

# Wall object design in maze game using cellular automata algorithm

Furqan Malem<sup>1</sup>, Purba Daru Kusuma<sup>1</sup>, Ashri Dinimaharawati<sup>1</sup>

<sup>1</sup>Department of Computer Engineering, School of Electrical Engineering, Telkom University, Indonesia

## Article Info

### Article history:

Received January 26, 2023

Revised February 16, 2023

Accepted February 19, 2023

### Keywords:

Educational Games  
Game-Based Learning  
Wall Object  
Maze Game  
Cellular Automata

## ABSTRACT

Games can be an excellent learning tool for kindergarten students considering that children are happy to play at that age. For this reason, this research tries to create a maze-type educational game with the theme "Environmental Cleanliness," which teaches children to clean up garbage and trains them in awareness of the importance of environmental cleanliness. The maze game uses the Cellular Automata Algorithm for designing wall objects so that the difficulty level of the maze can be increased at each level. The maze will change if it meets certain conditions, creating a more exciting and fun educational game for children. This implemented algorithm aims to provide challenges, improve strategy skills, and train directional and spatial awareness by knowing the spaces and paths that must be passed, pointing their location in the maze game. The result of this research is that all features in the game design developed have been successfully implemented and functioned properly. Especially ensuring the algorithm used to generate the maze wall can work according to the design.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Furqan Malem  
Department of Computer Engineering  
School of Electrical Engineering, Telkom University  
Bandung, Indonesia  
Email: [furqanmalem@student.telkomuniversity.ac.id](mailto:furqanmalem@student.telkomuniversity.ac.id)

## 1. INTRODUCTION

Educational games created stimulate thinking, including improving concentration and problem-solving. An effective early childhood interactive learning technique is educational games, as most children are curious about everything around them at a young age [1]. One of the vital education for children today is about cleanliness and the importance of protecting the environment.

The environment is where all living things live, grow, develop and maintain survival. Therefore, the environment must be kept clean. The call to always be environmentally friendly, especially for a clean and litter-free environment, has been widely implemented in schools and through campaigns and posters to make everyone aware of the importance of a clean environment. However, that is not enough to sensitize the public to always care about the environment [2]. For this reason, it needs more effective delivery and close to children, namely through early childhood educational games.

Therefore, this research focuses on making educational games for kindergarten and elementary school students with the theme of "environmental hygiene.". That is with a maze-type game that teaches children to throw garbage in its place and complete challenges on time. This maze game is expected to train children's awareness of the importance of protecting the environment. In this maze game, a cellular automata algorithm is applied to add challenges to play, teach children's ability to solve problems, and improve strategic skills, management skills, creativity, and responsibility.

## 2. THEORETICAL BASIS

### 2.1 Educational Games

Of the many types of games, there is a particular category called educational games with educational and entertainment purposes. Especially for kindergarten children who like to play, learn and introduce themselves, educational games are available tailored to early childhood characteristics. Educational games present learning or fun learning experiences to children [3]. The purpose of using educational games in the learning process is to increase students' enthusiasm and motivation in learning so that, in the end, it can improve students' desire for the learning process. This goal is achieved by integrating academic achievements with educational games and entertainment [4].

### 2.2 Game-Based Learning

Currently, game-based learning models have been widely adopted for children's learning. There are educationally valuable products on the market that are proven to enhance children's learning and understanding. Recently, game-based learning has also been proposed for adult education. A key characteristic of educational games is that instructional content is obscured with game characteristics. Games are becoming a new form of interactive content worth exploring for learning and educational purposes. Universities are also seeking a new position in the changing long-term learning setting. Universities must develop innovative forms of learning to deliver long-term learning to their primary customers, the students. Modern technology requires proficient employees in effective communication, teamwork, project management, and other soft skills such as responsibility, creativity, micro-entrepreneurship, and corporate culture. Game-based learning is an approach to address the above issues [5].

