American Journal of Advanced Research, 2024, 8–1
July. 2024, pages 01-06
doi: 10.5281/zenodo.12210679
http://www.ibii-us.org/Journals/AJAR/
ISBN 2572-8849 (Online), 2572-8830 (Print)

Automata Architect – a Game for Pattern Play

Javier Escobedo[1], Ling Xu[2,*]

[1,2] Department of Computer Science and Engineering Technology, University of Houston – Downtown, USA.

*Email: xul@uhd.edu

Received on 02/22/2024; revised on 06/20/2024; published on 06/21/2024

Abstract

This paper introduces our project for developing a serious computer game for cellular automaton patterns. Cellular automata as models of complexity have been studied by researchers in a wide variety of subjects including mathematics, physics, biology, and computer science. In this project we focus on the patterns from one-dimensional cellular automata and intend to construct a platform for game play that offers players experience for exploring abstract aesthetic patterns. At the same time, the game can also be used as a simulator for cellular automaton patterns by manipulating the rules and settings.

Keywords: Game Development, Cellular Automata, Procedural Textures, Serious Game

1 Introduction

With the growth of video game industries, computer game development has attracted attention and interest from students and educators in high education institutions. For the preparation of future professionals, many universities have integrated computer game development courses in Computer Science curriculum [14, 15, 22]. Most common commercial game genres, such as real-time strategy (RTS), first-person shooter (FPS), role-playing (RPG), action, adventure, and simulation games [2], serve for entertainment purposes. Due to their popularity, they are also commonly selected topics for student course/research projects. However, the role of games as a knowledge bearer is often ignored [7]. As Adams [1] pointed out, "games must seek to do more than provide fun". For certain topic-specific content such as educational applications and scientific simulations, commercial games may not be appropriate [16], whereas serious games can blend entertainment and learning and maximize player motivation and engagement [5]. Serious games usually refer to games "used for training, advertising, simulation, or education that are designed to run on personal computers or video game consoles" [20]. Figure 1 shows two examples of serious games, for kids' time learning and medical simulations, respectively. We are interested in serious games for game development education and practice for a few reasons. First, in this project we expect to apply the knowledge from game classes to construct a game prototype. It does not have to contain realistic visual effects or sophisticated designs for stories and challenges but focus on the implementation of a basic game interface and core mechanics. The complexity and scale of the development work should be feasible for undergraduate students. Second,

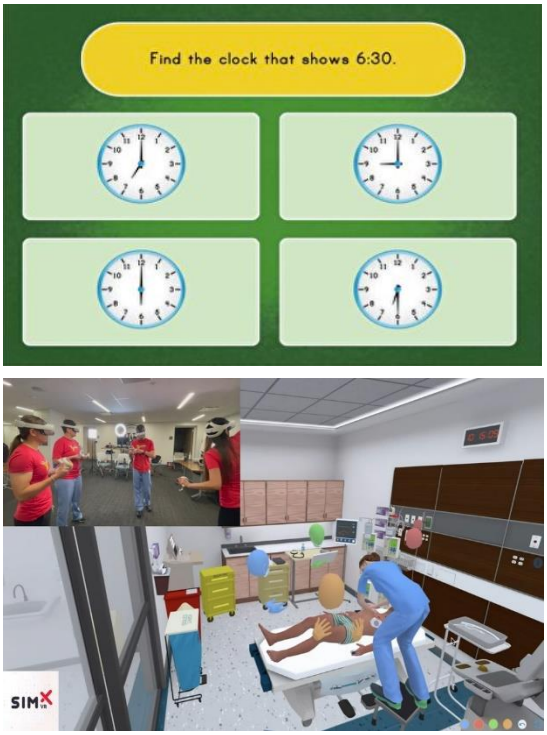

Figure 1. Examples of serious games. Upper: a game [21] for kids to learn reading clock and telling time. Lower: SimX [18, 19], a VR simulator for emergency care (SimWars).

we want to invoke the thoughts and explorations of applications for serious games – as computer science practitioners, what can we contribute to education communities? Last but not least, it would be beneficial to integrate the crossdisciplinarity subjects in a project to encourage students to think deeply and critically beyond the intrinsic circle of computer science.

