Architecture and Planning Journal (APJ)

Volume 28 Issue 3 *ASCAAD 2022 - Architecture in the Age of the Metaverse – Opportunities and Potentials*
ISSN: 2789-8547

Article 12

March 2023

PROGENY: A GRASSHOPPER PLUG-IN THAT AUGMENTS CELLULAR AUTOMATA ALGORITHMS FOR 3D FORM EXPLORATIONS

İNANÇ ŞENCAN
Istanbul Technical University, Turkey, sencani@it.edu.tr

DOI: https://doi.org/10.54729/2789-8547.1207

PROGENY: A GRASSHOPPER PLUG-IN THAT AUGMENTS CELLULAR AUTOMATA ALGORITHMS FOR 3D FORM EXPLORATIONS

Abstract

Cellular automata (CA) is a well-known computation method introduced by John von Neumann and Stanislaw Ulam in the 1940s. Since then, it has been studied in various fields such as computer science, biology, physics, chemistry, and art. The Classic CA algorithm is a calculation of a grid of cells' binary states based on neighboring cells and a set of rules. With the variation of these parameters, the CA algorithm has evolved into alternative versions such as 3D CA, Multiple neighborhood CA, Multiple rules CA, and Stochastic CA (Url-1). As a rule-based generative algorithm, CA has been used as a bottom-up design approach in the architectural design process in the search for form (Frazer,1995; Dinçer et al., 2014), in simulating the displacement of individuals in space, and in revealing complex relations at the urban scale (Güzelci, 2013). There are implementations of CA tools in 3D design software for designers as additional scripts or plug-ins. However, these often have limited ability to create customized CA algorithms by the designer. This study aims to create a customizable framework for 3D CA algorithms to be used in 3D form explorations by designers. Grasshopper3D, which is a visual scripting environment in Rhinoceros 3D, is used to implement the framework. The main difference between this work and the current Grasshopper3D plug-ins for CA simulation is the customizability and the real-time control of the framework. The parameters that allow the CA algorithm to be customized are; the initial state of the 3D grid, neighborhood conditions, cell states and rules. CA algorithms are created for each customizable parameter using the framework. Those algorithms are evaluated based on the ability to generate form. A voxel-based approach is used to generate geometry from the points created by the 3D cellular automata. In future, forms generated using this framework can be used as a form generating tool for digital environments.

Keywords

Cellular Automata, Grasshopper, Generative Design, Framework.

PROGENY
A GRASSHOPPER PLUG-IN THAT AUGMENTS CELLULAR AUTOMATA ALGORITHMS FOR 3D FORM EXPLORATIONS

İNANÇ ŞENCAN

Istanbul Technical University
sencani@itu.edu.tr

ABSTRACT

Cellular automata (CA) is a well-known computation method introduced by John von Neumann and Stanislaw Ulam in the 1940s. Since then, it has been studied in various fields such as computer science, biology, physics, chemistry, and art. The Classic CA algorithm is a calculation of a grid of cells' binary states based on neighboring cells and a set of rules. With the variation of these parameters, the CA algorithm has evolved into alternative versions such as 3D CA, Multiple neighborhood CA, Multiple rules CA, and Stochastic CA (Url-1). As a rule-based generative algorithm, CA has been used as a bottom-up design approach in the architectural design process in the search for form (Frazer,1995; Dinçer et al., 2014), in simulating the displacement of individuals in space, and in revealing complex relations at the urban scale (Güzelci, 2013). There are implementations of CA tools in 3D design software for designers as additional scripts or plug-ins. However, these often have limited ability to create customized CA algorithms by the designer. This study aims to create a customizable framework for 3D CA algorithms to be used in 3D form explorations by designers. Grasshopper3D, which is a visual scripting environment in Rhinoceros 3D, is used to implement the framework. The main difference between this work and the current Grasshopper3D plug-ins for CA simulation is the customizability and the real-time control of the framework. The parameters that allow the CA algorithm to be customized are the initial state of the 3D grid, neighbourhood conditions, cell states and rules. CA algorithms are created for each customizable parameter using the framework. Those algorithms are evaluated based on the ability to generate form. A voxel-based approach is used to generate geometry from the points created by the 3D cellular automata. In future, forms generated using this framework can be used as a form generating tool for digital environments.

Keywords: Cellular Automata, Grasshopper, Generative Design, Framework.