### 2.3 Cellular Automata

John von Neumann proposed cellular automata (CA) as a formal model of self-producing organisms. The learned structures are mostly one-two dimensional infinite grids, although higher dimensions can also be considered [6]. CA is one of the oldest natural computation models, dating back over half a century. John von Neumann first conducted the study of CA in the late 1940s, the goal being to design self-replicating artificial systems that are also computationally universal [7].

Cellular Automata can be said to be a set of identical cells that gather and neighbor each other. Each cell of the CA has a state and is given a particular rule. This CA model is represented in an array. CA is an algorithm that describes the discrete spatial evolution of a complex system by defining local or global deterministic or stochastic transition rules, usually in grid cells of arbitrary dimension. CA models are developed by applying specific transition rules that affect the state of each cell. Rules determine the state of a location based on the previous state and the state of neighboring cells (local rule) or the state of all cells (global rule) [8, 9].

Cellular Automata generally consist of several elements: cells, states, neighborhoods, transition rules, and time steps. Cells are the basic units; these cells are organized in spatial tessellation, a two-dimensional grid that is the most common form of cellular automata. Next, there are states which define the attributes of a system. In CA, each cell can only have one state from a set of conditions at a specific time. Then there is the neighborhood, a collection of interconnected cells, and the two most common types of the neighborhood are Von Neumann and Moore. There are transition rules which are response rules for changes in a cell that are influenced by the condition of its neighbors [7]. Figure 1. below gives an overview of cellular automata.

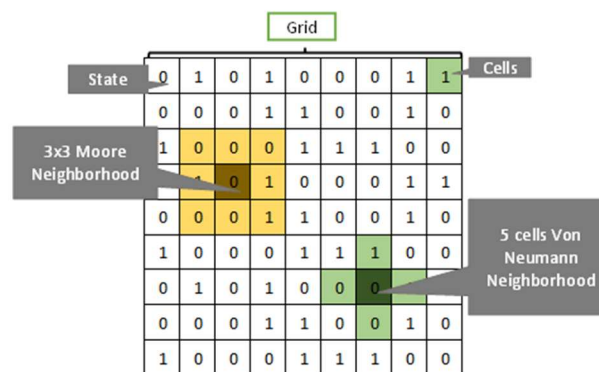


Figure 1. Cellular Automata Illustration

A cellular automaton model consists of a regular grid of cells. The cell's neighborhood can be specified by the local rule's radius of action  $r_{min}$ . Some standard neighborhood layouts are shown in Figure 2. The von Neumann and Moore layouts are used more frequently than the other layouts [10].

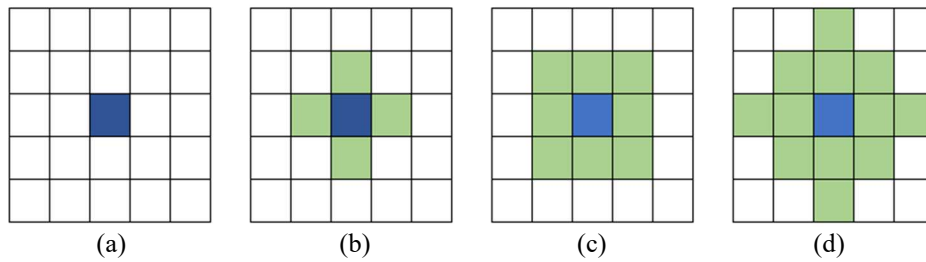


Figure 2. Cellular automata classical neighborhoods: (a) empty, (b) Von Neumann, (c) Moore, (d) Moore & Von Neumann

### 3. SYSTEM DESIGN

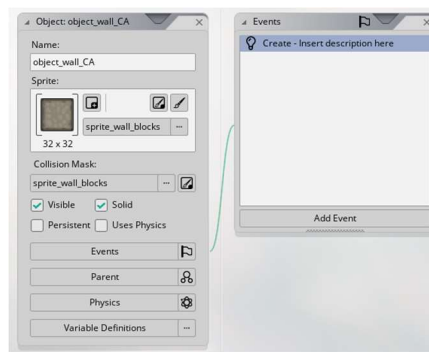


Figure 3. Wall object sprite design

Figure 3 above displays the object wall design tab in the Maze Cleaner game. The wall uses assets or sprites with a size of 32x32 pixels. Right next to it is the events tab of the object wall, which contains the create event, and the event includes the attributes of the object wall sprite.

