Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DEVELOPMENT OF PATHFINDING USING A-STAR AND D-STAR LITE ALGORITHMS IN VIDEO GAME

RISKA NURTANTYO SARBINI[1] , IRDAM AHMAD[2] , ROMIE OKTOVIANUS BURA[3] , LUHUT SIMBOLON[4]

[1]Student at Doctoral Program of Defense Science, Indonesia Defense University, Bogor, Indonesia.

[2,3,4]Lecturer at Doctoral Program of Defense Science, Indonesia Defense University, Bogor, Indonesia.

E-mail: [1]riskanurtantyosarbini@gmail.com, [2]irdam.ahmad@idu.ac.id, [3]romiebura@idu.ac.id, [4]lsimbolon427@gmail.com

ABSTRACT

Currently Artificial Intelligence (AI) is a very important component and is commonly used in video games. AI is used in games to make the player's experience more interesting and interactive. Artificial intelligence (AI) is utilized in a number of video games to enhance the player's experience and make it more interactive, particularly in role-playing games (RPG). Find a path (pathfinding) is essential to many computer games, especially role-playing games, and is required of the AI in the majority of games. Using the A* algorithm is one of the pathfinding methods used in video games to find the shortest path on the track to avoid static or dynamic obstacles, and The D* Lite algorithm is highly successful and capable of eliminating several difficulties and producing more ideal outcomes. The purpose of our research is to build and evaluate the performance of an artificial intelligence pathfinding system in searching for the fastest route using the A-star and D-star Lite algorithms. This work was carried out to collect data on how NPC movement works using path finding and study the results of this research. The path finding methods that will be studied are the A* algorithm and the D* Lite algorithm. The contribution of this research provides benefits to path finding problems which are commonly used by NPCs for technological games in the future, especially when using the A* algorithm and D* Lite algorithm in game technology. However, the pathfinding method is not only for games, but can also be implemented in other fields. The results of the experiment at five points under Pathfinding mechanism using the D-star Lite algorithm were faster in finding the closest route with a record time of 00:01.847 while using the A-star algorithm obtained 00:05.231.

Keywords: *Video Game, Artificial Intelligence, Pathfinding, A-Star Algorithm, D-Star Lite Algorithm*

1. INTRODUCTION

Currently Artificial Intelligence (AI) is a very important component and is commonly used in video games. AI is used in games to make the player's experience more interesting and interactive [1]. AI developed on a system application allowing players to compete against game objects that mimic other players [2]. AI, which is also commonly referred to as machine intelligence, is a form of simulation of human intelligence that is implemented and can be achieved through computer programming. AI can be activated by invoking/triggering human commands or independently based on AI experience. Therefore, AI is able to improve itself because it is programmed to learn from experience according to the program that has been included. AI can act like humans, think like humans, act rationally and think logically are the characteristics of an AI system [3].

Numerous strategies and algorithms including pathfinding method have been created to imitate the interaction between a computer system and human user. AI is predominantly employed to provide intelligent, adaptive, or responsive behavior in non-player characters (NPCs) in video games [4]. Due to the fact that AI in video games is typically customized for each game, AI tools in video games nowadays are designed to enable video game developers to generate particular AI efficiently and fast. One drawback with this technique is that it does not properly leverage the commonalities between video games media of the same genre, but also between video games of different genres, making it impossible to independently manage many parts of a complex ecosystem for each video game [5].

Diverse AI techniques has been used in video games, including decision making, pathfinding, intelligent narrative technology and character

intelligence [1]. Find a path (pathfinding) is essential to many computer games, especially role-playing games, and is required of the AI in the majority of games. Pathfinding system has been a significant area of research in video games platform for decades [6] and is generally utilized as the core of any AI movement system in video games. The role of intelligent pathfinding systems is always the focus of investigation. Intelligent pathfinding systems rely heavily on intelligent pathfinding algorithms [7]. Quite a few common pathfinding algorithms are implemented in a game [8], and the A* Algorithm and D* Lite Algorithm serve as basis for performance analysis in this paper.

The A* algorithm is one of the pathfinding methods used in video games to find the shortest path on the track to avoid static or dynamic obstacles [9]. It is a path-finding algorithm that has been utilized by the road-finding research community for a very long time. In comparison to other algorithms, its effectiveness, simplicity, and modularity are frequently cited as advantages. A* has become a popular choice for researchers trying to solve pathfinding problems [10] as a result of its widespread and pervasive use. The performance of the A* algorithm is mainly determined by the speed in planning the path search and the accuracy in determining the planned path. Pathfinding planning speed and path determination suitability are also being studied because they are related to the conventional algorithm effectiveness and speed function of the A* algorithm, determining expansion distances, and path equalization in the process [11].