ملخص

الأتمتة الخلوية (CA) هي طريقة حوسبية معروفة قدمها جون فون نيومان وستانيسلاف أولام في الأربعينيات. منذ ذلك الحين، تمت دراسة الأتمتة الخلوية في مجالات مختلفة مثل علوم الحاسبات والأحياء والفيزياء والكيمياء والفن. تعد خوارزمية الأتمتة الخلوية الكلاسيكية عبارة عن عملية حسابية لشبكة من الحالات الثنائية للخلايا بناءً على الخلايا المجاورة ومجموعة من القواعد. ومع اختلاف هذه المتغيرات، تطورت خوارزمية الأتمتة الخلوية إلى إصدارات بديلة مثل الأتمتة الخلوية ثلاثية الأبعاد، والأتمتة الخلوية متعددة الجوار، والأتمتة الخلوية متعددة القواعد، والأتمتة الخلوية العشوائية. وقد تم استخدام الأتمتة الخلوية – كخوارزمية توليد قائمة على القواعد – كنهج تصميم من أسفل إلى أعلى في عملية التصميم المعماري في البحث عن الشكل، وفي محاكاة حركة الأفراد في الفراغ، وفي الكشف عن العلاقات المعقدة على المستوى الحضري. وتوجد تطبيقات لأدوات الأتمتة الخلوية في برامج التصميم ثلاثية الأبعاد للمصممين وذلك فى هيئة نصوص أو مكونات حوسبية إضافية. ومع ذلك، غالبًا ما يكون لهذه المكونات قدرة محدودة على إنشاء خوارزميات الأتمتة الخلوية القابلة للتخصيص بواسطة المصمم. تهدف هذه الدراسة إلى إنشاء إطار عمل قابل للتخصيص لخوارزميات الأتمتة الخلوية ثلاثية الأبعاد لاستخدامها في الاستكشافات ثلاثية الأبعاد من قبل المصممين. تم استخدام برنامج Grasshopper3D، وهو بيئة برمجة نصية مرئية في Rhinoceros 3D، لتنفيذ إطار العمل. ويكمن الاختلاف الرئيسي بين هذا العمل والمكونات الإضافية الحالية لـ Grasshopper3D لمحاكاة الأتمتة الخلوية في إمكانية التخصيص والتحكم في إطار العمل بطريقة متزامنة. وتعتبر المتغيرات التي تسمح بتخصيص خوارزمية الأتمتة الخلوية هي: الحالة الأولية للشبكة ثلاثية الأبعاد وظروف الجوار وحالات الخلية وقواعد الأتمتة. تم إنشاء خوارزميات الأتمتة الخلوية لكل متغير قابل للتخصيص باستخدام إطار العمل. وتم تقييم تلك الخوارزميات بناءً على القدرة على توليد الشكل. وتم استخدام نهج يعتمد على voxel لإنشاء هندسة شكل عن طريق النقاط التي أنشأتها الأتمتة الخلوية ثلاثية الأبعاد. ويمكن مستقبلا استخدام الأشكال التي تم توليدها باستخدام هذا الإطار كأداة لتوليد الأشكال والنماذج للبيئات الرقمية.

الكلمات المفتاحية: الأتمتة الخلوية، برنامج جراسهوبر، التصميم التوليدى، إطار عمل.

1. INTRODUCTION

The word "design" can describe the process of designing an object and the object that is the result of this process. According to Dino (2012), a generative system is not a production system that defines the design product but a higher-level definition, or design process, that encodes the "making" phase of the design product. Therefore, generative systems allow the designer to "design a design tool" as an alternative to the task of designing an object. In this study, a design tool that uses Cellular Automata as a generative system has been produced. The contextual relationship with the concept of "rhythm" has been examined.

2. BACKGROUND

The theme of the work is "Rhythm". The difficulty in researching the concept of rhythm is the lack of a generally accepted, clear definition (Fraisse, 1982). Rhythm comes from the Greek words “ρυθμός” (rythmós) and “ρέω” (réo)(to flow) (Fraisse, 1982). Rhythm, which is defined as repetitive movements and symmetry (Liddell and Scott, 1996), is characterized as "the movement defined by the organized repetition of strong and weak elements or by opposing or different conditions" (Anon, 1971).

The most common use of the concept of rhythm is music. Additionally, in linguistics, the development of languages and their interaction with each other are used to describe situations. The word rhythm is also used outside of these contexts in the language we use daily. The concept of "Rhythm" is confused with "Tempo" in cases where a person defines the rhythm of his activities during the day, or the rhythm of a city is explained. Tempo describes the speed of a rhythm. In this work, while tempos are compared with each other in terms of speed, rhythms are compared with each other structurally.

2.1. Rhythm / Form / Form Finding

According to Benveniste (1951), the semantic connection of the words rhythm and flow was not inspired by the repetitive motion of the waves, as is supposed. "Rhytmos", which appears as one of the keywords of Ionian philosophy, is defined as a more developed, instantaneous, and variable "form" with a generalized meaning of "form" (Fraisse, 1982).

Form finding is one of the critical components of the design process. Different definitions have been made over time for the concept of form-finding. Haber and Abel (1982) define the form-finding problem as the "initial balance problem" (Veenendaal & Block, 2012). This definition has been further discussed later on. According to Lewis (2003), form finding is "the search for the optimal shape of a form-active structure at or near equilibrium". This definition has been accepted and used by most people. Looking at more recent definitions, the definition used by Coenenders and Bosnia (2016) is "finding the appropriate architectural and structural form". Basov e Del Grosso (2011) states that shape finding is "a structural optimization system that uses point coordinates and variables".

Form production methods have been applied primarily in physical and digital environments with the development of technology. Antoni Gaudi's experiments with hanging chains, Heinz Isler's experiments with hanging various membranes, and Frei Otto's experiments with soap bubbles can be shown as examples of form-seeking experiments in the physical environment (Veenendaal, D., & Block, P., 2012). With the development of digital tools, the variety of form-finding methods has increased. Just as gravity is the form-determining force in Gaudi's experiment by hanging chains, the laws of physics, the growth and development mechanisms of living things, and mathematical equations, such phenomena were recreated in the virtual environment and used in form-finding experiments in the digital environment. Cellular Automata (CA) is one such method. Within the scope of this study, CA methods are discussed to find form. The working logic of different CA algorithms has been examined, reproduced, and diversified. As a result, a design tool based on the CA algorithm has emerged.