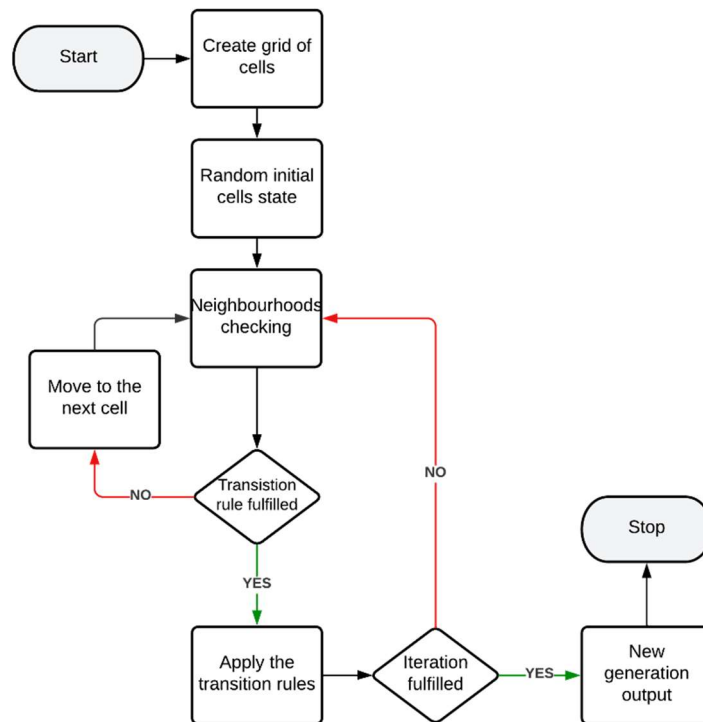
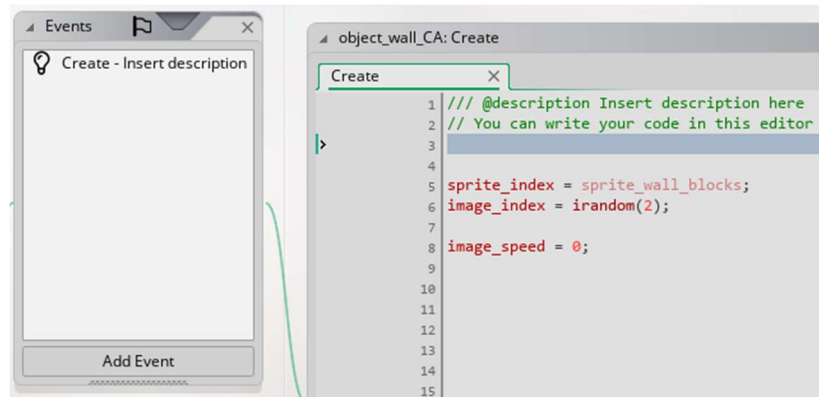


Figure 4. Cellular automata flowchart

Figure 4 above is a simple flow chart description of the design of the wall placement system using the cellular automata algorithm. In the initialization process, a random state value is placed in each cell and then checks each cell to apply the change rule. If the iteration is complete, the placement ends and produces output for the next generation cell.