In this project, we developed a serious game that focuses on abstract art and educational applications. The algorithm is based on the mechanism of cellular automaton (CA). A cellular automaton is a dynamic model governed by mathematical rules and algorithms that can be used to simulate a wide range of complex phenomena [23, 24]. Here we focus on the abstract aesthetic patterns created from cellular automata. The game offers two game play modes, for level challenges and free sandbox pattern play. The former challenges the player's skills for matching the rules with the given patterns. The later provides control handles to the player for exploring procedural settings and creating patterns freely.

Our work makes the following contributions. First, we provide an interactive platform for players to explore procedurally generated abstract aesthetic patterns. It can help the player for learning the topics in related areas including mathematics, computer science, and abstract arts. At the same time, our game is also a simulator for cellular automata. Compared with existing tools it is lightweighted without parameter specifications. The pattern matching features and functions for hints make the rules straightforward to understand. In addition, through our introduction to the design and development of this game, we expect our experience can be useful for CS students, educators, and game developers, and invoke more attentions and interest about serious games in computer science education. The rest of the paper is organized as follows. Following the introduction, the section of Background will introduce procedurally generated pattens and cellular automata. Next, we introduce the details of the game design and implementation. At the end we conclude the work and propose the future tasks.

2 Background

This section reviews the background of procedural methods for pattern generation. We especially focus on the mechanism of cellular automata.

2.1 Procedural Textures

Textures and pattern features play an important role in abstract art due to their influence on the human perception and emotions [9, 17]. How to generate textures (and patterns) has been an active research topic in the areas of computer graphics. Existing methods can be generally classified into three categories: example-based, simulation-based, and procedural ones [10]. Example-based methods require existing texture samples to synthesize new ones usually in larger scales. Simulation-based methods intend to simulate the patterns from natural phenomena such as crystal textures based on their forming processes and mechanisms. Procedural methods [12, 26, 27] use parameter-controllable algorithms or models to generate textures. The results can be controlled and varied by manipulating the parameters. Compared with other methods, procedural methods do not require texture samples or priori knowledge on the formation of natural patterns, but basic understanding of rules and parameter settings. They usually provide flexibility and convenience for generating a wide variety of patterns with low cost for time and human labor.

According to human perception of the procedural textures, procedural methods can be further classified into a few categories for generating structured textures, unstructured ones, and those needing filtering and post processing [6]. In this project we are more interested in structured textures with appearance of regularity from the method of cellular automata, for the regular structures used in the game for pattern play, and the target of a serious game for learning and practicing cellular automaton rules.

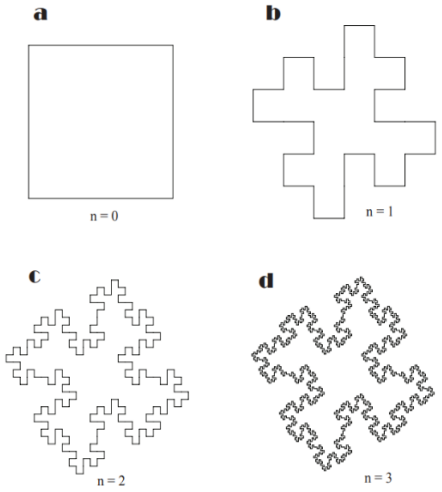

Figure 2. Starting from a square (a), a complicated pattern (d) can be generated after 3 replacement iterations. Image from [11].

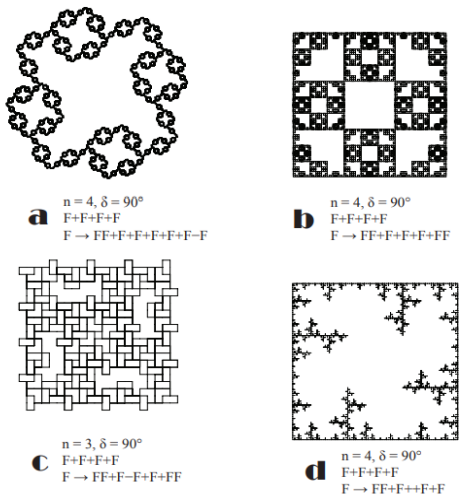

Figure 3. Fractal patterns from L-systems. Image from [11].

