Jurnal Pendidikan Multimedia (EDSENCE)

Journal homepage: https://ejournal.upi.edu/index.php/Edsence

Maze Game Development Using Procedural Content Generation for Maze Creation Automation

Aldy Nouval Kurnia[1], Ni Made Satvika Iswari [2], Made Adi Paramartha Putra[3]

[1,2,3] Universitas Primakara, Denpasar, Indonesia
Correspondence: E-mail: aldynvl38@gmail.com

ABSTRACT

This study implements a Depth First Search (DFS) algorithm for automatic perfect maze generation in a maze-themed game developed using Unity and the Game Development Life Cycle (GDLC) framework. Unlike previous studies that primarily focused on the correctness of maze generation, this study's tension evaluation includes algorithm performance testing, maze uniqueness evaluation, and user playability assessment in a fully playable game. Performance testing shows that the system consistently generates perfect mazes of various sizes up to 100x100 in less than three seconds. Maze uniqueness evaluation using different random seeds indicates a non-repeating maze structure and diverse solution paths. Furthermore, beta testing involving 31 users yielded a playability score of 91%, indicating high levels of usability and player acceptance. These results demonstrate that DFS-based PCG is not only computationally efficient but also suitable for practical game development and educational applications.

ARTICLE INFO

Article History:
Submitted/Received 05 Jul 2025
First Revised 10 Aug 2025
Accepted 10 Nov 2025
First Available online 11 Dec 2025
Publication Date 11 Dec 2025

Keyword:
Game Development,
Maze,
Procedural Content Generation,
Depth First Search.

1. INTRODUCTION

Over the past six decades, innovation in video game development has skyrocketed. From what were initially simple pixels that moved in two directions, we can now enter these virtual worlds with virtual reality technology (Banfi, 2021). According to data from 1970 to 2020, revenue generated from the video game industry reached US$2,548 billion. And by 2024, the

global video game industry is targeted to generate revenue of US$187.7 billion and is expected to increase annually (Buijsman, 2024). In Indonesia itself, where the video game industry is still relatively new, it managed to contribute 31.25 trillion rupiah to GDP in 2021 with a growth rate of 9.17 percent, this is the second-highest growth rate in the creative industry subsector (Kemenparekraf, 2023).

In 2024, video game platform segments were grouped into three: personal computer, console, and mobile. Mobile was the largest segment at 49 percent, followed by console at 28 percent, and personal computer at 23 percent. Revenue from mobile platforms was the largest, at US$9.2 billion, growing at 3 percent per year (Buijsman, 2024). The game market is also divided into various genres, one of which is the casual game genre. In 2023, the casual game genre accounted for 86.9 percent of installs, with the puzzle subgenre accounting for 31.3 percent (Liftoff, 2024).

Mazes are a popular theme within this puzzle subgenre (Li, 2023; Yuniasih et al., 2023). However, when it comes to creating custom mazes, there are often limitations in providing the uniqueness needed to maintain player focus. This makes mazes predictable and reduces the game's immersion(Juwiantho et al., 2023). Furthermore, mazes come in several types, one of which is the perfect maze (Setiadharma et al., 2020). According to a study titled " How to generate perfect mazes," a perfect maze requires a starting point, an ending point, a solution path, intersections, and dead ends (Bellot et al., 2021).

One way to address this problem is through automated Procedural Content Generation (PCG) (Lazaridis & Fragulis, 2024; Zafar et al., 2021). Implementing PCG in games has become a popular way to solve this problem through automated game content generation (Zhang et al., 2022). One use of PCG is to automatically create mazes using a maze generation algorithm. According to a study, several algorithms can be used to automatically create mazes: Kruskal's, Prim's, Aldous-Broder, Hunt and Kill, Wilson's, Eller's, and Depth First Search (DFS). DFS is found to be the best algorithm for creating mazes (Mane et al., 2023; Yang et al., 2024).

