Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

WALL PATTERN DETECTION WITH PRIM'S ALGORITHM TO CREATE PERFECT RANDOM MAZE

AGNESIA[1] , WIRAWAN ISTIONO[2]

[1,2]Universitas Multimedia Nusantara, Informatic Department, Scientia Boulevard, Curug Sangereng, Tangerang, Indonesia

E-mail: [1]agnesia@student.umn.ac.id, [2]wirawan.istiono@umn.ac.id

ABSTRACT

Replayability is one of the factors that determines how a video game played again by players, with one of the example is by offering new and unique content into a game. One of the methods that can be used is by implementing a map generation into the game level via Procedural Content Generation or PCG. With Prim's Algorithm as the base of the PCG, this research will design and develop a game with mazes as its map level. Focusing on their maze generation, this research will also on the result and will try to detect and display the data of mazes generated, while also trying to determine video game satisfaction from players that will be playing the game developed via Game User Experience Satisfaction Scale or GUESS. A Detect Wall Pattern method is developed to detect the pattern of each mazes' grids, where the data will be documented and then processed to determine the result of 250 maze generations of mazes with size 2x2, 3x3 and 4x4. Based on the research, MazeGame has succeeded on being developed with PCG feature based on Prim's Algorithm. Detect Wall Pattern method has also been developed successfully, where this method successfully detecting 4 unique patterns for 2x2 size mazes, 79 patterns for 3x3 size mazes, and 243 patterns for 4x4 size mazes from 250 maze generation on each size.

Keywords: *Replayability, Procedural Content Generator, Prim's Algorithm, MazeGame, GUESS, Detect Wall Pattern*

1. INTRODUCTION

Developments in video games that are in line with current technological developments provoke many innovations and new findings in the world of gaming. This encourage competition in the gaming industry, where game developers competing to make games with something unique enough in them to attract new players [1]. One of the factors that can determine this is the content in the game that can be replayed but not monotonous. Replay ability is an important factor that supports the value of player satisfaction in the long term when the game is played [2]. The existence of several components of the game that are made uniquely for each gameplay can guarantee a change in the way each player plays the game and get them to play the game again. One aspect that can support the replayability factor of a game is randomization factor [3], [4]. Games with randomized elements usually implement these elements on components that are not too prominent but still change the gameplay whatever small change it might offer.

Maze Game is a type of game that has a maze (labyrinth) element in it, usually this type of game is in the form of games with puzzle, adventure, and other genres [5], [6]. This type of game provides several design levels that are used during gameplay so that players don't get bored quickly. Game developers usually develop various new modes and maps periodically with the aim of adding content for players to prevent boredom on the gameplay [7]. Maps that are formed usually have a manual creation system, but sometimes there are those who use a method to build maps randomly, using the Procedural Content Generation method [8].

Procedural Content Generation is a method of forming a complex game object in a short time by carrying out procedures that are arranged in such a way that the resulting design is something as if it was made manually [9]. Several algorithms can be used as the basis for creating this Procedural Content Generation method, with the basic Maze Generation algorithm is the Depth First Search Algorithm, then followed by several more complex algorithms such as Prim's algorithm and Kruskal's algorithm [10]. Both algorithms are based on the Minimum Spanning Tree (MST) concept, but the results from Kruskal's algorithm emphasize

balanced weighted edges and tends to have a similar pattern so that it can be solved more easily [11]. While the results of the maze with Prim's algorithm can have variations that are much different from the results of the previous generation of genes [12], [13].

Based on this knowledge, the design and development of a 3D Maze Puzzle Game will be carried out with the game level will be created Procedural Content Generation method based on Prim's Algorithm. To see how this method is used in the User Experience of the designed game, a survey will be conducted to determine the level of player satisfaction using the Game User Experience Satisfaction Scaled (GUESS) survey method. In addition to testing User Experience, this research will also search for and display the results of maze patterns created using the Procedural Content Generation method. The difference of this research from previous research as well as the novelty of this research lies in making a perfect random maze using Prim's Algorithm, by detecting wall patterns. The meaning of the perfectly random Maze created in this research is, that there will always be paths connected to the destination although the maze paths are made randomly with many size, and the next maze path that is generated, it is very rarely generate same path as before.

