Dean&Francis
ISSN 2959-6157

Heuristic Pathfinding Algorithms in Video Games Indepth Analysis and Performance

Runjie Lu

Department of Art and Science, Santa Clara University, Santa Clara, United States
lurunjie1301@gmail.com

Abstract:

The creation of artificial intelligence in video games is mostly dependent on heuristic pathfinding algorithms. The ability to move of the agent is one of the biggest obstacles in the construction of accurate artificial intelligence (AI) in video games. Before the A* algorithm emerged as a provably optimum solution for pathfinding, several search algorithms, including Dijkstra's algorithm, the bread first search algorithm, and depth first search algorithm, were developed to tackle the shortest route issue. Since its inception, it has drawn the interest of several scholars who have decided to work on it. This paper provides an in-depth analysis of these algorithms, focusing on their development, evolution, and performance. Beginning with an overview of classical algorithms like A*, the paper explores recent advancements and optimizations. Performance metrics and practical applications are discussed, with case studies from contemporary video games. This analysis highlights the balance between computational efficiency and path optimality, guiding future developments in game AI.

Keywords: Heuristic Pathfinding, Hierarchical Pathfinding, Dijkstra's Algorithm, Jump Point Search (JPS), Bidirectional Search.

1. INTRODUCTION

The rise of video games demands a growing number of Artificial Intelligence researches which allows to approach AI problems such as Strategy and Pathfinding [1]. An important problem is that finding a path for non-player characters (NPCs) makes use of high CPU and memory resource rates [2]. For AI NPCs in games to seem remotely intelligent and responsive, pathfinding algorithms are crucial as they allow them to move much more dynamically through various game worlds giving a better gameplay experience. Heuristic based pathfinding algorithms are a set of heuristic finding best way between two points, in some case shortest and most efficient, which can be taken reference for applications where there is need to solve the graph theory or get the robot from one point (position A) to another (Position B). Any of them contain Dijkstra's algorithm that is a single source shortest path algorithm [1,2]. It starts with the node at the source and repeats selecting a vertex not visited yet that has minimal distance from this selected list, then marking it has done along its edges to all unvisited neighbors.

ISSN 2959-6157

Based over Dijkstra's algorithm, the A* (pronounced "A star") is yet another well-known pathfinding heuristic. In A*, a heuristic function is used to estimate the cost from that node ot goal and it ranks nodes according to sum of $g(n)$ (the actual way so far) adn $h(n)$ (estimated way fo additional operations for getting into te end). This permits A* to guide its search towards the lower bounds seenenberg and, as a result, is faster than Dijkstra in many practical situations.

More examples of heuristic-based pathfinding are different A* variants with modified heuristic functions, and methods that extend Dijkstra's algorithm by means of a heuristic. The main goal of these algorithms is to keep the optimal pathfinding solution while being efficient enough & flexible so it can be used in any real-life scenario. The solution algorithm chosen, according to special requirements of the application, such as size and structure of search space, need for optimality vs. speed, availability of cost Research is still being carried out to develop new heuristic techniques or hybridized algorithms to further advance the ability of pathfinding.

2. BACKGROUND

Movement planning algorithms have become mandatory for real-time strategy (RTS) games like the "StarCraft II" to provide means for units to move through environments effectively. In these games, classical pathfinding algorithms such as Dijkstra and A* have been applied to define the shortest path between two nodes [3, 4]. While effective for less complex game environments, the classical algorithms of path planning have the following shortcomings: Heuristic pathfinding algorithms present themselves as a solution to such limitations of classical approaches. These algorithms employ the use of heuristic functions to direct search, hence making the search more effective and faster than the other approaches [5,6]. Heuristic algorithms used in pathfinding are classic in the construction of video game AI as they ensure NPCs traverse through various levels or maps with efficiency and relevance. The only algorithm that is most associated with heuristic pathfinding is the A* algorithm which was developed by Hart, Nilsson and Raphael in 1968. . There are several heuristic pathfinding algorithms, of which one of the simplest and one of the most widely used is the A* algorithm that is based on the Dijkstra's algorithm and the greedy best-first search methods [7]. It does this by minimizing the total cost function $f(n) = g(n) + h(n)$, where $f(n)$ is the cost to be minimized, $g(n)$ is the direct cost, and $h(n)$ is the sum of the holding cost and the constraint cost. Here $f(n)$ denotes the total estimated cost including the cost of the cheapest solution up to n, $g(n)$ is the actual cost of the path from the start node to node n, h(n) on the other hand is the heuristic estimation of the remaining cost of the path from node n to the goal node. This approach has been used often in RTS games because it can find efficient paths in the map faster than classical algorithms [8]. It must be an admissible one indicating that it will never estimate the cost more than the actual cost to attain the specific goal, thus making the path that was identified by the A* algorithm to be optimal.