Although Depth First Search has been widely studied as a maze generation algorithm, most previous research emphasizes algorithmic correctness or theoretical comparison rather than practical implementation within a complete game development process. Recent studies have compared multiple maze generation methods in playable environments, highlighting performance differences and practical considerations for game-based applications and broader trends in PCG research (Čarapina et al., 2024; Viana & Dos Santos, 2021). Furthermore, comparative evaluations of PCG techniques have been conducted for diverse game map layouts, evidencing growing interest in empirical analysis of procedural algorithms (Alyaseri et al., 2025; Kalafatis et al., 2025). Therefore, there is a need for research that integrates DFS-based PCG into a full game development life cycle and evaluates both technical performance and user experience.

Based on the identified research gap, this study focuses on the implementation of Procedural Content Generation (PCG) using the Depth First Search (DFS) algorithm within a complete maze game developed using the Game Development Life Cycle (GDLC) framework. This research aims to evaluate the effectiveness of DFS-based PCG in terms of maze generation performance, maze uniqueness, and game usability. The results of this study are expected to provide practical insights for game developers and educators in implementing automated content generation for game applications.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

2. METHODS

2.1. Game Development Life Cycle (GDLC)

In this study, the research method used is the Game Development Life Cycle (GDLC). Game Development Life Cycle (GDLC) is a framework used to guide the game development process from the initial stage to completion (Rusmana et al., 2023) This study uses the GDLC method developed by Rido Ramadan and Yani Widyani as a framework in the game development process, this process begins with initiation, pre-production, production, testing, beta, and release.

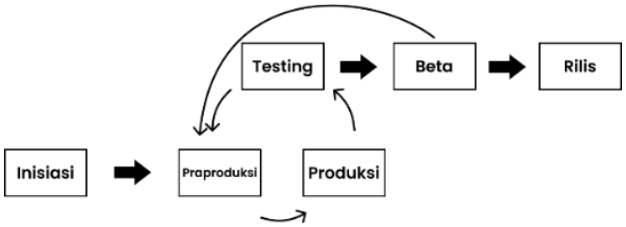


Figure 1. GDLC flow

a. Initiation

The initiation phase involves creating a brief description of the game's basic concept, determining the game type, defining the game's objectives, and selecting the game engine to be used.

b. Pre-production

Pre-production is a crucial stage in the production cycle. During this stage, the developer gathers references for the game to be developed. The pre-production phase begins with the creation of a Game Design Document (GDD) and the creation of a prototype.

c. Production

Production is the process where iterations in video game development occur. This process revolves around asset creation, scene management, and the integration of these elements into the game engine. This stage involves three processes: asset listing and scene creation.

d. Testing

Initial testing, or alpha testing, is the process of verifying operational functionality and game playability, ensuring the game runs without bugs. In this study, the testing method used was internal playtesting. Internal playtesting is a testing method conducted by the development team before the game is tested by external testers (Abeele et al., 2020).

e. Beta

Beta testing is a testing phase conducted by a third party to obtain player feedback and uncover bugs and errors in a game that may have been missed during development and alpha testing. The beta testing in this study used a closed beta playtesting method, a testing method in which only selected users can try the product (Eko Budi Susanto et al., 2023).

f. Release

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

Release is the final stage of game development where developers will release their game to the public.

2.2 Evaluation Metrics

The effectiveness of the DFS-based PCG implementation was evaluated using four metrics:

a. Generation Time, measured in milliseconds for different maze sizes;
b. Maze Validity, defined as the existence of exactly one solution path (perfect maze);
c. Maze Uniqueness, evaluated by comparing solution paths generated using different random seeds; and
d. Usability, measured through beta testing using a Likert-scale questionnaire and descriptive percentage analysis.

3. RESULTS AND DISCUSSION

3.1. Initiation

Based on the initiation, the following results were determined. The video game genre being developed is a casual game with a puzzle subgenre and a maze theme. In this game, players are tasked with navigating their character from the starting point to the end point.

