Journal of Information Systems and Informatics
Vol. 7, No. 1, March 2025 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v7i1.941 Published By DRPM-UBD

Heap Optimization in A* Pathfinding for Horror Games

Risaldi Angga Buana Putra[1], Ary Setijadi Prihatmanto[2], Rahadian Yusuf[3], Agus Sukoco[4]

[1,2,3,4]School Of Electrical Engineering and Informatics, Bandung Institute of Technology
Bandung, Indonesia
Email: [1]23222024@mahasiswa.itb.ac.id, [2]ary.setijadi@itb.ac.id, [3]yrahadian@itb.ac.id,
[4]agussukoco16@gmail.com

Abstract

This paper examines the implementation of the A* pathfinding algorithm with binary heap optimization in a horror game environment. The horror genre in gaming uniquely engages players by placing them at the center of fear-driven experiences, where intelligent and unpredictable enemy behavior is critical for immersion. To achieve this, adaptive AI—specifically for apparitions or monsters—is controlled using A*, an algorithm renowned for its efficiency in determining the shortest path. Heap optimization is introduced to enhance A* performance by reducing the time required to identify the lowest-cost node in the Open List. Experimental results from a Unity-based prototype demonstrate that the optimized A* achieves an average pathfinding time of 1.6 ms, compared to 3.16 ms without optimization—representing a 49.37% improvement. This speed increase allows for faster and more responsive enemy behavior, resulting in heightened difficulty and more dynamic, fear-inducing gameplay. The findings highlight the potential of algorithmic optimization to significantly enhance both technical performance and player immersion in horror game design.

Keywords: Horror, Horror Games, AI, A*Pathfinding, Binary Heap.

1. INTRODUCTION

Fear is a primal human emotion triggered by potential threats, often serving as a critical survival mechanism. It operates across the psychological and physiological domains of human experience, preparing individuals to confront danger or flee from it. In modern contexts, fear is no longer confined to real-world experiences. With the evolution of media and entertainment, it has found a new role in storytelling and digital experiences—particularly in the horror genre. This genre taps into deep-seated fears, allowing people to explore frightening scenarios in safe environments. The effect of fear on the brain and body has been well-documented, showing that even simulated threats can elicit intense emotional responses [1].

One of the most immersive forms of horror entertainment is video games. Unlike

passive mediums like films or books, horror games place the player in an active role, demanding interaction with environments and characters in real time. This interactivity significantly amplifies the fear experience. Horror games often utilize mysterious storylines, eerie settings, and frightening monsters to stimulate curiosity and anxiety. The psychological engagement of horror games stems largely from their unpredictability, a key element in maintaining tension and immersion. A growing body of research supports the notion that players seek this fear because it challenges their problem-solving and emotional resilience [2], [3].

However, despite the increasing sophistication of horror game design, a notable gap exists in how artificial intelligence (AI), particularly enemy behavior, is implemented. Many games rely on basic or engine-default pathfinding systems, which, though functional, often result in predictable enemy movements. This predictability can compromise the very suspense that horror games strive to create. The A* (A-Star) algorithm is one of the most commonly used pathfinding methods in game development due to its balance of efficiency and accuracy [4]. But without optimization, it can become a bottleneck especially when real-time decision-making is required in dynamic environments.

To address this issue, researchers and developers have explored optimizing A* with techniques such as binary heaps, which significantly improve the performance of the open list queue search one of A*'s most computationally expensive tasks. This optimization results in faster and more responsive enemy behaviors, making in-game adversaries less predictable and more lifelike [5], [6]. Such improvements are particularly vital in horror games where monsters or hostile NPCs must react quickly to the player's movements. Enhancing responsiveness allows the AI to simulate intelligence, adapt to player actions, and sustain tension throughout the gameplay experience.

Therefore, this study investigates the integration of binary heap optimization into the A* pathfinding algorithm and its specific impact on the horror game experience. The goal is to evaluate how improved AI performance contributes to gameplay immersion, difficulty, and fear generation. We also explore the development process of horror games and how realistic enemy behaviors driven by optimized AI can elevate the genre. By comparing traditional A* with the optimized version in a horror context, this study provides valuable insights for game developers aiming to enhance both performance and player engagement [7].

2. METHODS

In this experiment, the researcher proposes the development of a horror-themed video game that leverages artificial intelligence (AI), specifically the A* pathfinding algorithm, to enhance gameplay difficulty and immersion. The goal is to investigate

whether the inclusion of heap optimization in the A* algorithm can significantly improve pathfinding efficiency. The game is designed with traditional horror elements such as low lighting, atmospheric sounds, and the presence of monstrous entities, contributing to a suspenseful environment where player unpredictability and fast-paced AI responses are crucial to maintaining fear. The research methodology is structured and visualized in Figure 1.