The A* algorithm is the most commonly used algorithm for heuristic search; it finds the optimal path on a track and provides the optimum estimated calculation on the route that traverses target node. The node used as the starting point for this heuristic estimation sequence [12]. Utilizing the heuristic function as a calculation optimization by assigning a cost value to each node, is an advantage of A* which can provide the required solution. The basic terms commonly used in the A* algorithm are starting point, node, open list, closed list, cost (price), and bottleneck [13]. The A* search algorithm is the algorithm most frequently used in grid-based path searching [14].

Similar to A*, the D* Lite pathfinding algorithm is utilized in this game to locate the shortest path on the track by performing a heuristic search [15]. The D* Lite algorithm is three to seven times quicker than Generalized Adaptive A*, the fastest incremental search algorithm for resolving target search issues [16]. Under the D* Lite algorithm, any node can be transitioned into any position on the neighboring edge of the grid cell, and not simply the terminus of the edge node. In addition, the A* algorithm eliminates the heading limits provided on the route, enabling sustainable cut-off ranges and providing a less costly direct solution.

The D* Lite algorithm is highly successful and capable of eliminating several difficulties and producing more ideal outcomes [17] with a few small tweaks. It is an incremental heuristic pathfinding technique that could be used in circumstances when the environment is unknown in whole or in part to identify the shortest path. Historically, the bulk of pathfinding algorithms were constructed under the assumption that the environment model was complete and accurate [16]. This study proposes an Artificial Intelligence (AI) development strategy for performing pathfinding using the A* and D* Lite algorithms. The purpose of this study was to development and evaluate the time-related effectivity of an artificial intelligence system for pathfinding in an RPG Maker MV-based video game. This work was carried out to collect data on how NPC movement works using path finding and study the results of this research. The path finding methods that will be studied are the A* algorithm and the D* Lite algorithm. The scope of the work we do is to provide pathfinding type artificial intelligence capabilities to NPCs to search for the same target point using different algorithms, namely the A* and D* Lite algorithms. Researchers provide an overview of how NPCs can move in a situation by showing several examples of path finding algorithms for NPCs moving from one position to another.

2. RESEARCH METHODOLOGY

2.1. Pathfinding on Video Game

Many researches have studied the use of pathfinding, one of which is P. C. Iloh's study using Depth First Search, Breadth First Search, and A star Algorithm (A*) [18] is one of the numerous studies on the use of pathfinding. This study compares three different obstacle layouts for each search algorithm based on their path length and execution time. In their conclusion, in maze game can use Depth First Search, Breadth First Search, and A* algorithms to find the shortest path to the destination. In Maze games and grids, the A* algorithm is the best pathfinding method. This is supported by the fact that the execution procedure takes only a few milliseconds and the destination search takes a relatively short time. Other studies have proposed various types of video game

pathfinding. U. A. S. Iskandar et al. carried out one of these studies. According to the findings in their paper, the A* algorithm along with the Djikstra algorithm, are the most popular algorithms in video games. In addition, it is observed that the execution time, nodes, travelled distance, and the number of nodes calculated by the algorithm are the most crucial factors in determining an algorithm's effectiveness. Important is that all experiments are conducted in a 2D environment view, involve only horizontal movement, and do not account for the limited vertical movement typical of 2D platform games. The results of the algorithm experiments show that the A* algorithm will work more efficiently once the obstacles that the NPC has to pass pass a certain height value, because it will try to use the highest possible jump arc available to the NPC to reach the goal faster while requiring fewer nodes to be calculated. The results of the efficiency of the two algorithms show that A* is more suitable for long-distance movements in large areas. [6].

In a separate study, Hailong Huang investigates the traditional A* algorithm's use of particulars and proposes several optimization strategies to enhance it. Initially, heuristic coefficients are added to the heuristic function of the A* algorithm in order to optimize redundant pathfinding during the pathfinding procedure. Using a binary stack, the A* algorithm manages the open list. Next, the map's path weights are optimized. The produced path is smoothed using the median insertion technique, and the A* algorithm is further refined using the zone search technique, allowing it to be applied to a wider variety of game maps. By testing the Unity3D engine-generated simulation experiment software, the aforementioned approach was validated. The outcomes of this study indicate that by raising the standard of the intelligent pathfinding algorithm, the pathfinding efficiency is improved, the range of applications is expanded, the player experience is enhanced, and the approach is applicable to video games [7].

Additionally, Ade Chandra et al. created a 2D tile-based game with a grid map on the android platform as part of their research. This game contains a variety of elements, such as adversaries, barricades, trees, and the ground. A star method is used to identify the enemy's closest probable path to the tree. The player must then guard the tree from enemies by touching them or creating barriers in order for the tree to survive. The game will terminate when the enemy successfully fells a tree. This investigation will reveal the enemy's path to the tree, as well as the time required to traverse it [13].

