INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS

ISSN(print): 2643-9840, ISSN(online): 2643-9875

Volume 05 Issue 09 September 2022

DOI: 10.47191/ijmra/v5-i9-20, Impact Factor: 6.261

Page No. 2445-2453

Comparison of Flow Field and A-Star Algorithm for Pathfinding in Tower Defense Game

Graldy Tirta Kumala[1], Wirawan Istiono[2]

[1,2] Universitas Multimedia Nusantara & Jl. Scientia Boulevard, Curug Sangereng Kelapa dua, Tangerang, Indonesia

ABSTRACT: Tower Defense is a game genre that uses a pathfinding algorithm. Pathfinding is a way to find a path from one point to another; pathfinding itself has many variants with different scenarios. One of these algorithms is the A-Star algorithm, a well-known and commonly used method for game pathfinding. Another algorithm is the Flow Field algorithm, which is an algorithm that is not widely known and has become the topic in this following research. This research will be carried out by comparing the time required by both algorithms to reach the target point from starting point, and time comparison will be carried out in 3 different scenarios. The result of the research is that the Flow Field algorithm reaches the target faster than A-Star Pathfinding in every scenario carried out in the simulation. This research concludes that the Flow Field algorithm can compete with the A-Star algorithm to find paths in the Tower Defense game.

KEYWORDS: Flow field algorithm, A-Star, Pathfinding, Artificial Intelligence, Algorithm Comparison

I. INTRODUCTION

Real-time Strategic (RTS) Tower Defense (TD) is a strategy game with building and managing resources in a game that is moving in real-time [1]. RTS TD aims to protect a zone and eliminate the enemy or enemy zone. One of the challenges in a Tower Defense game is how to move the enemy units. To overcome the challenge, we can use pathfinding. Pathfinding is putting Artificial Intelligence into NPC (Non-Player Character) to find a way to move to the target. A-Star algorithm is one of the most popular pathfinding used [2], [3]. A-Star algorithm is also often used for comparison between algorithms in pathfinding because of its popularity and flexibility. Every Algorithm has its downsides. If a lot of units are involved in the game that runs the Algorithm, if the map is too small, or if the game environment is too dynamic, A-Star Algorithm would not be the best Algorithm to solve pathfinding problems [4], [5].

There are many alternative pathfinding algorithms, one of which is Flow Field pathfinding. Flow Field and A-Star are similar algorithms to each other, and both of them use informed search. According to Emerson [6], Flow Field Algorithm is worth studying because of its capability to solve paths for hundreds of units, detect obstacles, and it's heuristic properties [7]. The capacity to create a path from hundreds of units came from the grid-based Algorithm [8]. The downside of Flow Field Pathfinding is a high amount of resource usage if the map is too big [9].

This research compares the Flow Field algorithm with the A-Star algorithm in a Tower Defense game. This comparison aims to know if Flow Field Pathfinding algorithm could compete against the A-Star Pathfinding algorithm in an RTS TD game. Time to reach the target would be the comparison between the two Algorithms. This research is expected to add to the general insight into the Flow Field Pathfinding algorithm and add interest in making Tower Defense games using algorithms Flow Field Pathfinding.

II. LITERATURE STUDY

Tower Defense is a strategy game where the player stops or slows the enemy with an obstacle in-game environment. Tower Defense is a RTS game focused on managing resources and placing unit/tower to prevent the enemy from reaching a certain zone [10], [11]. From the two definition above, can be concluded that the tower defense game is a game where the player prevents the enemy units from reaching a certain zone with every means possible. The player creativity and skill would be tested on how to prevent the enemy from going into the zone.

