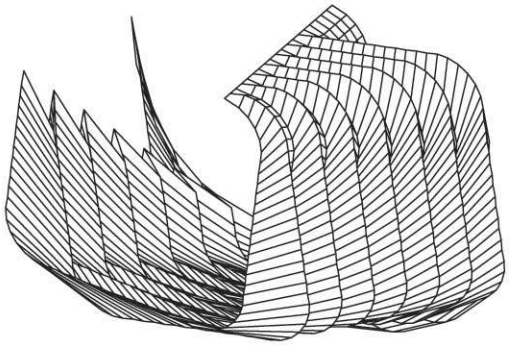


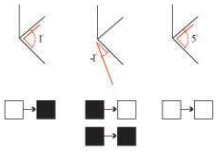
Relative Angular Interpreter

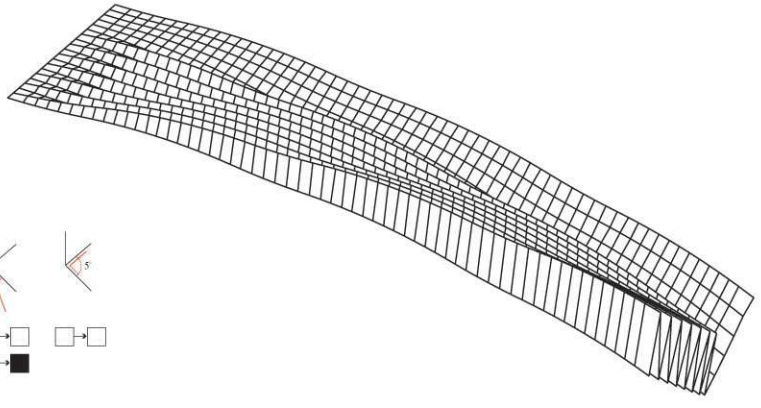




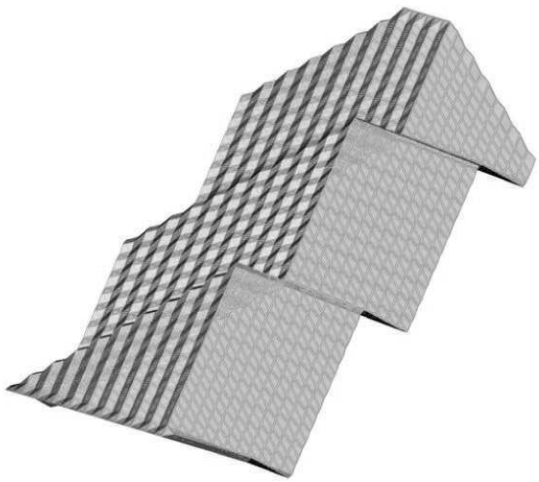




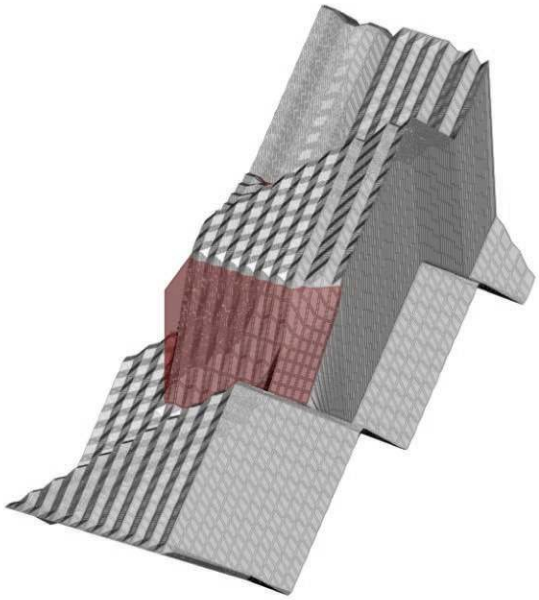




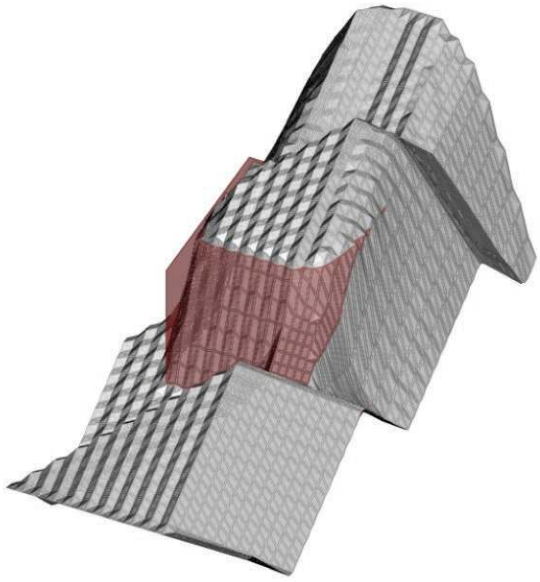
A simple Cellular Automata using rule 0111101 to generate a folding form supported via a differential space truss structure. The Cellular Automata generates a repeating pattern that continues indefinitely, unresponsive to any external factors.



The same Cellular Automata system and rule set as used above, but with an additional layer of intelligence, adding an environmental responsiveness. The structure responds to the influence of a rectangular volume in the environement. The Cellular Automata generates a folding form as above, however when it comes in contact with the volumetric obstacle in the environment it reacts by attempting to move up and out of the volume.



The third image in the series depicts the same Cellular Automata system described above, but with yet another layer of intelligence added to the generative system. This layer of intelligence provides for a 'flocking' logic, preventing extreme variations in the overall form that would compromise the structural integrity. The 'flocking' logic is a communication mechanism between neighboring elements, allowing them to monitor the distance between themselves and their nearest neighbors. This allows them to adjust and maintain a consistent spacing and angle between components that is necessary for the structural behavior of the system. The term 'flocking' refers to a similar algorithm that is employed in flocks of birds, allowing the individual birds to maintain uniform distances between each other within a moving flock.



optimal distance, one that corresponded to an optimal relative length for a structural element in a differential space-truss. The mechanisms employed in maintaining optimal distance is similar to those used in the flocking behavior described earlier.