2. LITERATURE STUDY

Procedural Content Generation (PCG) is way to produce unique game content each time the instance is run, so the results will be ranged from slightly to significantly different for each player [9]. Procedural Content Generation refers to the algorithm used to create content and can reduce the workload of game design and development. Some of the methods created have become common methods used in the gaming industry, although they are still applied to specific contexts and game elements in a game [14], [15]. In making game elements such as level design, or in the example of this research a maze, Procedural Content Generation will use maze generation algorithms, where one of the algorithms that can be implemented in this method is Prim's Algorithm.

With Prim's Algorithm, the implementation of Procedural Content Generation in creating a maze will start by creating a grid that will become the 'floor' of the maze (maze cell). Then a function will be carried out to initialize the parts that will be part of the maze, either as a wall of the maze (maze wall) or as a path of the maze (maze passage) [13].

Maze generation algorithm has several algorithms that can be used in the application of the method. The differences that these algorithms have are the length, shape, and number of corridors of the results created during the implementation of the difficulty, efficiency and comprehensive level of the algorithm. In this game that will be made, the algorithm chosen is Prim's Algorithm [14]. Prim's Algorithm is an algorithm that is usually used to find the Minimum Spanning Tree (MST) on a graph. This algorithm will look for a subset of edges on each vertex and will look for the smallest number of combinations.

3. METHODOLOGY

The research methodology that used in this study is the Procedural Content Generation method based on Prim's Algorithm. The steps taken in this research are in six steps, namely, literature study, analysis and design, implementation and development, testing, evaluation and documentation.

The initial step in this research is a literature study, where at this stage, research on the theory and methods needed and used during game design and development will be carried out. The materials sought are in-depth knowledge of the implementation of the Procedural Content Generation method, video game concepts, Prim's Algorithm, enemy AI and behaviors, and others.

The next proceed is analysis and design, where at the game design stage, steps are carried out to analyze and summarize all the required game elements, both formal and dramatic elements. Besides that, the concepts such as flowchart planning, procedural content generation method design, player behaviour, level objectives, mockup design, and game assets planning will also be carried out. Then the next step is the implementation and development step, where the prim's algorithm will be used to generate the perfect maze. The experiment will be carried out using the Unity Game Engine. Beside that a copy of the project is also made to carry out maze generation aimed at collecting data on the implementation of the Detect Wall Pattern method.

The next step is testing, where at this stage, the finished maze project will be tested in the testing phase. The testing phase is carried out by searching and displaying the results of maze patterns made by doing maze generations in large numbers, and looking for the presence or absence of similar mazes when generation is carried out. The results of the maze generation will be compared with data on

Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the number of possible spanning trees to be formed on the n x n grid to find out whether the resulting unique maze will exceed the spanning tree limit that is possible to form on the n x n grid. After The testing maze generation phase is carried out, other testing is also carried out involving several respondents, then the respondents will be given an online questionnaire based on questions based on the Game User Experience Satisfaction Scale (GUESS).

Based on the test results, both from the questionnaire that has been distributed and also through searching for unique maze results made by doing maze generation, the next stage that will be carried out is evaluating game results. Questionnaire results will be collected, calculated, and concluded to calculate the value of video game satisfaction with the Game User Experience Satisfaction Scales (GUESS) [16]. In addition, maze generation results will be searched and displayed after generating a large number. Then documentation is carried out during the research, where the documentation process is carried out in an appropriate manner. Documentation results will be in the form of code snippets, search results, and attachments to the results of the GUESS survey of respondents.

In the Generate Map procedure in the experiment, there are several functions involved in implementing Procedural Content Generation based on Prim's Algorithm into Maze Game. The figure 1 to figure 7 shows a flowchart for the Generate Map function in Maze Game.

