University for Business and Technology in Kosovo

UBT Knowledge Center

UBT International Conference

2023 UBT International Conference

Oct 28th, 8:00 AM - Oct 29th, 6:00 PM

A Comparative Analysis of Pathfinding Algorithms in NPC Movement Systems for Computer Games

Vesa Morina
University for Business and Technology - UBT, vesa.morina@ubt-uni.net

Rinesa Rafuna
University for Business and Technology (UBT) in Kosovo, rr47042@ubt-uni.net

Follow this and additional works at: https://knowledgecenter.ubt-uni.net/conference

Recommended Citation

Morina, Vesa and Rafuna, Rinesa, "A Comparative Analysis of Pathfinding Algorithms in NPC Movement Systems for Computer Games" (2023). *UBT International Conference*. 6.
https://knowledgecenter.ubt-uni.net/conference/IC/CS/6

This Event is brought to you for free and open access by the Publication and Journals at UBT Knowledge Center. It has been accepted for inclusion in UBT International Conference by an authorized administrator of UBT Knowledge Center. For more information, please contact knowledge.center@ubt-uni.net.

12th UBT Annual International Conference On Computer Science And Engineering

A Comparative Analysis of Pathfinding Algorithms in NPC Movement Systems for Computer Games

Vesa Morina[1], Rinesa Rafuna[2]

[1]UBT – Higher Education Institution,
Prishtine, Kosovo
vesa.morina@ubt-uni.net

[2]UBT – Higher Education Institution,
Prishtine, Kosovo
rr47042@ubt-uni.net

Abstract. Abstract. Non-Player Characters (NPCs) play a pivotal role in computer games, making them a focal point for game developers. The cornerstone of NPC behavior lies in the design of efficient pathfinding strategies, which are integral components of the underlying Artificial Intelligence (AI) models. This paper delves into the algorithms essential for realizing NPC pathfinding, with a specific emphasis on two prominent methods: the Dijkstra algorithm and the A* algorithm. Our study is rooted in an extensive review of existing research in this domain, followed by a comprehensive comparative analysis of these algorithms. Through this comparison, we aim to shed light on the advancements achieved in this field thus far. Furthermore, we provide a succinct summary of the limitations and challenges that warrant continued investigation and research efforts. By offering a nuanced exploration of pathfinding algorithms and their implications for NPC movement systems in computer games, this paper contributes to the ongoing evolution of AI in gaming and serves as a valuable resource for game developers and researchers alike.

Keywords: Pathfinding Algorithms, Non-Player Characters (NPCs), Artificial Intelligence (AI) in Gaming, Dijkstra Algorithm and A* Algorithm

1 Introduction

Since ancient times, games have been a significant source of entertainment for diverse generations, regardless of their type. In recent decades, with the continuous development of technology, especially personal computing devices, the popularity of games has grown to such an extent that today, the computer gaming industry is one of the billion-dollar industries. In the last year alone, it has generated revenues of up to 197 billion dollars in South America. Considering such statistics, which predict an approximately 36% increase in revenues in the next three years solely in South America's electronic gaming industry, game developers are increasingly interested in advancing technologies for development, as well as existing algorithms, which are essential for the development of the majority of games. The primary aim of this study is the analysis of the most widely used general algorithms for pathfinding, employed in the development of computer games. Observing the massive prevalence of these types of games, pathfinding algorithms have become a focal point of study for developers [1].

2 Identifying the problem

During the development of NPCs, two main problems are commonly encountered, which can often arise throughout the game development process. The first problem is the excessive CPU usage by pathfinding algorithms for NPCs, which can negatively impact the game's quality. This issue is particularly prevalent in computer games that feature a large number of NPCs, such as Call of Duty, Hogwarts Legacy, Elden Ring, Dark Souls, etc. A common phenomenon is that some algorithms may perform the required tasks in a specific game but have weaknesses, such as not guaranteeing the destination, not finding the shortest path to the destination, or unnecessarily traversing multiple routes before finding the correct path. Meanwhile, other algorithms may avoid these problems, but their implementation is not always the most optimal way to create NPCs. The second problem that may arise is the lack of inclusion of tactical information in decision-making. This means that even though an NPC calculates the shortest path to a point, that path may be entirely unnecessary from a player's perspective [2]. Multi-agent pathfinding, the study of finding paths for more than one NPC (Non-Player Character), is a critical aspect that specifically addresses the issue of route finding when there are multiple NPCs, a scenario prevalent in most games. These NPCs may need to behave in two ways: each having its own goal (having different destinations) or all having a common objective. In contrast to single NPC pathfinding studies, a study conducted by Botea [3] explicitly mentions two challenges that arise in cases of multi-agent pathfinding. The first challenge is the size of the joint action space, which becomes exponential with the number of NPCs. This complexity poses difficulties in finding a method for the global planning of this aspect. Additionally, considering that optimizing methods for a single NPC do not directly translate when there are multiple NPCs, collision management between NPCs becomes a crucial consideration. These challenges present significant hurdles, especially given that every exponential increase is a substantial CPU expense, impacting the overall gaming experience for the player.