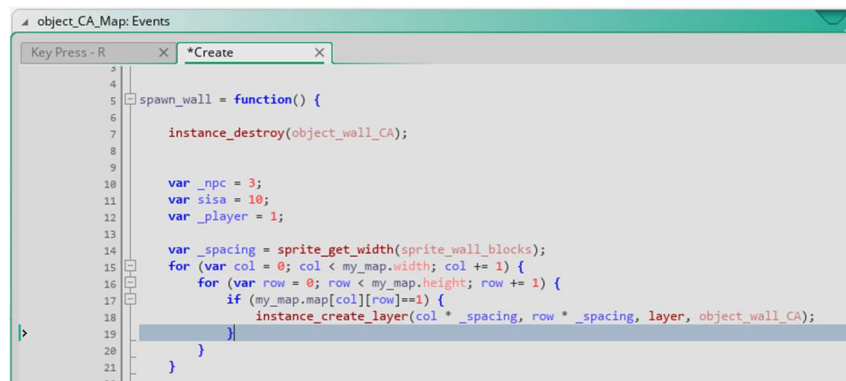
```

1  /// @description Insert description here
2  // You can write your code in this editor
3
4
5  sprite_index = sprite_wall_blocks;
6  image_index = irandom(2);
7
8  image_speed = 0;
9
10
11
12
13
14
15

```

Figure 5. Source code of object wall

Figure 5 above is the source code contained in the create event. The code above is the code for the wall object sprite. `Sprite_index` is a built-in variable that calls the wall sprite so that the desired sprite can appear in the room. `Image_index` is also a built-in variable that selects the sprite index (if the sprite has more than one frame).



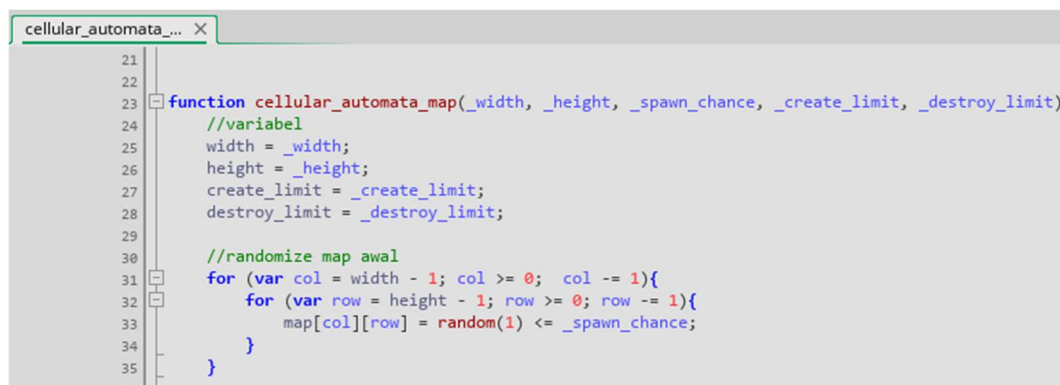
```

3
4
5  spawn_wall = function() {
6
7      instance_destroy(object_wall_CA);
8
9
10
11      var _npc = 3;
12      var _sisa = 10;
13      var _player = 1;
14
15      var _spacing = sprite_get_width(sprite_wall_blocks);
16      for (var col = 0; col < my_map.width; col += 1) {
17          for (var row = 0; row < my_map.height; row += 1) {
18              if (my_map.map[col][row]==1) {
19                  instance_create_layer(col * _spacing, row * _spacing, layer, object_wall_CA);
20              }
21          }
22      }
23  }
24
25
26
27
28
29
30

```

Figure 6. Source code for wall spawn

Figure 6 above is the code to display the wall CA in the room. In the spawn wall function, there are initialization variables, namely `_npc` for the number of NPCs, variable `_sisa` to show the remaining garbage, and variable `player` to indicate the number of players spawned along with the CA wall, as well as variable `spacing` for the grid size value. `Sprite_get_width()` is a syntax to get the width of a sprite. The width of this sprite is then used as a reference for the grid size for Cellular Automata.



```

21
22
23  function cellular_automata_map(_width, _height, _spawn_chance, _create_limit, _destroy_limit)
24      //variabel
25      width = _width;
26      height = _height;
27      create_limit = _create_limit;
28      destroy_limit = _destroy_limit;
29
30      //randomize map awal
31      for (var col = width - 1; col >= 0; col -= 1){
32          for (var row = height - 1; row >= 0; row -= 1){
33              map[col][row] = random(1) <= _spawn_chance;
34          }
35      }
36
37
38
39
40