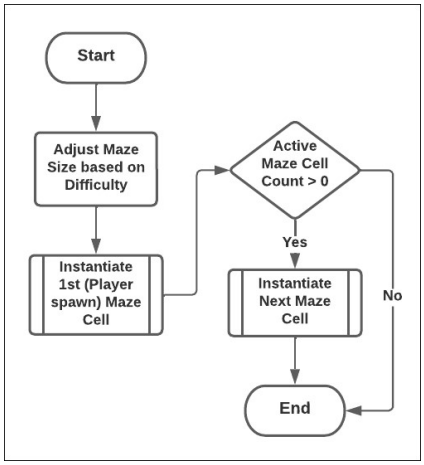

Figure 1: Maze Generation Flowchart

Figure 1 shown flowchart for Maze Generation. This function begins by determining the size of the maze that is generated, according to the difficulty selected by the user. After that, instantiate the first maze cell, which will later become the spawn point for the game object player. After that, according to the number of maze cells that generated, maze cells will be created via the Instantiate Next Maze Cell function. After the Maze Cell has been initialized along with the maze wall and maze passage in it, Maze Generation is finished, and will continue to other parts of the gameplay flowchart.

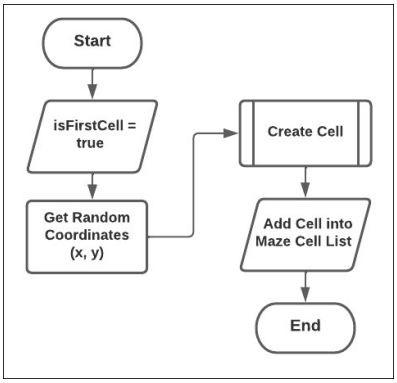

Figure 2: Maze Generation – Instantiate 1st Maze Cell Flowchart

Figure 2 shown the flowchart for the instantiate of the first maze cell in the maze generation. In this step, a boolean variable isFirst will be set to true before a random coordinate is retrieved and the Create Cell function is executed. Maze cells that have been created will be added to a MazeCell List to be used when initializing components from other mazes such as maze walls and maze passages.

Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

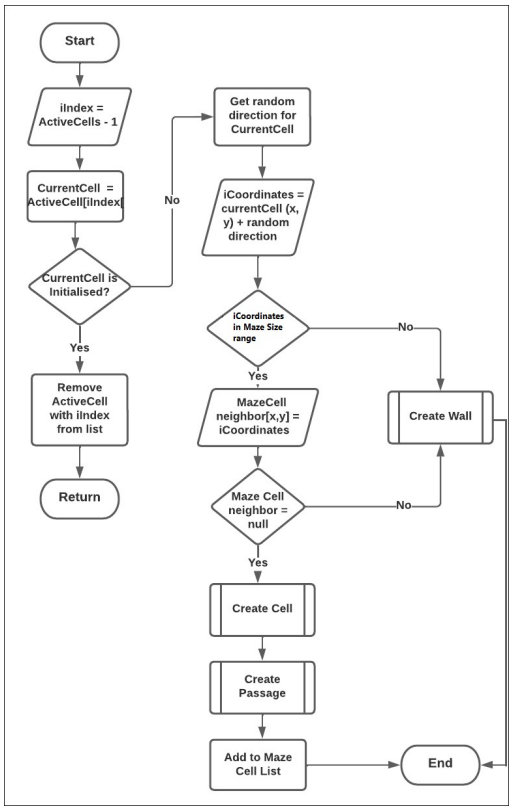

Figure 3: Maze Generation – Instantiate Maze Cells

The flowchart in Figure 3 shows the Maze Cell Instantiate function in Maze Generation. After selecting the last maze cell in the list, the maze cell will be checked whether it has finished initializing the position of the maze wall or the possible maze passage in that cell. If the maze cell has finished initializing, the maze cell will be deleted from ist and this function will finish with a return. If the maze cell has not finished initializing, the maze cell will check the condition of the neighbouring maze cells that are 4-adjacent to the maze cell. An iCoordinate variable will be created which contains the maze cell coordinates which add a random direction to the 4-adjacent position with the maze cell, if the icoordinate is outside the maze size, that means the initial maze cell is on the edge of the maze, and the function will only form a maze wall as the border of the maze. If iCoordinate is in the maze range, it will be checked whether the maze cell position already has a maze cell or not. If a new cell maze has not been formed, Create Cell and Create Maze Passage will be performed on the maze cell neighbour. If a new maze cell has been formed, the Create Wall function will be performed.