In addition to cellular automata, there are other procedural methods for generating structured textures, such as L-systems. L-system is a string rewriting system. The basic idea is to define a complex object from a simple initial object by applying the replacement rules for a few iterations [4, 11, 20]. For example, as shown in Figure 2, the process starts from a square defined by a string F + F + F + F, where F represents the action of moving forward a step of length (i.e., drawing an edge), and + represents turning right by δ = 90°. In each later iteration, we can apply the replacement rule

F → F + F − F − F F + F + F – F. Finally, a complex pattern is generated after three iterations. Different patterns can be generated given different initial strings and replacement rules (as shown in Figure 3).

2.2 Cellular Automata

A cellular automaton is a discrete model that contains a grid of cells, in which each cell evolves through a few time steps according to simple local rules based on the states of other cells in its neighborhood [24]. The simplest form of grid is a one-dimensional line, though higher dimensions are possible for cellular automata. In the one-dimensional form, also called elementary cellular automata, each cell has two possible values (0 or 1), and its state in the next generation after a time step is decided by a rule with inputs of its current state and the neighboring cells' states. Different rules define how the neighborhood affects the updated state of a cell. For example, in Figure 4, rule 30 tells eight possible states that a cell (at the center) may evolve in case of eight states of its neighborhood (i.e., its own state, and the states of its left and right cells). To be more specific, for instance, the leftmost scenario shows a cell at the center in black (or 1) with two black neighbors (at the top row) will evolve to a white (or 0) cell (at the lower row) in the next generation. Based on this rule, if we display the next generation of cells below the prior generation, we can get a matrix of grids that show the states (black or white) of each cell, forming some interesting patterns. Figure 4 shows the initial grids with a single black cell at the center (in the first row) and the evolution in 15 steps (from row 2 to row 16).

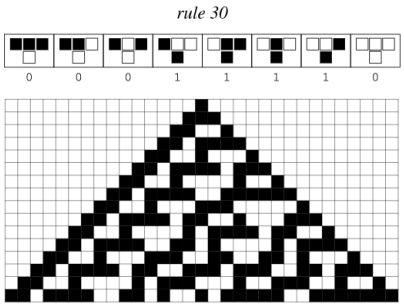

Figure 4. Rule 30 and its pattern. Image from [24] and [25].

Since different rules can generate different patterns (as shown in Figure 5), how to relate a rule with its corresponding resulting pattern is a challenge – this is also the challenge for our game players for their exploration and fun.

3 Game Design and Implementation

3.1 Game Overview

Automata Architect is a puzzle-simulator game in which the player must manipulate automata rules on a grid-based cellular automaton to achieve a specified result. The game uses elementary cellular automata rules to determine how the cells evolve with each step. The game offers varying levels of difficulty, a sandbox mode, and an in-depth help system for educating and assisting players throughout the game.

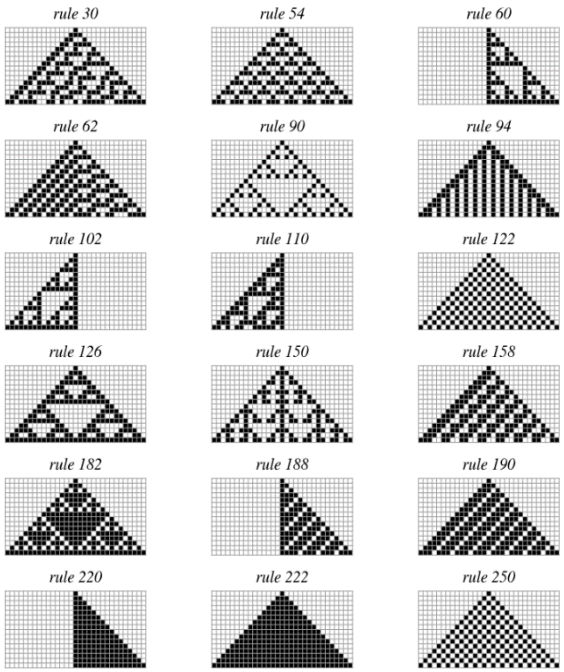

Figure 5. Different rules and their patterns. Image from [24] and [25].