```

Figure 7. Source code of Cellular automata script

Figure 7 above is the `cellular_automata_map` function, which has five arguments: `width`, `height`, `spawn_chance`, `create_limit`, and `destroy_limit`. There is also a `for` loop, which is used to generate the initial map randomly; this is useful as the initial state of the cellular automata.

```

37 //fungsi iterasi
38 static iterate = function(_num){
39
40     repeat(_num){
41
42         //membat map baru agar tidak menimpa data sebelumnya
43         var _new_map = [];
44
45         //loop melalui map lama, dan buat map baru dengan sel generasi selanjutnya
46         for (var col = 0; col < width; col += 1){
47             for (var row = 0; row < height; row +=1){
48
49                 //menghitung jumlah neighbours
50                 var _col_dif, _row_dif, _count;
51                 _count = 0;
52                 for (var _col_offset = -1; _col_offset < 2; _col_offset += 1){
53                     for (var _row_offset = -1; _row_offset < 2; _row_offset += 1){
54                         _col_dif = col + _col_offset;
55                         _row_dif = row + _row_offset;
56
57                         if (_col_dif < 0) || (_row_dif < 0) || (_col_dif >= width) || (_row_dif >= hei
58                             _count += 1;
59                         }else if (_col_dif == 0) && (_row_dif == 0){
60                             continue;
61                         }else if (map[_col_dif][_row_dif]){
62                             _count += 1;
63                         }
64                     }
65                 }

```

Figure 8. Iteration function

Figure 8 above is the source code for the iteration function in cellular automata. This iteration function is an essential part of cellular automata because it concerns the number of repetitions. In this function, there is also a variable initialization for the new map. In addition, there is also a `for` loop which helps loop the old map and then create a new map with the next generation of cells. In this section, code is inserted to calculate the number of neighbors.

```

65     }
66     //menerapkan aturan perubahan
67     if (map[col][row]){
68         _new_map[col][row] = _count > destroy_limit;
69     }else {
70         _new_map[col][row] = _count > create_limit;
71     }
72 }
73 }
74 }
75
76     map = _new_map;
77 }
78 }
79 }

```

Figure 9. Transition rules

Figure 9 above explains the transition rules used for cellular automata in the design of this Maze Cleaner game wall. The cell will remain active if the count exceeds the created limit. The cell will also remain active if the count exceeds the created limit.

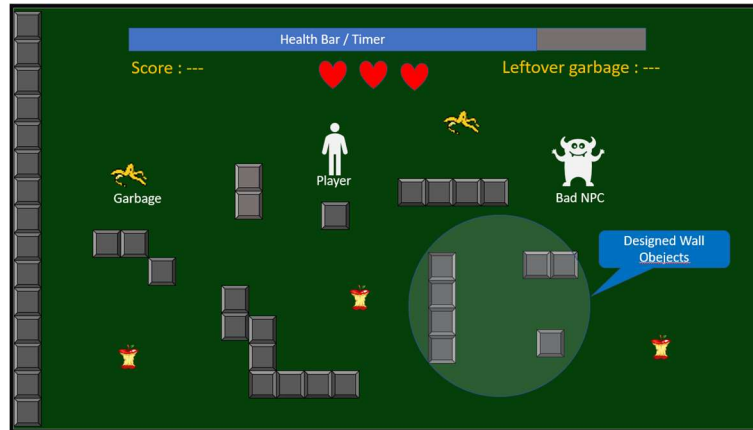


Figure 10. Interface Design Illustration

Figure 10 above illustrates the maze game's interface design for each level. In this research, the author focuses on designing wall objects, as in the highlighted image, using a cellular automata algorithm that has been adjusted.