Alongside memory consumption, a closely associated problem is the prolonged execution time. This is because we might have a game with a visually appealing and intricate map for the player's view, but the map takes a significant amount of time to load entirely. Additionally, character movements may experience delays as the CPU is occupied with executing background algorithms responsible for bringing NPC participants in the game to life. In such cases, the game loses its effectiveness, and the player's interest dwindles. Some studies have shown satisfactory results in scenarios with thousands of NPCs only when executed on supercomputers, which are impractical for widespread use. Nowadays, there has been a shift in approaching this problem by transitioning some algorithms from CPU to GPU, and performance analysis is now based on GPU [4].

3 Identifying the solution

Game developers are often tempted to create their characters based on AI, using the latest methods and algorithms that are increasingly trendy and widely used but often come with significant complexity. When, during development, AI has not yet learned to perform the intended tasks, developers frequently resort to basic algorithms, which have been in use since the Pacman game development era in 1979. The use of basic algorithms is always an efficient way to achieve the desired goal. However, even these algorithms do not guarantee success in every scenario. Therefore, a common solution is the combination of these algorithms and their modification in a way that minimizes errors and issues, making them virtually imperceptible to the player's eye. The algorithms under study will primarily focus on Dijkstra's algorithm and the A* algorithm. The emphasis lies on these algorithms because they are among the most commonly used in solving pathfinding problems in game development. Millington and Funge highlight in their book that the Dijkstra's algorithm is often employed in tactical decision-making rather than pathfinding. They assert that it serves as a simpler version of the A* algorithm, making it a starting point for understanding the latter.
To conduct a comprehensive analysis of each algorithm, it is essential to outline the problem each algorithm addresses, the approach it takes to solve that problem, the code or pseudocode representing the algorithm, the data structures used during its development, its performance, and any shortcomings observed during execution. This structured approach will facilitate a concise examination of these two algorithms [5].

In order to address the challenges related to memory consumption and execution time in gaming scenarios, a comparative study of these algorithms will be analysed. By analysing these studies and their performance metrics in order to identify potential optimizations, we aim to determine which algorithm proves more effective in enhancing overall gameplay experience.

4 Comparison of Algorithms

Efficiency Comparison

This study was conducted in three distinct environments, differing in obstacle positioning. The first environment arranges obstacles based on a grid model, the second adopts a symmetrical but non-grid pattern, and the third employs a random layout. Three different algorithms (A*, D*, Dijkstra) were tested, with the focus directed towards the comparative analysis of A* and Dijkstra.

For this study, the A* algorithm utilized the Manhattan distance as a heuristic. This choice was justified by the assumption that, for the majority of nodes, the distance to the destination must traverse at least one other non-destination node, causing the path to deviate from a straight line. The experiment comprised 50 trials for each algorithm, with results from each contributing to the overall conclusions drawn. According to the obtained results, Dijkstra exhibited the lowest efficiency due to its exhaustive exploration of the entire surface, lacking any space-constraining methods. The length of the paths found by all algorithms was similar, with the primary distinction lying in the number of visited nodes, where Dijkstra showed the highest count. Overall, the A* algorithm demonstrated significantly more satisfactory results than Dijkstra [6].

Comparative Analysis of A*, Dijkstra, and BFS Pathfinding Algorithms in the Maze Runner Game

This study was conducted in a game called Maze Runner, where the player has the ability to utilize obstacles and place them in the path of a Non-Player Character (NPC). The NPC's task is to navigate from the starting point to the destination, overcoming obstacles placed by the player in the process. Based on experiments of this study, it was concluded that both A* and Dijkstra algorithms can be employed to find the shortest path in the Maze Runner game. However, the A* algorithm, considering its minimal computation process and relatively short search time, proves to be particularly suitable for such games. This conclusion was drawn from evaluating the efficiency of both algorithms in the specific context of Maze Runner [7].