Mochamad Kholil conducted the other study, which proposed a simulation involving troop agents pursuing a moving enemy. The placement of the food is determined by chance. When an agent moves closer to a target or opponent, the agent's food source is considered its new position. The modified D* Lite method is intended to replicate the movement of an adversary's attacks. The agent will move through a three-dimensional space containing both static and dynamic obstacles that must be avoided. The avoidance movement consists of circling the obstacle and passing by its left and right sides. The agent will pursue and follow the target position. Depending on the optimal approach to the target for each agent [19], the optimal path to the objective for each agent can differ. Consequently, it is required to assess the performance of the A* and D* Lite pathfinding algorithms in the games developed for this study. So we proposed a research using of AI by the A* algorithm and the D* Lite algorithm requires us to test them on video games that we have created to determine which algorithm is more effective. This research also offers a number of contributions, as shown in figure 1 below.

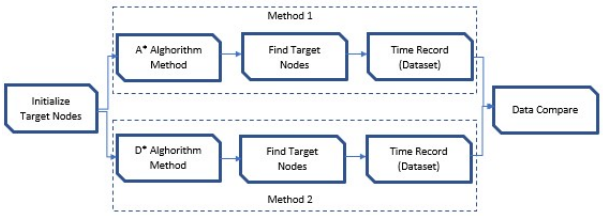

Figure 1. Proposed System

In this study, we build a video game in a Role Playing Game (RPG) design inside the game makers named "Anak Negeri" [25], with one area being a finding position by determining the target point to be the goal of finding the path. The procedure from the beginning of the process used is to determine the target point, then the process of finding the path to the specified point is carried out using one of the algorithms, the speed of time needed to reach the target point is an indicator that is used as a reference in performance. Furthermore, from the time data obtained, data collections will be carried out in the final results to be compared between the time required and the utilize of two algorithms. To get the desired recommendations we use JS Profiler application to generate execution time analysis.

2.2. Research Method

In this study, we proposed a system into two main parts. The first is the creation of a video game application and the next is a path-finding

experiment. This type of research is software engineering by SDLC Stages with a waterfall model. Software development is the most essential aspect of software project management, whereas software process models are a crucial means of obtaining standards, particularly for the production of software in the form of digital games. The period during which processes such as defining, developing, testing, deploying, operating, and maintaining application software or systems occur is known as the Software Development Life Cycle (SDLC) [20]. The productivity and quality of the development team depend on determining and assessing software process metrics throughout the SDLC can be seen on next figure 2.

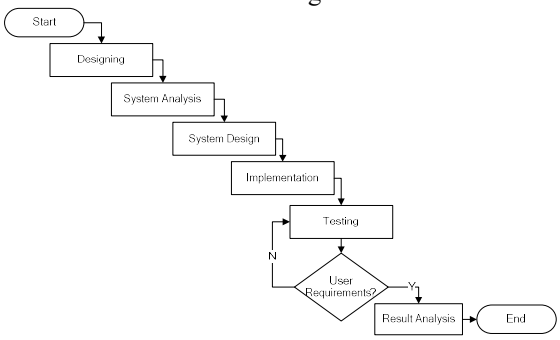

Figure 2. The development steps of Waterfall Model

The software engineering model employed is the SDLC method with the waterfall model [21], which is a sequential design frequently used in the best programming because each phase of analysis, design, implementation, construction, testing, and maintenance is executed sequentially and flows down like a waterfall. The waterfall display represents slow and progressive improvement. Database design begins with creating the necessary tables, determining the keys in each table, relations between tables, and so on using the database application program contained in the RPG Maker MV program. The system of producing this game used the RPG Maker MV application, further testing is aimed at determining the performance of the pathfinding mechanism in which there are functions of the Djikstra and A* algorithms to see the results obtained. The target of this research is to build a stable application on a smartphone and continue with the trial of finding the search location point using pathfinding mechanism with the A* and D* Lite algorithms. The research procedure carried out is to start with making an application and then analyzing the data for finding the closest route related to the time needed to search for the nearest route.

2.3. Research Procedure

In this study, the procedure carried out by the writer is to run a scenario with a single event, namely a map with a maze. The input of the same map run by two different algorithm mechanism in search model using the A* and D* Lite algorithm methods. Total time data will be obtained based on 5 points using the A* algorithm and 5 points using the D* Lite algorithm. Furthermore, it will be illustrated that the red dot target is the target point and the blue writing is the 5 points that will be tested against the required time. It can be seen in the next Figure 3, and the illustration, the artificial intelligence capability of pathfinding method will be placed on which NPC at node 1, coordinate (31, 14) which will search the path to the end point at coordinates (31, 18). Illustration can be seen from the next Figure 4.