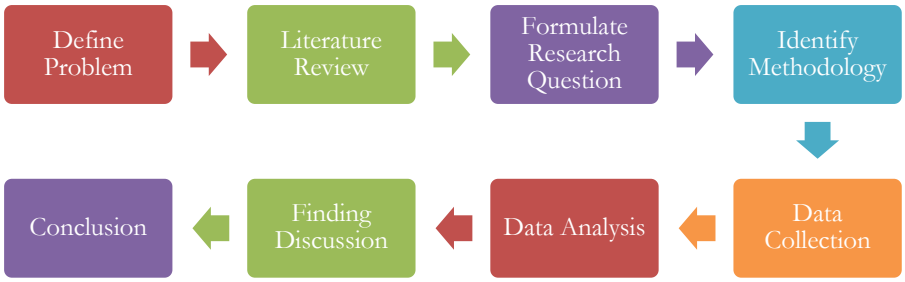

Figure 1. Research Flow

Figure 1 outlines the research process, starting from the identification of the core problem to the final conclusions. The steps are described in detail as follows.

1) Problem Definition: The initial phase identifies the core issue: although the A* algorithm is widely recognized for its efficiency in pathfinding tasks, it exhibits inconsistent performance in real-time gaming environments, particularly in scenarios requiring rapid target tracking. In this study, the observed limitation lies in the AI's fluctuating speed when pursuing the player. This inconsistency affects the suspense and realism in horror gameplay, making algorithm optimization a central objective.
2) Literature Review: A comprehensive literature review was conducted using keywords such as "A* pathfinding," "game AI," "horror games," and "pathfinding optimization." Relevant sources were examined to support the hypothesis. Key among them is the work of Liu [5], which highlights how integrating a binary heap structure can significantly improve the processing speed of A* by reducing the time complexity of priority queue operations.
3) Research Question Formulation: Based on the literature, the central research question emerged: Can heap optimization improve the efficiency and responsiveness of A pathfinding in a horror game environment, and how does this impact gameplay? * Liu's findings [5] suggesting a potential doubling in performance led to the hypothesis that similar gains could be realized in this study's custom game environment.
4) Research Methodology: The methodology involved designing a prototype horror game using the Unity game engine. The game world was constructed on a grid size of 241×331 units. Two versions of the A* algorithm was implemented: one standard and one incorporating heap optimization. These

versions were tested in identical conditions to measure differences in target acquisition speed by the AI.

5) Data Collection: For empirical validation, both algorithms were tested under the same gameplay scenario. The AI was tasked with tracking a moving target across 50 nodes. Multiple trials were conducted to record the time taken by each algorithm to reach the target, and the average pathfinding speed was calculated.
6) Discussion of Findings: Upon analyzing the data, the optimized A* algorithm demonstrated a significant improvement in efficiency. Specifically, the version utilizing heap optimization showed a 49.37% increase in performance speed compared to the standard implementation, validating previous claims in the literature [8].
7) Conclusion: The experiment confirms that integrating heap optimization into the A* pathfinding algorithm substantially improves AI responsiveness in horror games. Faster AI movements not only enhance gameplay difficulty but also increase unpredictability, which is vital in horror scenarios where tension and fear are driven by uncertainty. The optimized pathfinding allows the AI to better anticipate player movement, making chase scenes more intense and the overall game experience more immersive.

2.1. Game Prototype

The game was developed using the Unity game engine, chosen for its robust features and flexibility in creating interactive 3D environments. Unity is a widely adopted game development platform that provides a comprehensive suite of tools to support rendering, physics simulation, input handling, and more [9]. It offers a powerful framework suitable for both two-dimensional (2D) and three-dimensional (3D) game development, making it the ideal engine for our horror-themed game, which relies heavily on immersive 3D space. Unity's capabilities in designing models, managing scene transitions, and integrating custom scripts greatly facilitated the implementation of our study, which focuses on optimizing the A* pathfinding algorithm through a horror gameplay experience [10].

The core gameplay revolves around a player character who must escape from a locked, eerie environment following a mysterious and unsettling event. Players are required to explore the game world, collect essential items such as keys and hidden notes, and uncover clues that guide their escape. To enhance the engagement and difficulty of the gameplay, players must also gather information about various supernatural entities that pursue them. Understanding these apparitions' behaviors allows players to devise specific strategies for evasion, survival, and ultimately escape [11].

