JURNAL RISET INFORMATIKA **P-ISSN: 2656-1743 |E-ISSN: 2656-1735**
Vol. 4, No. 3. June 2022 DOI: https://doi.org/10.34288/jri.v4i3.384
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

COMPARATIVE ANALYSIS OF PATHFINDING ARTIFICIAL INTELLIGENCE USING DIJKSTRA AND A* ALGORITHMS BASED ON RPG MAKER MV

Riska Nurtantyo Sarbini-1*), Irdam Ahmad-2, Romie Oktovianus Bura-3, Luhut Simbolon-4

1,2,3,4Department of Defence Science, Universitas Pertahanan
https://www.idu.ac.id/
riskanurtantyosarbini@gmail.com-1, irdam.ahmad@idu.ac.id-2, romiebura@idu.ac.id-3, lsimbolon427@gmail.com-4

(*) Corresponding Author

Abstract
In most games, an artificial pathfinding intelligence is required for traversing the fastest discovery. It is essential for many video games, particularly Role Playing Games (RPGs). The algorithm pathfindings implemented in this game are A* and Dijkstra Algorithms. This study aims to test an artificial intelligence system for discovering routes using the A* and Dijkstra algorithms based on RPG Maker MV. The result showed that from the time obtained, in the experiment on eight nodes using the Pathfinding mechanism of A* algorithm has faster result in discovering the nearest route with the time 08:15:23 with format (mm:ss: ms) whereas Dijkstra Algorithm has a 34:47:43 time result. The time record needed represents the distance between the search nodes. It indicates that the multiple weighting in the impassable nodes caused the cost calculation process becomes faster and more efficient.

Keywords: Artificial Intelligence; Pathfinding; Dijkstra Algorithm; A* algorithm

Abstrak
Dalam sebagian besar game, kecerdasan buatan fungsi penemu jalan dibutuhkan untuk menemukan tercepat untuk dilalui hal tersebut penting untuk banyak permainan komputer, khususnya permainan Role Playing Game (RPG). Algoritma pathfinding yang diimplementasikan pada game ini adalah algoritma A dan algoritma Dijkstra. Tujuan dari penelitian ini adalah untuk menguji coba sistem kecerdasan buatan untuk melakukan pencarian rute menggunakan algoritma A* dan algoritma dijkstra berbasis RPG Maker MV. Hasil penelitian Dari waktu yang didapatkan, pada percobaan pada 8 titik dengan mekanisme Pathfinding menggunakan algoritma A* lebih cepat dalam menemukan rute terdekat dengan catatan waktu 08:15:23 format (mm:dd:md) sedangkan menggunakan algoritma Dijkstra 34:47:43, Hal tersebut dikarenakan pemberian bobot berlipat pada titik yang tidak dapat dilalui hal tersebut menyebabkan proses perhitungan biaya jalan menjadi lebih cepat dan efisien. Catatan waktu yang dibutuhkan menunjukkan jarak antar titik pencarian.*

*Kata kunci: Kecerdasan Buatan; Pathfinding; Algoritma Djikstra; Algoritma A**

INTRODUCTION

When artificial intelligence gets into various fields, especially game applications, it presents exciting user experiences (Zhao, 2020). Artificial Intelligence (AI) is used in games to provide more exciting and interactive experiences (Hammedi, Essalmi, Jemni, & Qaffas, 2020). Through the intelligent technology significant improvement, artificial intelligence (AI) has been the core technology for improving the capability in playing a game, and also as the principal value of the game promotion that can give more deep experience in playing game (Tang, Wang, Sima, & Zhang, 2020). AI is a game's leading component and needs to be carefully developed and adjusted regularly. The role affects toward capacity and memory used in a game. AI is an essential component that often impacts the success or failure of a game.

Artificial Intelligence (AI) needed for Pathfinding is assumed to be important in computer games, particularly in Role Playing games. It has been the main research area in video games for decades (Iskandar, Diah, & Ismail, 2020). Usually, it is used as the core of the artificial intelligence movement system in computer games. Various AI fun techniques include the search function, decision making, intelligent narrative technology, and character intelligence (Hammedi et al., 2020). In this situation, the algorithm pathfindings commonly implemented in a game are Dijkstra and A* Algorithms.

