Implementation of a Pathfinding-Based Intelligent Agent for Enemy Behavior in the "Run to BIPOL" Maze Tournament Game

Yoseph Satria Praka
Digital Multimedia Engineering
Jakarta State Polytehnic
Depok, Indonesia
yoseph.satria.praka.tik23@stu.pnj.ac.id

Candi Rahayu Tri Prameswari
Digital Multimedia Engineering
Jakarta State Polytehnic
Depok, Indonesia
candi.rahayu.tri.prameswari.tik23@stu.pnj.ac.id

Nurul Wahyuni
Digital Multimedia Engineering
Jakarta State Polytehnic
Depok, Indonesia
nurul.wahyuni.tik23@stu.pnj.ac.id

Ananda Maria
Digital Multimedia Engineering
Jakarta State Polytehnic
Depok, Indonesia
ananda.maria.tik23@stu.pnj.ac.id

Abstract*—This study aims to implement a pathfinding-based intelligent agent for autonomous enemy behavior in the web-based maze tournament game Run to BIPOL. The game features a competitive tournament in which one player competes against six AI-controlled opponents to reach limited objectives in procedurally generated maze environments. Development followed the Multimedia Development Life Cycle (MDLC), consisting of Concept, Design, Material Collecting, Assembly, Testing, and Distribution. The intelligent agent integrates Procedural Content Generation (PCG) for maze generation, Breadth-First Search (BFS) for accessibility validation, A* with the Manhattan Distance heuristic for pathfinding, and a Finite State Machine (FSM) for behavioral control. Black-box testing was conducted using Win and Lose scenarios to evaluate the functionality of the developed system. The results showed that the game components and intelligent agent operated successfully under both scenarios. The Win scenario recorded 123 A* executions, an average path length of 10.3 nodes, and 3,256 explored nodes, while the Lose scenario recorded 128 A* executions, an average path length of 8.5 nodes, and 2,156 explored nodes. These results demonstrate that the proposed intelligent agent can autonomously navigate dynamic maze environments and support competitive enemy behavior across multiple tournament stages. The study provides an implementation foundation for intelligent agents in browser-based competitive maze games.

***Keywords*—Artificial Intelligence, Maze Tournament Game, Breadth-First Search, Finite State Machine, A* Algorithm**

I. INTRODUCTION

Artificial intelligence (AI) has become an essential component in modern digital games, enabling non-player characters (NPCs) to exhibit autonomous behaviors that enhance player engagement and gameplay experience. In maze-based games, AI is commonly utilized to navigate complex environments, pursue objectives, and interact dynamically with players [1], [2], [3], [4]. Among various AI techniques, pathfinding has received significant attention because it enables game agents to determine efficient routes within grid-based environments while maintaining responsive movement [5], [6]. As web technologies continue to advance, browser-based games have become an attractive platform for implementing intelligent game agents due to their accessibility, cross-platform compatibility, and ease of deployment [7].

Previous studies have demonstrated the effectiveness of pathfinding algorithms in maze games. Arief et al. [8] implemented the A* algorithm in a web-based maze game with randomly generated maze patterns. Their study demonstrated that A* successfully identified the shortest path using the Manhattan Distance heuristic and improved navigation efficiency within procedurally generated mazes. However, the implementation primarily focused on assisting player navigation and visualizing the shortest path rather than controlling autonomous enemy behavior. The authors also suggested that future research could explore artificial intelligence for non-player characters (NPCs), adaptive gameplay, and comparisons of different pathfinding algorithms.

Similarly, Pratiwi [9] developed a desktop-based maze game using Unreal Engine in which an NPC employed the A* algorithm to determine the fastest route while competing with the main player. The implementation enabled the NPC to respond dynamically to player movement by pursuing the player through the calculated shortest route, demonstrating the potential of pathfinding to improve gameplay interaction. However, the NPC behavior primarily focused on pursuing a single player and did not address competitive scenarios involving multiple autonomous agents, dynamically changing objectives, or tournament-based decision-making.

Artificial intelligence has also been applied to educational maze games. Riganil et al. [10] developed an AI-based maze game as an interactive learning medium for geometric transformation. The game incorporated adaptive AI logic to present learning questions and achieved validity, practicality, and effectiveness scores of 82%, 85.6%, and 83.1%, respectively. However, the AI implementation primarily supported adaptive educational content rather than autonomous enemy behavior or competitive interactions among multiple game agents.

Received: July 03, 2026; received in revised form: July 06, 2026; accepted: August 26, 2026; available online: August 28, 2026.
*Corresponding Author

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

Based on these previous studies, existing maze games have predominantly utilized AI for navigation assistance, player pursuit, or educational adaptation. Research on intelligent agents operating in competitive maze environments remains limited, particularly in situations where multiple autonomous opponents compete simultaneously for dynamically generated objectives under tournament-based elimination rules. Such environments require agents to continuously adapt their navigation strategies, respond to changing game conditions, and make autonomous decisions throughout multiple stages of gameplay, making the problem more complex than conventional single-target pathfinding.