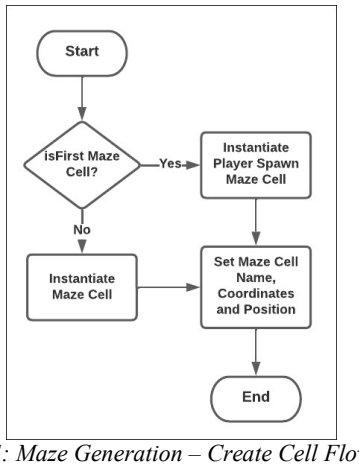

Figure 4: Maze Generation – Create Cell Flowchart

The flowchart in Figure 4 is a flowchart for the Create Maze Cell function in Maze Generation. For this function, the boolean variable isFirst is checked whether it contains a true or false value. If isFirst is true, then the maze cell to be initialized is the maze cell that contains the game object player spawner. If isFirst is false, the maze cell to be initialized is a normal maze cell. After the maze cell is initialized, the coordinates, name and position of the maze cell will be determined according to the value of the given variable.

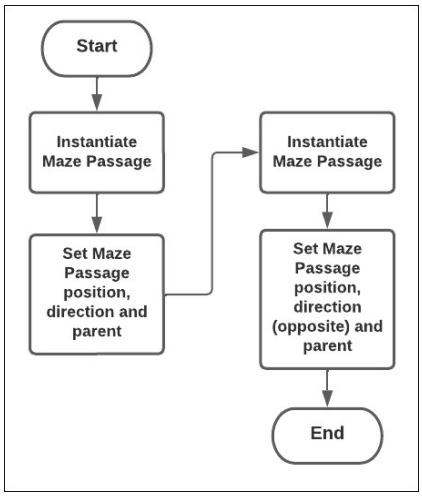

Figure 5: Maze Generation – Create Passage Flowchart

The flowchart in Figure 5 is a flowchart for the Create Maze Passage function in Maze Generation. Maze passage is part of a maze that does not contain a maze wall or is just an ordinary road in a maze. In this function the maze passage will be initialized in a maze cell, then the position, direction and parent of the maze passage will be determined. After that, a new maze passage will be initialized in the maze cell and the position, direction and parent will be determined, where the

Journal of Theoretical and Applied Information Technology
15th May 2023. Vol.101. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

direction specified this time is the opposite direction from the previous maze passage.

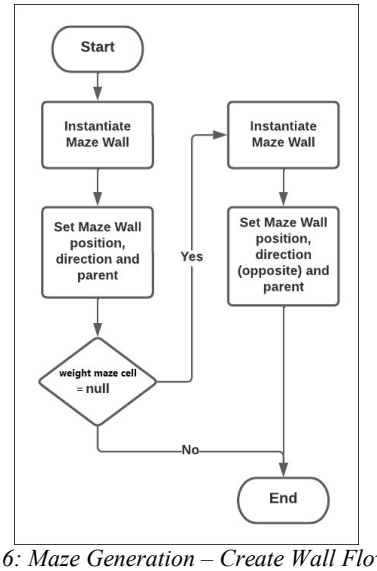

Figure 6: Maze Generation – Create Wall Flowchart

The flowchart in Figure 6 is a flowchart for the Create Maze Wall function in Maze Generation. In this function the maze wall will be initialized in a maze cell, then the position, direction and parent of the maze wall will be determined. After that, will be check the weight or mazecell neighbor of the maze cell with the 4-adjacent specified in the maze cell neighbor. If the weight is empty, a new maze wall will be initialized and placed in the same position and parent, but in the opposite direction.