To deliver a chilling and immersive horror experience, several elements were

incorporated into the game design. These include dynamic lighting, audio effects, and multiple types of apparitions, each with unique characteristics. Darkness plays a central role in creating tension and suspense. Players rely on a flashlight to navigate through dimly lit areas, though they can also activate light sources scattered throughout the environment to gain temporary relief from the obscurity [12]. Apparitions appear randomly and exhibit distinct behaviors depending on environmental conditions—some may toggle lights on and off, while others produce unsettling noises that disrupt the player's focus. Audio cues such as loud footsteps, screeches, and eerie roars are employed to amplify the fear factor and keep players in a state of heightened alertness [13].

For the purposes of testing and evaluating the heap optimization of the A* algorithm within this horror setting, a virtual environment with a grid size of 241 × 331 was created. This specific dimension was selected to ensure the inclusion of a sufficient number of grid nodes necessary for effective pathfinding while accommodating various obstacles and map complexity. The grid-based world design enables rigorous testing of the algorithm's efficiency and responsiveness in navigating complex paths, especially under the stress of in-game threats and dynamic environmental conditions.

2.2. Artificial Intelligence (AI)

Artificial Intelligence (AI) has long been a cornerstone of modern game development, particularly in single-player experiences where it serves to enrich gameplay by simulating intelligent behavior in non-player characters (NPCs). In the context of game design, AI refers not to visual or audio elements, but to systems and algorithms that govern interactions, pathfinding, learning behaviors, group coordination (e.g., flocking and formations), dynamic difficulty adjustment, and decision-making processes [14].

The primary objective of game AI is to control in-game characters in a manner that aligns with the environmental context and narrative of the game world. For example, in a hyper-realistic game environment, an AI that performs flawlessly such as demonstrating perfect accuracy in a first-person shooter can seem unnatural and immersion-breaking. This is because it lacks the imperfections typically associated with human behavior. Therefore, game developers often implement intentional limitations or "artificial stupidity" to make the AI's behavior appear more human and relatable. These compromises are often necessary due to the constraints of computational power and design complexity in maintaining consistent, believable AI across diverse scenarios [15].

In this horror game, AI plays a crucial role in enhancing the immersive and suspenseful atmosphere. The implementation of AI is focused primarily on

controlling the apparitions' supernatural entities that interact with and react to the player's presence. The AI is responsible for behaviors such as patrolling, searching for the player, initiating chases, and triggering scripted events based on the player's actions.

For the scouting behavior, the apparition AI is programmed to patrol specific areas for a set duration. During this time, the AI attempts to detect the player's location using predefined vision and proximity parameters. If no player presence is detected within the time frame, the AI dynamically shifts to another patrol route, simulating a realistic search behavior. Once the player is located, the AI transitions into chase mode, in which it uses the A* pathfinding algorithm to efficiently navigate the environment and pursue the player [16]. The use of AI in this context not only adds challenge but also increases the unpredictability of each gameplay session, as apparitions adapt their behavior based on the game state and player actions. This dynamic approach keeps players engaged, forcing them to remain alert and continuously adapt their strategies to survive.

2.3. A* Pathfinding Algorithm

A* (A-Star) Pathfinding is one of the most widely implemented algorithms in the domain of graph traversal and shortest-path discovery, particularly within game development. Originally introduced by Hart, Nilsson, and Raphael in 1968, the algorithm is designed to identify the most efficient path from a starting node to a target node by evaluating potential paths using a cost function [17]. Its blend of performance and reliability has made it a go-to method for managing AI movement in real-time scenarios such as video games. A* achieves this by evaluating three critical parameters:

a) G(n): The exact cost of traveling from the start node to the current node.
b) H(n): The heuristic estimated cost from the current node to the target node.
c) F(n): The total estimated cost of the path through the current node, calculated as shown in Equation 1.

$$F(n)=G(n)+H(n) \quad (1)$$

This equation allows the algorithm to make strategic choices by combining both the known cost of a path and an educated guess about the remaining distance, ensuring efficient and near-optimal solutions. Despite its heuristic nature, which doesn't always guarantee the perfect solution, A* generally delivers a highly efficient and practical path within a short computational time [18]. This makes it especially valuable in gaming environments where fast and believable AI movement is essential.

In the development of our horror game, A* was employed to govern the

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

pathfinding behavior of the AI-controlled apparitions. These entities use the algorithm to patrol, chase the player, and dynamically adapt to changing environments. The AI calculates the optimal paths in a 241 × 331 grid-based world filled with obstacles and triggers, ensuring that the apparitions move intelligently in pursuit of the player. The underlying logic of the A* algorithm is demonstrated in the pseudocode below, labeled as Code 1.