The game falls under the puzzle, simulator, and educational genres. It is targeted toward players who enjoy logic-based puzzles and have an interest in learning about computational theory and cellular automata. The game caters to both casual and dedicated players, but specifically to those who are willing to trudge through a sizable learning curve. The game challenges players to solve complex grid-based problems. It provides a simulation environment to study automata patterns, all while ideally teaching players about computational theory and cellular automata.

Automata Architect has a minimalist and visually appealing design that focuses on grid-based cellular automaton. The game interface is clean and straightforward, with easy-to-understand icons and buttons. The visual style of the game maintains a balance between simplicity and complexity, allowing players to focus on the intricate patterns and mechanisms involved in solving puzzles or generating odd patterns while still immersing them in a visually engaging environment. The animations of the cellular automaton running are smooth and mesmerizing, encouraging players to experiment and observe the emerging patterns. The overall feel of the game is intellectually stimulating and rewarding, as players progress through increasingly challenging levels or encounter something unexpected, all while gaining a deeper understanding of the concepts behind cellular automata and computational theory.

We use Processing for the development tool. Processing is an open-source computer language and integrated development environment [13]. We selected Processing for the benefits mentioned by Huang et al. [27]. As a popular tool "used by students, artists, designers, architects, and researchers for learning, prototyping, and production" [13], the learning curve for Processing is smoother compared with others such as XNA and Unity. Since our game focuses on greyscale geometric patterns, we prefer simpler and shorter code to render images in Processing. In addition,

the source code for our game is straightforward for reading and modifications in the IDE of Processing, making future augmentations and changes convenient.

Automata Architect benefits from its simplicity and doesn't require a high-end setup, which makes it accessible to a wide range of players. The game is designed to run smoothly on most current-generation personal computer hardware. It doesn't require a cutting-edge GPU or an overly powerful system. Instead, it focuses on delivering enjoyable gameplay, even on mid-range systems.

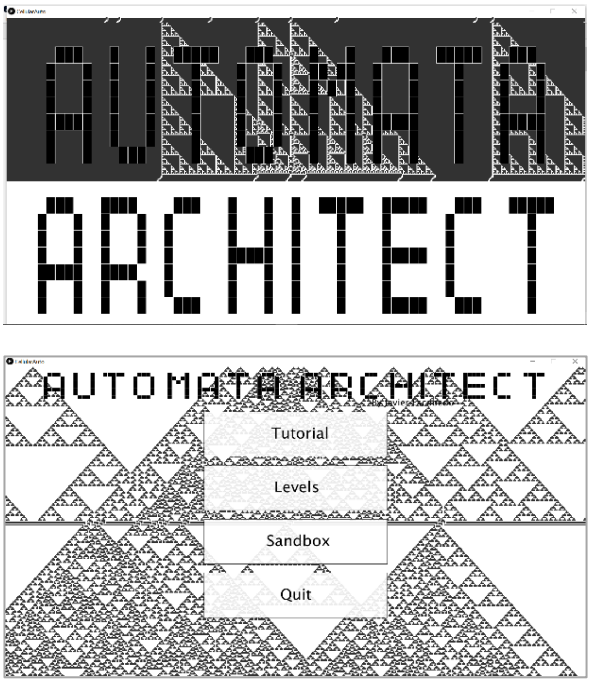

Figure 6. Main menu of Automata Architect.