There are also several series of functions designed to determine the wall position that each cell will have in the maze, where the data will be stored for the purposes of searching for maze generation results using Procedural Content Generation based on Prim's Algorithm as shown in Figure 7.

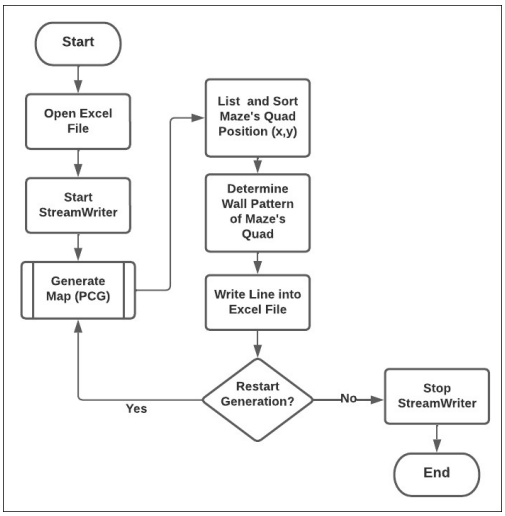

Figure 7: Detect Wall Pattern Method Flowchart

The flowchart in Figure 7 is a flowchart for commands used to search, analyze and store data on the number and position patterns of walls in each box in the maze generation results. The results of the analysis are stored in a comma-separated values file (.csv) which can then be processed in the form of a spreadsheet.

4. RESULT AND DISCUSSION

Figure 9 shows the implementation of prims algorithm to generate the maze in the game forms.

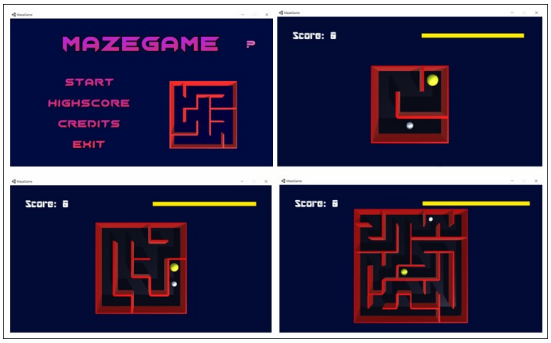

Figure 8: MazeGame Preview

Figure 8 shows the screenshots taken as the results of development of MazeGamze. In MazeGame, there are 3 difficulties, Easy mode, Medium mode, and Hard mode. With each difficulty, the size of the maze will increase accordingly, with 3 x 3 maze in Easy mode, 5 x 5 in Medium mode, and 8 x 8 in Hard mode.

MazeGame is a single player, 3D game in which the player's objective is to get to the highest score possible before the time runs out. As the

players collects more score, the time will also increase for a fraction to prolong the gameplay.

Table 1: Spanning Tree Number for n x n Grid.

Font: Times
Size: 9 pt
Style: italics

n x n grid	Spanning Trees
1x1	1
2x2	4
3x3	192
4x4	100352
5x5	557568000

Table 1 shows the data found about the possible spanning tree possible in n x n grid [12]. With this information, we can convert the number as the possible maze created in n x n grid, as the maze created is a spanning tree result. With that information, there will be testing phase using maze with the generation of 250 mazes with the size of 2x2, 3x3 and 4x4 to determine whether the Detect Wall Pattern method is successfully developed

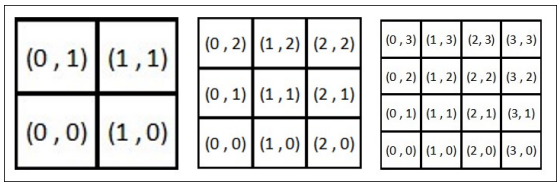

Figure 9: Maze's Coordinate of 2x2, 3x3 and 4x4 size.

Figure 9 shows the positions and coordinates for the maze cells in a maze with maze sizes of 2x2, 3x3, and 4x4 for each maze generation performed. This coordinate data will be collected and compiled in the Detect Wall Pattern Method before a function will be carried out to detect the number of maze walls and direction in each existing maze cell.