Therefore, this study proposes the implementation of a pathfinding-based intelligent agent for enemy behavior in a web-based maze tournament game entitled *Run to BIPOL*. Unlike previous studies, the proposed game introduces a tournament elimination system in which one player competes against multiple AI-controlled opponents to obtain randomly generated objectives within procedurally generated maze environments. Throughout each tournament stage, participants compete for limited objectives, resulting in the elimination of the slowest participant until only the player and one AI remain in the final stage. This competitive and dynamic environment requires intelligent agents to navigate efficiently, adapt to changing situations, and maintain autonomous behavior across multiple rounds. It is expected that the proposed implementation will contribute to the development of more adaptive enemy behavior and enrich the application of intelligent agents in competitive web-based maze games.

II. METHODS

This study employed the Multimedia Development Life Cycle (MDLC) as the research methodology to develop the web-based maze tournament game *Run to BIPOL*. MDLC was selected because it provides a systematic framework for multimedia application development, encompassing the planning, design, implementation, evaluation, and deployment phases. The model consists of six sequential stages: Concept, Design, Material Collecting, Assembly, Testing, and Distribution [11]. The overall research methodology is illustrated in Figure 1.

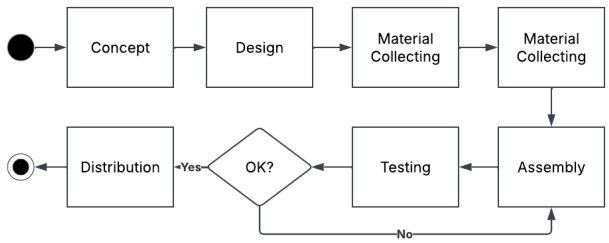


Fig. 1. Research Stage

A. Concept

The Concept stage defined the objectives, target users, gameplay mechanics, and intelligent agent implementation [12]. The primary objective of this study was to develop a web-based maze tournament game featuring autonomous enemy behavior through a pathfinding-based intelligent agent. The game was designed to provide a competitive environment in which one player competes against multiple AI-controlled opponents to reach randomly generated objectives within procedurally generated maze environments. During this stage, the functional requirements, gameplay rules, tournament system, and intelligent agent behavior were identified and documented as the foundation for subsequent development stages.

B. Design

The Design stage focused on designing the game architecture, gameplay flow, and intelligent agent mechanism [13]. The game interface was designed using a pixel-art style to provide a retro-modern visual experience while maintaining usability. System flowcharts, game state diagrams and navigation structures were prepared to describe user interaction and game processes. The intelligent agent behavior was also designed by defining its navigation strategy, target selection mechanism, and interaction with dynamically generated maze environments.

C. Material Collecting

During the Material Collecting stage, all multimedia assets required for game development were collected and prepared [14]. These assets included character sprites, maze tiles, background images, user interface components, audio files, and visual effects. All graphical assets were organized according to the game design specifications to ensure visual consistency throughout the application. Supporting resources such as pixel fonts and background music were also integrated into the asset collection.

D. Assembly

The Assembly stage involved implementing the entire game using HTML5, CSS3, and JavaScript with the Canvas API as the rendering engine. At this stage, all gameplay mechanics, including maze generation, player movement, intelligent enemy behavior, tournament progression, collision detection, user interface, and game state management, were integrated into a functional application. The intelligent agent continuously navigated the maze environment to compete for available objectives while adapting to changes occurring during gameplay. The application was developed in a modular structure to simplify implementation, maintenance, and future improvements.

E. Testing

Testing was carried out to assess whether the developed game functioned properly and whether the intelligent agent could operate as intended during gameplay. Black-box testing was used to examine the main features based on their expected inputs and outputs. The tested features included maze generation, player movement, tournament progression, enemy behavior, pause functionality, and match log generation [15]. Gameplay trials were also carried out to observe the behavior of the AI opponents across different maze layouts and tournament stages. Particular attention was given to the agents' ability to select targets, determine routes, move autonomously, and respond when the game situation changed. The resulting match logs were subsequently examined to obtain information about player and AI performance during the tournament.

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

F. Distribution

After the testing process was completed, the game was prepared for online deployment as a browser-based application. The source code, game assets, and supporting files were organized into a deployable package so that the game could be accessed directly through a modern web browser without requiring additional installation. The application was published through Netlify.

III. Result and Discussion

A. Concept

The Concept stage resulted in the definition of the game's objectives, target users, platform, gameplay mechanics, and intelligent agent requirements. The developed game, Run to BIPOL, is a web-based maze tournament game designed specifically for students of Politeknik Negeri Jakarta (PNJ). The game concept was inspired by the daily experience of PNJ students competing to catch the BIPOL (Bus Politeknik), a campus shuttle bus that is frequently crowded during class hours. This real-world phenomenon was transformed into a competitive gameplay scenario in which one player competes against multiple AI-controlled opponents to reach randomly generated BIPOL objectives within procedurally generated maze environments. In addition, the concept establishes a tournament-based elimination system, where one participant is eliminated in each stage until only the player and one AI opponent remain in the final round. The intelligent agent was designed to autonomously navigate the maze, compete for available objectives, and adapt its behavior to the dynamic tournament environment. The specifications produced during this stage served as the foundation for the subsequent design and implementation phases.