A part of the research we developed a software plugin for Maxon's Cinema4D to generate structures based on the parameters described above. The plugin has two major components, one titled 'Behavior', for controlling the Cellular Automata behavior and a second titled 'Renderer' for controlling the interpretation of that behavior into form. The two components are reflected in the interface shown in Figures 6 and 7 below. The Cellular Automata rules, iterations and initial state as well as its flocking behavior are all controlled via the Behavior interface. The Renderer portion allows a designer to associate the different states of the Cellular Automata with different angular directions for the interpretation into form, as shown in figure 4. The collision detection and avoidance behavior is integrated into the system so that any shapes defined within the modeling environment are treated as context and avoided as the Cellular Automata structure generates itself. The plugin generates both a surface model as well as a structure of lines and curves; these curves can provide a basis for the generation of laser cut structural models.

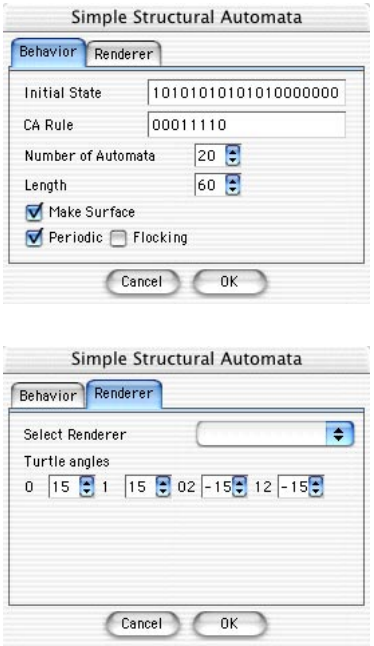


Figure 6 & 7 – Interface components for Behavior and Renderer control in the Cellular Automata system

We ran several tests of hypothetical scenarios, deploying Cellular Automata systems with specific rule sets within different urban and rural contexts. Figure 8 shows the results of one of these tests. In this test a pair of Cellular Automata, tuned towards different degrees of transparency and opacity, were deployed in an urban infill lot. The Cellular Automata plugin we developed generates ruled surfaces, which can be optimized for fabrication with linear tools such as laser cutters. Generating surfaces with a rule-based system that incorporates requirements of the fabrication method is similar to that employed in the Morphogenetic Surface Structures project from the Emergent Design Group (Testa, O'Reilly, 1999).