The A* algorithm is quite famous because it provides both the optimality of the paths and costs time compared to other search algorithms. The actual cost $(g(n))$ for reaching a node together with the estimated cost $(h(n))$ to the goal makes the application of A* very efficient in search space because it reduces the search area, which makes it ideal for real-time applications such as video games. The fact that the algorithm can come up with the best routes while at the same time consuming less computational space has made the algorithm be widely used in creating intelligent NPCs navigation systems. Dijkstra's algorithm is developed by Edsger Dijkstra in 1956 and unlike the path finding algorithms to be discussed this algorithm only looks for finding the shortest path from start node to all other nodes in the graph whether it considers every possible route in the search [9]. Although Dijkstra is certain it provides the shortest route, it does not use a heuristic to help the search, which can reduce the application for large maps often used in video games. Unlike the above, Greedy Best-First Search selects nodes with most likely to be nearer to the goal by only considering the heuristic estimate $(h(n))$. As it establishes, Greedy Best-First Search can work very fast in terms of searching for a path; however, it does not provide the shortest path and may therefore be considered significantly less efficient than A*.

Another greedy pathfinding algorithm that has been employed in RTS games is the Potential Field algorithm also. This algorithm then models the environment as a field in which attractive and repulsive forces are generated by obstacles and goals respectively that lead the units [10]. In addition, the Potential Field algorithm has especially used for organizing the motion of many units since it can prevent collision and keep the units close together. However, this algorithm has the issue of local minima, where the units can end up in some rather less than ideal locations.

When it comes to video games, Real-Time Strategy (RTS) like 'StarCraft II' is a good example which requires Heuristic Pathfinding Algorithms. RTS games entail players controlling many units that must move on the map while

avoiding hindrances and enemy forces while at the same time setting out and performing sophisticated strategies [11]. These navigational problems are often solved by applying the A* algorithm because it is fast and relatively accurate. If other optimization comprises in hierarchical pathfinding, games developers can organize the appropriate movement of units in an intelligent manner and thus improve game play. This approach of navigating through the game map can be regarded as the process of dividing the game map across several levels. This simplifies the work of path finding in large maps by giving the algorithm the small maps to work on to provide solutions for a large-scale map. This technique is more useful in games which have large environments which if, the whole map must be managed at once would be very expensive to process.

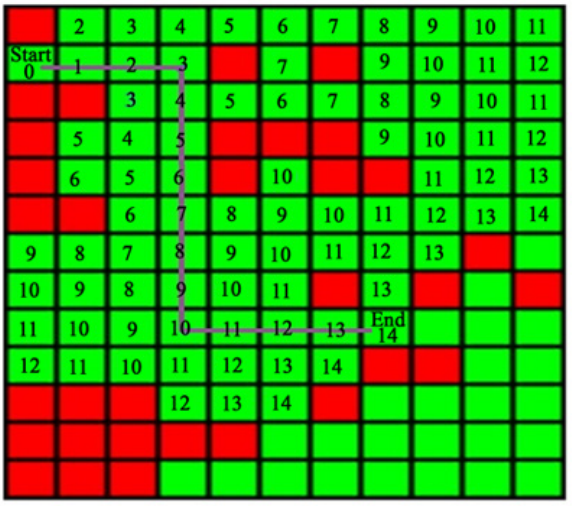

Figure 1. Shortest pathfinding from source to destination.