3.2. Pre-production

The pre-production phase begins with the creation of a Game Design Document (GDD) and the creation of a prototype.

a. Game Design Document

The Game Design Document (GDD) in this research covers the basic concepts, game mechanics, and production techniques. The GDD can be seen in Tables 1 to 3.

Table 1. Game overview

No	Aspect	Description
1	Title	Mazy
2	Platform	PC
3	Genre	Casual puzzle
4	Theme	Maze
5	Audience	Casual player

Table 2. Game mechanics

No	Game Mechanics	Description
1	Player control	Players navigate their characters up, down, left, and right using the keyboard (WASD)/arrow keys and gamepad/joystick.
2	Objective	Players navigate their characters from the starting point to the end point within the allotted time.
3	Rules	1. Players complete each maze within the allotted time. 2. The maze size increases by one row and column for each completed maze.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

No	Game Mechanics	Description
		3. The game ends and the player's score is calculated if the character gives up or is unable to find a way out of the maze within the allotted time. 4. The time limit is calculated based on the number of rooms/nodes from the starting point to the end point, calculated by the pathfinding algorithm, where 1 room/node corresponds to 1 second.

Table 3. Production tolls

No	Production Software	Software Used
1	*Game engine*	Unity 6 6000.0.24f1
2	*Programming tools*	Visual studio 2022
3	*Visual development tools*	Figma dan Illustrator
4	*Audio development tools*	Audacity

b. Prototyping

The prototyping phase of this research focuses on the development of a maze generation algorithm. The implementation of the maze generation algorithm consists of the following stages**.**

a) Grid Initiation

A maze grid consists of a 2D array represented by a collection of interconnected rooms. Each room has three states: unvisited, active, and visited. An illustration of the grid can be seen in Figure 2.

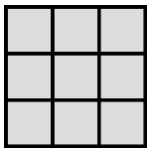


Figure 2. Grid Initiation

b) Initial Path Initiation

The algorithm starts with a random room, marked as visited, and pushed onto the stack. The algorithm then searches for nearby unvisited neighbors and randomly selects one to visit, mark as visited, and push onto the stack. An illustration of this process can be seen in Figure 3.

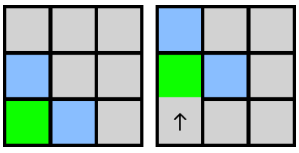


Figure 3. Initial Path Initiation

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

c) Looping Process

The algorithm then repeats the same process until there are no more unvisited neighbors.

d) Backtracking Process

When the algorithm reaches a dead end, where there are no more unvisited neighbors, the algorithm will backtrack by popping the top of the stack until it finds an unvisited neighbor. An illustration of this process can be seen in Figure 4.

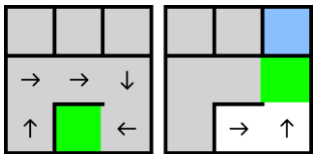


Figure 4. Backtracking Process

e) Maze Finish

When there are no more unvisited rooms, the algorithm will pop the remaining stack and the looping process ends. An illustration of this process can be seen in Figure 5.

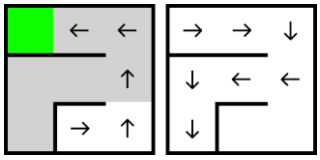


Figure 5. Maze Finish

3.3. Production

At this stage there are three processes in making this game, namely asset listing and scene creation.

a. Asset listing

There are three types of assets used in the game, as shown in Tables 4 through 6.

