SIMULATION AND COMPARISON OF EFFICENCY OF DIJKSTRA'S AND A* ALGORITHM

Anita, Department of Computer Science Engineering (CSE)
Prannath Parnami Institute science and Technology,
Chaudharywas,
Hisar, Haryana, India

Neeraj Verma, Department of Computer Science Engineering (CSE)
Prannath Parnami Institute science and technology,
Chaudharywas,
Hisar, Haryana, India

Abstract-There are a lots of paths to go from one place to another place i.e. point A to point B in real road maps and Driver need to pick the best path. To do this, the pathfinding calculations is utilized. At present, a few calculations have been proposed for steering in recreations so the general difficulties of them is high utilization of memory and a long Execution time. Because of these issues, the improvement and presentation of new calculations will be proceeded. At the initial segment of this article, notwithstanding essential and imperative utilized calculations, everyone knows the point where the driver or user is and where they want to go. The map has roads (they are called edges) that connect the nodes (places with coordinates).From every node, user can go to one or many edges. An edge has a cost (e.g. length or time it takes to travel it). For small maps, one could perhaps calculate all possible routes to the destination and select the shortest.

For these calculations in the different modes and Simulated calculations various algorithms are Dijkstra, Iddfs, Biddfs, Bfs (Breadth first search), Greedy Best First Search, Ida*, A*, Jump point seek, HPA*.

Keywords -Dijkstra algorithm, shortest path, small heap, passing point, heuristic, pathfinding.

I. INTRODUCTION

Way finding is characterized as the way toward moving a protest from its prior position to the last position. Distinctive application territories utilized Path Finding Algorithms (PFA). These incorporate Games and Virtual Tours, Driverless Vehicles, Robot Motion and Navigation.

Way finding is typically portrayed as a procedure of finding a way between two focuses in a specific domain. By and large the goal is to locate the briefest way conceivable, which would be ideal i.e., the most limited, least expensive or easiest. A few criteria, for example, way which emulates way picked by a man, way which requires the most reduced measure of fuel, or from two focuses A and B through point C is frequently discovered significant in numerous way discovering undertakings.

Finding the briefest way is the most troublesome issue in numerous fields, beginning with navigational frameworks, manmade brainpower and closure with PC reenactments. In spite of the fact that these fields have their own particular calculations, there are numerous universally useful way discovering calculations that are connected effectively. In any case, it stays misty what benefits certain calculation have in correlation with others.

Briefest way calculations are at present utilized generally. They are the premise of a few issues, for example, arrange stream issues, tree issues and other related issues. They choose the base cost of movement of the issues generation cycle, the briefest way in an electric circuit or the most dependable way.

The web is an immense field where the briefest way calculation is typically connected. The Internet issues contain information bundle transmissions with insignificant time or utilizing the most solid way.

The paths to reach the destination may consist of various kinds of obstacles which the NPC is to avoid. Also, it is feasible to expect that the NPC would take the shortest path possible to arrive at its destination while avoiding these obstacles. This arises an issue of the NPC finding the shortest path between these two end points while eluding obstacles. The technique used to resolve this issue is called pathfinding, which finds the shortest path between two locations for the computer-controlled player. The concept of pathfinding has become more and more popular as the gaming industry is gaining more and more importance. Dijkstra’s algorithm has been the solid foundation on which various pathfinding algorithms have been developed. Many conventional solutions to the pathfinding problem like Depth-first Search, Best- First Search and Breadth-First Search were overwhelmed by the increase in the complexity demands by the games. A* algorithm has become the most popular and provably the optimal solution to the pathfinding problem. Nevertheless, it presents a very promising field for future research by considering various improvements and optimization’s to the A* algorithm.

II. PROBLEM STATEMENT

In this paper, the problem under study is to analyze the shortest path and analyzing the performance using Dijkstra’s and A* algorithm and to implement on Microsoft visual studio using C# to mimic the real behavior of the system. To analyze the performance of shortest path using A* environment is much better than to test it on real system rather than Dijkstra’s. The main problems faced while routing are taken into consideration:

1. Calculating the shortest path between origin and destination. In section 3.2 step 3, the researcher used an application of his development in visual C# to calculate the shortest path between the origin and destination points based on Dijkstra and A* algorithm.