2.2. Cellular Automata

Cellular Automata is a subject studied in fields such as computer science, mathematics, physics, and theoretical biology. It was also considered a productive system and a research topic during the design process. The cellular automata approach allows the simulation of neighborhood relations, in the field of architecture, both in the search for form (Frazer, 1995; Dinçer et al., 2014), in simulating the displacement of individuals in space, and in revealing complex relationships at the urban scale (Guzelci, 2013). It has been the subject of practice and debate in the fields. It is the change of different viability states of cells in a set based on rules depending on the relationship of cells with other cells around them. 1, 2, 3, or more sizes of CA can be defined (Adamatzky, 2010). Although the first studies on CA were made in the 1950s, it was not at a level that would attract the attention of a broad audience until Conway's CA application named "Game of Life" (GoL) was published in the "Scientific America" magazine in 1970 (Adamatzky, 2010).

In this application, some cells can be found in "alive" or "dead" states on the grid formed by a two-dimensional matrix. For each cell, the cells which are interacting with it are called “neighbor cells”, and the definition of this interaction is called “neighborhood”. The "Moore" neighborhood format defined in the GoL application consists of 8 cells located 1 unit away from the cell. A "Generation" is the set of states of all cells at a given moment. The state of each cell in the next generation is determined by the living or dead state of the neighboring cells, according to the rules. According to the GoL application, a living cell can continue to live in the next generation if there are 2 or 3 cells in its neighborhood. Otherwise, it goes dead. A dead cell becomes alive in the next generation if three living cells are in its neighborhood. The system develops step by step depending on these rules and produces new geometries. The gradual intergenerational progression of the form generation process in CA can be described as rhythmic growth. In addition, the contrast of vitality and deadness gives a contextual reference to the definition of rhythm. Form generation methods can be diversified by differentiating the neighborhoods, rules, and living-dead situations, which are the basic features of CA.

2.2.1. Neighborhoods

In a CA algorithm, the set of surrounding cells used to determine whether a cell complies with the rule that will determine its status in the next generation is called "neighboring cells". These cells are usually located near the parent cell. As seen in Figure 2, Von Neumann and Moore's neighborhood can be shown as an example of neighborhood forms in 2D CA. Von Neumann neighborhood considers cells where the cell is adjacent in the horizontal and vertical directions (Adamatzky, 2010). On the other hand, Moore's neighborhood covers eight neighboring cells by taking the nearest cell in the diagonal direction and adjacent cells in the horizontal and vertical directions (Adamatzky, 2010).

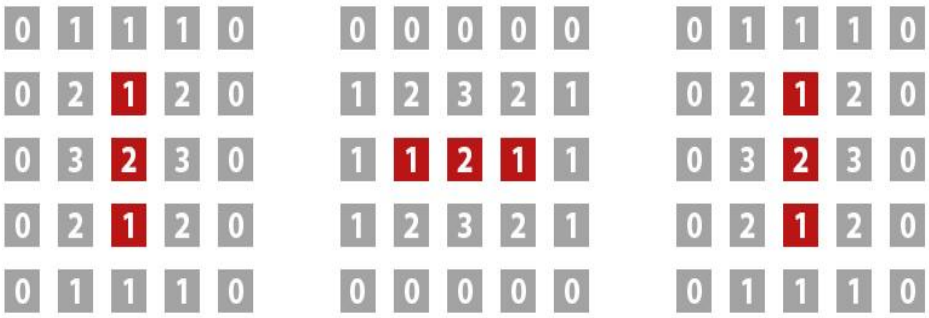

Fig.1: CA calculation between generations.

2.2.2. Generations

In the CA algorithm, each cell has a specific state. Whether this situation will change in each new generation is calculated according to the rule defined in the algorithm. For example, in GoL, cells are either "alive" or "dead". Various patterns are obtained by showing live and dead cells in different colors (live = black, dead = white) (Figure 1).

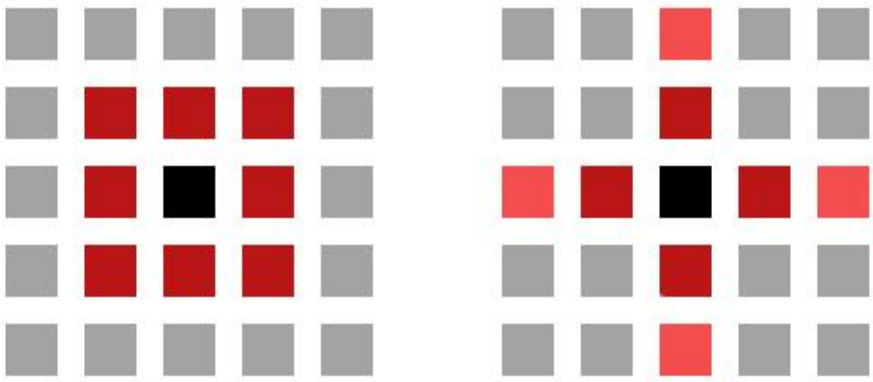

Fig.2: Neighbor cells in different neighborhood types.