5 CASE STUDY 2: Conway's Game of Life Variant

5.1 System Rules

Conway's Game of Life is a two-dimensional Cellular Automata a matrix of elements with variable states in two directions. The matrix of elements is typically arranged as a grid of black and white squares, and a set of four rules is used to change the states of the elements at each discrete moment in time. The rules for changing each element are, as with one-dimensional Cellular Automata, based on the immediate context of each element. However, since there are many more possible configurations of neighboring elements than in a one-dimensional Cellular Automata, the rules can operate on a

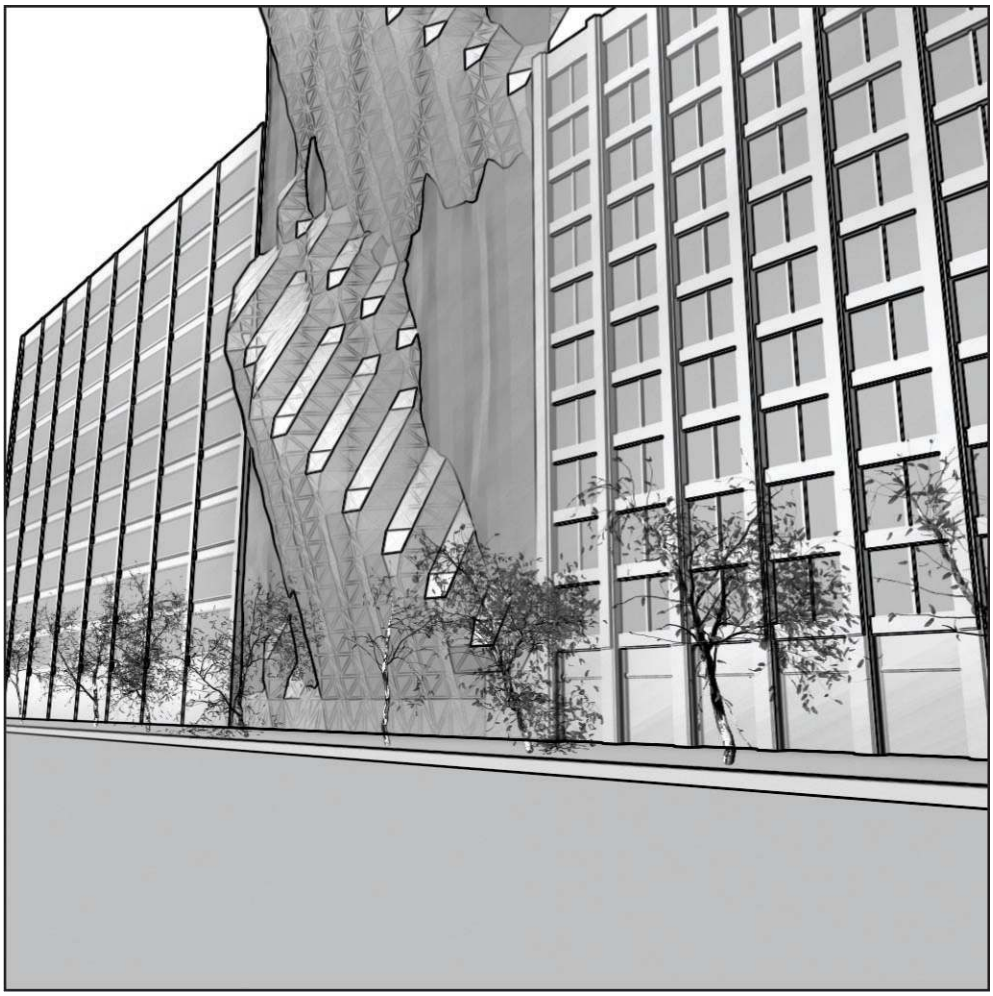
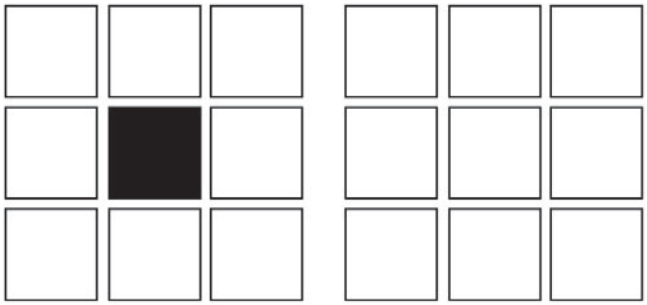
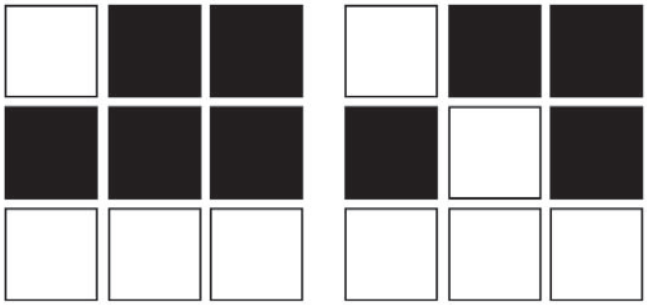