Code 1. A* Pathfinding Pseudocode

```
function A_Star(start_node, goal_node):
    open_set = {start_node}
    closed_set = {}

    g_score[start_node] = 0
    f_score[start_node] = heuristic(start_node, goal_node)

    while open_set is not empty:
        current = node in open_set with lowest f_score[]
        if current == goal_node:
            return reconstruct_path(current)

        open_set.remove(current)
        closed_set.add(current)

        for each neighbor of current:
            if neighbor in closed_set:
                continue

           tentative_g_score     =     g_score[current]     +
distance(current, neighbor)

            if neighbor not in open_set:
                open_set.add(neighbor)
            else if tentative_g_score >= g_score[neighbor]:
                continue

            came_from[neighbor] = current
            g_score[neighbor] = tentative_g_score
            f_score[neighbor]     =     g_score[neighbor]     +
heuristic(neighbor, goal_node)

   return failure
```

2.4. Binary Heap Optimization

A heap is a powerful priority queue data structure organized in the form of a binary tree, where each node is assigned a priority. The structure maintains an efficient hierarchy, allowing quick access to the most important element typically the smallest or largest value. In a min-heap, the smallest element is always at the root, while in a max-heap, the largest takes that position [19]. Heaps follow two essential

rules.

a) Ordering Rule: In a min-heap, the value of the parent node is always less than or equal to that of its children.
b) Completeness Rule: The binary tree must be complete; that is, all levels must be fully filled except possibly the last, which is filled from left to right.

When a node is inserted or removed, the heap reorganizes to maintain these rules, allowing consistent performance. This behaviour is particularly useful in applications like pathfinding algorithms where performance matters most. In A* pathfinding, the major computational bottleneck often occurs when the algorithm searches through the Open List to find the node with the lowest F(n). As the size of the map increases, so does the number of nodes, resulting in slower searches if the list is implemented with a basic array or list. These linear data structures require O(n) time to find the minimum value, which is inefficient at scale. To address this, a binary heap can be used to significantly improve the speed and responsiveness of the A* algorithm [5]. In this horror game, we implemented a min-heap to replace the linear Open List structure within the A* algorithm. By ensuring that the node with the smallest F(n) always bubbles to the top, the algorithm can access it instantly in O(1) time, while maintaining an efficient O(log n) time for insertions and deletions. The pseudocode for the binary heap implementation, structured as Code 2.

Code 2. Binary Heap Operations

```
1. Add(item):
   - Place 'item' at the end of the heap.
   - Assign HeapIndex(item) = currentItemCount.
   - Increment currentItemCount.
   - Call SortUp(item) to maintain heap order.

2. RemoveFirst():
   - Set firstItem = items[0] (node with the lowest F(n)).
   - Replace items[0] with the last item in the heap.
   - Decrement currentItemCount.
   - Assign HeapIndex(items[0]) = 0.
   - Call SortDown(items[0]) to maintain heap order.
   - Return firstItem.

3. UpdateItem(item):
   - Call SortUp(item) to adjust its position if its priority changes.

4. SortDown(item):
   while item has children:
      - leftChild = item.HeapIndex * 2 + 1
```

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

```
        - rightChild = item.HeapIndex * 2 + 2
        - Determine swapIndex as the child with the lower F(n)
        - If item.F(n) > items[swapIndex].F(n):
            Swap(item, items[swapIndex])
        - Else:
            break

5. SortUp(item):
    while item has a parent:
        - parentIndex = (item.HeapIndex - 1) / 2
        - If item.F(n) < items[parentIndex].F(n):
            Swap(item, items[parentIndex])
        - Else:
            break

6. Swap(itemA, itemB):
    - Swap itemA and itemB in the items array.
   - Update their HeapIndex to reflect new positions.
```

The binary heap structure ensures that the A* pathfinding algorithm runs with improved performance by reducing the computational load required to manage the Open List. This is particularly beneficial in our horror game, where fast and reactive AI pathfinding enhances player immersion and challenge. By applying this heap-based optimization, the AI can efficiently evaluate routes, respond to player actions, and create a dynamic gameplay experience.

3. RESULTS AND DISCUSSION

3.1. Gameplay Testing

Gameplay testing was conducted to evaluate the basic mechanics and core features implemented in the early development stages of the game using the Unity game engine. At this stage, the game includes two playable levels, each designed with distinct layouts and gameplay mechanics. Although the game remains in its testing phase, several foundational horror elements such as apparitions and ambient sound effects have been successfully integrated to establish the tone and atmosphere.