2. Detecting the source and destination points. For the path to be drawn, the user should determine the starting point (origin) and final point (destination) for routing. However, in general the origin and destination can be any point.

3. There is no direct path from origin and destination points.

4. Dealing with huge data. The transportation systems have huge datasets to represent destinations, transmit type, etc.; hence manipulation such as calculating shortest path requires handling huge data; the researcher has faced this problem while reading and processing in Microsoft Visual C#. The map has roads (they are called edges) that connect the nodes (places with coordinates).From every node, you can go to one or many edges. An edge has a cost (e.g. length or time it takes to travel it). For little maps, one could maybe compute every conceivable course to the goal and select the briefest. In any case, that isn't extremely useful for maps with numerous hubs as the mixes develop exponentially.

III. OBJECTIVES OF THE RESEARCH

The reason for inquire about is to find the response to inquiries through the use of logical techniques. The fundamental point of the exploration is to discover reality which is covered up and which has not been found up 'til now. Each exploration examine has its own particular reason, so this examination work likewise has its own particular targets. The noteworthy point of this exploration is to get the best way and going to least hubs by utilizing Dijkstra's and A*. The emphasis will be on getting the best way with least cost.

IV. PSEUDO CODES

a) Pseudo code for Dijkstra's calculation

The Dijkstra calculation was found in 1959 by Edsger Dijkstra. This is the manner by which it works:

(i) From the begin hub, add every associated hub to a need line.

(ii) Sort the need line by least cost and make the principal hub the present hub.

(iii) For each tyke hub, select the best that prompts the most limited way to begin.

(iv) When the sum total of what edges have been examined from a hub, that hub is “Went to" and you don't have to go there once more.
(v) Add every kid hub associated with the present hub to the need line.
(vi) Go to stage 2 until the point that the line is vacant.
(vii) Recursively make a rundown of every hub that leads the most limited way from end to begin.
(viii) Reverse the rundown and you have discovered the most limited way

b) Pseudo code for the A * calculation

(i) There are numerous upgrades of Dijkstra's calculation. A standout amongst the most widely recognized is called A*. It is essentially the same as Dijkstra with one straightforward alteration.

(ii) Edges are organized additionally as for how much closer that edge prompts a straight-line separation to the objective. So before running an A* seek, the straight-line separation to the last goal must be estimated for each hub, which is simple in the event that you know every hub facilitate. This is the easiest type of A* and its definition additionally takes into account enhancements of the heuristics work. (For this situation Straight Line Distance To End)

(iii) This calculation has a major execution advantage since it doesn't have to visit the same number of hubs when the course of the way end is known.

V. RESULTS

The following tests will illustrate how the Dijkstra’s and A* algorithm is used for path finding on huge data i.e. Network of roads in real life. These tests will show the effectiveness of the A* algorithm against the system running without A* i.e. working in Dijkstra’s Algorithm mode. Since it is possible to use both algorithms i.e. Dijkstra's or A* calculation, various test correlations will be done to demonstrate how A* calculation can enhance the directing when ways are longer legitimate and new routes have to be chosen.