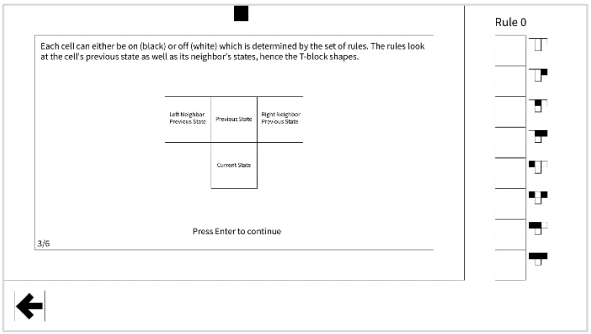

Figure 7. The interface for the tutorial mode.

3.2 Gameplay and Mechanics

Upon initiating the game, players are greeted with an engaging title sequence where a mesmerizing background animation of a cellular automaton is running. This background animation carries on as the main menu is drawn on the screen. The main menu (Figure 6) allows access to the tutorial section (Figure 7), the sandbox mode (Figure 8), the level selection menu (Figure 9), or exiting the game altogether.

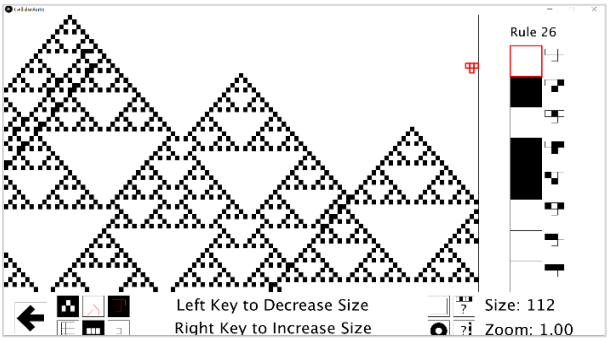

Figure 8. The interface for the sandbox mode.

3.2.1 Game Modes

The game provides three modes: tutorial mode providing practices with instructions, sandbox mode for flexible gameplay, and level mode that gives different difficult levels of challenges.

The tutorial is made up of tutorial states, each of which draws something different to explain the game's mechanics. All throughout the tutorial, players have the option of going back to the main menu via the back arrow in the lower left corner.

In the sandbox mode, the player can toggle cells, branches, render mode, rules, and initial conditions. They can select any rule and pick their own configuration of cells, among other things. The game will immediately run and generate the pattern after any change to the automaton is made, so players can observe the generated pattern. To customize the display of the patterns, the player can resize the automata by pressing the left or right arrow keys. The option to return to the main menu is always available as in the mode for level play.

The level mode contains the primary functions for gameplay experience. After entering the interface for levels, the player is offered three sections of levels that they can navigate through. Each section contains 30 levels, thus 90 levels in total. Players can start playing from level 1 through 90 by leveling up, or select and jump onto any level to start an unknown challenge. Generally, a higher level gives more intricate patterns and complex rules that requires the player to better understand automaton rules to manipulate and match patterns. While playing the game, players can opt to switch back to the main menu or level selection at any time by clicking on the designated icon on the lower left. Alternatively, they can refresh the grid and start with a clean slate or move to the next or previous level.

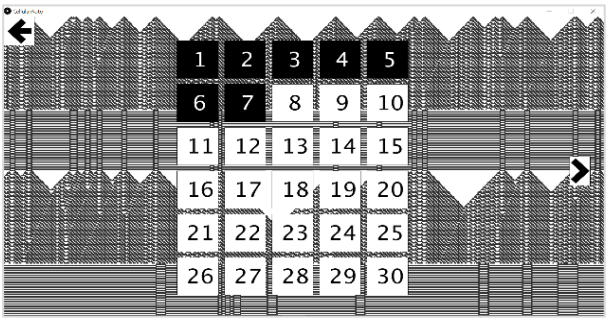

Figure 9. The interface for the level mode.