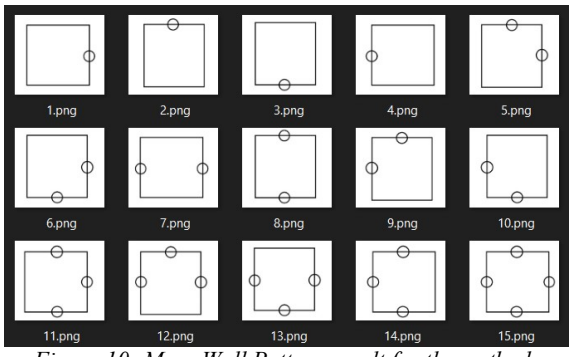

Figure 10: Maze Wall Pattern result for the method

Figure 10 shows a sequence of patterns on the maze wall that can be generated on every maze cell during the maze generation. The pattern is arranged based on the number of maze walls and the position of the maze walls owned by the maze cell. Once the pattern of a maze cell is determined, the data will be entered into a comma-separated e in the order that corresponds to the the maze cell in the maze generated by the maze.

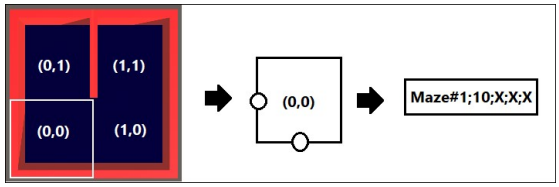

Figure 11: Detect Wall Pattern Method Steps

Figure 11 shows the simplified explanation on how Detect Wall Pattern method works. The coordinate of a maze cell will be determined, alongside that the number of maze wall on the cell will be counted. Depending on their position, there will be a number set on the maze cell as the result that will be written down into the maze data. This will be iterated until all maze cells are determined, and then will be re-iterated until all 250 mazes' data have been written down. The result will be then processed on a spreadsheet and will be determined how many unique mazes are generated on the 250 generation on each mazes size.

Table 2: Detect Wall Pattern Result

	Numbers maze pattern	Duplicates
250 Mazes (2x2) Generated	1	66
	2	64
	1	56
With total of 4 unique mazes generated		
250 mazes (3x3) Generated	1	9
	1	8
	5	7
	4	6
	6	5
	12	4
	17	3
	12	2
	21	0
With total of 79 unique mazes generated		
250 mazes (4x4) Generated	7	2
	236	0
With total of 243 unique mazes generated		

Table 2 shows the final result of Detect Wall Pattern method that was implemented on the maze generation. By comparing the calculation results with the data on Table 2, it can be determined that this method can detect 4 unique patterns in the 2x2 maze size according to the number that should be in 250 maze generation, at least 79 unique mazes with 3x3 maze; which are below the 192 patterns in the

data, and there are at least 243 unique mazes generated from the 4x4 mazes generated; which are far from the amount of data, 100352. From the results obtained, it is evident that the generated maze produces maze data that is in accordance with the number of unique patterns that should be or below the limit of the possible pattern.

Evaluating the level of video game satisfaction on MazeGame is done by doing a playtest and filling out a questionnaire with the GUESS scale as a form of review from MazeGame. The questionnaire mentioned was made through Google Form which contains 2 sections. The first section contains questions about general information of the respondents (name and email), and some brief questions about gameplay on MazeGame. Section 2 contains 37 questions from GUESS based on subscales that can be assessed in MazeGame, Usability/Playability, Play Engagement, Enjoyment, Audio Aesthetic, Personal Gratification, and Visual Aesthetic. The playtest was closed with 36 respondent data collected and from the answers obtained, an evaluation was carried out.