B. Flowchart

A flowchart is a graphical representation used to illustrate the sequence of processes, decisions, and interactions within a system. It provides a clear visualization of the operational workflow, making it easier to understand the logic and execution of an application. In game development, flowcharts are commonly used to describe the progression of gameplay, system decisions, and transitions between different game states [16]. The flowchart of the proposed Run to BIPOL game is presented in Figure 2.

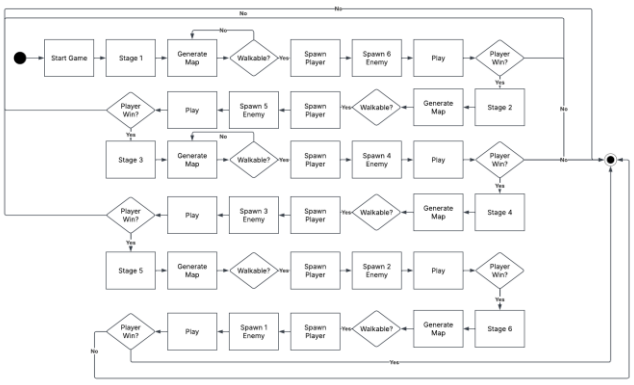


Fig. 2. Flowchart of "Run to BIPOL"

As illustrated in **Figure 2**, the gameplay starts after the player initiates the race and enters the first tournament stage. The system generates a maze according to the grid dimensions specified during setup. Before the race begins, the generated maze is checked using BFS to verify that every participant has access to at least one BIPOL destination. If the validation fails, the maze is discarded and a new layout is generated until a valid configuration is obtained. After a valid maze has been established, the player and six AI-controlled opponents are placed on different walkable tiles, while the BIPOL objectives are distributed within the maze. The race then begins, with the player controlled through keyboard input and the AI opponents navigating autonomously toward available BIPOLs. The stage continues until all available BIPOLs have been obtained, determining the participant who is eliminated before the game proceeds to the next stage.

At the end of each stage, the system evaluates whether the player successfully reaches a BIPOL and survives the elimination round. If the player fails to obtain a BIPOL, the game ends. Otherwise, the game proceeds to the next stage, where a new maze is generated and the number of AI opponents is reduced by one according to the tournament elimination system. This process continues through six tournament stages until the final round, where the player competes against a single remaining AI opponent. The player wins the game by surviving all elimination stages and successfully reaching the final BIPOL before the last AI opponent.

C. Breadth-First Search (BFS) Implementation

Breadth-First Search (BFS) is an uninformed graph traversal algorithm that explores neighboring nodes level by level to determine all reachable locations from a given starting point [17], [18]. In this study, BFS is implemented as a map validation mechanism to ensure that every participant has at least one valid path to a BIPOL before a tournament stage begins. By validating the generated maze prior to gameplay, the algorithm prevents the creation of unwinnable maps and guarantees a fair playing environment. The overall BFS validation process implemented in the proposed game is illustrated in Figure 3.

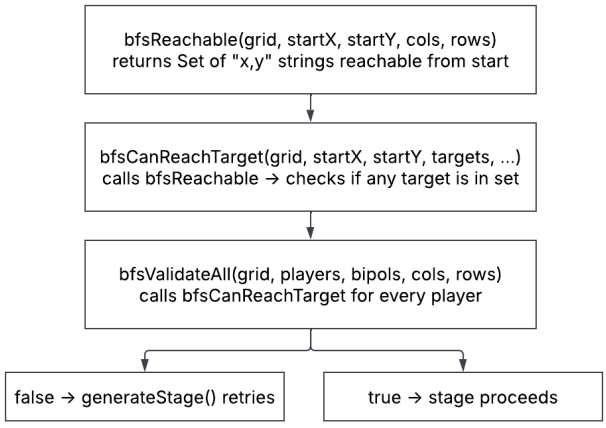


Fig. 3. BFS Validation of "Run to BIPOL"

As shown in Figure 3, the validation process begins with the bfsReachable() function, which receives the generated maze grid, the starting coordinates, and the grid dimensions as input. The function performs a breadth-first traversal using a queue to visit all walkable cells while maintaining a set of visited positions to avoid revisiting previously explored

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

nodes. For each iteration, the BFS process checks the four adjacent cells, namely the cells above, below, to the left, and to the right of the current position. A cell is added to the queue when it is within the grid, is not a wall, and has not been visited previously. The visited set therefore records the positions that can be reached from the participant's starting point. The bfsCanReachTarget() function then uses this set to check whether any available BIPOL is located in a reachable position. This check is performed for every participant through bfsValidateAll(). When all participants have access to at least one BIPOL, the generated maze is considered valid and the stage can begin. If one or more participants cannot reach a destination, the current maze is rejected and the system generates another layout. The bfsPath() function is also available to obtain a complete route between a starting position and a target, although this function is not the main navigation mechanism during gameplay. The AI opponents use A* to determine their movement paths, while BFS is mainly responsible for checking maze connectivity before a tournament stage starts.