3.2.2 Level Designs

The design of the levels is intricately woven around the principles of cellular automata, computational theory, and logic puzzles. Each level unfolds a new set of automaton rules and a distinctive target pattern for players to unravel. The foundational query each level poses to the player is, "Which set of rules will generate the target pattern?". In the beginning stages, levels are designed with simplicity to help users understand the fundamental mechanics of the game. For the most part, they feature straightforward automaton rules and clear-cut target patterns, making it easier for players to observe the links between the rules and their outcomes on the grid. For the first 30 levels. players are also shown what rule applies on the cell that their mouse hovers over, making it easier to choose the correct rule. As players progress, the complexity gradually amplifies. Mid-level stages introduce moderately complex automaton rules and patterns, thereby pushing players to broaden their strategic and logical thought process. The final stages are randomized to challenge the player's comprehension of the game mechanics and their ability to implement an astute strategy to achieve the target patterns. These levels present intricate rules and patterns that are not immediately apparent, thus demanding a seasoned understanding of cellular automata and an inventive approach to solving the puzzle.

Moreover, the levels are not solely developed to be harder as the sequences advance, they are also intended to be educational and immersive. Each level can be seen as a step forward in the learning curve, subtly teaching players about different automata patterns, behavior, and rules as they play along. For any given level, different players may approach the puzzle from various angles and employ different strategies, making each gameplay unique. This quality not only increases the game's replay value but also stimulates player creativity and problem-solving skills.

The player's progression is marked by how many levels have been completed with the fewest moves possible. A move is counted whenever the rule selected is different from the previous rule selected. If the moves performed exceed the minimum moves required and the level was completed, then that level is treated as a partially completed level. In the level selection menu, fully completed levels are marked in black, partially completed levels in grey, and incomplete levels in white.

Upon completing all levels, the main menu will present players with an additional feature, the endless levels. These levels are designed with an infinite nature, providing an unbounded gameplay experience. Unlike the finite levels, these do not conclude after reaching a certain point, but continue indefinitely, with each new level posing a fresh set of challenges. The striking feature of these endless levels is their dynamic scaling and randomized initial conditions. As players navigate through and solve each level, the size of the cellular automaton incrementally increases with a different initial configuration. This progress in scale corresponds with an elevation in complexity, providing players with increasingly intricate puzzles to solve. The emphasis in these limitless levels is placed upon sustained engagement and problem-solving prowess rather than reaching a definitive end. This allows for an extended play value as well as fostering deeper understanding and proficiency in handling the cellular automaton. This endless play feature amply rewards continued interaction, with each solution discovered and each level surpassed serving as a gratifying achievement in the player's limitless journey through Automata Architect.

3.2.3 Game Controls

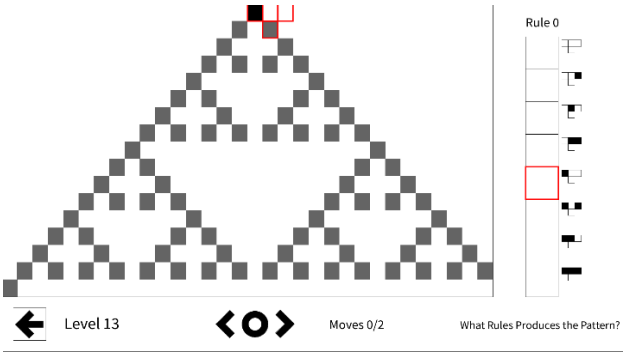

Figure 10. The interface of a level.

The primary method of controlling the game is through mouse clicks on the on-screen user interface (as shown in Figure 10), which grants an intuitive and user-friendly design. At the right of the screen, the various automaton rules are represented by clickable boxes next to the corresponding rule. Players select or deselect these rules by clicking on them, causing an immediate effect on the grid. Other buttons also include icons for returning to the main menu, proceeding to new levels, or refreshing the grid to start anew. In sandbox mode, players can manually change the setup of automaton cells by clicking within the grid, turning cells 'on' or 'off' as desired. Players can also use the right and left arrow keys to resize the automata and use the up and down key to increment or decrement the current rule. There are also buttons for randomizing the rules, randomizing initial conditions, and altering automaton settings, all of which can be found at the bottom of the screen. Also, by using a mouse wheel the player can zoom in to view their pattern more closely, they may also click and drag to move around while zoomed in, which offers fluid control over what the player can see.