Figure 1 shows a fundamental idea of finding the shortest path between two points on a map or grid: how different algorithms find the best route. It is a basic example to help understand pathfinding techniques.

3. EXTENSION AND EVOLUTION OF CLASSICAL HEURISTIC ALGORITHMS

The traditional A* search algorithm is still considered a fundamental tool for finding paths in video games due to its optimal and efficient nature [1]. Nonetheless, the current day video games complex and large requires extension and optimization of the particle system. Learned herein is that there exist solutions to the challenges faced by A* in large dynamic worlds hence enabling efficient pathfinding to take place.

3.1 Hierarchical Pathfinding

This is one of the biggest enhancements on top of A* called Hierarchical Pathfinding. The use this technique and decompose the game map into a hierarchical cluster [11]. The algorithm can then act to a smaller scale within each region, lowering the complexity of the pathfinding process.

Hierarchical Pathfinding works in practice by planning a high-level path within the larger areas first, then performing detailed search of sub regions. So for a very large open-world game the map will be split into cities -> neighborhoods, and then individual streets. It finds a route between cities with the pathfinding algorithm, through neighborhoods and finally amongst streets. This multi-layered mechanism drastically lowers the computational burden paths results in quicker and precise routing. This is quite useful in games where you have big, open environments and the amount of nodes to expand by A* becomes too slow or memory-heavy [12]. This allows it to process more complex maps and return a response quicker after the player makes an action.

3.2 Jump Point Search (JPS)

Another optimization technique which is based upon grid-based maps, those are commonly used in many video games. JPS makes A* better by eliminating some of the nodes that it expands. It does so by moving (``jumping'') several steps in direct lines, thus avoiding nodes to be explored. Traditional A* will expand all the nodes on the grid which results in many computations. The JPS solves this problem by recognizing “jump points”, critical nodes that change the path’s direction drastically. Instead of evaluating all the intermediate nodes, JPS skips over these and only evaluates critical points which can reduce the

ISSN 2959-6157

search space significantly [13]. Smarter AI will choose to not navigate through a straight path filled with unnecessary waypoints and jump directly from the initial point of that segment of travel right till its endpoint without having to evaluate every node in between, for example a grid based map. This optimization could be magnitudes faster in some cases, such as games with vast open grid maps where the path finding process would slow down.

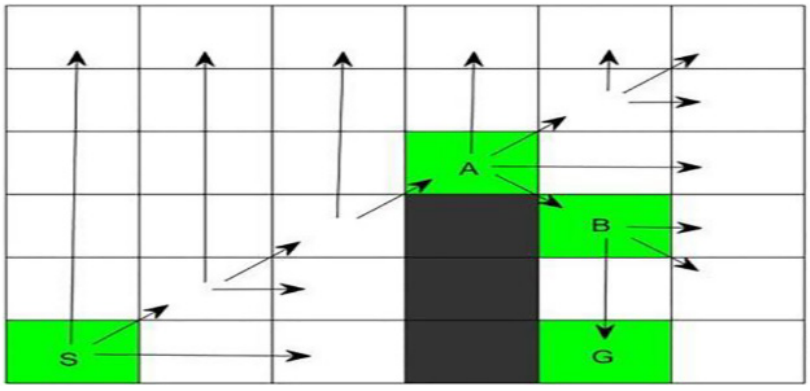

Figure 2. Jump Point Search. JPS-algorithm-pathfinding process JPS.

Figure 2 shows the JPS algorithm in action: Note how it "jumps" from one critical node to another methodically, thereby avoiding the need to consider redundant nodes as a way of routing or path-finding more efficiently. The efficiency gains by using JPS over other methods in such a diagram become clear.

4. Bidirectional Search

Bidirectional Search is an optimization that involves running two simultaneous searches: This is because each node in the graph corresponds to some activity and they will need one edge from the start node and one from the goal node. Both search's advance seeking each other and when they converge, essentially, they split the search space in half. Bidirectional Search is hence less expensive than unidirectional search because it constructs the paths from both ends of the problem and thus eliminates the nodes that are likely to be generated in both searches. This technique is suitable in large maps since the distance separating the start node and the goal node is likely to be large. As for the implementation, Bidirectional Search can be used along with other improvements like the hierarchical decomposition or JPS [14]. This is a convenient tool for determination of routes, which will help NPCs to move fast and without mistakes, even if the game world possesses maximal complexity.