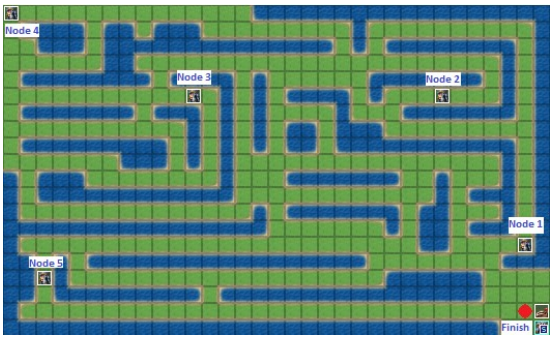

Figure 3. Map Scenario With 5 Nodes Target

Figure 4. Node 1 to Finish Node

2.4. A* Algorithm Mechanism

In the mechanism of A* Algorithm usage, the next step is to calculate the overall scoring stages, which is written in $F_{(n)}$ at each point from start to finish [22]. Eqs. (1) is the equation for calculating the step cost of each point. $G_{(n)}$ is the movement cost, $H_{(n)}$ is the estimated cost. We assume the value

Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of G=1 in each step towards the adjacent point, and the cost of passing the impassable point is 10 G and H which is the estimated cost of moving each point, in this case the start is the coordinates (31, 14) to the point (31, 18) without regard to actual costs. The following is the basic formula for the work performance of the A* algorithm points set to proceeds at Eqs (1), a variety of heuristic functions are also commonly used to increase the accuracy and speed of A* [27] and can work well when combined with manhattan distance [23] for the heuristic function at Eqs. (2) can be written as follows :

$$F_{(n)} - G_{(n)} + H_{(n)} \quad (1)$$

$$DXY = |X1 - Y1| + |X2 - Y2| \quad (2)$$

In this study, we use the Manhattan Distance (D) because of limitation in direction of movement application in vertical and horizontal movement (Straight), to move from one point to another [23]. Manhattan distance is also known as "city block distance" which is the number of blocks of points from a place all attributes. This can be proven as follows For two data points x and y in d-space dimensions [26]. Eqs. (2) is the equation for calculation in computes the absolute difference of two objects between coordinates. From Figure 5, it can be seen that the direction of the road traversed by the NPC which is at point 1 of the coordinates (31, 14) to the search point is the coordinates (31, 18) where the calculation basis is obtained from the sample values obtained as shown in the next table 1.

Figure 5. Result node 1 to finish node on method 1

Table 1. Method 1 Scoring of Adjacent Points

Scoring Between Nodes				
28, 13 G=13 H=8 F=21	29, 13 G=12 H=7 F=19	30, 13 G=11 H=8 F=19	31, 13 G=1 H=9 F=10	32, 13 G=11 H=10 F=21
28, 14 G=3 H=7 F=10	29, 14 G=2 H=6 F=8	30, 14 G=1 H=7 F=8	31, 14 (Node 1, Start Position)	32, 14 G=10 H=9 F=19
28, 15 G=4 H=6 F=10	29, 15 G=3 H=5 F=8	30, 15 G=11 H=4 F=15	31, 15 G=10 H=3 F=13	32, 15 G=20 H=4 F=24
28, 16 G=14 H=5 F=18	29, 16 G=4 H=4 F=8	30, 16 G=5 H=3 F=8	31, 16 G=6 H=2 F=8	32, 16 G=7 H=3 F=10
28, 17 G=15 H=4 F=19	29, 17 G=5 H=3 F=8	30, 17 G=6 H=2 F=8	31, 17 G=7 H=1 F=8	32, 17 G=8 H=2 F=10
28, 18 G=7 H=3 F=10	29, 18 G=6 H=2 F=8	30, 18 G=7 H=1 F=8	31, 18 G=8 H=0 F=8	32, 18 G=9 H=1 F=10
28, 19 G=8 H=4 F=12	29, 19 G=7 H=3 F=10	30, 19 G=8 H=2 F=10	31, 19 G=9 H=1 F=10	32, 19 G=10 H=2 F=12

2.5. D* Lite Algorithm Mechamism

In On the mechanism of the D* light method, g(s) and rhs(s) is the estimated minimum distance cost requirement from the initial node. The search orientation of the D*lite algorithm is opposite to the A* algorithm, the process of calculating the value of g(s) is recalculated continuously while expanding the eight adjacent edges around the node. Update the value of g(s) and select g(s) with the lowest value [24]. To calculate rhs(s) from the starting point g, use the following formula:

$$rhs(s) = \begin{cases} 0, s = s_{goal} \\ min_{s', pred(s)}(g(s') + c((s', s)), others \end{cases} \quad (3)$$