Flow Field is derived from the concept of fluid dynamics, namely the study of the movement of water surfaces. Before the Flow Field algorithm was used for video games, this Algorithm is used for water surfaces, this concept can also be identified as

vector fields. Flow Field Algorithm works in a vector plane, where each vector points to the neighboring Node closest to the destination. When a unit passes through a cell, it will know the vectors belonging to that one, so as the name implies, the vectors in the flow field will look like water flows [12]. The Flow Field Pathfinding algorithm has a time complexity of O(n2) notation; this notation is formed because the code checks all nodes on the grid in a nested if. A-Star algorithm uses mathematical equations and uses the lowest value overall as a pathway from the starting point to the endpoint [13]. The calculation of the lowest value is done using the heuristic formula [14] :

$$f(n) = h(n) + g(n).$$

Where the g(n) is a total value from starting point to point n, h(n) is an estimated value from point n to the endpoint, and f(n) is the total value of combined g(n) and h(n). The lowest value of f(n) would be the shortest path from the starting point to the endpoint. The A-Star Pathfinding algorithm has a time complexity of O(bd) or O(E). This is happen because, traveling to all nodes is a possibility for the Algorithm to reach the destination point from the starting point.

III. METHODOLOGY

Comparison between the Flow Field algorithm and A-Star algorithm is made by testing both Algorithms in 3 different scenarios. Each scenario would be tested ten times, and calculate the average time taken for the pathfinding algorithm to reach the endpoint. In this study, the Flow Field Pathfinding algorithm and the A-Star Pathfinding algorithm are the base code. For the A-Star Pathfinding algorithm, the base code is used by Seb Lague [15], while for the Flow Field Pathfinding algorithm, the base code from Turbo Makes Games is used which is then developed to meet the testing criteria to be carried out. System design is done by making flowcharts and map designs.