Dijkstra algorithm is often used to solve the pathfinding problem using the principle of determining the first node to the next that keeps connecting until the target node. The basic of this algorithm is based on the bandwidth allocation of nodes (Waleed, Faizan, Iqbal, & Anis, 2017). This algorithm is used to discover the shortest way based on the most negligible weight starting from the departure node to the others. For example, the building and monument as the point and the road as the lines, so the Dijkstra will calculate entire lines by the most negligible weight from the greedy algorithm. It includes the route finder algorithm used to solve the problem of the shortest way in one node source that has no negative side cost and produces the quickest way from the tree. This algorithm is often used in routing processes (Wahyuningsih & Syahreza, 2018).

The Dijkstra algorithm works by finding the most minimal weight of a weighted graph, the shortest distance will be obtained from two or more points of a graph, and the total value obtained is the least. For example, G is a directional graph labeled with points V(G) = {v1,v2,...,vn} and the shortest path searched is from v1 to vn. The Dijkstra algorithm starts from point v1. In its iteration, the algorithm will look for a single point whose weight amount is from the smallest point 1. The selected points are separated and are not noticed again in subsequent iterations (Maria, Hanna, & Yosefina, 2022).

The research (Harahap & Khairina, 2017) implemented the Dijkstra algorithm based on the shortest trajectory operating from the initial or source node to the destination node. With the node having a predetermined distance value, the fastest Path to travel to the target node is determined, as seen in figure 1 below.

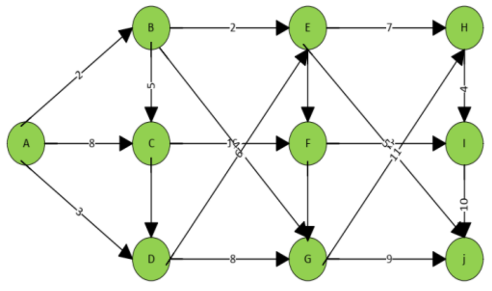

Figure 1. Graph Node Djikstra Aghorithm.

Like the Dijkstra algorithm, the A* algorithm is another pathfinding method used in this game to discover the shortest way to prevent static or dynamic obstacles (Sazaki, Primanita, & Syahroyni, 2018). A* algorithm is often used for heuristics finding of an optimal path on the track. "Heuristics prediction" h(x) provides the best route prediction through the knot. It visits the nodes in this order of heuristic estimate (Rachmawati & Gustin, 2020). To improve their pathfinding ability, some researchers used the A* algorithm in a Real-Time Strategy Game (Chen, Shih, & Chen, 2012).

The A* (A Star) algorithm in this study was used to determine the shortest route that can be traveled. The Formula that used in the A* (A Star) algorithm is as follows: F(n) = G(n) + H(n) Where, F(n) = cost required to move G(n) = costs traveled from the origin node H(n) = the approximate value from the current node to the destination node (Hu, Gen Wan, & Yu, 2012).

The research conducted about the Implementation of the A*(A Star) Algorithm in determining the shortest route that NPCs can pass in games that have succeeded in finding the shortest route by choosing a course based on the lowest F Cost. A case-based reasoning method in the A* algorithm process in multi-task Pathfinding saves path costs before executing node searches (Li, Su, & He, 2012). The Calculating Cost Path can be seen in figure 2 as follows :

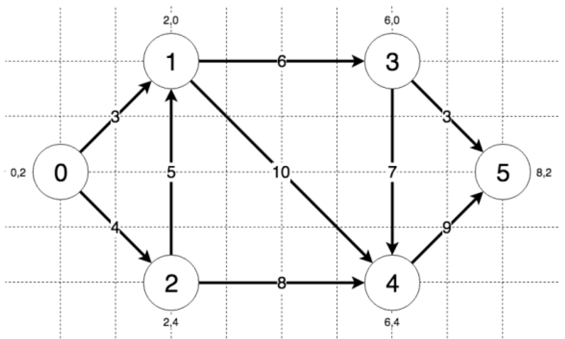

Figure 2. Calculating Cost Path A* Algorithm.

In this research, an improvement approach of Artificial Intelligence (AI) is proposed to analyze the Pathfinding using A* and Dijkstra algorithms. It aims to test the time effectiveness of artificial intelligence for Pathfinding in a Role-Playing game using these two algorithms.