5. Real-Time and Anytime Algorithms

Applications that are designed for dynamic environments where the map could be changing over time include the Real-Time and Anytime Algorithms; D*-Lite and Lifelong Planning A* (LPA*). These algorithms enable continual rehoming and are effective at responding to alterations on the agenda in real-time and as such, are perfect for enhanced video games with highly engaging environments.

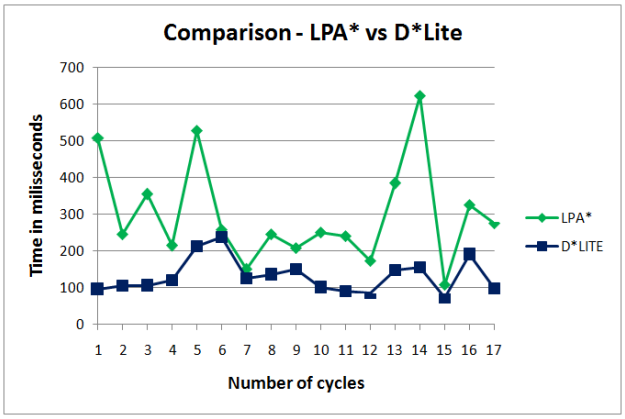

Figure 3.Time-efficiency-LPA-vs-DLite diagram.

Figure 3 shows time complexity comparison between LPA* and D*-Lite algorithms. It gives a good overview

of how these methods work in a dynamic environment. It shows the trade-offs for different pathfinding approaches. For example, D*-Lite is an incremental search algorithm that operates off prior searches to create new ones. If the map changes for some reason (e.g.) because of the appearance of a new obstacle or the movement of the target), D*-Lite does not calculate a new path but updates the existing one. This is a very effective strategy in organizations with fluctuating but not massive changes. LPA* is also similar in the sense that it emphasizes flexibility when there are frequent changes. There exists a list of active nodes that should be updated and the algorithm incrementally modifies the path as changes happen. This allows for constant updating when for instance, a blockage is met but no total recalculation of another whole path, thus making path finding continuous and smooth. Real-time algorithms are critical for the games which require dynamic environment such as destructible terrain, moving objects or the changing goal. In this way, they guarantee that NPCs are ready to face new challenges and changes within the game environment and can find themselves in it, thus improving the game environment.

6. ANALYSIS OF THE LATEST HEURISTIC PATHFINDING TECHNOLOGY

The current developments of heuristic pathfinding have greatly improved the efficiency and flexibility of AI technologies in complex worlds. These enhancements are prompted by the problem of requiring better pathfinding algorithms to suit the dynamics of today's video games. Over the past year, two of the most distinctive improvements can be considered: Flow Field Pathfinding, and Machine Learning Approaches.

6.1 Flow Field Pathfinding

The concept labeled Flow Field Pathfinding is a new important breakthrough in the field of pathfinding algorithms was delivered particularly in relation to the real time strategy genre and other games in which movement of many units is an important factor. One of the games, where this technique was applied, is the "Supreme Commander" game and it proved its effectiveness and versatility.

The basic idea of Flow Field Pathfinding allows for the calculation of paths before a game and then creating a vector field that will lead units to their target [15]. Unlike the conventional pathfinding algorithms that calculate the paths to each unit separately, Flow Field Pathfinding generates a grid interlinking the game map. Every square in this grid is a vector out to the goal. The objects that traverse through this grid just follow these vectors, although these vectors together make a flow field.

Thus, the ability of Flow Field Pathfinding to find a way for big groups of units is one of its major advantages. Suppose the game is a strategy game where players directly manage hundreds of units on the map, such as 'Supreme Commander'. That is moderated by the Flow Field Pathfinding since it enables all the units to see a precomputed vector field instead of computing it.