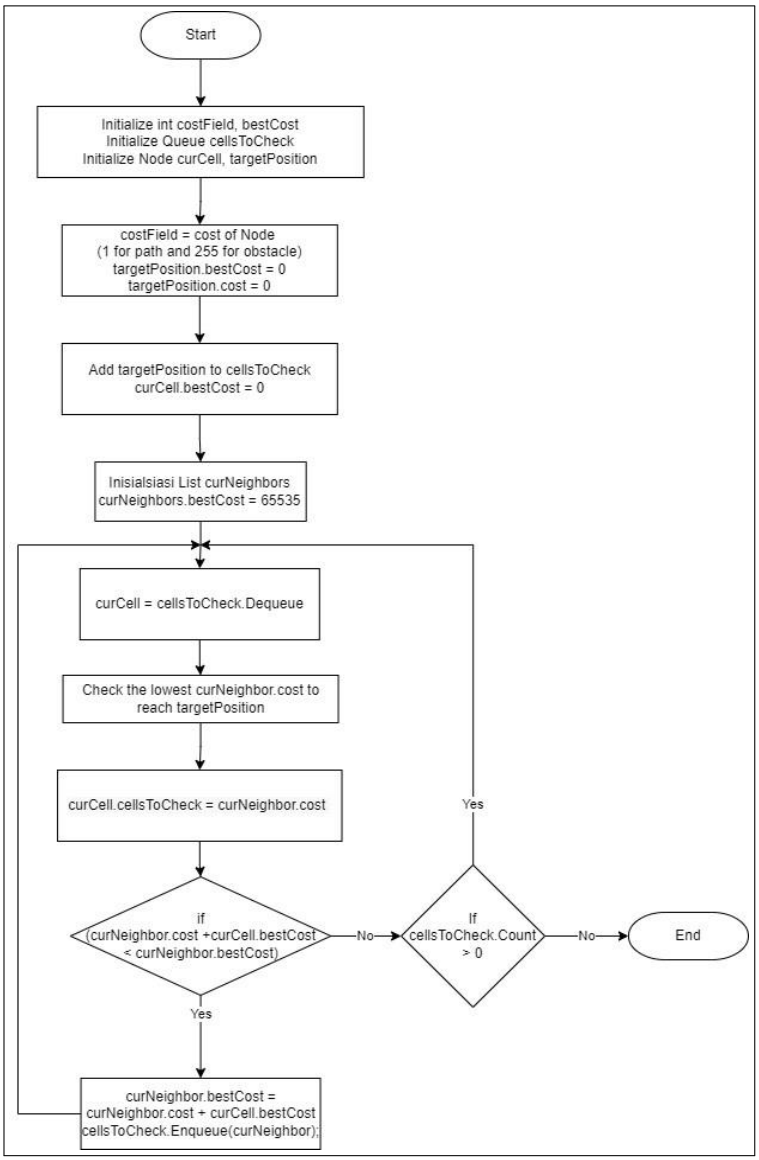


Fig 1. Flow field pathfinding modul

Figure 1 shows the flow for Flow Field Pathfinding algorithm. The first step in this Algorithm is to initialize several things, there are integer initialization costField and bestCost, then Queue cellsToCheck initialization, and Node initialization curCell and targetPosition. After that the costField value of each Node in the grid will be set, the value is 1 if the Node can be used as a path, whereas if the Node cannot be passed, then the value will be 255. The targetPosition node will also be set targetPosition.bestCost = 0 and targetPosition.cost= 0, this value serves to distinguish the Node value between the destination and the path and the initial stage is to find the shortest path to reach the goal. Then add targetPosition to Queue cellsToCheck and fill the value curCell.bestCost = 0 so that the destination node will be processed first by the Algorithm.

Next, initialize the curNeighbors List and fill in the curNeighbors.bestCost value with a very high value (65535) so that when compared to other bestCost (paths) which have lower values, that path will be used. The next thing is to fill in the Node values one by one starting from curCell, then check to find the lowest curNeighbor.cost value to reach the target, by comparing the value of curNeighbor.cost plus curCell.bestCost with the value of curNeighbor.bestCost. If the value is smaller, it will be continued by determining the Node value curNeighbor.bestCost with the value curNeighbor.cost + curCell.bestCost and adding the Node into the Queue which will then loop back to read the Nodes contained in the Queue. If the value is greater, it will check whether there are still Queues left from cellsToCheck, if so, the code will loop when the Algorithm reads cellsToCheck, if the cellsToCheck Queue is empty, this means that all Nodes have been read and have values. At this stage the Flow Field algorithm has been successfully created and the closest path made between the starting point and the end point is through the Node with the lowest total value.