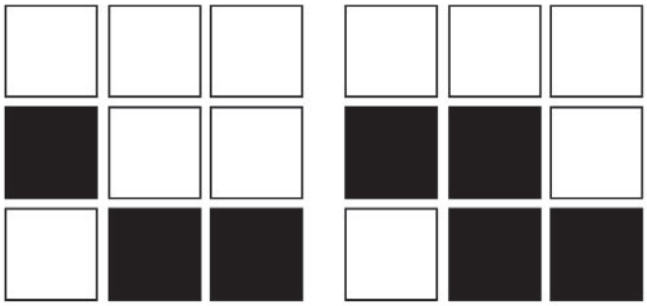
Figure 8 - Deployment of a Cellular Automata architectural design system in an urban infill context



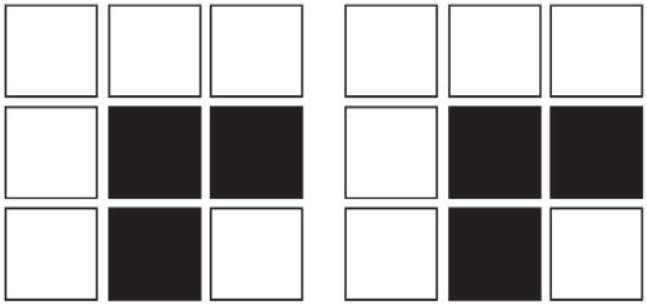
Loneliness
A cell with less than 2 adjoining cells dies.



Overcrowding
A cell with more than 3 adjoining cells dies.



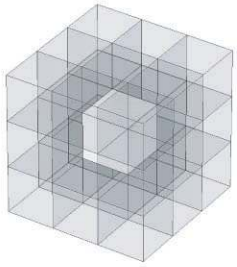
Reproduction
An empty cell with more than 3 adjoining cells comes alive.



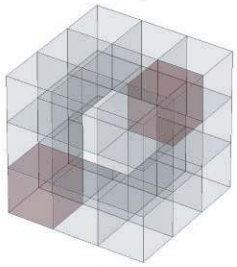
Stasis
A cell with exactly 2 adjoining cells remains the same.

The rule table below illustrates one set of possible rules for a 3D cellular automat based on the Game of Life. The 3D CA uses a three-dimensional grid as its matrix of cells, giving each cell a neighborhood of 26 adjoining cells. With the addition of exponentially more neighbors, the criteria for 'Loneliness' seen in 2D cellular automata is removed, leaving 3 rules. The 3 rules described below set criteria for reproduction, death from over-crowding and statis.

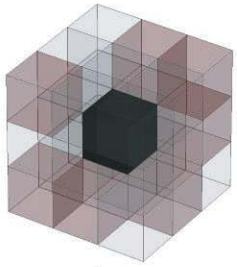
On the right is pictured a typical sequence of iterations of the 3D game of life. The entire matrix appears to pulse from iteration to iteration. Similar to the 2D game of life, recurring coherent organizations occur within the patterns of behavior.



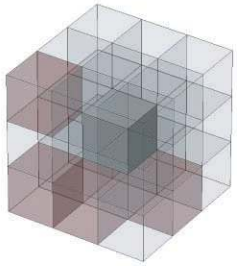
3D neighborhood, comprised of the active cell and its 26 neighbors



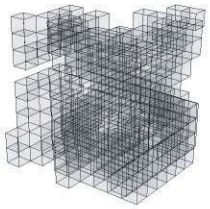
Reproduction - An empty cell with 2 neighbors comes to life

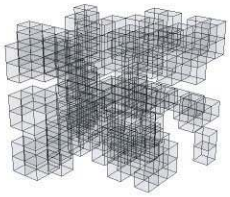


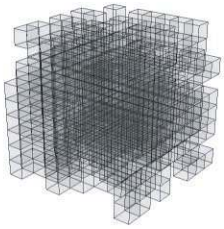
Overcrowding - A cell with more than 7 neighbors dies