Flow Field Pathfinding also has outstanding performance whenever there are alterations in the conditions of the surroundings. That is why the path planning can be made incrementally when the obstacles appear or disappear, instead of recalculation of the vector field for the all units. That is why it is optimal for games with destructible environments or moving objects on the level. Moreover, the conveniently executed according to the flow field smooth and resulting motion contributes to better progressing of units' maneuvers as well as providing a strategically deeper exposition of the battlefield layout which also adds to the general experience and its plausibility.

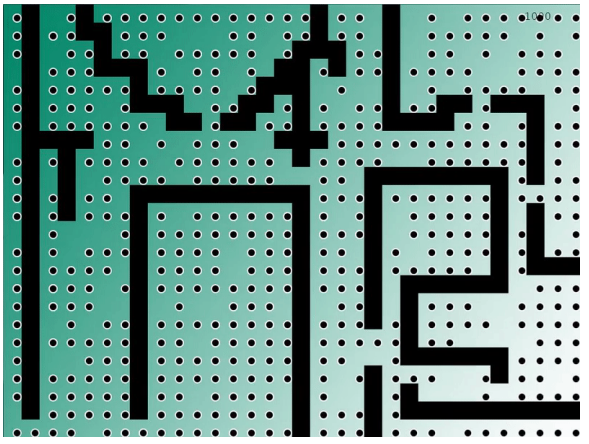

Figure 4. Flow Field Pathfinding image.

Figure 4 shows Flow Field Pathfinding the Flow Field Pathfinding technique guides units through a vector field across a grid-based game map. The Flow Field Pathfinding technique enables efficient movement of the units in real-time strategy games. This figure shows that this approach can handle thousands of unit movements.

7. Machine Learning Approaches

Increasingly, pathfinding algorithms are incorporating machine learning (ML) techniques to improve performance and adaptability. These methods use the massive amounts of data generated during gameplay to train models that can predict optimal paths or dynamically adjust heuristics based on the state of the game. One of the main uses of machine learning in pathfinding is the creation of mod-

ISSN 2959-6157

els that can predict the optimal path in a given scenario. These models are trained on large datasets that contain many successful pathfinding outcomes. Through examples like this, the model learns and can generalize to new unseen situations with high quality path finding decisions in real-time. This premonition ability leads to a considerable increase in the efficiency of pathfinding algorithms.

More generally, one could think of other uses cases where the heuristics are dynamically altered based on various part/situation in game. Traditional pathfinding algorithms use constant heuristics in a game that do not always apply. These models need to update the heuristic function as they are learning how to detect patterns in certain states of game. If, for instance, a game has enemy units that spawn on some paths (obstacle), the machine learning model will learn to predict these obstacles and adapt its heuristic as well. This kind of flexibility makes it easier and faster to identify obstacles in non-stationary environments.

Pathfinding using reinforcement learning (RL) is one of such branches [16]. In the RL, some rewards or penalties are given to an agent to make decisions basing on how it acts. The agent employs trial-and-error techniques in identifying best approaches of maneuvering tough environments within which they exist. In pathfinding, reinforcement learning agents learn how to get through maps faster by checking different paths as well as evaluating the outcomes associated with such choices. Eventually, this means the agent will have found the shortest or best possible way towards the goal.

There are challenges in combining machine learning and pathfinding algorithms. Among them is computational burden concerned with training machine learning models that can be considerable. Nevertheless, after this phase of model training is over, game playing benefits become very significant for these models. It is also important for these models to generalize well across diverse scenarios and not overfit on the training data.

Despite these challenges, machine learning in pathfinding has immense potential benefits. These techniques make AI systems for video games more effective and adaptable, thus leading to more responsive and intelligent NPC behavior during the game. The possibility of dealing with complex and dynamic environments better allows for creating deeper gaming worlds [16].

The latest progress in heuristic pathfinding technology is a great example of this, especially Flow Field Pathfinding and machine learning approaches. For example, Flow Field Pathfinding enables for management of large numbers of units and can cater for changing dynamics of the environment. On the other hand, predictive models as well as reinforcement learning which are types of machine learning offer improved performance and adaptability so that AI systems can successfully navigate ever more complex and dynamic environments. This is important because current video games require such advances to its path finding algorithms to maintain high levels of performance and realism.