RESEARCH METHODS

System Development Life Cycle (SDLC) is used in this system improvement. It is the process of

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v4i3.384
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

designing, developing, and testing high-quality software. SDLC aims to provide a structured flow in assisting high-quality software production, fulfill the user expectation, acquire the software life cycle model, and compare its performance (Saravanan, Jha, Sabharwal, & Narayan, 2020). One of the most critical phases of SDLC is the quality assurance or testing phase (Sinha & Das, 2021).

Types of research

This research is a kind of software engineering through the SDLC phases of the waterfall model. The phases are designing, analyzing, testing, implementing, and analyzing observation data results (quantitative).

Time and Place of Research

This research was done in March 2022, and the software construction used computer laboratory PC desktop on the application was ready. Next are the installation steps on the smartphone.

Research Target / Subject

The research target is to construct a stable application on the smartphone and then continue to test the location node search using the pathfinding mechanism of A* and Dijkstra algorithms.

Procedure

Research procedures started from the application making and continue to analyze the nearest route search data that related to the duration needed to discover the nearest route, and the explanation can be seen as follows:

Software making is the most critical project management. The steps that get through the game-making involve designing, analyzing, designing, implementing, and testing. Besides, the software process model is essential to get standards, especially in digital software making. Creating a software model for the entire life cycle of software development (SDLC) by the Waterfall model must be efficient for the software team to quickly get productivity (output) of 80% through reduced software development. Ultimately, it can increase the software process performance (Iqbal & Rizwan, 2009). The software engineering model used the SDLC with the waterfall model (Trivedi & Sharma, 2013). It can be seen in figure 3 as follows:

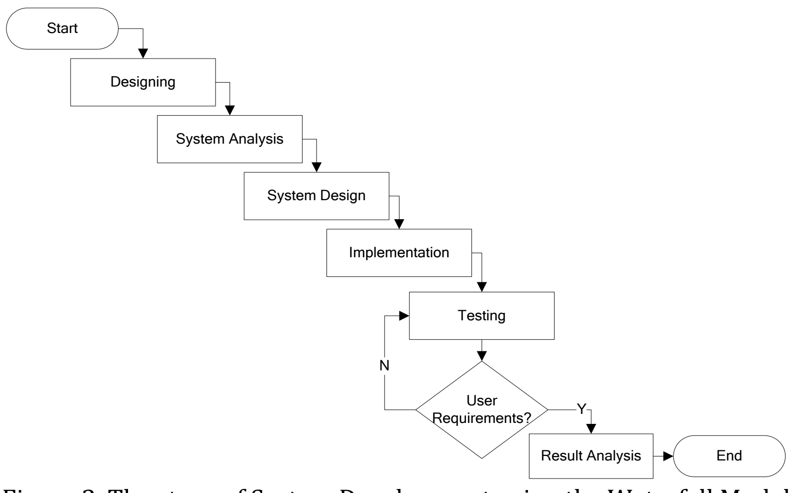

Figure 3. The steps of System Development using the Waterfall Model

Explanation from figure 3 is as follows:

1. Designing

In this step, system specifications will be considered and constructed based on the user requirements. There are some tasks classification that must be done, as follows:

a. It collected information requirements related to the constructed system, such as game observation.
b. Determining the program's objectives by focusing on the specific problems to be solved, that is, designing a stable game.
c. We are determining the components inside the game related to the artificial intelligence ability using A* and Dijkstra algorithms.

2. System Analysis

The steps involve:

a. We are identifying the problems of system description and running system explanation.
b. Identifying and process analyzing the requirements of the Role-Playing mechanism.
c. Identifying the need analysis, such as doing a system requirements checklist for functional and non-functional needs towards implementing artificial game intelligence.

d. It identifies and analyzes alternative solutions to the system that will be constructed.

3. System Design

In this step, there are some activities as follows:

a. Application architecture design or sitemap.
b. Input and output designs involve identification and layout making.
c. Process design requires process identification and system process scenario, then continued in modeling.
d. Process design involves process identification and system process scenario, then modeled using DFD (Data Flow Diagram).
e. Database design involves the identification of table, entity, and ERD making (Entity Relation Diagram).
f. Interface design involves interface identification and creating a layout used.
g. Formulate the Implementation of RPG through a pathfinding mechanism.

4. Implementation

The database is used to save the input and output data from the game system. The database from this application uses RPG Maker MV. The database design as the knowledge base is the storage basic concept and game storyline. Database design starts from creating the tables, determining the keys in each table, and relating one table to another using the RPG Maker MV program database application program. This Game creation includes the database system and editor map.