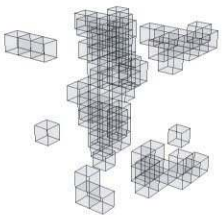


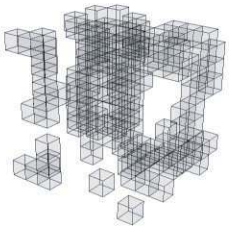
Statis - A cell with more than 2 but less than 7 neighbors continues unchanged

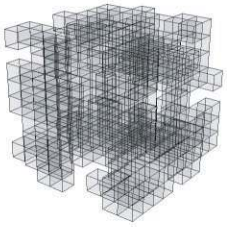












more qualitative level. For instance, the typical rules describe conditions of loneliness, over-population, stasis and reproduction. These rules are shown in Figure 9.

To adapt Conway's Game of Life for architectural application, we derived a variant of the rules and system that works on a three-dimensional matrix of elements. Figure 10 shows how the Game of Life can be thought of as a three-dimensional system. In this case study we extended the system of Conway's Game of Life so that its implementation is not limited to a Cartesian spatial matrix so it can produce spatial conditions that are complex and more specific to the structural morphology of the differential space-truss.

5.2 Outcome

In experiments with our variant of Conway's Game of Life (Conway Variant) we sought to make the ideas of contextual awareness and structural/tectonic logic explored in the one-dimensional Cellular Automata more central to the generation of form. The basic rules of the system, those of loneliness, overcrowding and reproduction, integrate contextual awareness, without layering on additional behavioral logic as was done in the Cellular Automata studies. The architectural context is described in the same manner as the systems elements, allowing the growth of the elements to react to boundaries and environmental constraints. The forms generated by the system also act as contextual constraints for the next phase of growth, allowing the system to limit its growth in a more controlled manner.

To embed structural and formal logic into the system, studies were made in using space-filling geometries other than the typical 90-degree grid (Nooshin, H., Disney, P. L., and Champion, O. C, 1997). With one-dimensional Cellular Automata and Conway's Game of Life, the system is typically represented as a grid of square tiles. In moving into three dimensions it is simple to mimic the behavior of these systems by extending those square tiles into three dimensions as a matrix of cubes, giving extremely limited formal possibilities and imparting no structural logic to the system. Instead, we looked towards more complex and variable space-filling geometries, such as the geometry of cells and crystals (Pearce, 1990), that matched more effectively the formal logic of the differential space-truss. Coupling different space-filling geometries with variants on the rules of Conway's Game of Life we were able to produce a diverse range of possible forms including complex spatial and structural configurations.

As with the Cellular Automata case study, we developed a custom plugin for the Conway Variant case study that would generate sample structures. The plugin provides an interface for controlling all the aspects of our variant of Conway's Game of Life rules such as 'reproduction' and 'overcrowding'. The plugin interface is pictured in Figure 11. Instead of operating within a simple Cartesian grid, as shown in Figure 10, the system grows based on a spatial morphology of a differential space-truss.

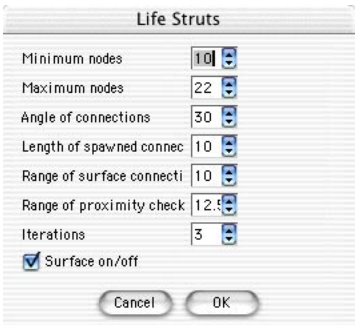


Figure 11 - Software interface for generating Game of Life structures

As elements within the Game of Life, the plugin uses 'nodes' within a space-truss. However, instead of being located within a fixed three-dimensional grid the position and location of these nodes are determined by the 'angle' and 'length' parameters of the plugin. As the system grows, each node in the structure spawns new nodes and struts, or 'kills' them off, based on the rules of reproduction and overcrowding from our variant on the Game of Life.

Figure 12 shows a highly constrained Conway Variant system deployed with constraints that limit its growth to a primarily vertical direction. The rules and space-filling geometry used produce a variation on a typical office tower structure with much more structural redundancy and spatial variety at the scale of the individual floor and unit.

Figure 13 shows the same system used to produce the tower, but deployed with a different initial context and geometric structure. With these different parameters the system produces a series of punctured domes with varying degrees of enclosure.