#### 4. RESULTS AND DISCUSSION



Figure 11. Implementation of cellular automata

Figure 11 shows cellular automata's implementation on the placement of wall objects for level 2. The object wall in this Maze Game is placed in each room level, starting from level 1 to level 3, and uses cellular automata algorithm to generate wall position randomly. Each level has a different complexity and increases as the level increases, which aims to increase the challenge and excitement of playing. The object wall at each level uses a different theme to make it look more varied. Figure 12 below shows the implementation of the wall object design for each level. The number of walls and the maze's difficulty increase as the game level increases.



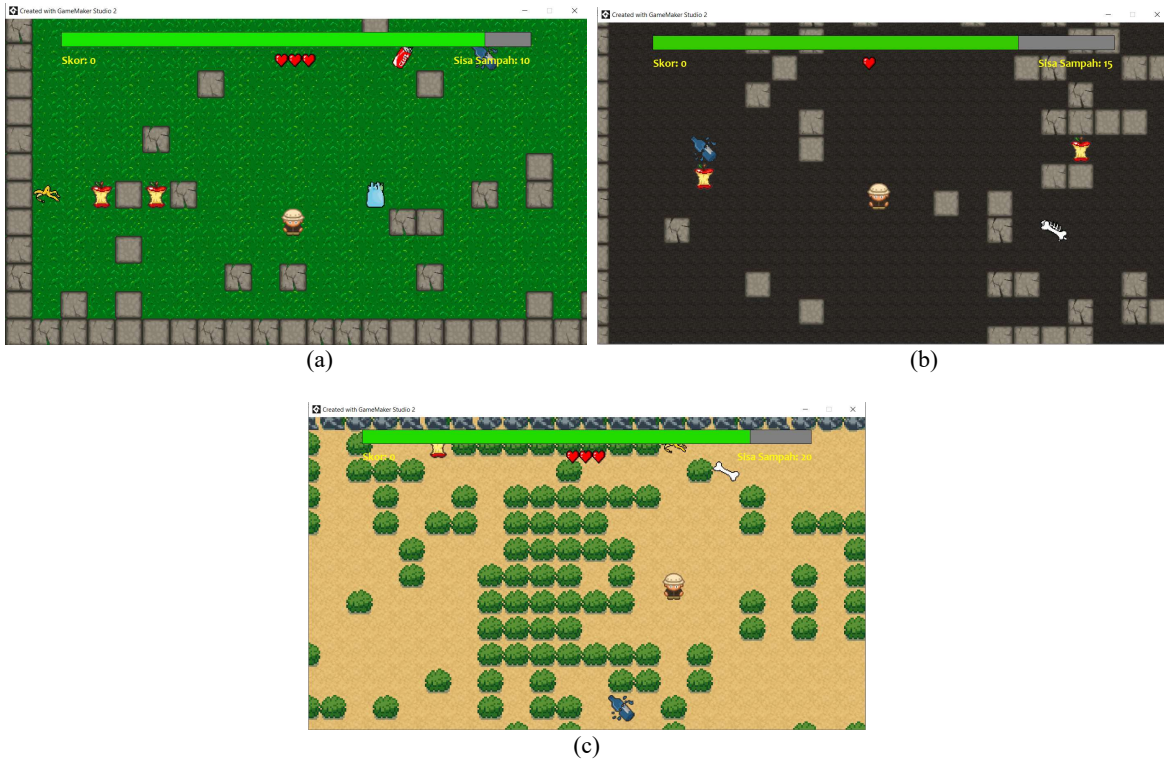


Figure 12. Room Level Interfaces: (a) Level 1 Interface Page, (b) Level 2 Interface Page, (c) Level 3 Interface Page

### 4.1 Alpha Testing

Table 1. Alpha Testing

No	Description	Input	Expected results	Output	Conformity
1	Start the game	Click the start button	Direct the view to room level 1 and generate the maze wall.	Room Level 1 appears, and the maze wall generate as expected	Match
2	Randomly generate wall	Player caught by NPC The health bar runs out / time ends	Restart the room, and regenerate the maze wall.	The room restarted successfully, and the position of the maze wall changed.	Match
4	Restart after the game over	Click the restart button and repeat the level	Directs the view to the previous level and regenerates the maze wall.	The room successfully returned to the previous level, and the position of the maze wall changed.	Match
5	Next Level	Click the next button	Direct the view to the next level and generate a random wall.	Successfully redirect the display to the next level and generate a random wall	Match