D. A Pathfinding Implementation*

The A* algorithm is implemented as the primary navigation method for the intelligent agents to determine the shortest path toward a BIPOL destination. Compared with Breadth-First Search (BFS), which is used only for validating map accessibility before gameplay begins, A* is executed continuously during gameplay to support autonomous enemy movement. By combining the actual movement cost with a heuristic estimation, A* efficiently identifies an optimal route while reducing unnecessary node exploration. The overall implementation of the A* pathfinding algorithm in the proposed game is illustrated in Figure 4.

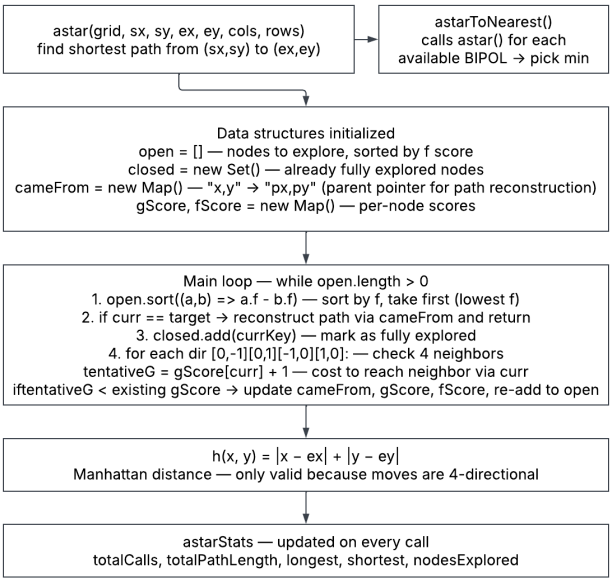


Fig. 4. Pathfinding Validation of "Run to BIPOL"

As illustrated in Figure 4, the A* process is handled by the astar() function, which takes the maze grid, the starting position, the target position, and the grid dimensions as input. At the beginning of the search, the algorithm prepares the data required to manage the explored nodes and reconstruct the route. The open list contains candidate nodes for exploration, while the closed set keeps track of nodes that have already been processed. The cameFrom map records the predecessor of each node so that the route can be reconstructed once the destination is reached. The gScore and fScore maps store the actual movement cost and the estimated total cost, respectively. The search then proceeds iteratively by selecting the node with the smallest fScore and examining its four adjacent cells to determine the next possible movements. The heuristic function applies the Manhattan Distance, $h(x, y) = | x - ex | + | y - ey |$, which is suitable for the game's four-directional grid movement [19]. Once the destination is found, the route is rebuilt by tracing the predecessor nodes stored in the cameFrom map until the starting position is reached. The implementation also keeps several statistics from the search process, such as the number of A* calls, average route length, longest and shortest routes, and the total number of nodes examined. These values are stored in the gameplay log and can be used to review the agent's pathfinding behavior. The astarToNearest() function extends this process by checking each BIPOL that is still available and comparing the resulting routes. The route with the fewest steps is selected as the agent's current target. Through this mechanism, each AI opponent can independently choose an available BIPOL and adjust its target as the tournament progresses.

E. Finite State Machine (FSM) Implementation

A Finite State Machine (FSM) is a behavioral model that controls the decision-making process of an intelligent agent by defining a finite number of states and the transitions between them [20]. Unlike the A* algorithm, which is responsible for determining the shortest navigation path, the FSM governs the overall behavior of the AI by deciding what action should be performed at each stage of the game. In the proposed game, every AI-controlled opponent operates through seven states: Idle, Find_Target, Find_Path, Move, Recalculate, Win, and Eliminated. The state transition model implemented in the intelligent agent is illustrated in Figure 5.

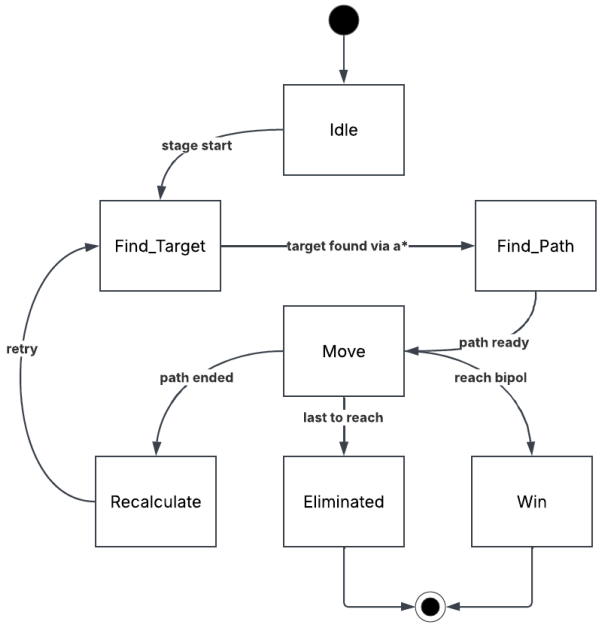


Fig. 5. FSM Diagram of "Run to BIPOL"