Eqs. (3) Where c(s', s) is the edge of nodes s ' to s, It is normally represented by 1. S'∈pred(s) represents the set of predecessors of vertex s ∈ S. When g(s)=rhs(s), consistent nodes; alternatively, the graph is not uniform. Among them, g(s)>rhs(s) is called local over-consistency, and g(s)<rhs(s) is called local under-conformity. When evaluating the estimated value of grid points, D* Lite introduces the key(s) value for comparison when estimating the value of grid points, where key(s) comprises two values key(s)=[PrimaryKey(s) ; SecondaryKey(s)], which satisfy the following formulas:

$$PrimaryKey(s) = \min(g(s), rhs(s)) + h(s) \quad (4)$$

$$SecondaryKey(s) = \min(g(s), rhs(s)) \quad (5)$$

Eqs. (4), (5) PrimaryKey and SecondaryKey is a parameter for the priority order structure. The key-value size is used as the expansion priority, which determines the order in which sequence nodes expand. Initially, compare the size of the PrimaryKey value, then choose the grid with the smallest SecondaryKey value, and if the PrimaryKey values are equal, compare the size of the SecondaryKey value. The h(s) heuristic function is the same as the heuristic function in the Algorithm A*, which reflects the estimated value between the current node and the starting point and continues to advance the object towards the destination point. Without calculating actual costs, the initial and final locations in this situation are (31, 14) and (31, 18). Due to the limited application of the direction of movement in vertical and horizontal (Straight) motion, we continue to use Manhattan Distance (D) to travel from one point to another in this approach. At the same point from figure 6, it can be seen that the direction of the road traversed by the NPC which is at point 1 of the coordinates (31, 14) to the search point is the coordinates (31, 18) where the calculation basis is obtained from the sample values obtained as shown in the next table 2.

Figure 6. Result node 1 to finish node on method 2

Table 2. Method 2 Scoring of Adjacent Points

Scoring Between Nodes				
28, 13 g=∞ rhs=∞	29, 13 g=∞ rhs=∞	30, 13 g=∞ rhs=∞	31, 13 g=9 rhs=9	32, 13 g=∞ rhs=∞
28, 14 g=7 rhs=7	29, 14 g=6 rhs=6	30, 14 g=7 rhs=7	31, 14 (Start) g=8, rhs=8	32, 14 g=∞ rhs=∞
28, 15 g=6 rhs=6	29, 15 g=5 rhs=5	30, 15 g=∞ rhs=∞	31, 15 g=∞ rhs=∞	32, 15 g=∞ rhs=∞
28, 16 g=∞ rhs=∞	29, 16 g=4 rhs=4	30, 16 g=3 rhs=3	31, 16 g=2 rhs=2	32, 16 g=3 rhs=3
28, 17 g=∞ rhs=∞	29, 17 g=3 rhs=3	30, 17 g=2 rhs=2	31, 17 g=1 rhs=1	32, 17 g=2 rhs=2
28, 18 g=3 rhs=3	29, 18 g=2 rhs=2	30, 18 g=1 rhs=1	31, 18 (Finish) g=0, rhs=0	32, 18 g=1 rhs=1
28, 19 g=4 rhs=4	29, 19 g=5 rhs=5	30, 19 g=6 rhs=6	31, 19 g=1 rhs=1	32, 19 g=2 rhs=2

3. RESULT AND DISCUSSION

The results of the pathfinding experiment in the test aimed to determine the efficiency of the time required to perform the search process. The comparison was based on the A* algorithm (Method 1) and the D* Lite algorithm (Method 2). Technically, the writer built 5 points and tested all parts of the map on an NPC with pathfinding capabilities based on methods one and two as a comparison. The data to be processed was a time in the form of mm: ss,000 (minute: second, millisecond) obtained from the search results for the route of the point to be searched. These values were compared to determine the efficiency of the time required to perform the search process.

3.1. Profiler Analysis Result

Using data analysis techniques, specifically the results of the time required to reach the target point. The artificial intelligence capability of the pathfinding method will be embedded within the NPC, which will conduct a path search to the endpoint located at the coordinates (31, 18). Using a comparison of the A* and D* Lite algorithms, the experiment results of the study to determine the required level of time-efficiency in the mechanism's process are observed. JS-Profiler, a tool used to perform execution analysis of the game's script, was employed to retrieve the data. Figure 7 and Figure 8

represent, respectively, the outcomes of the analysis at node 1 using the two methods where "Map's get() method" description is t_s, and "access object properly" is t_m and t_{total} is t_s+t_m. The A* algorithm revealed in Figure 7 that the execution time of the method on the map was 202 milliseconds (ms), the time to reach the target was 2.528 ms, and the time to move the NPC to the target node was 2.528 ms, for a total of 2.730 ms at node 1. Using D* Lite Algorithm, Figure 8 demonstrates that the execution method required 53 milliseconds and the time to reach the target was 2.528 milliseconds.