Figure 2 illustrates the spatial arrangement of the first level. In this level, the player's main objective is to escape by locating the goal, which is hidden in the final room of the map. However, this task is complicated by the constant pursuit of a hostile apparition that begins chasing the player once triggered. Players can interact with various in-game objects, including doors, keys, flashlights, and light switches.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Interactable objects are visually indicated with an on-screen prompt displaying "[E]", which appears when the player is within proximity. This interaction system ensures intuitive gameplay and helps guide player progression through the environment.

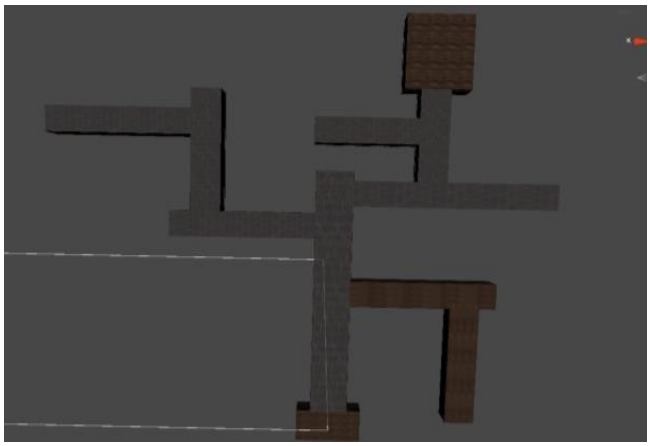

Figure 2. First Level Layout

Figure 3. Example of Interactable Object

As shown in Figure 3, objects such as doors can be opened though not all doors are unlocked by default. Some require specific keys to access, making key collection an essential sub-goal within the gameplay. These keys are scattered throughout the level and must be located to progress. Figure 4 shows an example of a spawned apparition. Once active, the apparition relentlessly chases the player, creating pressure and tension throughout the gameplay session. The AI guiding the apparition is programmed to locate and pursue the player, with behaviors that include chasing, capturing, and emitting disturbing audio cues. These audio elements, including mysterious whispers and sudden loud noises, are intended to

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

enhance immersion and provoke fear, encouraging the player to remain cautious and alert.

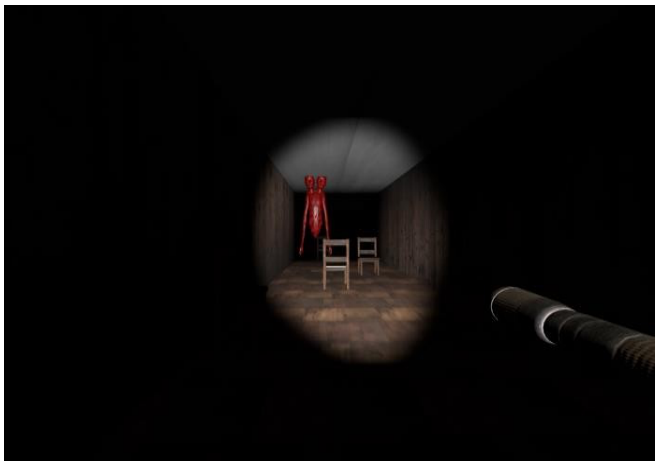

Figure 4. Apparition Encounter

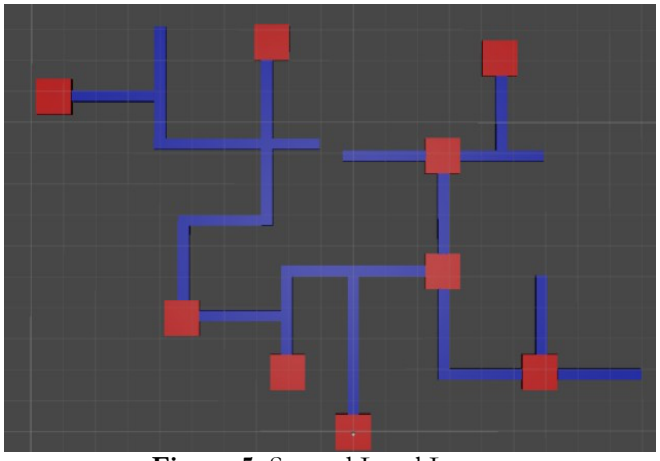

Figure 5. Second Level Layout

In the second level, the core objective remains similar—finding a way to escape the environment but the gameplay path is more complex. The map is designed in a maze-like structure, with multiple misleading routes and decoy objectives. Several keys are placed throughout the level, though only one is usable for unlocking the final escape route. Adding to the complexity, the map includes a trap room, which contains multiple dormant apparitions. If the player inadvertently opens this room, several apparitions will be unleashed, increasing the difficulty and requiring the player to evade multiple threats simultaneously.