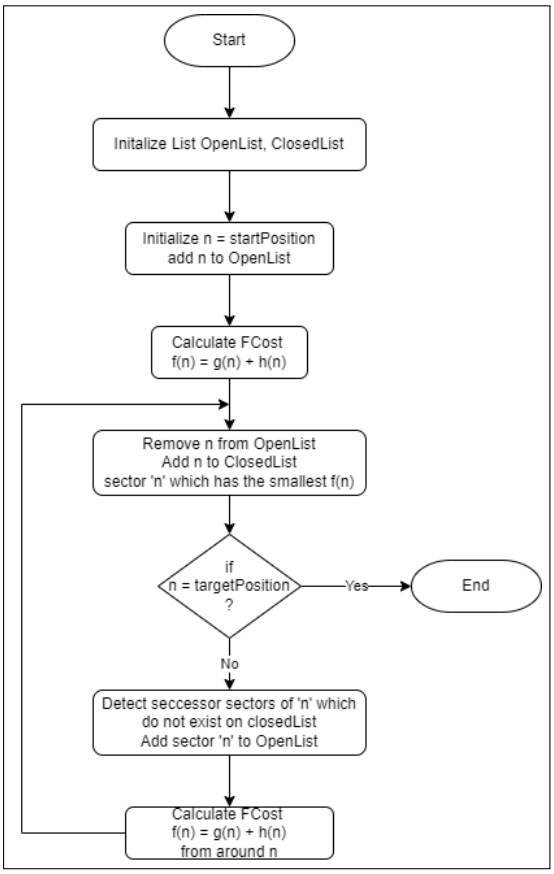


Fig 2. A-Star pathfinding modul

Figure 2 shows the flow of the A* pathfinding algorithm. The A* algorithm works by creating a list named open list and closed list, open list are nodes that would, but not yet calculated. Meanwhile closed list are nodes that has been calculated. After the two list are created, the Algorithm would calculate the f cost with equation of f(n) = g(n) + h(n), where f(n) is the f cost, g(n) would represent as the value from starting position to Node n, and h(n) is the heuristic value, the estimated value needed to reach end Node from Node n. The A* algorithm would search for the lowest value of f(n), then to find out whether n is the target Node, if it is not the target position the Algorithm would keep running to search for n that are not in the closed list, add n to the open list then the Algorithm would calculate the value of f(n) then add it to the closed list, and check the next n. If n is the target Node, then the Algorithm has been run successfully and the path would be created by following the closed list from the starting point to the end point.

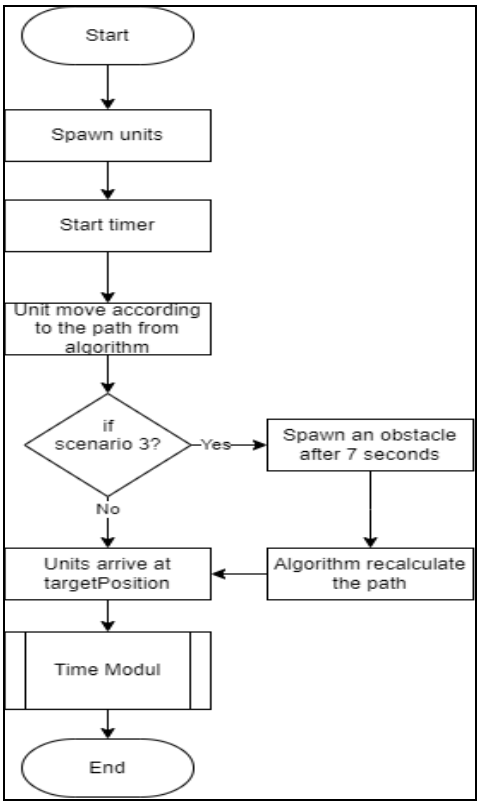


Fig 3. Main modul Scenario flow

Figure 3 is the flow for the main module. When the 'Play' button is clicked, 10 entities will spawn from starting point, and the timer will start. The entity would start to move to the end point based from the path the Algorithm creates. If the process is in scenario 3, after 7 seconds, an obstacle would appear on the map causing the Algorithm to recalculate a new path to avoid the obstacle.

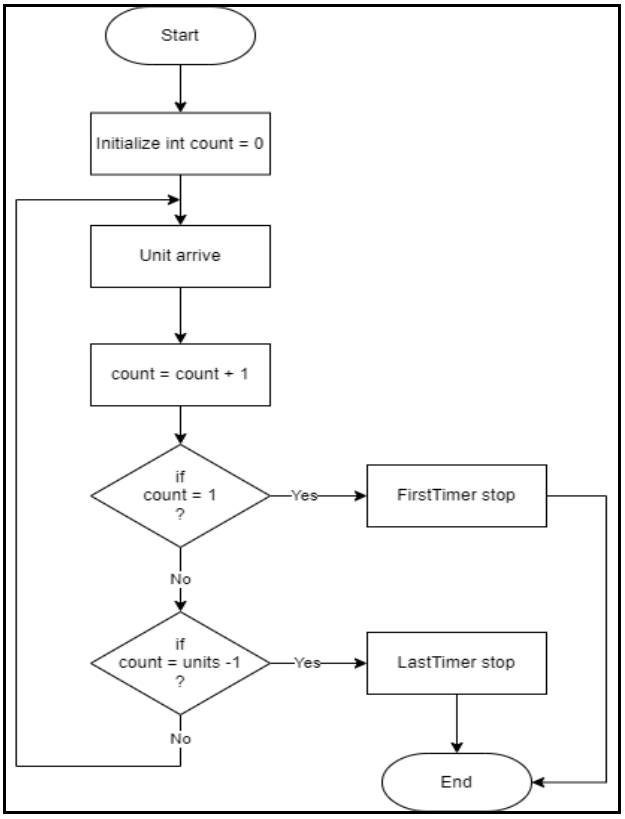


Fig 4. Time modul flowchart

Figure 4 is the flowchart for time modul. Count variable is initialized to detect and calculate unit. When a unit reached the destination, the amount of unit would be checked, if it is the first unit that reaches the destination, the first timer will stop. When the last unit reached the destination, the last timer would stop, recording the time needed to get to the destination from the start.