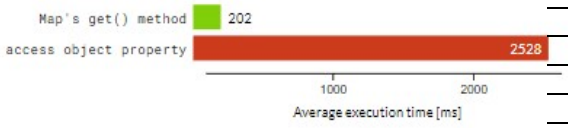

Figure 7. The A algorithm's execution time at node 1*

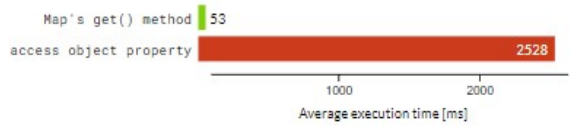

Figure 8. The D algorithm's execution time at node 1*

3.2. Data Comparative Results

The results of the pathfinding experiment in the test aimed to determine the efficiency of the time required to perform the search procedure. In this instance, the findings acquired from the five points and positions of the NPCs are displayed in Figure 3 before obtaining the comparison results using the A* algorithm method in table 3 there are 5 search points (n) with coordinates (x ; y), a summary of the results using the A* algorithm where the value of t_m = is the execution time of searching for the target point, then the value of t_s = is the travel time from the starting point to the target point, and t_{total} value = is the total time t_m + t_s. Next the D* Lite algorithm in table 4 in the table there are 5 search points (n) with coordinates (x ; y), a summary of the results using the D* Lite algorithm where the value of t_m = is the execution time of searching for the target point, then the value of t_s = is the travel time from the starting point to the target point, and t_{total} value = is the total time t_m + t_s.

Table 3. A time results were obtained from 5 nodes*

Node (n)	Coord. (x ; y)	A* (ss.ms)		
		t_m	t_s	t_{total}
1.	31 ; 14	00.202	02.528	02.730
2.	21 ; 05	00.737	08.245	08.982
3.	11 ; 05	01.381	20.682	22.063
4.	00 ; 00	02.202	26.722	28.924
5.	21 ; 16	00.709	08.213	08.922
Total		05.231	66.390	71.621

Table 4. D Lite time result obtained from 5 nodes*

Node (n)	Coord. (x ; y)	D* Lite (ss.ms)		
		t_m	t_s	t_{total}
1.	31 ; 14	00.053	02.528	02.581
2.	21 ; 05	00.198	08.245	08.443
3.	11 ; 05	00.443	20.682	21.125
4.	00 ; 00	00.961	26.722	27.683
5.	21 ; 16	00.192	08.213	08.405
Total		01.847	66.390	68.237

Node 1 was the NPC closest to the target point, whose barrier level consisted solely of other NPC vehicles, with several points being terrains. Point 4 was the most distant, and the process took a long time due to the long distance and the relatively large amount of terrain. The t_{total} value was determined by the following calculations:

$$t_{total} = t_m + t_s \quad (6)$$

Eqs. (6) is the calculation from the starting point to the target point. t_m is the time required to calculate road costs to the target point, and ts is the time required after calculating from the starting point to the target point. The t_s value obtained using two method is demonstrated in figure 7 and 8, the time obtained from the time required to perform calculations as a projection and performance evaluation methods compared and illustrated on the next figure 9. while the t_m value of a is the same where the travel time from the starting point to the target point value had the same cost. The t_{total} value obtained using two method is demonstrated in figure 7 and 8, the time obtained from the time required to perform calculations as a projection and performance evaluation methods compared and illustrated on the next figure 10 as follows:

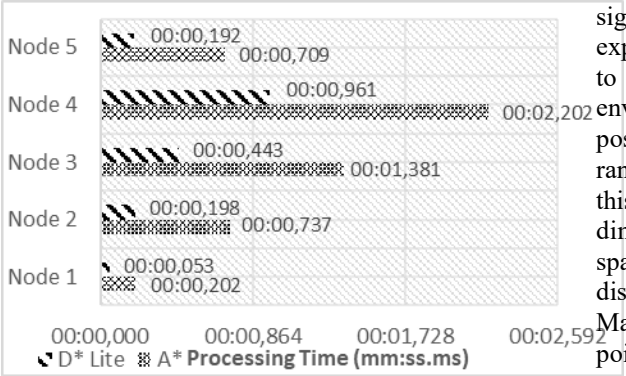

Figure 9. Graph of comparison of the search function execution time (t_s) in the map of the two methods