While the game currently supports consistent player interactions, object collection,

and a basic AI chase system, several more advanced features are still under development. For instance, the future vision of the game includes adaptive AI that dynamically adjusts to player behavior, along with more sophisticated horror triggers that respond to exploration and decision-making patterns. At this stage, horror events are activated through basic in-game triggers such as picking up a key, opening specific doors, or entering predefined areas. Overall, the testing phase confirms that the core gameplay mechanics and horror elements function as intended, though refinement and additional complexity are planned for future iterations. This foundation lays the groundwork for implementing more dynamic AI behaviors and immersive horror scenarios.

3.2. A* Algorithm Heap Optimization Test

This section evaluates the performance impact of integrating heap optimization into the A* pathfinding algorithm. The primary objective is to measure the time efficiency gained in locating a target node when using a binary heap to manage the algorithm's Open List, compared to using a basic array or list. Testing was conducted on a controlled environment a flat level layout with dimensions of 241 × 311 grid units, and a grid node radius of 2. For consistency, both versions of the A* algorithm (with and without heap optimization) were tested under identical conditions. Each trial involved a total of 50 movement calculations, where the AI was tasked with finding a dynamically moving target that followed a predefined path across the map.

Table 1. Speed Comparison Between A With and Without Heap Optimization*

Algorithm	Nodes Evaluated	Average Speed (ms)
A* without heap optimization	50	3.16 ms
A* with heap optimization	50	1.60 ms

As shown in Table 1, the A* algorithm without heap optimization recorded an average target acquisition speed of 3.16 milliseconds, while the optimized version utilizing a binary heap achieved a significantly faster average speed of 1.60 milliseconds. To quantify the improvement in performance, we used the following formula to calculate the percentage increase in speed, as shown in Equation 2.

$$\text{Performance Increase (\%)} = \left(\frac{(\text{Avg Speed Without Heap} - \text{Avg Speed With Heap})}{\text{Avg Speed Without Heap}}\right) \times 100 \quad (2)$$

$$\text{Performance Increase (\%)} = \left(\frac{(3.16 - 1.60)}{3.16}\right) \times 100 \approx 49.37\%$$

This result demonstrates a 49.37% improvement in pathfinding speed when using heap optimization. This performance boost significantly enhances AI

responsiveness, particularly in games where real-time chasing mechanics are crucial to the player's experience.

In the context of a horror game, where being pursued by AI-controlled entities is a core gameplay element, faster and more efficient pathfinding can elevate both the challenge and tension. By reducing the delay in how quickly enemies can respond and track the player's location, the game can deliver more intense and dynamic encounters. This not only increases difficulty but also encourages players to engage more thoughtfully with the environment—strategically planning their routes, interactions, and escape paths to avoid being caught by the AI. Ultimately, heap optimization contributes not just to technical performance but to a more immersive and adrenaline-driven gameplay experience.

3.3. Discussion

The results of this study confirm that the integration of heap optimization into the A* pathfinding algorithm significantly enhances performance. Our experimental data shows that the optimized A* algorithm achieves an average speed of 1.60 ms in locating the target node, compared to 3.16 ms without the optimization. This equates to a 49.37% increase in speed, which has direct implications for improving AI responsiveness within the game.

These findings are consistent with previous research. As reported by [20], and echoed in the study by [19], utilizing data structures such as binary heaps in pathfinding can substantially reduce the time complexity of operations like searching and inserting into the Open List. In a relevant study, [5] observed that heap optimization can make the A* algorithm twice to thrice faster, aligning with our measured improvement which approaches double the efficiency.

Further supporting this, [21] emphasized that the efficiency of A* is not solely dependent on the algorithm's implementation but is also affected by the game environment particularly the number of obstacles and map layout. More obstacles increase the number of node evaluations required, which in turn can slow down performance. Our results suggest that even in complex, obstacle-rich environments (such as our maze-like second level), heap optimization significantly mitigates these delays by streamlining node selection.

Another important study by [22] explored combining whale optimization with A* for monster path planning. While their approach favors more strategic positioning and interception by intentionally using longer, roundabout paths, it sacrifices speed in target acquisition. In contrast, our heap optimization approach focuses on maximizing the efficiency of direct pursuit, making it an excellent complement to [22]'s work. A hybrid AI, leveraging both whale optimization for strategic

positioning and heap-optimized A* for quick reaction, could provide a more complex and intelligent AI opponent in future iterations of our game.