8. PERFORMANCE EVALUATION AND PRACTICAL APPLICATION CASES

The evaluation of heuristic pathfinding algorithms involves a set of criteria which are used as the base for measurement [15]. Time complexity, crucial in real-time scenarios where quick decision making is mandatory, defines an upper bound on the amount of computational time it will take for the algorithm to find a path [15]. Space Complexity: Calculates how much memory is required for a method or function, and in which situation can use this proper means if input space is low. To guarantee efficient navigation, there is also need to demonstrate path optimality between how close the discovered route is to an actual shortest path. As a third criterion, one can take the ability of an algorithm to solve the same problem for more complex conditions and larger maps. This is directly related to modern video games; whose worlds are so vast. Case studies also confirm the benefits of more advanced pathfinding. “The usage of such advanced techniques is backed up by several case studies. In “The Legend of Zelda: Breath of the Wild”’s NPC pathfinding, an heavily modified version of A* is used for instance. Characters in the original implementation computed paths directly in world space, which could be a heck of work given such large and intricate environment. This new version uses hierarchical pathfinding, which vastly improves performance. Layered architecture of the game world so that some level or logic (like AI) can run in a smaller part. This will not only reduce the overhead of pathfinding but also enable to use different kind of algorithms depending on each surface type for traversing nodes. For example, the smoother terrains will use simpler traversal and rugged areas would ting to more complex one. The way hierarchical pathfinding works is a two-level system - on the high level it decides in which zones to go, while at low levels how exactly move from point A to point B within one specific zone.

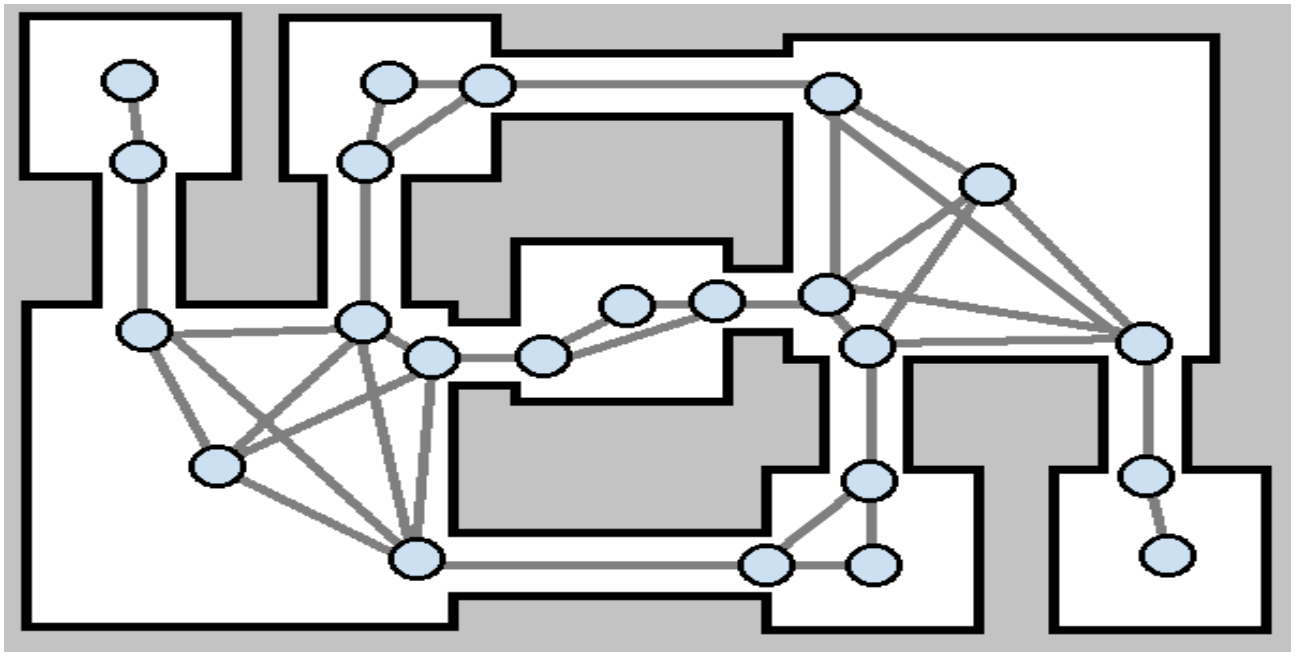

Figure 5. Pathfinding Illustration diagram.