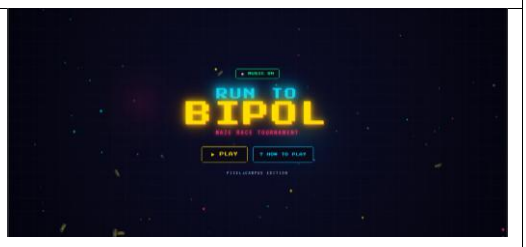
Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

As shown in Figure 5, the AI behavior is managed through several states that determine its actions during the tournament. At the beginning of a stage, each AI starts in the Idle state and then moves to Find_Target to look for an available BIPOL. After selecting a target, the agent enters Find_Path and uses A* to determine a route to that location. The agent then moves through the maze in the Move state, following the calculated route step by step. The target is checked during movement, so when another participant takes the selected BIPOL or the route can no longer be followed, the AI switches to Recalculate and searches for another target. When the AI reaches a BIPOL, it enters the Win state and continues to the next stage. An AI that fails to obtain a BIPOL and is eliminated from the tournament enters the Eliminated state. The agents also use different movement intervals, resulting in different movement speeds. These intervals are reduced as the tournament progresses, making the later stages more challenging and creating greater competition among the remaining participants.

F. Result of The Game

The final implementation of *Run to BIPOL* is summarized in Table 1, which presents the main interfaces used throughout the game. The game opens with the Hero Page, where players can access the Play and How to Play options. The Home Page then provides the initial game settings, including grid width, grid height, player name, gender, and character selection. Players can choose from 14 characters representing seven academic departments at Politeknik Negeri Jakarta, namely Accounting (AK), Business Administration (AN), Mechanical Engineering (TM), Electrical Engineering (TE), Civil Engineering (TS), Information and Computer Engineering (TIK), and Graphic Engineering and Publishing (TGP), with both male and female versions available. After the player selects a character and starts the race, the tournament proceeds from Stage 1 through Stage 6, with one participant eliminated after each stage. The competition eventually reaches the final stage, where the player faces the remaining AI opponent. The game ends with the Win screen when the player wins the final race or the Lose screen when the player is eliminated. These interfaces represent the complete gameplay sequence, from initial setup and character selection to the final tournament result.

TABLE I. USER INTERFACE OF "RUN TO BIPOL"

Interface	*Vizualitation*
Hero Page	
Home Page	
Stage 1	
Stage 2	
Stage 3	
Stage 4	
Stage 5	
Stage 6	
Win Screen	

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

Interface	*Vizualitation*
Lose Screen	
Gameplay Grid 20 x 30	
Gameplay Grid 10 x 20	

G. *Black-box Testing*

Black-box testing is a software testing method that evaluates the functionality of an application by examining its inputs and outputs without considering the internal implementation or source code. This testing approach is commonly used to verify whether each functional component performs according to the specified system requirements from the user's perspective [21]. In this study, black-box testing was conducted using two primary gameplay scenarios, namely the Win and Lose conditions, to validate the functionality of the developed game. The testing covered all major game components, including user interface navigation, game configuration, procedural maze generation, BFS validation, intelligent agent behavior, tournament progression, and gameplay logging. The results of the black-box testing are summarized in Table 2.

TABLE II. BLACK BOX TESTING RESULTS

Component	*Win Scenario*	*Lose Scenario*
Result	WIN	LOSE
Final Stage	Stage 6	Stage 4
Grid Size	15 × 15	10 × 15
Total Tiles	225	150
Wall Tiles	88	59
Walkable Tiles	137	91
Wall Density	39.1%	39.3%
Game Duration	30.335 s	21.187 s
Total A* Calls	123	128
Average Path Length	10.3 nodes	8.5 nodes
Longest Path	21 nodes	14 nodes
Shortest Path	2 nodes	2 nodes
Nodes Explored	3,256	2,156
Player Total Steps	63	48
Player Input Count	63	48
Stage Time 1	4.844 s	5.862 s
Stage Time 2	2.397 s	3.334 s
Stage Time 3	3.723 s	1.955 s
Stage Time 4	2.623 s	Eliminated
Stage Time 5	1.819 s	-
Stage Time 6	0.889 s	-
Stage Steps 1	13	18
Stage Steps 2	9	10
Stage Steps 3	14	12
Stage Steps 4	12	Eliminated
Stage Steps 5	8	-
Stage Steps 6	4	-
First Eliminated Participant	AI-SIPIL	AI-TGP
Second Eliminated Participant	AI-TGP	AI-SIPIL
Third Eliminated Participant	AI-MESIN	AI-MESIN
Fourth Eliminated Participant	AI-ELEKTRO	PLAYER
Fifth Eliminated Participant	AI-AK	-
Final Opponent	AI-AN	AI-ELEKTRO, AI-AN, AI-AK survived