Table 4. Visual Assets

Name	Format	Size	Description
Character	png	946 x 946 (21 KB)	Spriterenderer2d for characters
Wall Vertical	png	818 x 144 (11 KB)	Spriterenderer2d for vertical room walls (top, bottom)
Wall Horizontal	png	129 x 824 (9 KB)	Spriterenderer2d for horizontal room walls (right, left)
ButtonUI	png	1080 x 1080 (20 KB)	Spriterenderer2d for ButtonUI
Trigger Finish	png	1080 x 1080 (40 KB)	Spriterenderer2d for finish trigger
Trigger Stop	png	1080 x 1080 (36 KB)	Spriterenderer2d for stop trigger
Stopwatch	png	1080 x 1080 (51 KB)	Spriterenderer2d for timeout indicator

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

Table 5. Audio Assets

Name	Format	Duration	Description
Button Hover	mp3	00:01 (5 KB)	Audio Source for ButtonUI buttons on hover
Button Click	mp3	00:01 (5 KB)	Audio Source for ButtonUI buttons on click
Background Music	mp3	02:47 (3,3 MB)	Audio Source for background music
Character Moving	mp3	00:01 (3 KB)	Audio Source for character movement

Table 6. Font Assets

Name	Type	Size	Source
Beauty Faces	Handwriting	20 KB	Dafont.com
Liberation Sans	Sans Serif	131 KB	Unity (*default*)

b. Scene Creation

This game uses two scenes: the Main Scene and the In-Game Scene.

a) Main Scene

The Main Scene is the first scene displayed when the player opens the game. The Main Scene contains four buttons that can be interacted with: Continue, New Game, Settings, and Exit. This scene contains the GameObject canvas, which consists of the main UI display, the settings display, and the new game confirmation display.

Figure 6. Main menu display

b) In-Game Scene

The In-Game Scene is the primary environment where gameplay takes place. It contains essential game components, including the maze generation, player character, camera, level controller, and audio controller. Integrating Procedural Content Generation (PCG) into the game architecture enables dynamic maze generation without the need for manual

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

level design, thereby reducing development time while preserving replayability (Hafis et al., 2019). By generating maze structures at runtime, the game minimizes repetitive layouts and progressively adjusts difficulty through maze size scaling. This design approach enhances player engagement and provides a scalable foundation for future content expansion.

Figure 7. Ingame display

3.4. Testing Result

In this game testing, the things tested were the functionality of the maze generation function using an artificial intelligent pathfinding algorithm (Putra & Utami, 2025; Satriantoro & Iswari, 2016), the uniqueness of the maze and the function of the components in the game. The test results can be seen in tables 7 to 9.

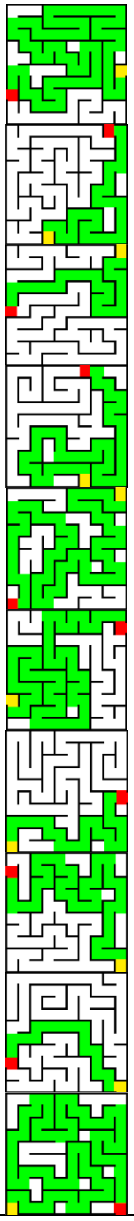
Table 7. Maze generation test

No	Maze Size	Generate time (millisecond)	Pathfinding	Result
1	5x5	1	9	Success
2	10x10	1	27	Success
3	15x15	3	83	Success
4	20x20	6	193	Success
5	25x25	12	301	Success
6	30x30	20	377	Success
7	35x35	34	515	Success
8	40x40	56	225	Success
9	45x45	85	1071	Success
10	50x50	138	575	Success
11	55x55	260	1329	Success
12	60x60	266	935	Success
13	65x65	365	1345	Success
14	70x70	494	1073	Success
15	75x75	648	1699	Success
16	80x80	805	1757	Success
17	85x85	1470	1523	Success
18	90x90	1232	1429	Success
19	95x95	1515	3009	Success
20	100x100	1958	1889	Success

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

The results of the maze generation test in Table 7 can be concluded that the algorithm was successfully implemented and was able to create mazes of various sizes, had one valid solution path, and was able to create mazes in less than 3 seconds. A direct experimental comparison with alternative maze generation algorithms was not conducted, as this research focuses on validating DFS performance within a complete game implementation rather than algorithm benchmarking. Previous comparative analyses have shown DFS to achieve competitive or superior generation performance compared to algorithms such as Prim's and Kruskal's for perfect maze construction (Mane et al., 2023; Vijaya et al., 2024).