2.2.3. Rules

In the CA algorithm, each cell has a specific state. Whether this situation will change in each new generation is calculated according to the rule defined in the algorithm (Figure 3). For example, in GoL, cells are either "alive" or "dead". Various patterns are obtained by showing live and dead cells in different colors (live = black, dead = white).

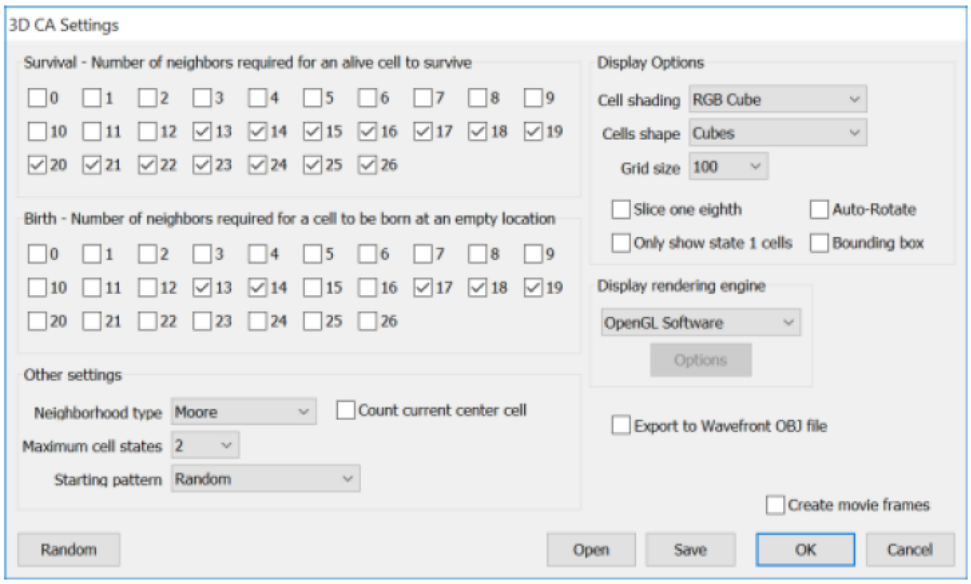

Fig.3: Rules for 3D Cellular Automata (Visions of Chaos).

3. RESEARCH BACKGROUND

The first studies of the Cellular Automata algorithm were performed without computer support by drawing each generation separately, moving objects on a grid. There are tried-and-tested examples available, such as Rabbit (Url-2) and Vision of Chaos (Url-1). However, these methods are slow and limited. With the development of computer technology, the number and quality of studies in this field have increased by calculations being made by computers.

Today, running the CA algorithm in most programming interfaces with visual output is possible. While it is possible to see the outputs of only specific rules in some more advanced applications, it is possible to define rules and print out a 3D model besides the visual output. Additionally, there are programs that allow multiple dimensional CA algorithms, different neighborhoods, and different cell states are numbered and inserted in the text after the first reference to it.

3.1. Processing: Game of Life

In this study, first, the "Game of Life" application, which has an essential role in the spread of CA applications, was examined, and the working principles of the CA algorithm were studied. In the "Processing" (Url-3) programming interface, the GoL application was reproduced using the "Java" programming language (Figure 4). This study is vital in learning the CA algorithm's features to be defined in computer language.