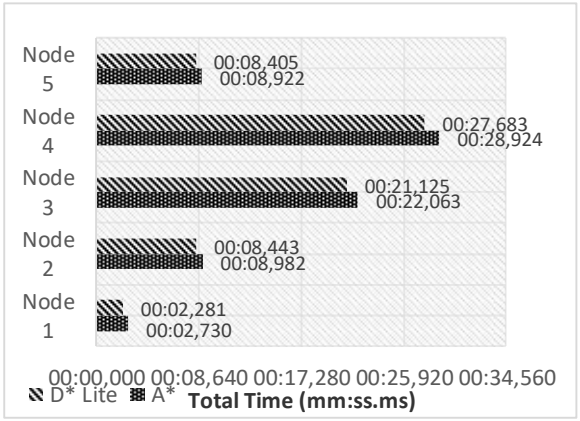

Figure 10. Comparison graph of the total completion time (t_{total}) for each point from the two methods

According to the statistical data obtained from the 5 points in the table 3 and table 4 using the A* alghorithm and D* Lite algorithm methods, the t_{total} value is 71.621 seconds which is part of the t_m value 05.231 seconds + t_s value of 66.390 seconds in the first method and 68.237 seconds with details of the t_m value of 01.847 seconds + the t_s value of 66.390 seconds in the second method. The same result is obtained at t_s value is 66.390 seconds, this is the step cost time by default from the game. while the t_m value has a significant comparison with a difference in comparison of 05.231 seconds and 01.847 seconds for all tested nodes.

4. CONCLUSION

The main focus of this research is time needed to find the target point. According to the total time acquired, the experiment at 5 points with the Pathfinding mechanism using the D* Lite algorithm finds the nearest route faster with a total processing time of 00:01.847 than the A* method with a total processing time of 00:05.231 that value was very significant result. based on the source code experience, might produce different result if testing to the other study. different of features, method, environtment, map, size, distributing obstacles possible ways and then generate the data using random start and goal positions. At the same time, this paper critically analyzes these algorithms in dimensions that include computational time and space efficiency as well as advantages and disadvantages in implementation using the RPG Maker MV application. Path search method at node points in the game. The contribution of this research provides benefits to path finding problems which are commonly used by NPCs for technological games in the future, especially when using the A* algorithm and D* Lite algorithm in game technology. However, the pathfinding method is not only for games, but can also be implemented in other fields.

REFERENCES:

[1]. Hammedi S., Essalmi F., Jemni M., & Qaffas A. A. An investigation of AI in games: educational intelligent games vs non-educational games. Proc. 2020 International Organization of Knowledge and Advanced Technologies (OCTA). 2020; pp. 1-4. doi: 10.1109/OCTA49274.2020.9151738.

[2]. Yakan S. A. Analysis Of Development Of Artificial Intelligence Game Industry. International Journal of Cyber and IT Service Management (IJCITSM). 2022; Vol. 2; No. 2; pp. 111-116. doi:10.34306/ijcitsm.v2i2.100

[3]. Kühl N., Goutier M., Hirt R., & Satzger G. Machine Learning in Artificial Intelligence: Towards a Common Understanding. Proc. of the 52nd Hawaii International Conference on System Sciences. 2019; pp. 5236 – 5245.

[4]. Ranjitha M., Nathan K. & Joseph L. Artificial Intelligence Algorithms and Techniques in the computation of Player-Adaptive Games. Proc. Third National Conference on Computational Intelligence (NCCI 2019) J. Phys.: Conf. Ser. 2019; Vol. 1427; Article id. 012006. doi: 10.1088/1742-6596/1427/1/012006

[5]. Safadi F., Fonteneau R., & Ernst D. Artificial Intelligence in Video Games: Towards a Unified Framework. International Journal of Computer Games Technology. 2015; Vol. 2015; Article ID 271296. doi: 10.1155/2015/271296

[6]. Iskandar U. A. S., Diah N. M., & Ismail M. Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games. Proc. 2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS). 2020; pp.74-80. doi: 10.1109/I2CACIS49202.2020.9140177.

[7]. Huang H. Intelligent Pathfinding Algorithm in Web Games. Proc. CSIA 2020: Cyber Security Intelligence and Analytics. 2020; pp 250–257. doi: 10.1007/978-3-030-43306-2_36

[8]. Pardede S. L., Athallah F. R., Huda Y. N., & Zain F. D. A Review of Pathfinding in Game Development. Journal of Computer Engineering. Progress, Application and Technology. 2022; Vol. 1; No. 1; pp. 47-56. doi: https://doi.org/10.25124/cepat.v1i01.4863

[9]. Sazaki Y., Primanita A., & Syahroyni M. Pathfinding car racing game using dynamic pathfinding algorithm and algorithm A*. Proc. 2017 3rd International Conference on Wireless and Telematics (ICWT). 2017; pp. 164-169. doi: 10.1109/ICWT.2017.8284160.