As shown in Table 2, the black-box testing was carried out using two gameplay conditions, namely Win and Lose. In the Win condition, the player finished the tournament up to Stage 6 using a 15 × 15 grid. The recorded game duration was 30.335 seconds, with 225 total tiles consisting of 88 walls and 137 walkable tiles. The resulting wall density was 39.1%. During this session, A* was called 123 times, with an average path length of 10.3 nodes. The longest and shortest paths were 21 and 2 nodes, respectively, while the total number of explored nodes reached 3,256. The player recorded 63 steps from 63 inputs, distributed across the six stages as 13, 9, 14, 12, 8, and 4 steps, with completion times of 4.844, 2.397, 3.723, 2.623, 1.819, and 0.889 seconds. The AI results also show the expected elimination sequence: AI-SIPIL was eliminated in Stage 1, AI-TGP in Stage 2, AI-MESIN in Stage 3, AI-ELEKTRO in Stage 4, and AI-AK in Stage 5, leaving AI-AN as the opponent in the final stage.

For the Lose condition, the player used a 10 × 15 grid and was eliminated during Stage 4 after 21.187 seconds. The maze contained 150 tiles, including 59 wall tiles and 91 walkable tiles, giving a density of 39.3%. In this session, A* was executed 128 times and produced an average path length of 8.5 nodes, with 14 nodes as the longest path and 2 nodes as

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

the shortest path. The total number of explored nodes was 2,156. Before elimination, the player completed 48 steps from 48 inputs, with stage times of 5.862, 3.334, and 1.955 seconds and stage steps of 18, 10, and 12. The AI results recorded 12 total steps for AI-TGP, 22 for AI-SIPIL, 17 for AI-MESIN, 25 for AI-ELEKTRO, 21 for AI-AN, and 17 for AI-AK. Their target switches were 3, 6, 4, 2, 1, and 1, respectively, while all six AIs recorded zero recalculations. AI-TGP, AI-SIPIL, and AI-MESIN were eliminated in Stages 1, 2, and 3, respectively. AI-ELEKTRO, AI-AN, and AI-AK remained active when the player was eliminated in Stage 4.

The two test results also show that A* activity was affected by the conditions of each match rather than simply by the number of executions. The Lose condition produced five more A* calls than the Win condition, with 128 calls compared with 123, but the number of explored nodes was lower, at 2,156 compared with 3,256. One reason is the difference in grid size. The Win session used a 15 × 15 grid with 225 tiles, whereas the Lose session used a 10 × 15 grid containing 150 tiles. The average path was also shorter in the Lose session, at 8.5 nodes compared with 10.3 nodes in the Win session. A smaller search area and shorter routes can reduce the number of nodes examined during individual searches, even when A* is invoked more frequently. Meanwhile, the Win session continued through all six stages, giving the agents more opportunities to search from different starting positions toward different BIPOL locations. The results therefore indicate that A* workload in *Run to BIPOL* is influenced by grid dimensions, the distance between an agent and its selected target, and the changing target conditions during each stage. Thus, the number of A* calls alone is not sufficient to describe the pathfinding workload, since the number of nodes explored and the resulting path characteristics also need to be considered.

H. Distribution

The Distribution stage represents the final phase of the Multimedia Development Life Cycle (MDLC), in which the completed application is deployed for public access and practical use. In this study, the Run to BIPOL game was published as a web-based application using the Netlify hosting platform, allowing users to access the game directly through a web browser without requiring installation. The deployed game is publicly available at s.pnj.ac.id/RunToBipol.

I. Discussion

The results of this study demonstrate that the proposed pathfinding-based intelligent agent was successfully implemented in the Run to BIPOL web-based maze tournament game. The combination of procedural maze generation, BFS validation, A* pathfinding, and a Finite State Machine enabled the AI-controlled opponents to navigate autonomously while adapting to dynamic gameplay conditions throughout the tournament. The black-box testing results confirmed that all functional components operated correctly under both Win and Lose scenarios, indicating that the intelligent agent consistently performed its navigation, target selection, and tournament progression tasks as designed. Furthermore, the generated gameplay logs provide quantitative information regarding map characteristics, pathfinding performance, player behavior, and AI performance, which can be utilized for future analysis and optimization of intelligent game agents.