Figure 5 summarizes various pathfinding algorithms in practice, comparing the more classical approach of A* to other modern machine learning-based methods. This is more of a high-level overview of what kind of strategy each one uses within game AI.

Figure 6. "The Legend of Zelda: Breath of the Wild".

Figure 6 shows Pathfinding in the "Legend of Zelda: Breath of the Wild" likely uses hierarchical pathfinding to handle such complex environments. It should be underlined that this figure allows a better understanding of how vital pathfinding is in open-world games. StarCraft II presents an example of Flow Field Pathfinding to deal with numerous units [17,18]. This method generates a vector field that efficiently guides automatons to their

ISSN 2959-6157

goals by precomputing paths. To maintain strategic depth and make sure there is still good movement and fluidity in units pathing during the chaos of large-scale fights, the game uses a flow field that many entities reference for fast movement & collision avoidance.

Figure 7. "StarCraft II" In-game view.

Figure 7 shows Flow Field Pathfinding technique implemented in "StarCraft II": how the game manages many units traversing the battlefield. This is an actual application of pathfinding algorithms in real-time strategy games. A* and machine learning algorithms are combined in "Red Dead Redemption 2" to provide dynamic and realistic pathfinding for both NPCs and wildlife [19]. A more realistic and adaptable navigation system is made possible by the game's ability to dynamically modify heuristics based on the status of the game with the inclusion of machine learning [20]. This combination makes sure that animals and characters move organically in the intricately detailed landscape, giving players a more immersive experience.

Figure 8. A screenxhot of "Red Dead Redemption 2".

Figure 8 shows Dynamic pathfinding is also quite realistic in "Red Dead Redemption 2", a hybrid technique of

the A* algorithm combined with machine learning. This screenshot, taken from the game, shows how advanced AI techniques enhance the realism of game environments

9. CONCLUSION

The heuristic pathfinding has improved greatly over time to handle the needs of modern video games, robotics simulations and other dynamically changing environments. While the base A* algorithm still exists, along with a slew of extensions and optimizations (Hierarchical Pathfinding, Jump Point Search, Bidirectional Search) that have helped make it faster and more scalable. The most recent advances, with Flow Field Pathfinding and machine learning techniques have more advanced performance for managing large groups of units adapting to real-time changes. How they form a "backbone on which the rest of our games middleware can be built," and how canon-breaking case studies from The Legend of Zelda: Breath of The Wild, to StarCraft II, to Red Dead Redemption 2 prove these technologies aren't just theory. These are highly adoptative methods which utilized pro-actively during gaming enable efficient navigation, realistic movement and strategic depth within the game-play experience. Given that gaming environments are becoming more complicated, the marriage of machine learning with old-fashioned heuristic models will inevitably push matters further providing AI systems that are even smarter and quicker. Overall, improving the existing heuristic pathfinding will continue to play a critical role in building smarter, adaptive and efficient navigation systems into video games creating richer virtual worlds.

REFERENCES

[1] Mathew, G.E., 2015. Direction based heuristic for pathfinding in video games. Procedia Computer Science, 47, pp.262-271. https://www.sciencedirect.com/science/article/pii/S1877050915004743

[2] Rafiq, A., Kadir, T.A.A. and Ihsan, S.N., 2020, February. Pathfinding algorithms in game development. In IOP Conference Series: Materials Science and Engineering (Vol. 769, No. 1, p. 012021). IOP Publishing. https://iopscience.iop.org/article/10.1088/1757-899X/769/1/012021/meta

[3] Monzonís Laparra, D., 2019. Pathfinding algorithms in graphs and applications. https://diposit.ub.edu/dspace/handle/2445/140466