Based on the data collected, it was noted that 18 out of 36 respondents were more likely to play or prefer to play Medium difficulty on MazeGame. In addition, 10 out of 36 respondents chose Easy difficulty, and 8 out of 36 respondents chose Hard difficulty. 26 out of 36 respondents liked the MazeGame difficulty mode variation. 24 out of 36 like maze generation in gameplay, and 20 out of 36 respondents like the replayability factor given by maze generation and difficulty in MazeGame. 16 out of 36 respondents liked the UI appearance on MazeGame, with 14 out of 36 respondents liked the Audio Game and 11 out of 36 respondents liked the MazeGame Visual Game. For another additional opinion by respondents, 1 respondent added the name MazeGame as one of the things they like about MazeGame. 11 out of 36 respondents stated that difficulty mode and maze generation are still not sufficient as factors for the replayability of MazeGame. 9 out of 36 respondents did not like Visual Games and 5 out of 36 respondents did not like Audio Games. 6 out of 36 respondents did not like the look of the MazeGame UI or thought the navigation from MazeGame was still too confusing. 7 of 36 respondents considered the results of maze generation to make gameplay too difficult, and 6 of 36 respondents considered that difficulty mode makes gameplay between too difficult, too easy, or does not change the game experience they have. In addition, some additional opinions from other respondents are that there is no option to adjust game audio, lack of variations such as the implementation of buffs or nerfs during gameplay, timers that are too fast, and color selection (visuals) in gameplay. In addition, 2 out of 36 respondents chose to answer 'nothing' in what they disliked in MazeGame.

The questionnaire in section 1 has questions regarding suggestions that can be given to MazeGame. From the answers, the suggestions given include improvements to the UI, game visuals, game audio, difficulty, and gameplay. The most dominant suggestion regarding UI and navigation is about the placement of the How to Play button which is too easy to miss if the player doesn't pay attention in the Main Menu. For Game Visuals, respondents suggested a variety of colors or materials for existing mazes, game object players and pellets to reduce monotonous gameplay. Control on Game Audio is also recommended, so that respondents can adjust the volume of the audio (BGM and SFX). The implementation of enemy and obstacle is also suggested as a form of adding replayability into the gameplay, along with suggestions to provide buffs and nerfs in the maze during gameplay to increase or decrease the level of difficulty in the gameplay. In addition, respondents also gave suggestions to add game modes in MazeGame and add competitive game modes with other players. At the end of section 1, 18 stated that there was a possibility to play MazeGame again, 9 stated that they would play MazeGame again, 8 chose neutral, and 1 stated that they might not play MazeGame again.

Table 3: Guess Result

Subscale	Result
Usability/Playability	88.13% (Very Good)
Pay Engrossment	60.12% (Pretty Good)
Enjoyment	73.02% (Good)
Audio Aesthetic	80.75% (Good)
Personal Gratification	78.77% (Good)
Visual Aesthetic	80.16% (Good)
GUESS Result	76.82% (Good)

Based on the result on Table 3 and from the data collected at the section 2 of the questionnaire it is concluded that the Usability/Playability construct has the highest value of 88.13% and the Play Engagement construct has the lowest value of 60.12%. With a value of 76.82% as the final result of game user satisfaction, which is calculated through the results of the questionnaire, it can be judged that MazeGame gets the 'Good' predicate on the GUESS scale.

ACKNOWLEDGEMENT

Thank you to the Universitas Multimedia Nusantara, Indonesia which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia.

REFERENCES:

[1] A. W. Istiono and A. Waworuntu, "What element that influence preschool and elementary school children to enjoy playing education games ?," *International Journal of Advanced Studies*, vol. 9, no. 12, pp. 9–13, 2021.