IV. RESULT

A comparison of the flow field algorithm and the A-Star algorithm in pathfinding is carried out with three mapping scenarios. At the same time, the effectiveness is calculated by measuring the speed of each NPC in achieving the goal. In the first scenario, a simple map looks like in Figure 4, where the path is made simple with a few obstacles.

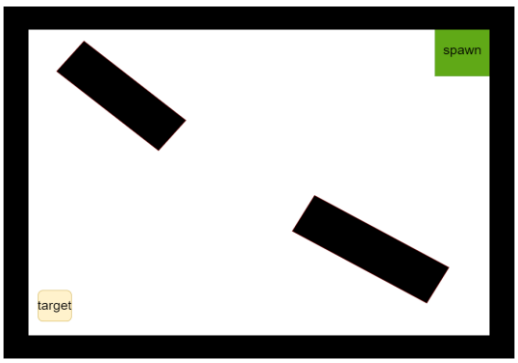


Fig 5. Mapping Path in Scenario 1

Figure 5 is the implementation result of scenario 1. In this scenario, the display will remain the same in the Flow Field Pathfinding algorithm and the A* Pathfinding algorithm. Beside Figure 6 is the implementation result of scenario 2. In this scenario, the display will remain the same in the Flow Field Pathfinding algorithm and the A* Pathfinding algorithm.

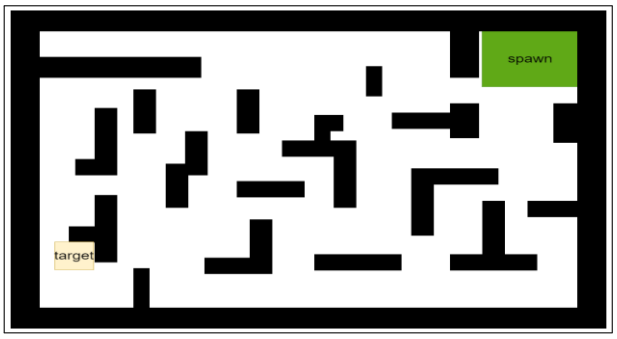
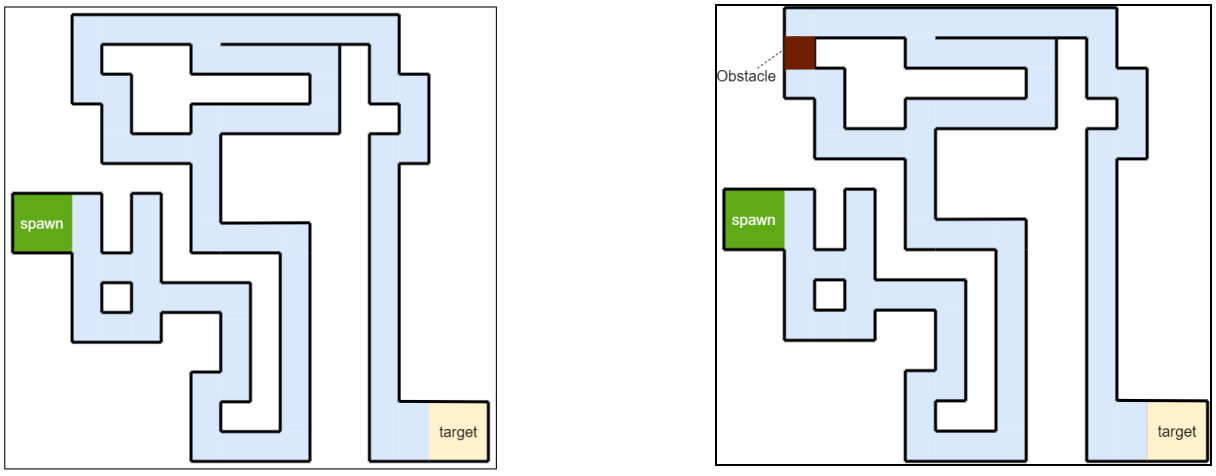