[4] Foead, D., Ghifari, A., Kusuma, M.B., Hanafiah, N. and Gunawan, E., 2021. A systematic literature review of A* pathfinding. Procedia Computer Science, 179, pp.507-514. https://www.sciencedirect.com/science/article/pii/S1877050921000399

[5] Barnouti, N.H., Al-Dabbagh, S.S.M. and Naser, M.A.S., 2016. Pathfinding in strategy games and maze solving using A* search algorithm. Journal of Computer and Communications, 4(11), pp.15-25. https://www.scirp.org/journal/paperinformation?paperid=70460

[6] Sidhu, H.K., 2020. Performance Evaluation of Pathfinding Algorithms (Master's thesis, University of Windsor (Canada). https://search.proquest.com/openview/3998aa7641a5c816dde91ddf8fe9b8f0/1?pq-origsite=gscholar&cbl=18750&diss=y

[7] Kapi, A.Y., Sunar, M.S. and Zamri, M.N., 2020. A review on informed search algorithms for video games pathfinding. International Journal, 9(3). https://web.pdx.edu/~arhodes/ai8.pdf

[8] Duarte, F.F., Lau, N., Pereira, A. and Reis, L.P., 2020. A survey of planning and learning in games. Applied Sciences, 10(13), p.4529. https://www.mdpi.com/2076-3417/10/13/4529

[9] Dijkstra's algorithm, developed by Edsger Dijkstra in 1956, focuses solely on finding the shortest path from a starting node to all other nodes in the graph, evaluating every possible route. https://www.academia.edu/download/65634054/G0810014047.pdf

[10] Yao, Q., Zheng, Z., Qi, L., Yuan, H., Guo, X., Zhao, M., Liu, Z. and Yang, T., 2020. Path planning method with improved artificial potential field—a reinforcement learning perspective. IEEE access, 8, pp.135513-135523. https://ieeexplore.ieee.org/abstract/document/9146273/

[11] Zhou, W.J., Subagdja, B., Tan, A.H. and Ong, D.W.S., 2021. Hierarchical control of multi-agent reinforcement learning team in real-time strategy (RTS) games. Expert Systems with Applications, 186, p.115707. https://www.sciencedirect.com/science/article/pii/S0957417421010897

[12] Černý, M., 2016. Reducing Complexity of AI in Open-World Games by Combining Search-based and Reactive Techniques. https://dspace.cuni.cz/handle/20.500.11956/82390

[13] Sharma, M. and Jindal, H., 2022. Pathfinding Visualizer. http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/3757/1/Pathfinding%20Visualizer.pdf

[14] Harabor, D. and Stuckey, P., 2018. Forward search in contraction hierarchies. In Proceedings of the International Symposium on Combinatorial Search (Vol. 9, No. 1, pp. 55-62). https://ojs.aaai.org/index.php/SOCS/article/view/18454

[15] Lawande, S.R., Jasmine, G., Anbarasi, J. and Izhar, L.I., 2022. A systematic review and analysis of intelligence-based pathfinding algorithms in the field of video games. Applied Sciences, 12(11), p.5499. https://www.mdpi.com/2076-3417/12/11/5499

[16] Alkazzi, J.M. and Okumura, K., 2024. A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. IEEE Access. https://ieeexplore.ieee.org/abstract/document/10506521/

[17] Chan, L., Hogaboam, L. and Cao, R., 2022. Artificial intelligence in video games and esports. In Applied Artificial Intelligence in Business: Concepts and Cases (pp. 335-352).

ISSN 2959-6157

Cham: Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-031-05740-3_22

[18] Churchill, D.G., 2016. Heuristic search techniques for real-time strategy games. https://era.library.ualberta.ca/items/c3589c0d-9b9e-46d5-b1a1-b4004379faad

[19] Ennabili, T.Y., 2023. A Comparison of Traditional Game Design vs. AI-Driven Game Design. https://www.theseus.fi/handle/10024/816173

[20] Zeng, J., Ju, R., Qin, L., Hu, Y., Yin, Q. and Hu, C., 2019. Navigation in unknown dynamic environments based on deep reinforcement learning. Sensors, 19(18), p.3837. https://www.mdpi.com/1424-8220/19/18/3837.