5. Testing

The testing steps of this research are:

a. Verification: observing the suitability between design and result.
b. Validation: testing the game function suitability of RPG and the correlation between artificial intelligence utilization with pathfinding mechanism.
c. The testing is intended to acquire the performance of the pathfinding mechanism that has A* and Dijkstra algorithms function and observe the results.

Data, Instruments, and Data Collection Techniques

Processed data are HH:MM: SS, obtained by the node route search result. There will be discovering and comparing processes using A* and Dijkstra algorithms. These aim to observe the level of time efficiency that is required in the discovery process. Technically, the comparison is through the author creating eight nodes to be tested in an entire map on a mini-game of transportation and also the NPC that has pathfinding abilities based on A* and Dijkstra algorithms in the same game but different algorithms. These eight nodes and NPC positions can be seen in Picture 4 as follows:

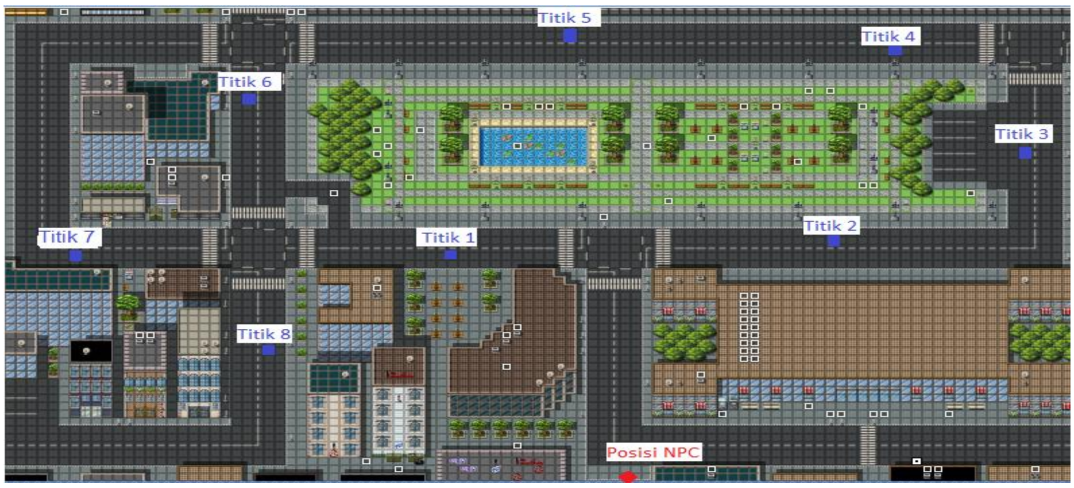

Figure 4. Analyze eight-node targets on the map.

Figure 2 shows that the red sentence is the first node of NPC, and the blue one is eight nodes that will be tested on time required.

Data analysis technique

Quantitative research is the priority analysis focusing on numbers, from collecting data, interpreting the data obtained and presenting the result (Arikunto, 2006). In this research, the author used a quantitative technique to analyze the data through the time data required to reach the point target. Sixteen data will be obtained based on the distribution of 8 nodes overall map using the Dijkstra algorithm and eight nodes using the A* algorithm.

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v4i3.384
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

RESULTS AND DISCUSSION

The ability of artificial intelligence using the pathfinding method will be located on NPC in discovering Path, as the first and last nodes where NPC located on the coordinate of (22,51).

Then, it continues to the initialization process of value discovery from the start node to the target (open_nodes, total_cost).

Next, adding node lists to the Open list is expected to be traversed. It can be seen in Table 1 as follows.

Table 1. List of Open Nodes Traversed (Coordinate Nodes)