Table 8. maze similiarity test

No	Seed	Start	End	Path Length	Maze form
1	-386683321	(9,4)	(0,2)	68	
2	1965309184	(3,9)	(8,0)	29	
3	-1703715913	(9,9)	(0,4)	25	
4	72131745	(6,0)	(6,9)	42	
5	-669310641	(9,9)	(0,0)	63	
6	370642451	(0,2)	(9,8)	58	
7	655939688	(0,0)	(9,4)	24	
8	892327005	(9,0)	(0,8)	44	
9	-746262399	(9,0)	(0,2)	18	
10	-740311060	(0,0)	(9,0)	74	

From the tests in Table 8 using 10 different random seeds, the algorithm generates varied starting and ending positions and produces distinct solution paths for each maze instance.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

The “Maze form” column represents a visual snapshot of each generated maze, yellow as start point, red as end point, and green as solution path. In these visualizations, identical maze structures or paths are not observed across different seeds, indicating that the procedural generation process does not repeat maze configurations and confirms that each seed produces a unique maze structure and solution.

Table 9. Usability testing

No	Component	Funtion	Metric of succes	Status
1	Character	Navigation	Move based on input keys	Pass
2	Level controller	Creating Maze	Succesfully generates a perfect maze	Pass
		Saving Game Progress	Preserves level progress after exiting	Pass
		Spawning Characters	Places the character at the starting point of the maze	Pass
		Setting Time Limits	Provides a timer after the maze is generated	Pass
3	Camera	Displaying Visuals	Show visual display of user interface	Pass
		Tracking Character Targets	Camera follows the moving character	Pass
4	Audio controller	Background Music	Background music plays	Pass
		Hovering Buttons	Button plays a sound on mouse hover	Pass
		Clicking Buttons	Button plays a sound on mouse click	Pass
		Character Movement	plays sound as character moving	Pass

From the results of testing the supporting components in Table 9, it can be concluded that these components are able to run well. No functional errors or crashes were observed during testing, indicating stable component interaction throughout gameplay.

3.5. Beta Testing Result

In the beta testing phase, the measurement tool used was a questionnaire with a Likert scale, which is an attitude measurement in which respondents are asked to state their level of agreement with a statement. Selected testers will play a game and complete a four-question questionnaire with a firm scale from strongly agree (4) to strongly disagree (1). The results of the examiner's responses were then analyzed using a descriptive percentage method using formula (1).

$$\text{Eligibility} = \frac{\text{Test result score}}{\text{Maximum score}} \times 100\% \qquad (1)$$

Furthermore, the results of the test score calculations will be used to assess the feasibility of the game that has been created using the benchmarks in table 10.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

Table 10. Eligibility benchmark

No	Percentage	Eligibility
1	76% – 100%	Very playable
2	51% – 75%	Playable
3	26% – 50%	Unplayable
4	<25%	Very unplayable

Table 11. Beta test result

Questions	Response				Total Score	Maximum Score	Percentage
	1	2	3	4			
Is the game's objective easy to understand?	0	0	8	23	116	124	94%
Does the game feel casual to play?	0	0	10	21	114	124	92%
Is the character control responsive?	0	0	15	16	109	124	88%
Is the user interface easy to understand?	0	0	10	21	114	124	92%
Average							91%

Based on testing conducted on 31 testers, the game was deemed highly playable, with a 91% pass rate. Responses to the questionnaire given to testers are shown in Table 11.

3.6. Release Platform

After completing its testing phases, the game Mazy was released publicly on the itch.io platform. During this launch period, it received 89 page views and 51 downloads. Research in game analytics highlights that user acquisition metrics—such as the ratio of views to downloads—are valuable indicators of early user interest and the viability of public release, specially for indie titles where feedback volume is limited (Su et al., 2021). Based on these results, Mazy demonstrates strong potential and is well-suited for public distribution.