[10]. Foead D., Ghifari A., Kusuma M. B., Hanafiah N., & Gunawan E. A Systematic Literature Review of A* Pathfinding. Procedia Computer Science. 2021; Vol. 179; pp. 507-514. doi: 10.1016/j.procs.2021.01.034

[11]. Wang H., Lou S., Jing J., Wang Y., Liu W., & Liu T. The EBS-A* algorithm: An improved A* algorithm for path planning. PLoS ONE Journal. 2022; Vol.17; No. 2; pp. e0263841. doi: 10.1371/journal.pone.0263841

[12]. Rachmawati D., & Gustin L. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem. Journal Physics: Conference Series. 2020; Vol. 15; No. 6; pp. 12-18. doi: 10.1088/1742-6596/1566/1/01206

[13]. Candra A., Budiman M. A., & Pohan R. I. Application of A-Star Algorithm on Pathfinding Game. Proc. 5 th International Conference on Computing and Applied Informatics (ICCAI 2020). 2020; pp. 1-6. doi: 10.1088/1742-6596/1898/1/012047

[14]. Suryadibrata A., Young J. C., & Luhulima R. Review of Various A* Pathfinding Implementations in Game Autonomous Agent. International Journal of New Media Technology. 2019; Vol. 6; No. 1; pp.43-49. doi: 10.31937/ijnmt.v6i1.1075

[15]. Raheem F. A., & Ibrahim U. Heuristic D* Algorithm Based on Particle Swarm Optimization for Path Planning of Two-Link Robot Arm in Dynamic Environment. Al-Khwarizmi Engineering Journal. 2019; Vol. 15; No. 2; pp. 108-123. doi: 10.22153/kej.2019.01.004.

[16]. Lawande S. R., Jasmine G., Anbarasi J., and Izhar L. I. A. Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games. A Journal ppl. Sci. 2022; Vol. 12; No. 11; Article Number. 5499. doi: 10.3390/app12115499

[17]. Kadry S., Alferov G., Fedorov V., & Khokhriakova A. Path optimization for D-star algorithm modification. Proc. AIP Conference Proceedings. 2022; pp. 2425. doi: 10.1063/5.0085608

[18]. Iloh P. C. A Comprehensive And Comparative Study Of DFS, BFS, And A* Search Algorithms In A Solving The Maze Transversal Problem. International Journal of Social Sciences and Scientific Studies. 2022; Vol. 2; No. 2; pp. 482-490.

[19]. Kholil M. Troop Movement to Pursue Moving Enemies Using D* Lite Based on Pathfinding Algorithm. Journal of Animation & Games Studies. 2017; Vol.2; No.1; pp. 45 - 68. doi: 10.24821/jags.v2i1.1413

[20]. Ergasheva S., & Kruglov A. Software Development Life Cycle early phases and quality metrics: A Systematic Literature Review. Journal of Physics: Conference Series. 2020; Vol. 1694; Article id. 012007. doi: 10.1088/1742-6596/1694/1/012007

[21]. Malleswari D. N., Kumar M. P., sathvika D., & Kumar B. S. A Study on SDLC For Water Fall and Agile. International Journal of Engineering & Technology. 2018; Vol. 7; no. 2; pp. 10-13. doi: 10.14419/ijet.v7i2.32.13516

[22]. Rafiq A., Kadir T. A. S., & Ihsan S. N., Pathfinding Algorithms in Game Development. Proc. The 6th International Conference on Software Engineering & Computer Systems - IOP Conf. Series: Materials Science and Engineering. 2020; Vol. 769; Article id. 012021. doi:10.1088/1757-899X/769/1/012021

[23]. Sharma S. K, & Kumar S. Comparative Analysis Of Manhattan And Euclidean Distance Metrics Using A* Algorithm. Journal of Research in Engineering and Applied Sciences. 2016; Vol. 01; No. 04; pp.196-198. doi: 10.46565/jreas.2016.v01i04.007

[24]. Xie K., Qiang J., & Yang H. Research and Optimization of D-Start Lite Algorithm in Track Planning. IEEE Access. 2020; vol. 8; pp. 161920-161928. doi: 10.1109/ACCESS.2020.3021073.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[25]. Sarbini R. N., Ahmad I., Bura R. O., & Simbolon L., Design Of Game-Based Learning Defend The Country Using Role Playing Games Mechanism. Journal of Theoretical and Applied Information Technology. 2023; 101(13).

[26]. Suwanda R., Syahputra Z., & Zamzami E. M. Analysis of Euclidean Distance and Manhattan Distance in the K-Means Algorithm for Variations Number of Centroid K. J. Phys.: Conf. Ser. 1566 012058, 2020. DOI 10.1088/1742-6596/1566/1/012058.

[27]. Liu D. Research of the Path Finding Algorithm A* in Video Games. Highlights in Science, Engineering and Technology. 2023; vol. 39; pp. 763–768. doi: doi.org/10.54097/hset.v39i.6642