[2] D. Callele, E. Neufeld, and K. Schneider, "Emotional requirements in video games," *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 299–302, 2006, doi: 10.1109/RE.2006.19.doi:10.1109/RE.2006.19

[3] R. Adellin, C. T. Khuan, and L. D. Gertrude, "Conceptual Framework Puzzle Game with High Replayability," *Journal of Physics: Conference Series*, vol. 1228, no. 1, 2019, doi: 10.1088/1742-6596/1228/1/012070.doi:10.1088/1742-6596/1228/1/012070

[4] M. Nkadimeng and P. Ankiewicz, "The Affordances of Minecraft Education as a Game-Based Learning Tool for Atomic Structure in Junior High School Science Education," *Journal of Science Education and Technology*, vol. 31, no. 5, pp. 605–620, 2022, doi: 10.1007/s10956-022-09981-0.doi:10.1007/s10956-022-09981-0

[5] R. yang and C. He, "Maze Adventure: An Application of Maze Algorithm in Role-playing Game Development by Python," 2022, doi: 10.4108/eai.17-6-2022.2322876.doi:10.4108/eai.17-6-2022.2322876

[6] A. Koesnaedi and W. Istiono, "Implementation Drunkard ' s Walk Algorithm to Generate Random Level in Roguelike Games," *International Journal of Multidisciplinary Research and Publications*, vol. 5, no. 2, pp. 97–103, 2022, [Online]. Available: Drunkard's Walk, Guest User Satisfaction Scale, Procedural Content Generation, Video game.

[7] B. M. F. Viana and S. R. Dos Santos, "A Survey of Procedural Dungeon Generation," *Brazilian Symposium on Games and Digital Entertainment, SBGAMES*, vol. 2019-Octob, pp. 29–38, 2019, doi: 10.1109/SBGames.2019.00015.doi:10.1109/SBGames.2019.00015

[8] N. Brewer, "Computerized Dungeons and Randomly Generated Worlds: From Rogue to Minecraft," *Proceedings of the IEEE*, vol. 105, no. 5, pp. 970–977, 2017, doi: 10.1109/JPROC.2017.2684358.doi:10.1109/JPROC.2017.2684358

[9] M. González-Hermida, E. Costa-Montenegro, B. Legerén-Lago, and A. Pena-Giménez, "Study of Artificial Intelligent Algorithms Applied in Procedural Content Generation in Video Games," *Eludamos: Journal for Computer Game Culture*, vol. 10, no. 1, pp. 39–54, 2020, doi: 10.7557/23.6171.doi:10.7557/23.6171

[10] A. Łukaszewski and Ł. Nogal, "Multi-sourced power system restoration strategy based on modified Prim's algorithm," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 69, no. 5, pp. 1–12, 2021, doi: 10.24425/bpasts.2021.137942.doi:10.24425/bpasts.2021.137942

[11] Y. Nagata, A. Imamiya, and N. Ono, "A genetic algorithm for the picture maze generation problem," *Computers and Operations Research*, vol. 115, p. 104860, 2020, doi: 10.1016/j.cor.2019.104860.doi:10.1016/j.cor.2019.104860

[12] Paryati and S. Krit, "Utilization of the Prim Algorithm to Determine the Nearest Path Car Transportation Problems of Goods Carrier Box," *ITM Web of Conferences*, vol. 43, p. 01005, 2022, doi: 10.1051/itmconf/20224301005.doi:10.1051/itmconf/20224301005

[13] A. Łukaszewski, Ł. Nogal, and M. Januszewski, "The Application of the Modified Prim's Algorithm to Restore the Power System Using Renewable Energy Sources," *Symmetry*, vol. 14, no. 5, 2022, doi: 10.3390/sym14051012.doi:10.3390/sym14051012

[14] P. Gabrovsek, "Analysis of Maze Generating Algorithms," *The IPSI BgD Transactions on Internet Research*, vol. 15, no. 1, pp. 26–33, 2019.

[15] P. H. Kim, J. Grove, S. Wurster, and R. Crawfis, "Design-centric maze generation," *ACM International Conference Proceeding Series*, 2019, doi: 10.1145/3337722.3341854.doi:10.1145/3337722.3341854

[16] J. R. Keebler Assoc, W. J. Shelstad, D. C. S. Google, B. S. Chaparro, and M. H. Phan Google, "Validation of the GUESS-18: A Short Version of the Game User Experience Satisfaction Scale (GUESS)," *Journal of Usability Studies*, vol. 16, no. 1, pp. 49–62, 2020.