List of Open Nodes Traversed (Coordinate Nodes)								
22, 22	23, 22	24, 22	25, 22	26, 22	27, 22	28, 22	29, 22	30, 22
22, 23	23, 23	24, 23	25, 23	26, 23	27, 23	28, 23	29, 23	30, 23
22, 24	23, 24	24, 24	25, 24	26, 24	27, 24	28, 24	29, 24	30, 24
22, 25	23, 25	24, 25	25, 25	26, 25	27, 25	28, 25	29, 25	30, 25
22, 26	23, 26	24, 26	25, 26	26, 26	27, 26	28, 26	29, 26	30, 26
22, 27	23, 27	24, 27	25, 27	26, 27	27, 27	28, 27	29, 27	30, 27
22, 28	23, 28	24, 28	25, 28	26, 28	27, 28	28, 28	29, 28	30, 28
22, 29	23, 29	24, 29	25, 29	26, 29	27, 29	28, 29	29, 29	30, 29
22, 30	23, 30	24, 30	25, 30	26, 30	27, 30	28, 30	29, 30	30, 30
22, 31	23, 31	24, 31	25, 31	26, 31	27, 31	28, 31	29, 31	30, 31
22, 32	23, 32	24, 32	25, 32	26, 32	27, 32	28, 32	29, 32	30, 32
22, 33	23, 33	24, 33	25, 33	26, 33	27, 33	28, 33	29, 33	30, 33
22, 34	23, 34	24, 34	25, 34	26, 34	27, 34	28, 34	29, 34	30, 34
22, 35	23, 35	24, 35	25, 35	26, 35	27, 35	28, 35	29, 35	30, 35
22, 36	23, 36	24, 36	25, 36	26, 36	27, 36	28, 36	29, 36	30, 36
22, 37	23, 37	24, 37	25, 37	26, 37	27, 37	28, 37	29, 37	30, 37
22, 38	23, 38	24, 38	25, 38	26, 38	27, 38	28, 38	29, 38	30, 38
22, 39	23, 39	24, 39	25, 39	26, 39	27, 39	28, 39	29, 39	30, 39
22, 40	23, 40	24, 40	25, 40	26, 40	27, 40	28, 40	29, 40	30, 40
22, 41	23, 41	24, 41	25, 41	26, 41	27, 41	28, 41	29, 41	30, 41
22, 42	23, 42	24, 42	25, 42	26, 42	27, 42	28, 42	29, 42	30, 42
22, 43	23, 43	24, 43	25, 43	26, 43	27, 43	28, 43	29, 43	30, 43
22, 44	23, 44	24, 44	25, 44	26, 44	27, 44	28, 44	29, 44	30, 44
22, 45	23, 45	24, 45	25, 45	26, 45	27, 45	28, 45	29, 45	30, 45
22, 46	23, 46	24, 46	25, 46	26, 46	27, 46	28, 46	29, 46	30, 46
22, 47	23, 47	24, 47	25, 47	26, 47	27, 47	28, 47	29, 47	30, 47
22, 48	23, 48	24, 48	25, 48	26, 48	27, 48	28, 48	29, 48	30, 48
22, 49	23, 49	24, 49	25, 49	26, 49	27, 49	28, 49	29, 49	30, 49
22, 50	23, 50	24, 50	25, 50	26, 50	27, 50	28, 50	29, 50	30, 50
22, 51	23, 51	24, 51	25, 51	26, 51	27, 51	28, 51	29, 51	30, 51

The next process is getting the target nodes and create direction of MoveRoute (while path_x!=src_x|| path_y!=src_y), that is path = (2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4,), where the values are given DOWN = 2, LEFT = 4, RIGHT = 6, UP = 8.

The experiment result of Pathfinding to observe the level of time-efficient needed in the discovery process uses the comparison between A* and Dijkstra algorithms. The analysis results at node 1 using the Djikstra and A* algorithm were shown in Figure 5 and Figure 6, respectively. Time data retrieval was using the JS-Profiler script, which scripts that serves to perform execution analysis of the hand used in the game.

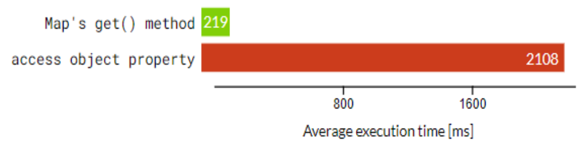

Figure 5. Execution time at node 1 with Dijkstra algorithm

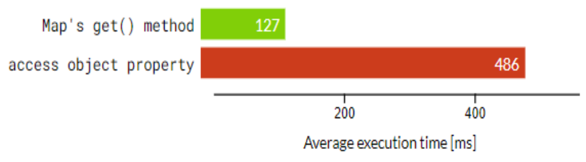

Figure 6. Execution time at node 1 with A* algorithm

Figure 5 exposed that the execution of the method took 219 ms and the time to reach the target was 2108 ms. Thus, the total time needed was 2327 ms at node 1. Meanwhile, Figure 6 showed that the execution method, which took 127 ms and the time to reach the target, was 486 ms accordingly if the total time required was 613 ms at node 1.