Fig 6. Mapping Path in Scenario 2

In the second scenario, the path to reach the target is more complicated than the first scenario to see the effectiveness of the speed of the A-Star algorithm and the flow field algorithm in the path search. However, in this scenario, the path from where the NPC appears to the target destination on the map does not change.



(a) (b)

Fig 7. Mapping Path in Scenario 3

In the third scenario shown in Figure 7a, the path on the map is made more complicated, and obstacles can suddenly appear to hinder the path search. The appearance of obstacles can be seen in Figure 7b, where obstacles can suddenly appear to hinder the path search. Still, in this comparative test, the appearance of obstacles in each test will be made simultaneously.

Test results were obtained and calculated by using the *timer*form Unity. Application testing is done ten times for each Algorithm and scenario to get the average time taken. The following tables are the results of the trial that has been carried out.

Table 1. Result of Scenario 1 Flow Field Pathfinding

Test number	First Timer	Last Timer	Average
1	3.0099	3.4152	3.2126
2	2.8994	3.3857	3.1426
3	2.9305	3.3216	3.1261
4	2.9462	3.3203	3.1332
5	2.9462	3.3643	3.1553
6	2.9271	3.4018	3.1645
7	2.9572	3.3730	3.1651
8	2.9189	3.675	3.1432
9	3.0095	3.3397	3.1746
10	2.8974	3.3223	3.1099

Table 1 is the result of testing using Flow Field Pathfinding to find the shortest time required to reach the target. The average result of all trials conducted was (3.2126 + 3.1426 + 3.1261 + 3.1332 + 3.1553 + 3.1645 + 3.1651 + 3.1432 + 3.1746 + 3.1099) / 10 = 3.1527.

Table 2. Result of Scenario 1 A* Pathfinding

Test number	First Timer	Last Timer	Average
1	3.5307	3.8710	3.7009
2	3.4046	3.9029	3.6538
3	3.4858	3.9975	3.7417
4	3.4465	3.9424	3.6945
5	3.4361	4.0604	3.7483
6	3.5163	4.0054	3.7609
7	3.4501	4.1039	3.7770
8	3.5495	4.1754	3.8625
9	3.5221	4.0333	3.7777
10	3.4616	4.0996	3.7806

Table 2 shows the result of the testing using A* Pathfinding to find the shortest amount of time required to reach the target. The average result of all trials conducted was (3.7009 + 3.6538 + 3.7417 + 3.6945 + 3.7483 + 3.7609 + 3.7770 + 3.8625 + 3.7777 + 3.7806) / 10 = 3.7498. The result of the comparison average of Table 1 with Table 2 for scenario 1 shows that the average time of the Flow Field (3,1527) is faster than A* (3,7498) and has a difference of 0.5971 seconds.

Table 3. Result of Scenario 2 Flow Field Pathfinding

Test number	First Timer	Last Timer	Average
1	3.3368	3.8011	3.5690
2	3.3598	3.7603	3.5601
3	3.3923	3.8280	3.6102
4	3.3980	3.8258	3.6119

5	3.3250	3.8013	3.5767
6	3.3903	3.7070	3.5487
7	3.3313	3.7858	3.5586
8	3.3652	3.7724	3.5688
9	3.3620	3.7855	3.5738
10	3.4944	3.7956	3.6450

Table 3 shows the result of testing using Flow Field Pathfinding to find the shortest time required to reach the target. The average result of all trials conducted was (3.5690 + 3.5601 + 3.6102 + 3.6119 + 3.5767 + 3.5487 + 3.5586 + 3.5688 + 3.5738 + 3.6450) / 10 = 3.5822.