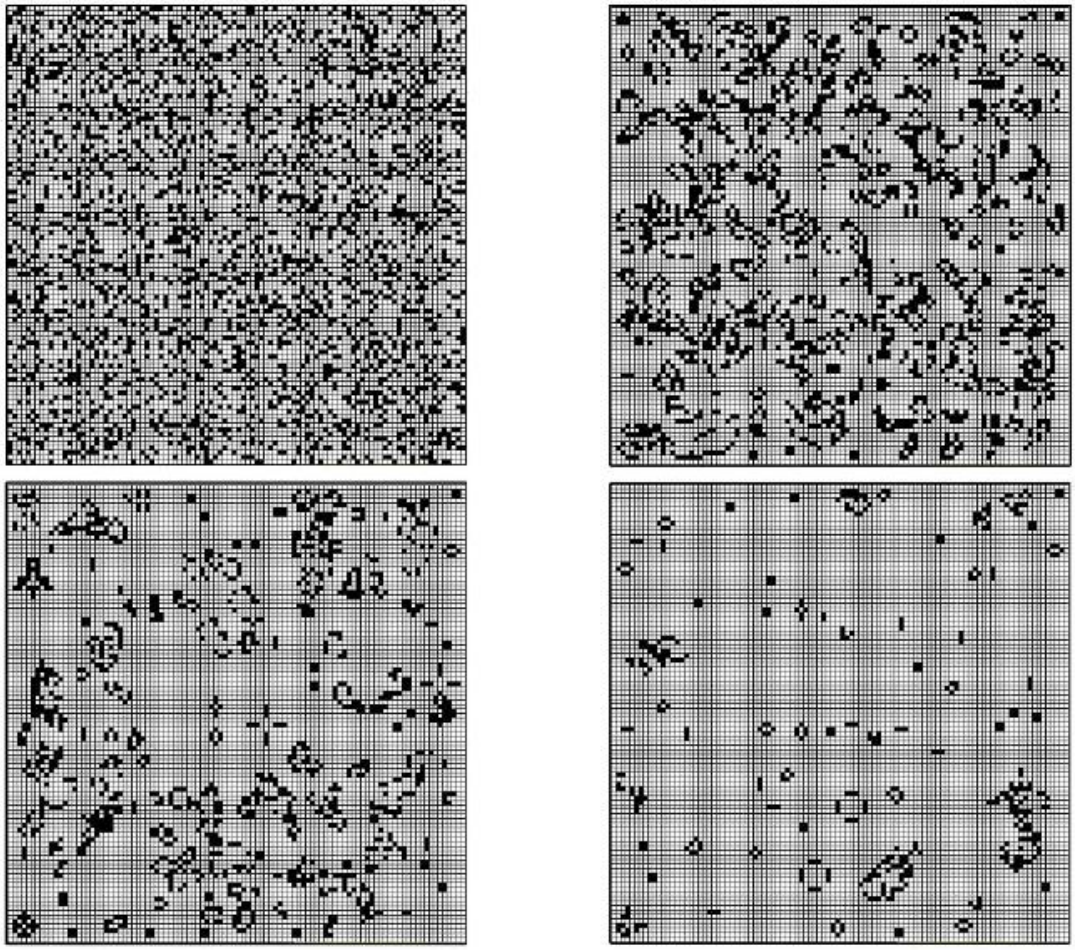

Fig.4: A screenshot from “Game of Life” algorithm in Processing.

3.2. Vısıons of Chaos

This application (Url-1) runs algorithms such as Fractals, L-systems, multi-agent systems, Cellular Automata, and Diffusion Limited Aggregation. It is a versatile program that can run various generative algorithms (Figure 5). In addition to 1D, 2D, 3D, and 4D applications of CA, there are methods in which different neighborhoods, different vitality states, and different rules can be applied. There is a user interface where these features can be defined. While it is easy to use, 3D models and video output are its positive features. Its negative feature is that the algorithm has a limited choice of a sphere, cube, or a filled environment as the initial form. This disadvantage makes it difficult to use the program as a design tool. In the work to be done, it is aimed to produce an interface that allows the user to determine not only the rules and CA type but also the initial state, and in this way, get rid of randomness and make his design.

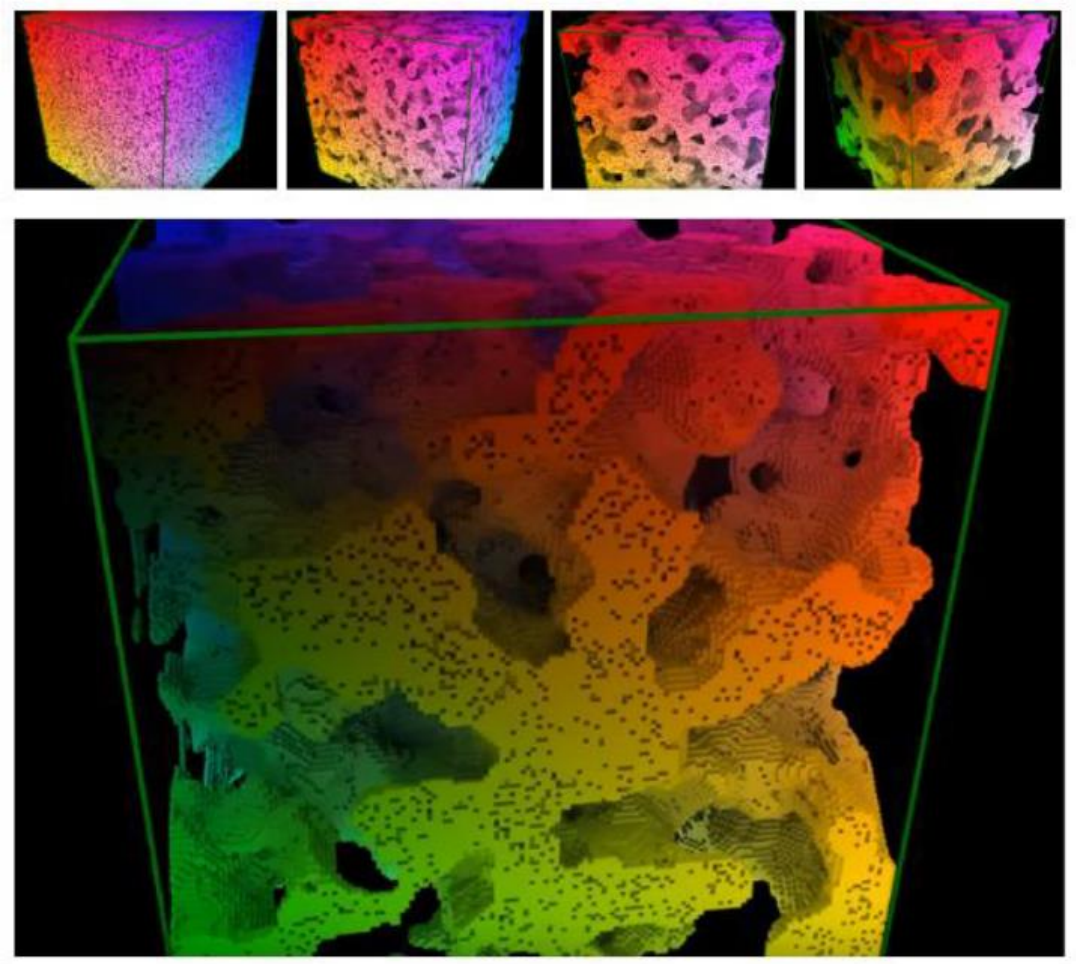

Fig.5: Screenshots from Visions of Chaos with 3D Cellular Automata. (Url-1)