Test: Number of nodes to be taken 3000

Result of 3000 nodes using Dijkstra’s mode

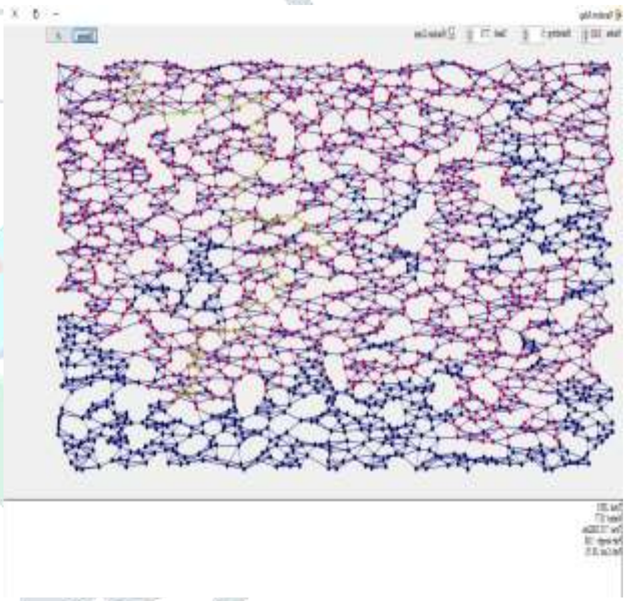
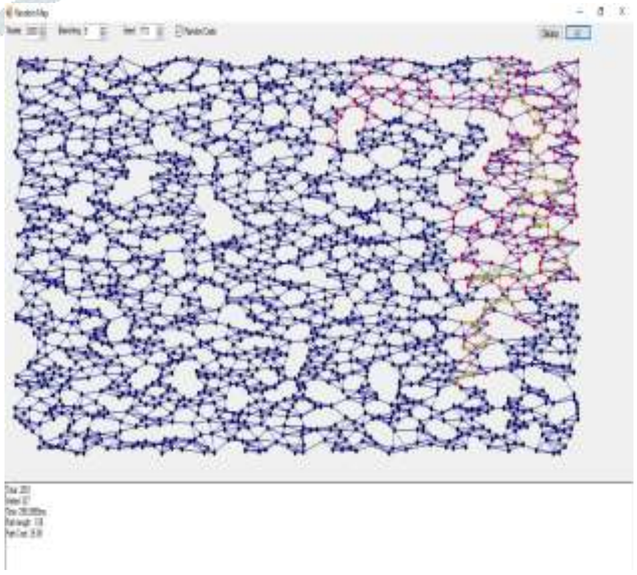


Fig.5.1 Result of 3000 nodes using Dijkstra’s



Algorithm

Result of 3000 nodes using A* Algorithm mode

Fig.5.2Result of 3000 nodes using A* Algorithm

The test contains two results:

- GUI 1 (**Dijkstra's Mode**)
- GUI 2 (**A* Mode**)

From this, it is clear that even by the first 3000 nodes visits completed; Dijkstra's mode has higher no. of visited nodes to reach the destination but A* reduced the average number of nodes by approximately 45 times. At the end of the analysis optimal route is obtained, resulting in A* improving performance by almost 45 times visiting and it is in very small data of 3000 nodes.

Now analysis tests will be performed by increasing number of nodes to be made in successive tests and we will observe the number of nodes visited to complete the route, then we will compare the simulation results of Dijkstra's vs. A*algorithm.

CONCLUSION

This paper gives a short portrayal of the essential pathfinding calculations which fill in as an establishment for the effective execution of A* calculation. It at that point displays an unpleasant draw of the A* calculation outlining how it clubs the benefits of Dijkstra's calculation and

Best-First-Search calculation, killing their disadvantages. The paper finishes up by talking about different improvement procedures for the A* calculation and future research scope around there.

REFERENCES

1. Xiao Cui and Hao Shi, "A*-based Path-finding in Modern Computer Games" Melbourne, Australia: IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011.
2. Tristan Cazenave, Labo IA, "Optimizations of data structures, heuristics and algorithms for path-finding on maps", France: Computational Intelligence and Games, 2006 IEEE Symposium, May 2006.
3. S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, "Artificial and computational intelligence in games (dagstuhl seminar 12191)." Dagstuhl Reports, vol. 2, no. 5, pp. 43–70, 2012.
4. Artificial and Computational Intelligence in Games.SchlossDagstuhlZentrumf¨urInformatik GmbH, 2013.
5. C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 4, no. 1, pp. 1–43, 2012.
6. G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," Affective Computing, IEEE Transactions on, vol. 2, no. 3, pp. 147–161, 2011.
7. J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 3, no. 3, pp. 172–186, 2011.
8. G. N. Yannakakis, P. Spronck, D. Loiacono, and E. Andre, "Player modeling," Dagstuhl Follow-Ups, vol. 6, 2013.
9. M. Riedl and V. Bulitko, "Interactive narrative: A novel application of artificial intelligence for computer games." in AAAI, 2012.
10. M. O. Riedl and A. Zook, "AI for game production," in Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE, 2013, pp. 1–8.
11. P. Hingston, C. B. Congdon, and G. Kendall, "Mobile games with intelligence: A killer application?" in Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE, 2013, pp. 1–7.
12. G. N. Yannakakis, "Game AI revisited," in Proceedings of the 9th conference on Computing Frontiers, ser. CF '12. New York, NY, USA: ACM, 2012, pp. 285–292.