Table 4. Result of Scenario 2 A* Pathfinding

Test number	First Timer	Last Timer	Average
1	3.3368	3.8011	3.5690
2	3.3598	3.7603	3.5601
3	3.3923	3.8280	3.6102
4	3.3980	3.8258	3.6119
5	3.3250	3.8013	3.5767
6	3.3903	3.7070	3.5487
7	3.3313	3.7858	3.5586
8	3.3652	3.7724	3.5688
9	3.3620	3.7855	3.5738
10	3.4944	3.7956	3.6450

Table 4 shows the result of testing using A* Pathfinding to find the shortest amount of time required to reach the target. The average result of all trials conducted was (4.4292 + 4.4347 + 4.3888 + 4.4840 + 4.3405 + 4.3667 + 4.4258 + 4.4445 + 4.4041 + 4.4225) / 10 = 4.4140. When the result compared to the average results of Table 3 with Table 4 for scenario 2, it can be seen that the average Flow Field time (3,5822) is faster than A* (4,4140) and has a difference of 0.8318 seconds.

Table 5. Results of Scenario 3 Flow Field Pathfinding

Test number	First Timer	Last Timer	Average
1	13.5495	14.1580	13.8538
2	13.5022	15.0643	14.2833
3	13.5506	14.0927	13.8217
4	13.5424	14.7615	14.1520
5	13.5153	14.1116	13.8135
6	13.5989	17.1915	15.3952
7	13.5591	17.3906	15.4749
8	13.6293	15.0069	14.3181
9	13.7016	14.4805	14.0911
10	13.6289	17.3056	15.4673

Table 5 shows the result of testing using Flow Field Pathfinding to find the shortest time required to reach the target. The average result of all trials conducted is (13.8538 + 14.2833 + 13.8217 + 14.1520 + 13.8135 + 15.3952 + 15.4749 + 14.3181 + 14.0911 + 15.4673) / 10 = 14.4671.

Table 6. Result of Scenario 3 A* Pathfinding

Test number	First Timer	Last Timer	Average
1	14.7301	15.3122	15.0212
2	14.7364	15.3767	15.0566
3	14.7859	15.1677	14.9768
4	14.6179	15.3490	14.9835
5	14.8016	15.1858	14.9937
6	14.7462	15.2847	15.0155
7	14.8541	15.1946	15.0244
8	14.8757	15.3147	15.0952
9	14.7276	15.2667	14.9972
10	14.8954	15.2926	15.0940

Table 6 shows the result of testing using A* Pathfinding to find the shortest amount of time required to reach the target. The average result of all trials conducted is (15.0212 + 15.0566 + 14.9768 + 14.9835 + 14.9937 + 15.0155 + 15.0244 + 15.0952 + 14.9972 + 15.0940) / 10 = 15.0258. When compared to the average results of Table 5 with Table 6 for scenario 3, it can be seen that the average time of Flow Field (14,4671) is faster than A* (15.0258) and has a difference of 0.5587 seconds

V. CONCLUSION

Based on the research that has been done, the following conclusions are obtained, that has successfully implemented the Flow Field Pathfinding algorithm and the A* Pathfinding algorithm into the Tower Defense game simulation. Flow Field Pathfinding can be used and has faster results than A* Pathfinding for finding paths in Tower Defense games. This is evident from the scenarios that have been simulated. From the simulation results carried out in scenario one, the average time required by the Flow Field Pathfinding algorithm is 0.5971 seconds faster. In scenario two simulation, the average time required by the Flow Field Pathfinding algorithm is 0.8318 seconds faster. And for scenario three simulation, the average time required by the Flow Field Pathfinding algorithm is 0.5587 seconds.