Technically, the author will compare by creating eight nodes that will be tested in the entire map of transportation mini-game and the NPC that has the pathfinding ability based on A* and Dijkstra algorithms in a similar game but different algorithms. The eight nodes of NPC position can be seen in Picture 4. The recapitulation result of Table 2 is as follows.

Table 2. Time results were obtained from the eight target nodes using A* and Dijkstra algorithms.

Numb.	Coordinate	Djikstra	A*
1	41, 31	00:23:27	00:06:13
2	76, 29	01:05:46	00:14:08
3	94, 18	05:09:01	00:55:12
4	82, 5	10:00:00	02:23:11
5	52, 3	10:00:00	03:03:00
6	22, 11	04:07:11	00:41:18
7	6, 31	02:03:51	00:17:35
8	24, 43	01:59:07	00:15:26
Total Time (mm:ss:ms)		34:47:43	08:15:23

From Table 2, the testing result of the eight nodes experiment can be concluded that on the NPC using A* and Dijkstra algorithms methods in 4,5 nodes experiment, and the discovery process needs extended time. The result may cause more extensive terrain (obstacle) of a pretty expansive park among these nodes.

The NPC Dijkstra algorithm of the 4,5 nodes experiment is stopped (not continued). It is because of spending a too long time. The experiment process stopped at 10 minutes because it was not efficient and would disturb the game application. It might be caused by the more extensive terrain (obstacle) of a vast park among these nodes. The graphic comparison lines can be seen in figure 7 as follows.

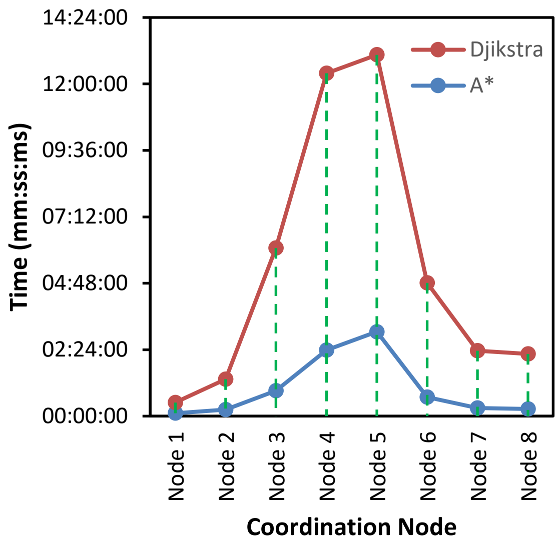

Figure 7. Graphic Comparison Using A* and Dijkstra algorithms.

Nodes 1,2 were the nearest NPC nodes where the obstacle level was only other NPC transportation and some terrain nodes. The A* algorithm NPC needed some seconds to find the target coordinate. A similar case was also found on Dijkstra algorithm NPC. The process nodes of 3,4,5,6 needed a longer time due to the nearest nodes with the city park in the game. Park has many terrains of objects NPC characters or transportation and objects of plants, chairs, pool, and trees.

Meanwhile, using the Dijkstra algorithm took a long time, so the process stopped at 10 minutes. It was because of spending a very long time. 7,8 nodes were not quite close to the park but had long routes. Therefore, the Dijkstra algorithm needed a longer time than the A* algorithm.

CONCLUSIONS AND SUGGESTIONS

Conclusion

According to the time obtained, the experiment of 8 nodes using the A* algorithm Pathfinding mechanism had a faster route in discovering the nearest road, with the time record of 08:15:23 with format (mm:ss: ms), whereas the Dijkstra algorithm with the time record of 34:47:43. The reason was that multiple weighing on the nodes could not be traversed. It caused the road cost calculation process to be faster and more efficient.

The time record needed by NPC did not represent the distance between NPC and the discovery nodes, but it was the route discovery process to the target. It could be seen in the Lag process in a game before finding.

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v4i3.384
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

Suggestion

Technically, the game-based RPG Maker MV application, where the artificial intelligence implemented was based on the author's source code, might produce different results if testing the other author. It needs improvement using other applications, such as unity3D and game maker.