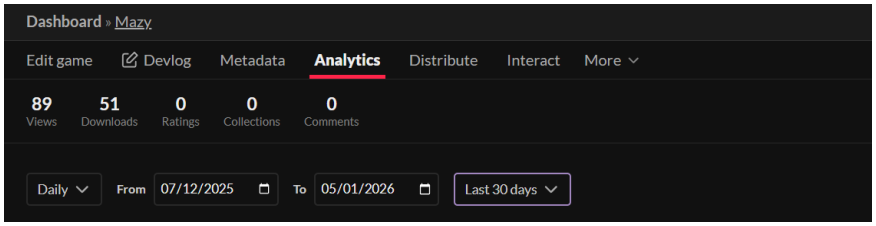


Figure 8. Post-release analytics of Mazy on the itch.io

4. CONCLUSION

This study demonstrates that Procedural Content Generation using the Depth First Search algorithm can be effectively applied to automate perfect maze creation in maze-themed games. Through integration within the Game Development Life Cycle framework, the

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

proposed system successfully generated unique and solvable mazes with consistent performance under three seconds for grids up to 100×100.

The main contribution of this research lies in validating DFS-based PCG not only from an algorithmic standpoint but also within a complete game development and usability context. User testing results indicate that the generated content supports a positive player experience, confirming its suitability for casual puzzle games.

However, this study is limited to a single maze generation algorithm and does not include direct empirical comparison with alternative approaches such as Prim's or Kruskal's algorithms. Future research may incorporate comparative performance analysis, adaptive difficulty mechanisms, and player behavior analytics to further optimize PCG-based maze design. Practically, this research provides a reference for game developers and educators seeking efficient level generation techniques for interactive learning and entertainment applications.

REFERENCES

Abeele, V. Vanden, Spiel, K., Nacke, L., Johnson, D., & Gerling, K. (2020). Development and validation of the player experience inventory: A scale to measure player experiences at the level of functional and psychosocial consequences. *International Journal of Human-Computer Studies*, *135*, 102370.

Alyaseri, S., Connor, A., & Sinha, R. (2025). Exploring Metaheuristic Algorithms for Enhanced Game Map Generation in Procedural Content Generation. *International Journal of Applied Metaheuristic Computing*, *16*(1), 1–33.

Banfi, F. (2021). The Evolution of Interactivity, Immersion and Interoperability in HBIM: Digital Model Uses, VR and AR for Built Cultural Heritage. *ISPRS International Journal of Geo-Information*, *10*(10), 685.

Bellot, V., Cautrès, M., Favreau, J.-M., Gonzalez-Thauvin, M., Lafourcade, P., Le Cornec, K., Mosnier, B., & Rivière-Wekstein, S. (2021). How to generate perfect mazes? *Information Sciences*, *572*, 444–459.

Buijsman, M. (2024, August 13). The global games market will generate $187.7 billion in 2024. *Newzoo*. https://newzoo.com/resources/blog/global-games-market-revenue-estimates-and-forecasts-in-2024

Čarapina, M., Staničić, O., Dodig, I., & Cafuta, D. (2024). A Comparative Study of Maze Generation Algorithms in a Game-Based Mobile Learning Application for Learning Basic Programming Concepts. *Algorithms*, *17*(9), 404.

Eko Budi Susanto, Paminto Agung Christianto, Mohammad Reza Maulana, & Nur Fadhilah. (2023). Closed Beta Testing on Filariasis Treatment Monitoring Applications. *International Journal of Information Technology and Business*, *5*(2), 06–11.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

Hafis, M., Tolle, H., & Supianto, A. A. (2019). A literature review of Empirical Evidence on Procedural Content Generation in Game-Related Implementation. *Journal of Information Technology and Computer Science*, *4*(3), 308–328.

Juwiantho, H., Liliana, L., & Budiono, M. (2023). Procedural Content Generation pada Game Tower Defense menggunakan Perlin Noise dan Algoritma Floyd Warshall. *Journal of Animation and Games Studies*, *9*(1), 11–28.