3.3. Rabbıt (Grasshopper Plug-In)

It is included in the Grasshopper3D, which is a plugin of the Rhinoceros3D program (Url-2). Running the CA algorithm inside a design program means better user control when using this algorithm for design purposes. In addition to graphically defining the initial form of the algorithm, it is also possible to intervene in the environment in real-time. Rabbit plugin includes 2D CA algorithm and 3D CA algorithms Stacked Cellular Automata (SCA - Stacked Cellular Automata). SCA is 3-dimensional geometry obtained by superimposing each generation of 2-dimensional CA. Rabbit is a good example of using 2D CA and its 3D extended version SCA as a design tool. However, this method is insufficient because it does not contain other varieties of CA. This study aims to create a Grasshopper3D plugin in which different methods of CA can be defined.

3.4. Comparison

The purpose of trying these three programs before writing a script for 3D CA algorithm in Grasshopper is to learn how the algorithm works in multiple environments and conditions. While writing a 2D CA algorithm in Processing was easier than the 3D version, its aid to the research was the understanding of how the algorithm works.

"Visions of Chaos" (VoC) helped visualizing 3D cellular automata and how the parameters of the algorithm worked. However, its downsides are also revealed while using it, like not being able to control the initial shape. Using Grasshopper to simulate 3D CA algorithm is thought to solve these problems. Rabbit, which is a Grasshopper plug-in, can solve 3D CA problems. However, this 3D interaction is 2,5D, because of how the plug-in works. As a conclusion, a 3D CA algorithm with the working system of VoC in Grasshopper is written.

4. 3D CELLULAR AUTOMATA PLUG-IN AS A C# SCRIPT IN GRASSHOPPER

4.1. The Algorithm

The algorithm is defined as a Grasshoper3D plug-in (Figure 6). The plug-in written in C# scripting language. The algorithm is planned to have user-defined parameters. These parameters are:

Rules: Each rule defines a new format generation system in CA algorithms. While some rule sets expand from a minor point, some rule sets can result in an entire format diminishing to an optimum state. These rules can be discovered and serve different purposes in design.

Neighborhood: Changing the definition of neighboring cells provides diversity in the development of the system.

Cell states: At the basic level, there are two alternative states of cells in CA, live and dead. The transition of a cell from a live state to a dead state occurs in a generational change. Alternatively, the viability state of a cell can be defined by a number. For example, a cell is dead when it is "0" and alive when it is "1". If the maximum number of states of CA is 4, cells can be found in states "0,1,2,3". When a living cell in the "1" state dies, it first changes to the "2" state, then to the "3" state, then to the "0" state. It can switch from a "dead" state to an "alive" state only when it is "0". In the "2" or "3" state, it is considered dead in the calculations of neighboring cells. Another example can be taken Rock-Paper-Scissors CA. In this case, the rules between cells change according to the type of neighbor cells.

Initial conditions: In classic CA applications, the startup format is usually defined. The difference in shape finding comes from the random differentiation of the states of the cells that define the initial shape. However, this randomness must be controllable for a designer.