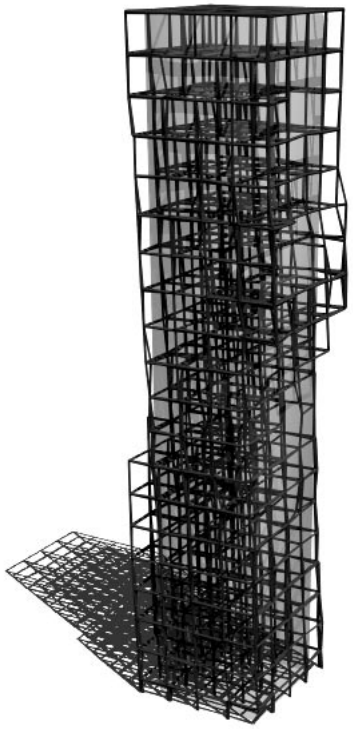


Figure 12 - Tower generated with an architectural design algorithm based on Conway's Game of Life.

6 FUTURE WORK

We intend to further extend the rudimentary explorations in Autonomic Architecture along several different trajectories.

6.1 Geometry

The use of various space-filling geometries in Conway Variant produced a range of interesting formal results, opening up several research opportunities. As one portion of our continued research we propose to investigate more rigorously the logic of these different spatial patterns and their formal range. We also hope to investigate these space-filling geometries combinatorially, testing applications of combinations of the geometries within a single structure to produce structures supportive of different programs and uses.

The association between ideas of structure and geometry in our case studies has remained primarily formal; in further investigations we plan to investigate "real-world" collaborative applications with structural engineers and architectural firms. There has been interest expressed within the community of practicing architects to utilize our current prototype for designing building components.