Table 1 above is the result of alpha testing, and this test aims to ensure the functions and features of the object wall are running correctly. Alpha testing is done with the white box method to ensure that the wall objects developed have run according to the initial design of the author. This test has five conditions: starting the game, randomly generating walls, restarting after the game is over, and continuing to the next level. And based on the test results, we can conclude that the design of wall objects using the cellular automata algorithm has followed the initial plan and can work in increasing the difficulty by changing the wall every time it meets the conditions.

## 4.2 Beta Testing

Beta testing was conducted on 10 respondents regarding using cellular automata algorithms in the Maze Cleaner game. This test is carried out by giving a simple questionnaire to respondents after playing the Maze Cleaner game and then giving an assessment with a score of 1-5 for each question.

Table 2. Beta Test Questions Score

Number	Respondent's Name	Question's score							Total
		x1	x2	x3	x4	x5	x6	x7	
1	Wirna	5	4	5	5	5	5	5	34
2	Hafidz	5	5	5	5	4	5	5	34
3	Ilham	2	3	3	3	3	3	3	20
4	Restu	5	5	5	5	4	5	5	34
5	Fajri	4	4	5	5	5	4	5	32
6	Angga	3	4	4	5	3	5	5	29
7	Teguh	5	5	3	5	5	5	5	33
8	Zidan	5	3	5	5	3	5	3	29
9	Fajar	5	4	5	5	5	5	5	34
10	Farhan	4	4	3	2	3	3	3	22

x1-x7 : Question items 1 to 7

1-5 : Rating points (minimum 1, maximum 5)

Total : The total value of each respondent's answer

Table 3. The average value of the question score

Number	Questions	Average Score (1-5)
1	How is the visualization of the maze created by the cellular automata algorithm?	4.3
2	Does the cellular automata algorithm make the maze feel random and challenging?	4.1
3	How easy is it to play a maze game using cellular automata algorithm?	4.3
4	How effective is the cellular automata algorithm in creating mazes?	4.5
5	Did you find the maze game more fun after using the cellular automata algorithm?	4.0
6	How is the experience of playing a maze game created using the cellular automata algorithm?	4.5
7	Compared to other mazes, how unique is the maze created with the cellular automata algorithm?	4.4

### 4.2.1 Validity Test

The validity test aims to determine whether the questionnaire used in obtaining data from respondents is valid or not. This validity test uses a significance value of 5% with 10 respondents, then the  $r_{table}$  value used is 0.632. If the value of  $r_{count} > r_{table}$  = valid, and if the value of  $r_{count} < r_{table}$  = invalid.

$$validity\ test \begin{cases} r_{count} > r_{table} = valid \\ r_{count} < r_{table} = invalid \end{cases} \quad (1)$$

$$r_{count} = \frac{n\Sigma - (\Sigma x)(\Sigma y)}{\sqrt{\{n\Sigma x^2 - (\Sigma x)^2\}\{n\Sigma y^2 - (\Sigma y)^2\}}} \quad (2)$$

Table 4. Validity Test

Item	rcount	rtable	Description
1	0.781333636	0,632	Valid
2	0.881154	0,632	Valid
3	0.714666	0,632	Valid



Item	rcount	rtable	Description
4	0.868258	0,632	Valid
5	0.748609	0,632	Valid
6	0.854542967	0,632	Valid
7	0.634793	0,632	Valid

Table 4 above is a calculation using the Excel and SPSS applications. The results of the calculation of the validity test value of the questionnaire are declared valid because it meets the requirements, namely,  $r_{count} > r_{table}$ .