Compared with previous studies, the proposed implementation extends the application of intelligent agents beyond conventional maze navigation. First research demonstrated that the A* algorithm effectively identifies the shortest path in randomly generated maze environments, but its implementation mainly focused on assisting player navigation and visualizing optimal paths [8]. Second research applied A* to enable an NPC to pursue a player in a desktop-based maze game; however, the NPC behavior was limited to following a single moving target [9]. Likewise, other research employed artificial intelligence to support adaptive educational content rather than autonomous competition among multiple agents [10]. In contrast, the proposed Run to BIPOL game introduces a tournament-based environment in which six AI-controlled opponents simultaneously compete with the player for limited BIPOL objectives across multiple elimination stages. This environment requires each intelligent agent not only to compute an optimal path but also to continuously select new targets, respond to changing game conditions, and survive throughout successive tournament rounds.

The proposed implementation also demonstrates that intelligent agent behavior can be effectively integrated into browser-based games using lightweight web technologies without sacrificing autonomous decision-making capabilities. By combining procedural content generation for dynamic maze creation, BFS for accessibility validation, A* for efficient navigation, and FSM for behavioral control, the developed system creates AI opponents capable of making independent decisions in real time. This approach produces a more competitive gameplay experience compared with traditional maze games in which AI behavior is limited to static movement or simple player chasing. Consequently, this study contributes to the growing body of knowledge on intelligent agents in digital games by presenting a practical framework for implementing autonomous enemy behavior in competitive, tournament-based maze environments. Moreover, the developed game provides a foundation for future research on adaptive game AI, multi-agent competition, and browser-based intelligent gaming applications.

IV. CONCLUSION

This study successfully implemented a pathfinding-based intelligent agent for enemy behavior in the web-based maze tournament game Run to BIPOL. The developed system integrates Procedural Content Generation (PCG) to generate dynamic maze layouts, Breadth-First Search (BFS) to validate map accessibility, the A* algorithm to determine efficient navigation paths, and a Finite State Machine (FSM) to control the autonomous behavior of AI opponents throughout multiple tournament stages. The implementation enables six AI-controlled opponents to compete simultaneously with the player in a dynamic elimination-based environment, where agents continuously select targets, navigate toward available objectives, and respond to changing game conditions. Black-box testing demonstrated that the developed game and intelligent agent functioned as intended under both Win and Lose scenarios.

However, this study has several limitations. The functional validation was conducted using only two black-box gameplay runs, consisting of one Win scenario and one Lose scenario, so the results do not yet represent repeated trials across a wider

Cite this article as: Y. S. Praka, N. Wahyuni, C. R. T. Prameswari, and A. Maria, "Implementation of a pathfinding-based intelligent agent for enemy behavior in the 'Run to BIPOL' maze tournament game", Journal of Games, Game Art, and Gamification (JGGAG) 11(2), 38-45, 2026.

range of maze configurations and gameplay conditions. In addition, the study did not compare A* performance with alternative pathfinding algorithms or evaluate the player experience through user testing. Future research can address these limitations by conducting repeated experimental trials, comparing different pathfinding approaches, and incorporating user-experience evaluation. Further development may also implement more advanced decision-making techniques to enable AI agents to learn and adapt their strategies.

REFERENCES

[1] J. Keshavelal *et al*., "EMBEDDING ARTIFICIAL INTELLIGENCE IN GAMES NPC AND EFFECTS ON EMOTIONAL HEALTH," *Spectr. Eng. Sci. 3(8), 1032–1042*, vol. 3, no. 8, pp. 1032–1042, 2025, [Online]. Available: https://thesesjournal.com/index.php/1/article/view/915

[2] J. Zhang, H. Li, Y. Teng, R. Zhang, Q. Chen, and G. Chen, "Research on the Application of Artificial Intelligence in Games," in *2022 9th International Conference on Digital Home (ICDH)*, IEEE, Oct. 2022, pp. 207–212. doi: 10.1109/ICDH57206.2022.00039.

[3] V. Levytskyi, M. Tsiutsiura, A. Yerukaiev, N. Rusan, and T. Li, "The Working Principle of Artificial Intelligence in Video Games," in *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, IEEE, May 2023, pp. 246–250. doi: 10.1109/SIST58284.2023.10223491.

[4] S. A. Yakan, "Analysis of Development of Artificial Intelligence in the Game Industry," *Int. J. Cyber IT Serv. Manag.*, vol. 2, no. 2, pp. 111–116, May 2022, doi: 10.34306/ijcitsm.v2i2.100.

[5] N. Salem, H. Haneya, H. Balbaid, and M. Asrar, "Exploring the Maze: A Comparative Study of Path Finding Algorithms for PAC-Man Game," in *2024 21st Learning and Technology Conference (L&T)*, IEEE, Jan. 2024, pp. 92–97. doi: 10.1109/LT60077.2024.10469459.

[6] A. Biju, G. MP, J. N, A. Chris, and N. R. Thaleeparambil, "Efficient Pathfinding in a Maze to overcome Challenges in Robotics and AI Using Breadth-First Search," in *2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE, Mar. 2025, pp. 727–732. doi: 10.1109/CSNT64827.2025.10968383.