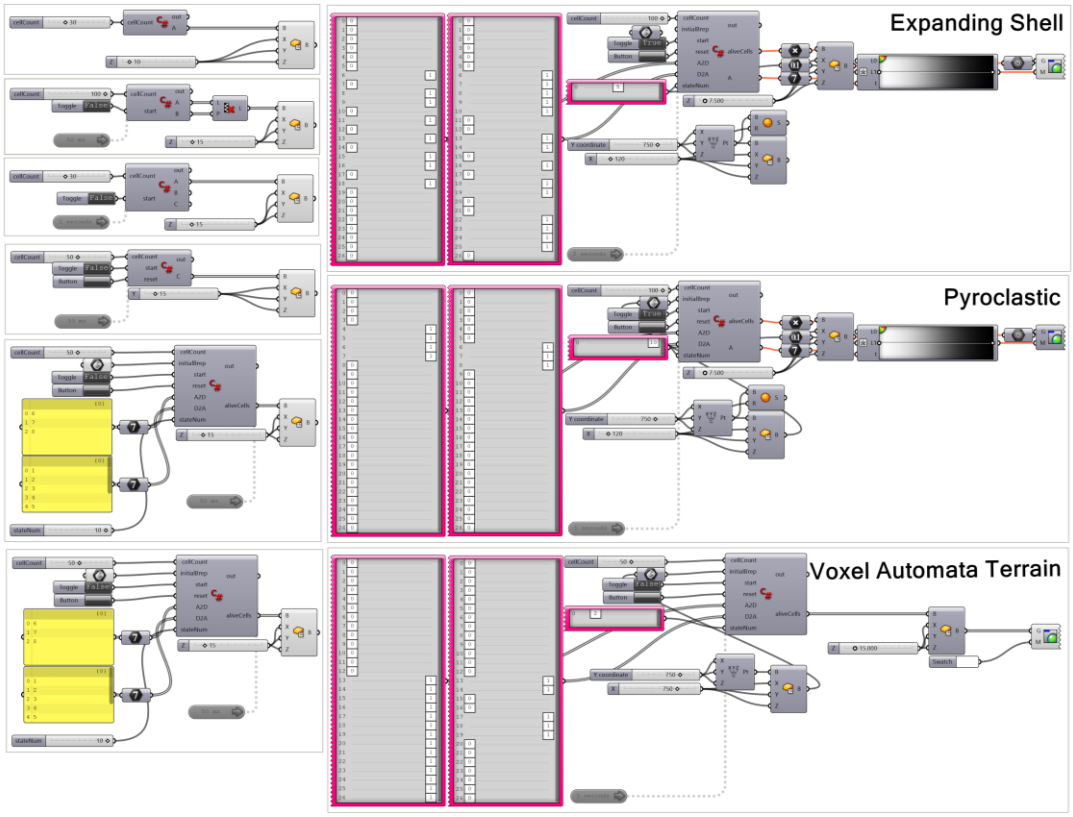

Fig.6: 3D CA algorithms using the plug-in; Expanding Shell, Pyroclastic, Cloud.

4.2. Usıng The Algorıthm

The presented 3D CA plug-in can create multiple generative systems, based on the variation in initial geometry, rules and model formation. The plug-in is tested with four different algorithms, which are named Expanding Shell, Pyroclastic, Cloud, and Voxel Terrain Automata.

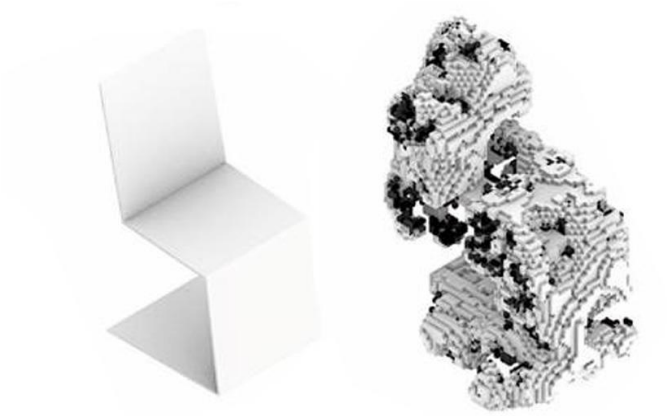

Fig.7: Initial condition (Left), generated model using 3D CA (Right).

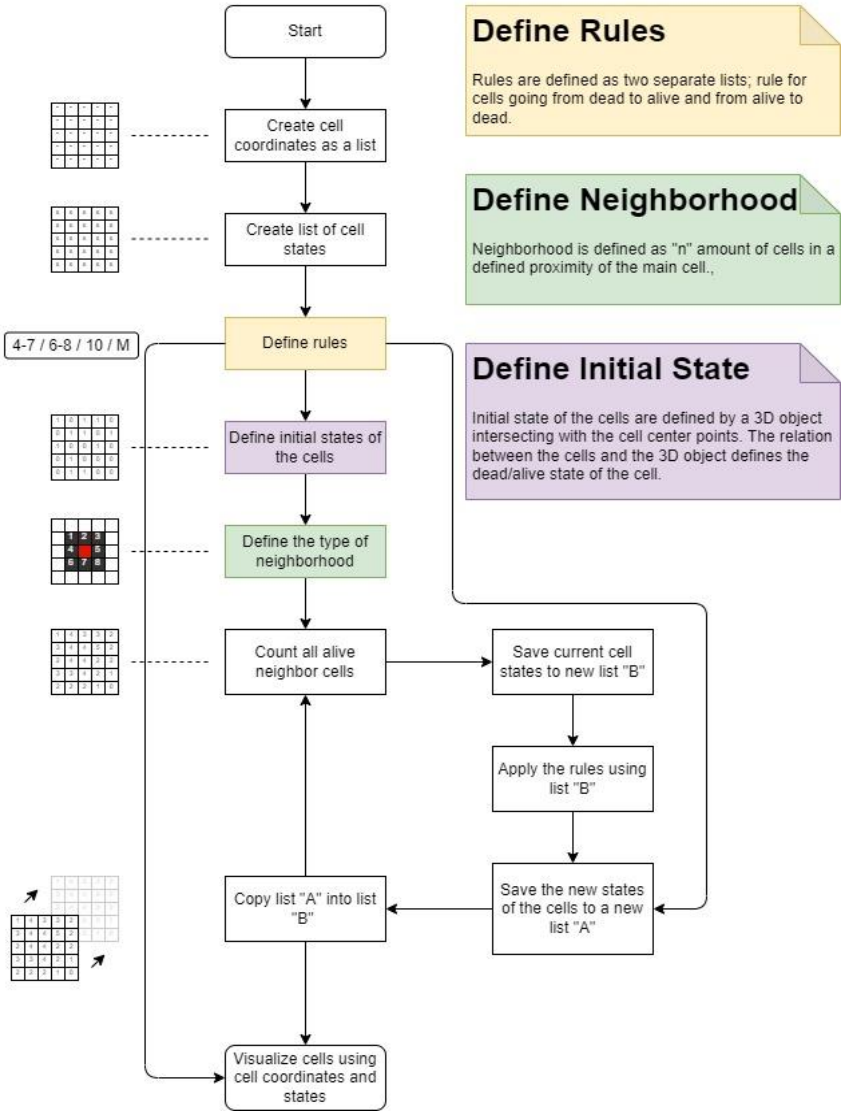

Fig.8: 3D CA algorithm flow chart.