Kalafatis, E., Mitsis, K., Zarkogianni, K., Athanasiou, M., & Nikita, K. (2025). A Modular Framework for Automated Evaluation of Procedural Content Generation in Serious Games With Deep Reinforcement Learning Agents. *IEEE Transactions on Games*, *17*(4), 1003–1012.

Kemenparekraf. (2023, June 9). Perkembangan Industri Gim di Indonesia, Raih Penghargaan Internasional. *Kemenparekraf*. https://www.kemenparekraf.go.id/hasil-pencarian/perkembangan-industri-gim-di-indonesia-raih-penghargaan-internasional

Lazaridis, L., & Fragulis, G. F. (2024). Creating a Newer and Improved Procedural Content Generation (PCG) Algorithm with Minimal Human Intervention for Computer Gaming Development. *Computers*, *13*(11), 304.

Li, Y. (2023). Analyzing the Influences of Puzzle Games on Learners' Learning. *Journal of Education, Humanities and Social Sciences*, *22*, 111–116.

Liftoff. (2024). *Casual Gaming Apps Report*.

Mane, D., Harne, R., Pol, T., Asthagi, R., Shine, S., & Zope, B. (2023). An Extensive Comparative Analysis on Different Maze Generation Algorithms. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(2), 37–47.

Putra, M. A. P., & Utami, N. W. (2025). Pelatihan Pengembangan Model Artificial Intelligence Berbasis Website Bagi Siswa di SMK Negeri 1 Mas Ubud. *BERNAS: Jurnal Pengabdian Kepada Masyarakat*, *6*(2), 1663–1670.

Rusmana, R. A., Asriyanik, A., & Setiawan, I. R. (2023). Penggunaan Metode Game Development Life Cycle (GDLC) Untuk Memudahkan Belajar Bahasa Inggris Dalam Media Game. *Journal of Information System Research (JOSH)*, *4*(4), 1402–1412.

Satriantoro, B., & Iswari, N. M. S. (2016). Sistem Pendukung Keputusan Pemilihan Taksi Berbasis Web dengan Algoritma Pencarian Linear. *Ultimatics : Jurnal Teknik Informatika*, *7*(2), 148–154.

Setiadharma, E., Husniah, L., & Kholimi, A. S. (2020). Algoritma Maze Generator Recursive Backtracking Untuk Membuat Prosedural Labirin Pada Game Petualangan Labirin 3D. *Jurnal Repositor*, *2*(3), 373–384.

Su, Y., Backlund, P., & Engström, H. (2021). Comprehensive review and classification of game analytics. *Service Oriented Computing and Applications*, *15*(2), 141–156.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535

Viana, B. M. F., & Dos Santos, S. R. (2021). Procedural Dungeon Generation: A Survey. *Journal on Interactive Systems*, *12*(1), 83–101.

Vijaya, J., Gopu, A., Vinayak, K. V. S. G., Singh, T. J., & Malhotra, K. (2024). Analysis of Maze Generation Algorithms. *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 1–6.

Yang, K., Lin, S., Dai, Y., & Li, W. (2024). Optimization and comparative analysis of maze generation algorithm hybrid. *Applied and Computational Engineering*, *79*(1), 20–33.

Yuniasih, Loita, A., & Aprily, N. M. (2023). Permainan Maze Dalam Meningkatkan Kemampuan Membaca Pada Anak Usia Dini. *JECIE (Journal of Early Childhood and Inclusive Education)*, *7*(1), 64–71.

Zafar, A., Mujtaba, H., & Beg, M. (2021). Procedural Content Generation for General Video Game Level Generation. *International Journal of Intelligent Systems and Applications in Engineering*, *24*(68), 33–36.

Zhang, Y., Zhang, G., & Huang, X. (2022). A Survey of Procedural Content Generation for Games. *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, 186–190.

DOI: https://doi.org/10.17509/edsence.v7i2.88783
p-ISSN:2685-2489 | e-ISSN:2685-2535