[7] N. Mehanna and W. Rudametkin, "Caught in the Game: On the History and Evolution of Web Browser Gaming," in *Companion Proceedings of the ACM Web Conference 2023*, New York, NY, USA: ACM, Apr. 2023, pp. 601–609. doi: 10.1145/3543873.3585572.

[8] F. K. Arief, I. Ismail, and D. Utami, "Application of the A * Algorithm for an Adaptive Pathfinding System in a Web-Based Maze Game with Random Maze Patterns," *Digit. NEXUS Syst. J.*, vol. 1, no. 2, pp. 9–14, 2025.

[9] I. Pratiwi, "Smart Adaptive NPC AI pada Permainan Labirin Menggunakan Algoritma A*," *J. Pengemb. Sist. Inf. dan Inform.*, vol. 4, no. 3, pp. 35–40, Feb. 2024, doi: 10.47747/jpsii.v4i3.1658.

[10] F. Rigani, N. Zahra, P. F. B. Tarigan, and F. R. Suwanto, "Pengembangan Game Labirin Berbasis Artificial Intelligence Sebagai Media Pembelajaran Transformasi Geometri," *Katalis Pendidik. J. Ilmu Pendidik. dan Mat.*, vol. 2, no. 4, pp. 159–170, Dec. 2025, doi: 10.62383/katalis.v2i4.2671.

[11] M. Rosalina, A. B. Prakoso, and N. Marcheta, "Development of Interactive 3D Map Using Unity Engine for Location-Aware Digital Museum Exploration," in *2025 9th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, IEEE, Dec. 2025, pp. 323–328. doi: 10.1109/ICITISEE68184.2025.11355134.

[12] I. Sonjaya, M. Rosalina, T. D. Ariyanti, A. S. Azmi, and D. A. Martono, "Development Of Educational Media Based On Virtual Reality Multiplayer Games For Basic Communication Learning For Students With Intellectual Disabilities," in *2025 8th International Conference of Computer and Informatics Engineering (IC2IE)*, IEEE, Sep. 2025, pp. 1–6. doi: 10.1109/IC2IE67206.2025.11283385.

[13] M. L. Haryanti, G. Chardaputeri, K. Wangsa, R. Figo, R. Indradjadja, and K. Sutanto, "PENERAPAN MULTIMEDIA DEVELOPMENT LIFE CYCLE DALAM MODEL PENGEMBANGAN PERMAINAN VR THE ARCHER," *J. Algoritm. Log. dan Komputasi*, vol. 8, no. 2, pp. 830–838, 2025, [Online]. Available: https://journal.ubm.ac.id/index.php/alu/article/view/8223

[14] I. Benawan, D. M. K. Nugraheni, B. Noranita, and G. Aryotejo, "Digital Education Game for TK-A Level Students Using Multimedia Development Life Cycle Method," *INTENSIF J. Ilm. Penelit. dan Penerapan Teknol. Sist. Inf.*, vol. 7, no. 1, pp. 68–83, Feb. 2023, doi: 10.29407/intensif.v7i1.18671.

[15] H. Nurul, D. M. K. Nugraheni, B. Noranita, and N. Bahtiar, "Pengembangan Game Edukasi Digital Tema Seni dan Bahasa dengan Metode Multimedia Development Life Cycle," *J. Sist. Inf. Bisnis*, vol. 14, no. 3, pp. 302–310, Aug. 2024, doi: 10.21456/vol14iss3pp302-310.

[16] I. Al Mahdi, S. Bukhori, and M. A. Furqon, "Implementation of Finite State Machine in Flowchart-Based Visual Programming Game," *J. Games, Game Art, Gamification*, vol. 10, no. 3, pp. 93–105, Dec. 2025, doi: 10.21512/jggag.v10i3.12133.

[17] Y. Wang, "A Comparative Study of Maze-Solving Algorithms: Breadth-First Search, A and Binary Tree with Depth-First Search," *Highlights Sci. Eng. Technol.*, vol. 161, pp. 26–29, 2026.

[18] S. L. Pardede *et al.*, "A Review of Pathfinding in Game Development," *[CEPAT] J. Comput. Eng. Progress, Appl. Technol.*, vol. 1, no. 01, p. 47, May 2022, doi: 10.25124/cepat.v1i01.4863.

[19] A. Patel, "Heuristics," thoery.standford.edu. Accessed: Jul. 02, 2026. [Online]. Available: https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[20] M. A. A. Syandana, H. Z. Zahro', and S. Achmadi, "Perancangan Game Mutes Dengan Menggunakan Metode Finite State Machine Berbasis Android," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 4, pp. 2167–2174, 2023, doi: 10.36040/jati.v7i4.7441.

[21] R. Janata, A. T. Priandika, and R. D. Gunawan, "PENGEMBANGAN GAME PETUALANGAN EDUKASI PENGENALAN SATWA DILINDUNGI DI INDONESIA MENGGUNAKAN CONSTRUCT 2," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 3, no. 3, pp. 286–294, Oct. 2022, doi: 10.33365/jatika.v3i3.2035.