6.2 Fabrication

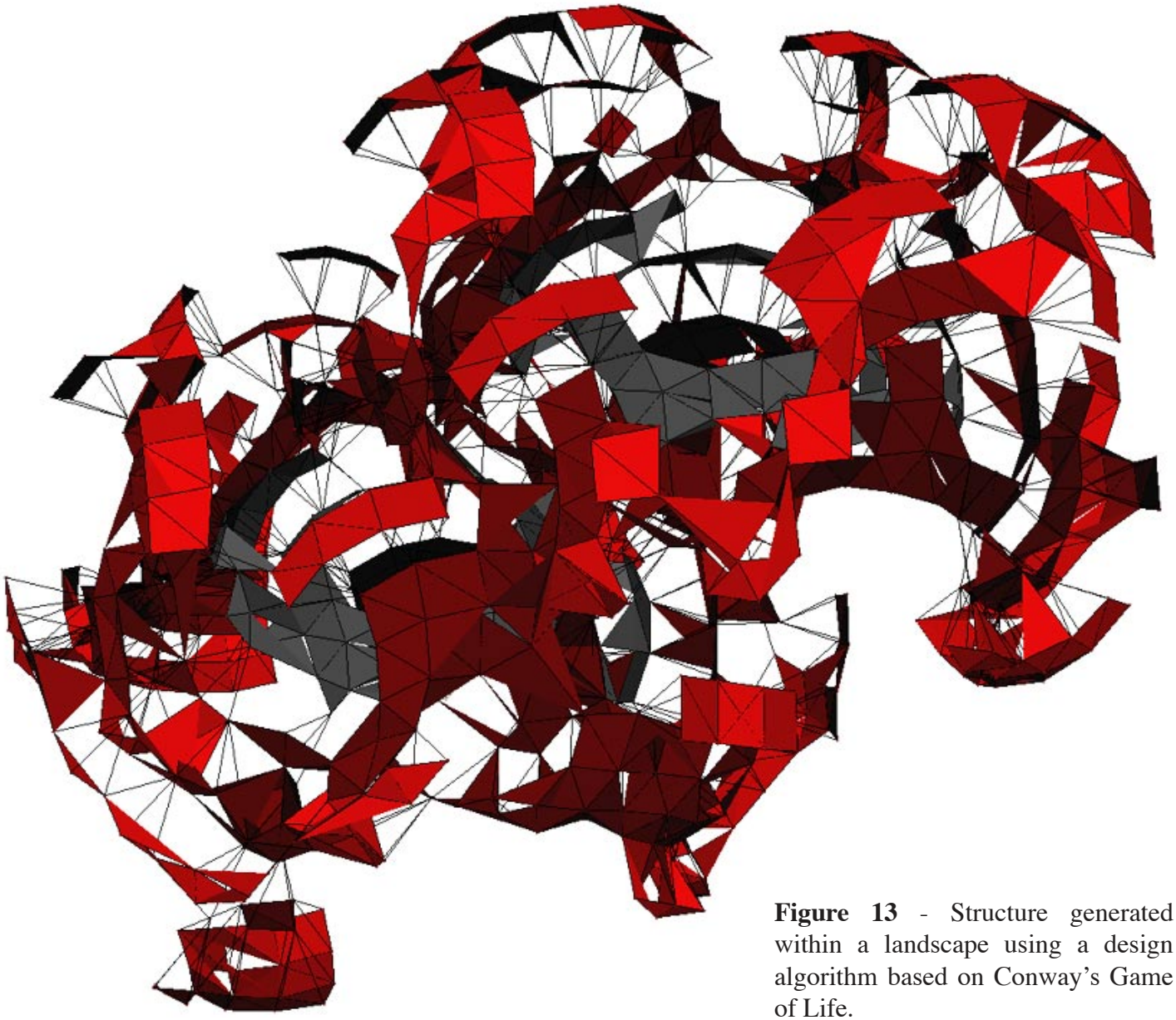


Figure 13 - Structure generated within a landscape using a design algorithm based on Conway's Game of Life.

The application of CASs in the manner we propose offers the ability to control the size and scale of every architectural element within a structure with mathematical precision, as it is designed. This degree of control would necessitate further research in the coupling of Computer Numerically Controlled fabrication techniques with the morphology of the architectural design system. As part of our continued research we intend to develop fabrication techniques for a differential space-frame truss that would allow the limitations of the fabrication and construction process to be added as control parameters to the design system.

6.3 Algorithms

The CASs explored in our case studies, one-dimensional Cellular Automata and Conway Variant, use some of the simplest of the CAS algorithms. In further research we intend to pursue additional CAS algorithms based on the following criteria:

The CAS systems we have employed so far use only two states within their organizational structure, other algorithms such as Turing Machines and 3-state Cellular Automata are capable of supporting multiple states. This multi-state structure could provide for additional behaviors relating to program, site, and aesthetics – factors that have not yet been addressed in case studies.

The systems used in the case studies thus far have had limited capacity for testing local conditions. For elements of both the one-dimensional Cellular Automata and the Conway Variant behaviors were based solely upon the state of each element's immediate neighbors. We propose to pursue methods of building behaviors based on an awareness of more than the immediate proximity, using methods similar to those developed for Neural Networks.

The environment and context of the CASs in the case studies have been described as fixed, not changing over time. To better simulate real-world applications of these systems, as well as to encourage evolution within the algorithms, we propose to pursue methods of constructing and testing these systems in continually varying environments.

BIBLIOGRAPHY

Books

Coxeter, H. S. M. Regular Polytopes. New York: Dover Publications, Inc., 1973.

Hybers, P. "The Polyhedral World" In Beyond the cube: The Architecture of Spaceframes and Polyhedra., edited by J. F. Gabriel. New York: John Wiley & Sons, Inc., 1997.

Neumann, J von and Arthur W. Burks. Theory of Self-reproducing Automata. Urbana, Ill: University of Illinois Press, 1966.

Nooshin, H., Disney, P. L., and Champion, O. C. "Computer-Aided Processing of Polyhedric Configurations" In Beyond the cube: The Architecture of Spaceframes and Polyhedra. edited by J. F. Gabriel. New York: John Wiley & Sons, Inc., 1997.

Pearce, Peter. Structure in nature is a Strategy for Design. Cambridge, MA: MIT Press, 1990.

Wuensche, Andrew and Mike Lesser. The Global Dynamics of Cellular Automata. New York: Addison-Welsey, 1992.

Wolfram, Stephen. A New Kind of Science. New York: Wolfram Media, Inc., 2002.

Articles

Gardner, Martin. " MATHEMATICAL GAMES: The fantastic combinations of John Conway's new solitaire game 'life'". Scientific American 223 (October 1970): 120-123.

Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model". Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) 25-34.

Testa, P. and O'Reilly, U. "MoSS: Morphogenic Surface Structures". Greenwich 2000 Conference Proceedings. London: Interscience Communications.