Fig.9: Resulting 3D objects. From left to right, Expanding Shell, Cloud, Pyroclastic, Voxel Terrain Automata.

5. CONCLUSION AND FUTURE POSSIBILITIES

With the presented Grasshopper3D plug-in "proGeny", new productive systems can be discovered by experimenting with the rules of various cellular automata algorithms, their neighborhoods, and the viability of cells. With these systems, new form finding experiments can be done which result in emerging forms.

The aspects of this program that are open to development can be listed under the following main headings:

Visual interface: Although Grasshopper's visual interface is sufficient to start these experiments, a more user-friendly interface can be added using C#.

User interaction: In Grasshopper3D, interaction with the computer, keyboard, and mouse is defined. However, other Grasshopper3D plugins have examples (Firefly, Quokka) like Kinect or LeapMotion. User interaction can reach a different dimension through motion sensors, Arduino, and various sensors.

AR/VR applications: Visual outputs of the products obtained with the developed program can be taken on the computer screen. As a design tool, AR/VR applications can help experience this design process in the same environment as other factors affecting design.

In today's world, where digital design has become widespread, it is seen that the role of the designer is gradually replaced by the role of "designer of tools" or "designer of systems". Within the scope of this study, a tool called "progeny" that the designer can use in the design process has been developed. Using the program, the user can generate a design system on various versions of the CA with the rules, initial format, and neighborhood parameters that he has determined and can use this system in the design process. Since it is an application with development potential, it can provide opportunities for new research topics.

REFERENCES

- ADAMATZKY, A. (2010). Game of life cellular automata (Vol. 1). London: Springer. Url: https://link.springer.com/content/pdf/10.1007/978-1-84996-217-9.pdf Erişim Tarihi: 27.05.2019)
- ANON. The Compact Edition of the Oxford English Dictionary II. Oxford and New York: Oxford University Press, 1971.
- BASSO, P., & DEL GROSSO, A. (2011). Form-finding methods for structural frameworks: a review. Proceedings of the International Association of Shells and Spatial Structures, London.
- BENVENISTE, E. La notion de "rythme" dans son expression linguistique. Journal de Psychologie Normale et Pathologique, 1915, 44, 401-411.
- BRENNAN, A., ALHADIDI, S., KIMM, G. (2013). Quokka: Programming for Real Time Digital Design Platform. International Conference on Computer-Aided Architectural Design Research in Asia, 18, pp. 261-270, Hong Kong.

- COENDERS, J., & BOSIA, D. (2006). Computational tools for design and engineering of complex geometrical structures: From a theoretical and a practical point of view. Game Set and Match II. On Computer Games, Advanced Geometries, and Digital Technologies. Episode Publishers, 006.
- DINÇER, A. E., ÇAĞDAŞ, G., & TONG, H. (2014). A computational model for mass housing design as a decision-support tool. Procedia Environmental Sciences, 22, 270-279.
- DINO, I. (2012). Creative design exploration by parametric generative systems in architecture. METU Journal of Faculty of Architecture, 29(1), 207-224.
- FRAISSE, P. (1982). Rhythm and tempo. The psychology of music, 1, 149-180.
- FRAZER, J. (1995). An evolutionary architecture.
- GÜZELCI, O. (2013), Bütünleşik Üretken Tasarım Sistemi ile MVRDV Silodam Projesi İçin Cephe Üretken Sistem Önerisi, E.Gürer, S.Alaçam, Z.Bacınoğlu (Ed.), VII. Mimarlıkta Sayısal Tasarım Ulusal Sempozyumu: Sayısal Tasarım, Entropi, Yaratıcılık, 2013, Istanbul.
- HABER, R. B., & Abel, J. F. (1982). Initial equilibrium solution methods for cable reinforced membranes part I—formulations. Computer Methods in Applied Mechanics and Engineering, 30(3), 263-284.
- LEWIS, W. J. (2003). Tension structures: form and behaviour. Thomas Telford.
- LIDDELL, HENRY GEORGE, AND ROBERT SCOTT. "ῥυθμός", in A Greek-English Lexicon, revised edition, combining the text of the ninth edition with an extensively revised and expanded Supplement. Oxford and New York: Oxford University Press, 1996. Online, Perseus Project.
- VEENENDAAL, D., & BLOCK, P. (2012). An overview and comparison of structural form finding methods for general networks. International Journal of Solids and Structures, 49(26), 3741-3753.
- WOLFRAM, S. (1984). Universality and complexity in cellular automata. Physica D: Nonlinear Phenomena, 10(1-2), 1-35.
- URL-1: https://softologyblog.wordpress.com/category/cellular-automata-2/
- URL -2: https://parametrichouse.com/rabbit/
- URL -3: https://processing.org/