Performance Evaluation of A* Algorithms

This study was conducted on a single map, where each algorithm was applied only once, a choice justified by the researcher based on the primarily deterministic nature of the results. The tests conducted for this study measured the algorithm's execution time, the number of expanded nodes, the length of the path, and the number of nodes in the found path. The algorithm's execution time and the length of the found path are key determinants of algorithm efficiency. The results of this study are categorized into different algorithms, the outcomes of which will be compared with those of the A* algorithm. Among these algorithms, Dijkstra exhibited the least satisfactory results, while among the others, only Theta* was slightly slower compared to A*, whereas IDA* and HPA* were not recommended for use in such cases due to unsatisfactory results. It was emphasized that HPA* could be beneficial in scenarios where the developed application is time-sensitive and requires frequent Pathfinding usage [8].

5 Importance of NPCs

Non-Player Characters (NPCs) play a pivotal role in the gaming experience, shaping the virtual world and influencing the player's journey. Their significance lies in adding depth, complexity, and realism to the game environment. NPCs contribute to the narrative, offering quests, challenges, and interactions that enhance the overall gameplay. Moreover, NPCs often serve as crucial elements in multiplayer and role-playing games, facilitating dynamic social interactions and strategic collaborations. The importance of well-designed NPCs becomes evident in players' preferences, with certain characters becoming favorites and influencing the overall enjoyment of the gaming experience [9].

6 Conclusion

Across all studies, the primary goal is to create as many Non-Player Characters (NPCs) as possible, ones that are realistic enough for the game, capable of making intelligent strategic choices, and not distinguishable from other characters. This is presented as achievable through algorithms and the implementation of AI in the game. However, when it comes to increasing the number of these characters or expanding their intelligent activities, a significant drop in game quality has been observed. One of the simplest activities that NPCs should perform is to move from one position to another in the same way as a player-controlled character. Studies in this aspect have concluded that this can be achieved through search algorithms such as Dijkstra, A*, BFS, DFS, etc., but not all yield satisfactory results. Thus far, the most successful algorithm in this regard appears to be the A* algorithm and its variants. However, even this algorithm, in certain cases, does not optimize computer performance to the desired extent. The primary approach noted is the optimization and modification of the conditions under which this algorithm is applied, such as changing graphs and other data structures, combining algorithms, etc., which have been positive but not to the desired extent. Another relatively recent approach is the choice of the processor on which the algorithm will be executed. This approach, as a new idea, has shown positive results but is currently limited to certain GPUs (specifically those of NVIDIA). Additionally, due to the vast number of possibilities and maps, conclusive results on efficiency are yet to be determined. From all the studies, we can conclude that there are positive developments in this direction, but there is ample room for optimization. Therefore, there is still no definitive answer to what the ultimate solution for this problem might be.

References

1. Clement, J. (2023, January 16). Video game market value worldwide 2015. Statista. Retrieved April 3, 2023, from https://www.statista.com/statistics/292056/video-game-market-value-worldwide/

2. Yap, P., & Calway, A. (2006). Path planning for computer games. ACM Computers in Entertainment (CIE), 4(3), 1-24.

3. Botea, A., Müller, M., Schaeffer, J., & Teng, C. (2013). "Near Optimal Hierarchical Pathfinding." Journal of Artificial Intelligence Research, 46, 607-648.

4. Silva, A. M., Rocha, F., Santos, A. C. F., Ramalho, G., & Teichrieb, V. (2011). GPU Pathfinding Optimization. 2011 Brazilian Symposium on Games and Digital Entertainment. https://doi.org/10.1109/sbgames.2011.35

5. Millington, I., & Funge, J. (2009). Artificial Intelligence for Games. Taylor & Francis.

6. Krishnaswamy, Nikhil. (2009) Comparison of Efficiency in Pathfinding Algorithms in Game Development.

7. Permana, S. D. H., Bintoro, K. B. Y., Arifitama, B., & Syahputra, A. (2018). Comparative Analysis of Pathfinding Algorithms A *, Dijkstra, and BFS on Maze Runner Game. IJISTECH (International Journal of Information System and Technology), 1(2), 1. https://doi.org/10.30645/ijistech.v1i2.7

8. Martell, V., & Sandberg, A. (n.d.). Performance Evaluation of A* Algorithms. DiVA portal. http://www.diva-portal.org/smash/get/diva2:949638/FULLTEXT02.pdf

9. Millington, I., & Funge, J. (2009). Artificial Intelligence for Games. CRC Press.