#### 4.2.2 Realibility Test

This reliability test was carried out to determine the level of consistency of the questionnaire. The reliability test is carried out after the question items are declared valid. According to Imam Ghozali, the variable is reliable if the Cronbach Alpha value is  $> 0.70$ . This reliability test uses the SPSS application.

Table 5. Realibility statistic

Cronbach's Alpha	N of Items
.895	7

Based on the reliability statistics table above, Cronbach's Alpha value is 0.895 with 7 items. The value obtained is  $> 0.70$ , and it can be concluded that the test conducted is reliable.

## 5. CONCLUSION

This work has demonstrated the development of the game Maze Cleaner and the tests performed. We can conclude that the development of the object wall using the Cellular Automata method for random maze successfully runs well and has been following the expected design. The Cellular Automata algorithm is considered effective by respondents in creating mazes and providing a better gaming experience, with an assessment score of 4.5 from a maximum score of 5. The Cellular Automata algorithm can create a maze wall object that feels unique and different based on the assessment of respondents with a score of 4.4, as well as create a maze that feels random and challenging with an assessment score of 4.1 from a maximum score of 5.

This research uses a relatively simple cellular automata transition rule. In the future, research on wall design using cellular automata algorithms in maze games can be improved by applying more complex cell change transition rules to produce more exciting maze shapes and add more difficulty levels to the maze.

## REFERENCES

- [1] R. A. Rahman and D. Tresnawati, "Pengembangan game edukasi pengenalan nama hewan dan habitatnya dalam 3 bahasa sebagai media pembelajaran berbasis multimedia," *Jurnal Algoritma Sekolah Tinggi Teknologi Garut*, vol. 13, no. 1, pp. 184–190, 2016.
- [2] E. J. Hutagaluh, "Game kebersihan lingkungan menggunakan metode finite state machine," *Jurnal Mahasiswa Teknik Informatika*, vol. 1, no. 2, 2017.
- [3] P. W. Wijayanto and Y. Siradj, "The educational game 'Indonesian Tribes' for the kindergarten students," *IJPTE: International Journal of Pedagogy and Teacher Education*, vol. 1, no. 1, pp. 27–36, 2017, doi: 10.20961/ijpte.v1i1.8456.
- [4] S. H. K. Dafalla, "The impact of educational games on the academic achievement of fifth grade students in science. (an experimental study on the elementary level, afif province)," *International Journal of Education and Research*, vol. 4, no. 12, pp. 173–188, 2016.
- [5] M. Pivec, O. Dziabenko, and I. Schinnerl, "Aspect of game-based learning," in *Proceedings of I-KNOW '03*, Jul. 2003, pp. 216–225.
- [6] P. Sarkar, "A brief history of cellular automata," *ACM Comput Surv*, vol. 32, no. 1, pp. 80–107, 2000, [Online]. Available: <http://alife.santafe.edu/alife/topics/>
- [7] J. Kari, "Theory of cellular automata: a survey," *Theor Comput Sci*, vol. 334, no. 1–3, pp. 3–33, Apr. 2005, doi: 10.1016/j.tcs.2004.11.021.
- [8] S. N. Kudrat, Y. Sibaroni, and E. B. Setiawan, "Traffic light control simulation using cellular automata and fuzzy inference system," in *e-Proceeding of Engineering*, Apr. 2015, pp. 1884–1891.
- [9] O. Ziaee, N. Zolfaghari, M. Baghani, M. Baniassadi, and K. Wang, "Energy equipment and systems a modified cellular automaton model for simulating ion dynamics in a li-ion battery electrode," *Energy Equip. Sys*, vol. 10, no. 1, pp. 41–49, 2022.
- [10] B. Hassani and M. Tavakkoli, "A multi-objective structural optimization using optimality criteria and cellular automata isogeometric topology optimization for stress, material and structures via level-set method," *Asian Journal of Civil Engineering (Building and Housing)*, vol. 8, no. 1, 2007, [Online]. Available: <https://www.researchgate.net/publication/228522976>