To further lighten the learning curve, the game includes tooltips that provide clear, concise explanations whenever a player hovers their mouse over a particular rule or button. The tooltips are designed to seamlessly blend with the gameplay without disrupting the immersive experience. They provide real-time explanations of both frequently used and less familiar game controls, making it easy for the players to understand the significance of each button and rule.

The game puzzles offer a blend of easy to extremely complex automata rules and target patterns. The puzzles require logical thinking and strategic manipulation of the automaton rules. The complexity of each level is mostly dependent on the target pattern, as it requires intricate knowledge of the automaton rules to produce that specific pattern. Players must experiment and find the right configuration of rules to reach the target pattern. Since the target pattern and the player's pattern are overlayed with each other, a puzzle (or level) is solved when the target pattern and the player's pattern are the same.

Most of the play flow involves using the mouse to click things on the program screen. While playing a level, some players will usually observe the target pattern before selecting any rules while others will randomly see what works. When the target pattern is generated, the level is complete, and the game will take the player to the next level.

3.3 Game Art

While Automata Architect utilizes simplicity in its visually appealing design, there are art assets that enhance the game's quality. These assets include different icons that denote various functions, such as menu buttons for exiting the game or icons for advancing to the next level, and symbols indicating automaton rules. Minimalistic designs are used to represent these icons, ensuring they are easily identifiable, yet unobtrusive. The backdrop of the game also acts as an art asset, composed of different combinations of cellular automata patterns which create an appealing and dynamic background display. It should be noted that no external game assets like images, sprites, or even external code was used. All icons and patterns were made using the functionality that Processing [13] provides.

The art style of Automata Architect is a fusion of abstract minimalism and intricate geometrical design, much inspired by cellular automata principles themselves. The use of stark black and white for the cells, along with shades of grey for overlaying two automatons over each other, gives a monochromatic theme that intensifies the focus on patterns and formations rather than distractions from colors. The rationale behind this decision is to maintain the association with classical cellular automata simulation, typically presented in binary colors, and to stay true to the mathematical roots of the concept. The simple geometrical shapes and patterns that keep changing based on the rules provide a sense of order and randomness at the same time, making the gaming experience captivating. The patterns can range from repetitive stripes to intricate fractals, all formed by a simple grid of black and white squares. This interaction between simplicity and complexity is the core focus of the visual style of Automata Architect.

The real highlight of the game's art is in the fluid animations of cellular automaton patterns forming, transforming, and shifting across the grid. The use of elementary cellular automata rules in the game means the smallest shifts in rules can lead to drastically different animations. Each time the automaton runs, the resulting patterns move and evolve across the screen, creating a relaxing and intellectually stimulating spectacle. By simply activating a rule or setting up cell configurations in the sandbox mode, the player can set off an awe-inspiring chain of movement and animation. The entire visual interactive experience created through running cellular automata infuses life into the simple black and white grid, showcasing the full potential of this minimalist art style.

4 Conclusion and Future Work

Procedural textures often appear in abstract arts. However due to challenges to connect rules and generated patterns, using procedural textures for art applications is not an easy task. In this paper we introduce our work for developing Automata Architect, a serious computer game for exploring cellular automaton rules and patterns. In this game, we designed a few interactive challenges to motivate the player’s observation, interpretation, and memorization abilities, thus helping the player get familiar with the working mechanism of procedural methods. We expect this work can invoke the interest from players for learning procedural patterns and creativity of developing applications for computer generated arts. A few possible extensions may be made in our future work. For example, to provide a function for player free sketched patterns. We also expect to develop an integrated system for multiple procedural methods such as Voronoi diagrams [3] and L-systems.

Funding

This work has been supported by the Organized Research and Creative Activities (ORCA) Program of University of Houston-Downtown.

Conflict of Interest: none declared.