ACKNOWLEDGMENT

Thank you to the Universitas Multimedia Nusantara, Indonesia which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia

REFERENCES

1) P. Avery, J. Togelius, E. Alistar, and R. P. Van Leeuwen, “Computational intelligence and tower defence games,” *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, no. July 2011, pp. 1084–1091, 2011, doi: 10.1109/CEC.2011.5949738.
2) S. L. Pardede, F. R. Athallah, Y. N. Huda, and F. D. Zain, “A Review of Pathfinding in Game Development,” *[CEPAT] Journal of Computer Engineering: Progress, Application and Technology*, vol. 1, no. 01, p. 47, 2022, doi: 10.25124/cepat.v1i01.4863.
3) N. H. Barnouti, S. S. M. Al-Dabbagh, and M. A. Sahib Naser, “Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm,” *Journal of Computer and Communications*, vol. 04, no. 11, pp. 15–25, 2016, doi: 10.4236/jcc.2016.411002.
4) H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, “The EBS-A* algorithm: An improved A* algorithm for path planning,” *PLoS ONE*, vol. 17, no. 2 February, pp. 1–27, 2022, doi: 10.1371/journal.pone.0263841.
5) S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, “An improved A-Star based path planning algorithm for autonomous land vehicles,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 5, pp. 1–13, 2020, doi: 10.1177/1729881420962263.
6) E. Emerson, “Crowd Pathfinding and Steering Using Flow Field Tiles,” *Game AI Pro 360*, pp. 67–76, 2019, doi: 10.1201/9780429055096-7.
7) A. Koesnaedi and W. Istiono, “Implementation Drunkard ’ s Walk Algorithm to Generate Random Level in Roguelike Games,” *International Journal of Multidisciplinary Research and Publications*, vol. 5, no. 2, pp. 97–103, 2022, [Online].

Available: Drunkard's Walk, Guest User Satisfaction Scale, Procedural Content Generation, Video game.

8) J. Zhang, Q. Chen, M. Shi, H. Zhou, and L. Xu, "Interaction and influence of a flow field and particleboard particles in an airflow forming machine with a coupled Euler-DPM model," *PLoS ONE*, vol. 16, no. 6 June, pp. 1–25, 2021, doi: 10.1371/journal.pone.0253311.
9) S. R. Lawande, G. Jasmine, J. Anbarasi, and L. I. Izhar, "A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games," *Applied Sciences (Switzerland)*, vol. 12, no. 11, 2022, doi: 10.3390/app12115499.
10) A. Hernàndez-Sabaté, M. Joanpere, N. Gorgorió, and L. Albarracín, "Mathematics learning opportunities when playing a Tower Defense Game," *International Journal of Serious Games*, vol. 2, no. 4, 2015, doi: 10.17083/ijsg.v2i4.82.
11) C.-M. C. Steven K.C. Lo, Huan-Chao Keh, "A Multi-agents Coodination Mechanism to Improving Real-time Strategy on Tower Defense Game," *Journal of Applied Sciences*, vol. 13, no. 5, pp. 683–691, 2013, doi: 10.3923/jas.2013.683.691.
12) S. L. Yin Fan, Chang-Hui, "12-Real-time visualization algorithm of 2D unsteady flow field.pdf," *Journal of Computer Applications*, vol. 30, no. 9, p. 2434, 2010, doi: 10.3724/SP.J.1087.2010.02434.
13) A. Suryadibrata, J. C. Young, and R. Luhulima, "Review of Various A* Pathfinding Implementations in Game Autonomous Agent," *IJNMT (International Journal of New Media Technology)*, vol. 6, no. 1, pp. 43–49, 2019, doi: 10.31937/ijnmt.v6i1.1075.
14) N. L. Manuel, N. İnanç, and M. Y. Erten, "Control of mobile robot formations using A-star algorithm and artificial potential fields," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 12, no. 2, pp. 57–67, 2021, doi: 10.14203/j.mev.2021.v12.57-67.
15) M. R. Wayahdi, S. H. N. Ginting, and D. Syahputra, "Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path," *International Journal of Advances in Data and Information Systems*, vol. 2, no. 1, pp. 45–52, 2021, doi: 10.25008/ijadis.v2i1.1206.