Additionally, our findings align with [23], who implemented A* pathfinding with heap optimization in a similar game environment. Their algorithm achieved a pathfinding speed of under 3 ms, corroborating our result of 1.60 ms average speed for the heap-optimized version. This convergence across studies reinforces the value of using binary heaps in performance-critical game AI systems.

In the context of gameplay, especially within horror games where AI-driven pursuit is a critical element, these performance gains are not just technical enhancements they directly impact player experience. Faster pathfinding allows AI entities (like apparitions) to respond more rapidly to player movements, increasing both the difficulty and tension. Players are thus required to think more strategically, use environmental cues wisely, and remain alert, heightening immersion and emotional engagement.

Moreover, the success of basic AI in our initial test levels combined with the speed improvement through heap optimization lays a solid foundation for future expansion. Planned features such as adaptive AI behaviors, more intelligent enemy pathing, and dynamic environmental responses will rely heavily on the speed and efficiency that heap-optimized A* can provide.

In conclusion, the integration of heap optimization into the A* algorithm presents a significant advantage in game AI performance. Not only does it enhance computational efficiency, but it also enriches the gameplay experience by enabling faster, more responsive, and more challenging enemy behavior essential qualities in any horror game aiming to captivate and challenge its players.

4. CONCLUSION

This study set out to explore how binary heap optimization of the A* pathfinding algorithm can enhance AI behavior in horror games, ultimately contributing to a more immersive and fear-inducing player experience. Rooted in the psychological foundations of fear and its crucial role in interactive media, horror games demand not just atmospheric tension but also intelligent, responsive adversaries to maintain unpredictability and engagement. Our implementation of A* pathfinding with heap optimization resulted in a 49.37% improvement in algorithm speed, reducing average pathfinding time from 3.16 ms to 1.60 ms. This seemingly small technical enhancement had a significant impact on gameplay. AI-controlled entities, such as apparitions, were able to react faster to player movements, making chases more intense and less predictable. This increased speed not only improves computational efficiency but also directly enhances the player's psychological experience by

raising the tension and urgency inherent to survival scenarios.

Compared to traditional implementations where enemy movement can become formulaic and easy to predict, the optimized A* system introduces variability and speed that more closely mimic lifelike behavior. This aligns with the genre's goal of creating emotionally charged gameplay rooted in uncertainty, anticipation, and sudden threat core ingredients in the recipe of fear. Moreover, our findings reinforce and build upon prior studies, demonstrating the clear advantages of optimizing foundational AI systems in gaming. While our focus was on a horror setting, the implications of this work extend beyond genre. Any game that relies on enemy pursuit, real-time player interaction, or complex navigation systems can benefit from similar optimization techniques.

In future work, we recommend expanding the testing to include larger maps, different genres, and multiple AI agents acting simultaneously to simulate crowd dynamics or group intelligence. Additionally, combining this optimization with advanced behavior models—such as predictive interception or emotional AI—could lead to the development of enemies that are not only faster but smarter and more strategically adaptive. Ultimately, this research illustrates how a technical enhancement at the algorithmic level can transform the experiential quality of a game. By enabling faster, more lifelike AI responses, binary heap optimization of A* pathfinding stands as a valuable tool in pushing the boundaries of interactive horror storytelling, delivering more immersive, challenging, and emotionally resonant gaming experiences.

REFERENCES

[1] L. Nummenmaa, "Psychology and neurobiology of horror movies," *PsyArXiv Prepr.*, Turku PET Centre, Dept. of Psychology and Turku Univ. Hospital, Univ. of Turku, Finland, 2020.

[2] E. S. de Lima, B. M. C. Silva, and G. T. Galam, "Towards the design of adaptive virtual reality horror games: A model of players' fears using machine learning and player modeling," in *Proc. Brazilian Symp. Games Digit. Entertain. (SBGAMES)*, Nov. 2020, pp. 171–177, doi: 10.1109/SBGames51465.2020.00031.

[3] G. A. da Silva and M. W. de Souza Ribeiro, "Development of Non-Player Character with Believable Behavior: a systematic literature review," in *Proc. XX Brazilian Symp. Games Digit. Entertain.*, 2021, pp. 319–323, doi: 10.5753/sbgames_estendido.2021.19660.

[4] J.-H. Kim, J. Lee, and S.-J. Kim, "Navigating Non-Playable Characters Based on User Trajectories with Accumulation Map and Path Similarity," *Symmetry*, vol. 12, no. 10, p. 1592, 2020, doi: 10.3390/sym12101592.