References

[1] E. Adams. Will computer games ever be a legitimate art form?. Journal of Media Practice. 7. 10.1386/jmpr.7.1.67/1. 2006
[2] E. Adams, Fundamentals of Game Design (3rd edition). Publisher: New Riders. 2014.
[3] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. 23, 3 (Sept. 1991), 345–405. https://doi.org/10.1145/116873.116880
[4] P. Bourke. L-System manual. http://paulbourke.net/fractals/lsys/
[5] A. De Gloria, F. Bellotti, and R. Berta. Serious Games for education and training. International Journal of Serious Games. 1. 10.17083/ijsg.v1i1.11. 2014.
[6] J. Dong, J. Liu, K. Yao, M. Chantler, L. Qi, H. Yu, and M. Jian. Survey of Procedural Methods for Two-Dimensional Texture Generation. Sensors 20, no. 4: 1135. https://doi.org/10.3390/s20041135. 2020.
[7] I. Gintere. Towards a new digital game of contemporary aesthetics: research and knowledge transfer. Society. Technology. Solutions. Proceedings of the International Scientific Conference. 1. 10.35363/ViA.sts. 2019.14.
[8] Y. Huang, T. Zhang, and L. Xu. The Development of a Game with Applications of Object-oriented Programming Concepts. American Journal of Advanced Research, 2(1), 7–13. https://doi.org/10.5281/zenodo.1407482. 2018.
[9] E. Kim, S. Kim, H. Koo, K. Jeong, and J. Kim. Emotion-Based Textile Indexing Using Colors and Texture. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science). 3613. 1077-1080. 10.1007/11539506_133. 2005.
[10] H. Kim, J. Dischler, and H. Rushmeier & B. Benes. Edge-based procedural textures. The Visual Computer. 37. 10.1007/s00371-021-02212-4. 2021.
[11] P. Prusinkiewicz and J. Hanan. Lindenmayer Systems, Fractals, and Plants. Lecture Notes in Biomathematics.1989.
[12] P. Prusinkiewicz and A. Lindenmayer. The Algorithmic Beauty of Plants. 10.1007/978-1-4613-8476-2. 1996.
[13] C. Reas and B. Fry. Processing: A Programming Handbook for Visual Designers, Second Edition. MIT Press. ISBN: 0-262-02828-X. 2014.
[14] T. Roden and R. LeGrand. Growing a computer science program with a focus on game development. SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education.
[15] C. Ron, M. Krembs, A. Labouseur, and J. Weir. Game design & programming concentration within the computer science curriculum. In Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05). Association for Computing Machinery, New York, NY, USA, 545–550. https://doi.org/10.1145/1047344.1047514
[16] M. Rüth and K. Kaspar. Commercial Video Games in School Teaching: Two Mixed Methods Case Studies on Students' Reflection Processes. Frontiers in psychology, 11, 594013. 2021. https://doi.org/10.3389/fpsyg.2020.594013.
[17] A. Sartori, B. Senyazar, A. Akdag Salah, A. Salah, and N. Sebe. Emotions in Abstract Art: Does Texture Matter?. 9279. 10.1007/978-3-319-23231-7_60. 2015.
[18] SimX Featured at SAEM SimWars. https://youtu.be/torkeVv1qok
[19] SimX: Virtual Reality Medical Simulation. https://www.simxvr.com/
[20] T. Susi, M. Johannesson, and P. Backlund. Serious Games - An Overview. 2015.
[21] A telling time game for kids. https://www.education.com/game/telling-time-quiz/
[22] D. Vlachopoulos and A. Makri. The effect of games and simulations on higher education: a systematic literature review. Int J Educ Technol High Educ 14, 22 (2017). https://doi.org/10.1186/s41239-017-0062-1
[23] S. Wolfram. Cellular automata as models of complexity. Nature 311, 419-424, 1984.
[24] S. Wolfram. Cellular Automata and Complexity: Collected Papers (1st ed.). CRC Press, 1994. https://doi.org/10.1201/9780429494093
[25] Wolfram MathWorld. https://mathworld.wolfram.com/CellularAutomaton.html
[26] L. Xu and D. Mould. Magnetic Curves: Curvature-Controlled Aesthetic Curves Using Magnetic Fields. 1-8. 10.2312/COMPAESTH/COMPAESTH09/001-008. 2009.
[27] L. Xu and D. Mould. Modeling dendritic shapes - using path planning. 29-36. 2007.