REFERENCES

Arikunto, S. (2006). *Prosedur penelitian suatu pendekatan praktik* (IV). Jakarta: Penerbit PT Rineka Cipta.

Chen, J. H., Shih, T. K., & Chen, J. Y. (2012). To develop the ubiquitous adventure RPG (role play game) game-based learning system. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 2973–2978. https://doi.org/10.1109/ICSMC.2012.6378247

Hammedi, S., Essalmi, F., Jemni, M., & Qaffas, A. A. (2020). An investigation of AI in games: Educational intelligent games vs non-educational games. *Proceedings of 2020 International Multi-Conference on: Organization of Knowledge and Advanced Technologies, OCTA 2020*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/OCTA49274.2020.9151738

Harahap, M. K., & Khairina, N. (2017). Pencarian Jalur Terpendek dengan Algoritma Dijkstra. *SinkrOn*, *2*(2), 18. https://doi.org/10.33395/sinkron.v2i2.61

Hu, J., Gen Wan, W., & Yu, X. (2012). A pathfinding algorithm in real-time strategy game based on Unity3D. *ICALIP 2012 - 2012 International Conference on Audio, Language and Image Processing, Proceedings*, 1159–1162. https://doi.org/10.1109/ICALIP.2012.6376792

Iqbal, M., & Rizwan, M. (2009). Application of 80/20 rule in software engineering Waterfall Model. *2009 International Conference on Information and Communication Technologies, ICICT 2009*, 223–228. https://doi.org/10.1109/ICICT.2009.5267186

Iskandar, U. A. S., Diah, N. M., & Ismail, M. (2020). Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games. *2020 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2020 - Proceedings*, 74–80. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/I2CACIS49202.2020.9140177

Maria, B., Hanna, P., & Yosefina, R. (2022). Penerapan algoritma dijkstra untuk menentukan rute terpendek dari pusat kota surabaya ke tempat bersejarah. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *4*(1), 213–223. https://doi.org/10.47233/jteksis.v4i1.407

Rachmawati, D., & Gustin, L. (2020). Analysis of Dijkstra's Algorithm and A∗ Algorithm in Shortest Path Problem. *Journal of Physics: Conference Series*, *1566*(1), 012061. Institute of Physics Publishing. https://doi.org/10.1088/1742-6596/1566/1/012061

Saravanan, T., Jha, S., Sabharwal, G., & Narayan, S. (2020). Comparative Analysis of Software Life Cycle Models. *Proceedings - IEEE 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2020*, 906–909. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICACCCN51052.2020.9362931

Sazaki, Y., Primanita, A., & Syahroyni, M. (2018). Pathfinding car racing game using dynamic pathfinding algorithm and algorithm A∗. *Proceedings - ICWT 2017: 3rd International Conference on Wireless and Telematics 2017*, *2017-July*, 164–169. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICWT.2017.8284160

Sinha, A., & Das, P. (2021). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. *2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. Kolkata, India: IEEE. https://doi.org/10.1109/IEMENTech53263.2021.9614779

Tang, C., Wang, Z., Sima, X., & Zhang, L. (2020). Research on artificial intelligence algorithm and its application in games. *Proceedings - 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture, AIAM 2020*, 386–389. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/AIAM50918.2020.00085

Trivedi, P., & Sharma, A. (2013). A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation. *Proceedings of the 2013 2nd International Conference on Information Management in the Knowledge Economy : 19-20 December 2013*. Punjab, India: Chitkara University. Retrieved from https://ieeexplore.ieee.org/abstract/document/6915096/authors#authors

Wahyuningsih, D., & Syahreza, E. (2018). Shortest Path Search Futsal Field Location With Dijkstra Algorithm. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *12*(2), 161–170. https://doi.org/10.22146/IJCCS.34513

Waleed, S., Faizan, M., Iqbal, M., & Anis, M. I. (2017). Demonstration of single link failure recovery using Bellman Ford and Dijikstra algorithm in SDN. *ICIEECT 2017 - International Conference on Innovations in Electrical Engineering and Computational Technologies 2017, Proceedings*. https://doi.org/10.1109/ICIEECT.2017.7916533

Zhao, M. (2020). Analysis on the connection between nonplayer character and artificial intelligence. *Proceedings - 2020 International Conference on Intelligent Computing and Human-Computer Interaction, ICHCI 2020*, 105–108. https://doi.org/10.1109/ICHCI51889.2020.00030