[5] D. Liu, "Research of the Path Finding Algorithm A* in Video Games," *Highlights Sci. Eng. Technol.*, vol. 39, 2023.
[6] S. R. Lawande, G. Jasmine, J. Anbarasi, and L. I. Izhar, "A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games," *Appl. Sci.*, vol. 12, no. 11, p. 5499, 2022, doi: 10.3390/app12115499.
[7] O. R. Chandra and W. Istiono, "A-star Optimization with Heap-sort Algorithm on NPC Character," *Indian J. Sci. Technol.*, vol. 15, no. 35, pp. 1722–1731, 2022, doi: 10.17485/IJST/v15i35.857.
[8] Z. Zhang, J. Jiang, J. Wu, and X. Zhu, "Efficient and optimal penetration path planning for stealth unmanned aerial vehicle using minimal radar cross-section tactics and modified A-Star algorithm," *ISA Trans.*, vol. 134, pp. 42–57, 2023, doi: 10.1016/j.isatra.2022.07.032.
[9] M. Foxman, "United We Stand: Platforms, Tools and Innovation with the Unity Game Engine," *Soc. Media Soc.*, vol. 5, no. 4, 2019, doi: 10.1177/2056305119880177.
[10] [10] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6932–6939, 2020, doi: 10.1109/LRA.2020.3026638.
[11] R. Subari, N. Radita, and B. Prakoso, "The Implementation of A* Algorithm for Developing Non-Player Characteristics of Enemy in A Video Game Adopted from Javanese Folklore 'Golden Orange,'" *Teknika*, vol. 13, no. 2, pp. 164–174, 2024.
[12] R. A. Gunawan, W. Istiono, J. Scientia Boulevard Gading, C. Sangereng, and K. Tangerang, "Optimizing RPG Pathfinding: A Hybrid Approach for Static and Dynamic Obstacle Avoidance," *Int. J. Inf. Syst. Comput. Sci. (IJISCS)*, vol. 7, no. 3, 2023.
[13] A. Andreychuk, K. Yakovlev, D. Atzmon, and R. Stern, "Multi-Agent Pathfinding with Continuous Time," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 39–45, doi: 10.24963/ijcai.2019/6.
[14] T. Lynch and N. Martins, "Nothing to Fear? An Analysis of College Students' Fear Experiences with Video Games," *J. Broadcast. Electron. Media*, vol. 59, no. 2, pp. 298–317, 2015, doi: 10.1080/08838151.2015.1029128.
[15] M. Ponsen and P. Spronck, "Improving Adaptive Game AI with Evolutionary Learning," unpublished.
[16] V. Morina and R. Rafuna, "A Comparative Analysis of Pathfinding Algorithms in NPC Movement Systems for Computer Games," unpublished.
[17] S. R. Lawande, G. Jasmine, J. Anbarasi, and L. I. Izhar, "A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games," *Appl. Sci. (Switz.)*, vol. 12, no. 11, 2022, doi: 10.3390/app12115499.

[18] E. B. Aydogan and Y. Atay, "Unity Based A∗ Algorithm Used in Shortest Path Finding Problem for Helicopters," in *2021 Int. Conf. Control, Autom. Diagn. (ICCAD)*, 2021, doi: 10.1109/ICCAD52417.2021.9638762.

[19] J. Moss, C. O'Brien, J. Cook, and P. Camargo, "Performance effects of heap structure choice for path finding: A traffic modelling perspective," in *Proc. 44th Australas. Transp. Res. Forum (ATRF)*, Perth, WA, Australia, Nov. 2023.

[20] T. Lynch and N. Martins, "Nothing to fear? An analysis of college students' fear experiences with video games," *J. Broadcast. Electron. Media*, vol. 59, no. 2, pp. 298–317, 2015, doi: 10.1080/08838151.2015.1029128.

[21] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-Star Algorithm on Pathfinding Game," *J. Phys.: Conf. Ser.*, vol. 1898, no. 1, 2021, doi: 10.1088/1742-6596/1898/1/012047.

[22] C. Ying, C. Yuhang, and W. Yaodong, "Research on Monster Path Planning Scheme in Horror Game Based on Whale Optimization Algorithm," in *Proc. 6th IEEE Int. Conf. Intell. Comput. Signal Process. (ICSP)*, 2021, pp. 111–114, doi: 10.1109/ICSP51882.2021.9408889.

[23] R. Subari, N. Radita, and B. Prakoso, "The Implementation of A* Algorithm for Developing Non-Player Characteristics of Enemy in A Video Game Adopted from Javanese Folklore 'Golden Orange,'" *Teknika*, vol. 13, no. 2, pp. 164–174, 2024, doi: 10.34148/teknika.v13i2.799.