13. B. Stout, “Smart moves: intelligent path-finding,” in Game Developer Magazine, pp.28-35, 1996.
14. Stanford Theory Group, “Amit’s A* page”, October 12, 2010.
15. N.Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann Publishers, San Francisco, 1998.
16. P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Trans.Syst.Sci.Cybernet., vol.4, no.2, pp.100-107, 1968
17. B. Stout, “The basics of A* for path planning,” in Game Programming GEMS, pp.254-262, Charles River Meida, America, 2000.
18. A.Botea, M.Mueller, and J.Schaeffer, “Near optimal hierarchical path-finding,” J. GD, vol.1, no.1, pp.7-28, 2004.
19. Unreal Developer Network, “Navigation mesh reference”, Jan.13, 2011.
20. Game/AI, “Fixing pathfinding once and for all”, September 23, 2010.
21. P. Tozour, “Building a near-optimal navigation mesh,” in AI Game Programming Wisdom, pp.171-185, Charles River Media, America, 2002.
22. S. Rabin, “A* speed optimizations,” in Game Programming GEMS, pp.264-271, Charles River Media, America, 2000.
23. B. D. Bryant, “Evolving visibly intelligent behavior for embedded game agents,” Ph.D. dissertation, Department of Computer Sciences, The University of Texas at Austin, 2006.
24. R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong, “Computational intelligence in games,” in Computational Intelligence: Principles and Practice, G. Y. Yen and D. B. Fogel, Eds. IEEE Computational Intelligence Society, 2006.
25. R. C. Arkin, Behavior-Based Robotics. CA: MIT Press, 1998.
26. M. J. Mataric`, “Behavior-based control: Examples from navigation, learning, and group behavior,” Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2-3, pp. 323–336, 1997, journal of Experimental and Theoretical Artificial Intelligence.
27. Adam A. Razavian, Sun J. (2005). Cognitive Based Adaptive Path Planning Algorithmfor Autonomous Robotic Vehicles, Southeast Con 2005 Proceedings, 8-10.
28. Amemiya T., Yamashita J., Hirota K. and Hirose M. (2004). Virtual leading blocks for thedeaf-visually impaired: a real-time way-finder by verbal-nonverbal hybridinterface and high-density RFID tag space.
29. BenaichaRamzi, Taibi Mahmoud. (2013).Dijkstra Algorithm Implementation On Fpga Card For Telecom Calculations.
30. Benjamin Chong Min Fui.(2012). A Comparative Study Of Maze Solving Algorithm For An Autonomous Mobile Robot.
31. BusraOzdenizci, Kerem , VedatCoskun, Mehmet N. Aydin. (2011). “Development of an Indoor Navigation System Using NFC Technology”.
32. Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). Single-source shortest Paths: Introduction to algorithms. 2nd ed. Cambridge, MA: MIT Press,581-635.
33. Carsten J. (2007).Global Path Planning on board the Mars Exploration Rovers. IEEE Aerospace Conference.
34. David M. Bourg, Seeman G.(2004). AI for Game Developers, O'Reilly, Chapter 7.
35. Edward M. Measure, David Knapp, Terry Jameson, and Andrew Butler. (2009). Automated Routing of U\nmanned Aircraft Systems (UAS).
36. Hart P. E., Nilsson N. J., Raphael B. (1968). “A Formal Basis for the Heuristic Determination of Minimum Cost Paths.
37. Jacob R., Marathe M. and Nagel K. (1999). A Computational Study of Routing Algorithms for Realistic Transportation Networks, ACM Journal of Experimental Algorithms.