A REVIEW PAPER ON EXAMINATION OF DIJKSTRA'S AND A* ALGORITHM TO FIND THE SHORTEST PATH

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

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

Mr. Abhishek Bansal, HARSAC, Department of Science & technology, CCS HAU, 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), 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. Literature Review

Way discovering calculations are helpful in theareaof automated control, as it can be utilized to control a robot around troublesome landscape without the need of human intercession (Carsten J, 2007). It would be productive if the robot were ondifferent planet like Mars, in which some geology must be dodged, however because of the extreme distances include, guiding it totally through remote control would be troublesome (a lot of deferral in the radio transmission)(Obara T et al., 1994).

It could likewise be valuable if the robots are to be worked submerged, where radio waves couldn't get to it. It additionally discover applications for any situation where a vehicle needs to go someplace, while maintaining a strategic distance from hindrances, without human intervention.(David M. et al., 2004)other utilize is in PC recreations where something should be moved starting with one place then onto the next keeping away from any dividers or different troubles in the way.

The calculations may likewise be utilized as a part of determining the most limited way to drive betweentwopointson a guide. Which is the most ideal approach to highway an email through PC arrange or briefest way to run phone wires through existing circuits.al. [11].

Path finding is an important part of game programming. Game typesmove according to the path they (or computer) calculate. There are some algorithms used in the gamechanging with the complexity and purpose of the path calculation. Themostalgorithm found currently in games todayare the A* Algorithm.

"What makes the A* algorithm so appealing is that it is guaranteed to find the best path between any initial point and any ending point, assuming that a path exists." (David M. et al., 2004).

Users directed in a virtual building or virtual duplicateof a real building, for example a museum, can see the artworks without having to go to a physical place. This is moralfor eluding traveling long distances and attaining more people to show the works. Users can arrange their visit path and algorithm calculates the route thenthe virtual tour begins.Shafie& Hassan (2004) work on this scenario and developed an application capable of doing path planning process. They used an A* algorithm to find the path. The final path can be seen in a 3D environment (Figure 2.1).

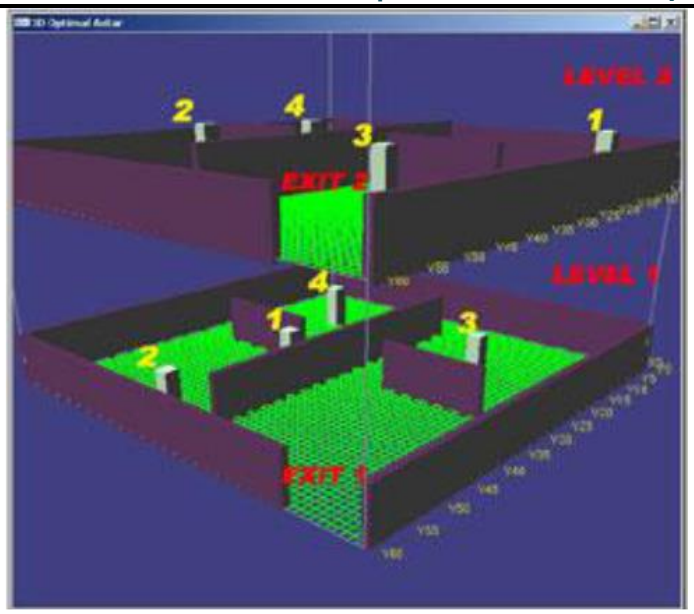

Figure 2.1: 3D Model of Latiff and Hassan 2004
(M. Shafie, & R. Hassan, 2004)

Self-governing triangulation of virtual characters, is likewise an essential errand to work on. It is required to counteract snags and continue to their ways with no impairment. In a virtual environment not like the automated route, there is no information originating from sensors, there exist the site database as it were. (Fr̈hlich and Kullmann, 2002)haveusedA* calculation ponders, on the natural displaying of the territory and settled that the uniform-sized matrix cell system is a superior decision, alsostatedthat A* calculation functions admirably in the situations where the world isflat,whichsuits our condition well.

Robot Motion and Navigation

Versatile robots moving in an open air or indoor condition must have their course planning and navigational plans keeping in mind the end goal to have the capacity to discover their heading. Generally, the route and pathfinding units are set on the robots so that can move without anyone else's input. It is basic to have this property, if the use of these robots is considered. In the military, identification of unsafe or unstable materials and revelation and examination of obscure zones are exceptionally reasonable for this sort of robots. Razavian and Sun (2005) proposed another calculation called Cognitive Based Adaptive Path Planning Algorithm (CBAPPA) contrasting it and A* Algorithm and Focused D* Algorithm (D*, aDynamic A* calculation which is coming about because of the A* Algorithm and has some enhanced included abilities utilized for self-ruling units), used it to watch the conduct of natural units and focused on the conduct of overlooking the immaterial data from the environment, attempting toreachthetarget rapidly. This technique may not utilize ideal ways, but rather the outcomes were productive. This calculation is likewise arranged for self-preparing units, which can be a decent case for our future works.

Koenig and Likhachev (2002) declares another new calculation called "D* Lite" which is a variation of D* Algorithm for enhanced quick arranging in obscure territory. D* calculation improves the situation than A* in obscure zones when utilized as a part of self-coordinated robots. D* Lite is more powerful than D* in light of the fact that it is more straightforward, shorter and less demanding to get it. D* and D* Lite calculations assess their insight about the landscape when they move into it and persistently perform arranging of the track. This strategyis very unique in relation to our condition, however can be exceptionally helpful when it is actualized framework in different (might be obscure) conditions.

It is expressed in Artificial Intelligence (AI) for the Game Developers book(David M. et al., 2004) that: "Pathfinding issue is thought as unraveled, future works are for influencing the calculation to quick and more productive." Studies give more viable and quicker calculations to utilize and execute in various zones.

a) Foundation Of A* Algorithm

A* algorithm is based on two conventional algorithms, namely Dijkstra's algorithm and Greedy Best-First-Search algorithm. In this section, the paper presents a brief description on these two basic algorithms. The idea behind both these basic algorithms is to keep track of the vertices in the form of an expanding ring-like structure called a frontier. The basic difference between these algorithms is how and on what terms the frontier expands.

b) Dijkstra's Algorithm

Dijkstra's algorithm visits vertices in the graph one by one starting with the object's starting point. It then examines the closest vertex which is yet to be examined and this process runs in an outer loop which terminates when either the vertex examined happens to be the target or else if the target is not found even after all the vertices have been examined. Otherwise, the closest vertices to the examined vertex is then added to the collection of vertices to be examined. In this fashion, it expands outwards from the starting point until it reaches the goal. When the target is found, the loop terminates and then the algorithm backtraces its way to the start remembering the required path.

Dijkstra's algorithm unfailingly finds the shortest path from the starting point to the goal. However, when searching for a single target or goal, the use of this algorithm is not recommended because it consumes extra time and resources due to the additional number of nodes this algorithm inspects. On the other hand, if there are multiple targets to be searched for, then this algorithm serves as the quickest option.

c) Covetous Best-First Search Algorithm

The Greedy Best-First-Search calculation additionally monitors a boondocks to find the objective. In any case, there is a noteworthy distinction in the way this wilderness grows when contrasted with that of Dijkstra's calculation. This calculation makes utilization of a heuristic capacity which decides roughly how a long way from the objective a specific vertex is. The Dijkstra's calculation chooses the hub closest to the beginning stage, while here the vertex nearest to the objective is chosen and given higher need than those hubs which are away. Subsequently, if the objective is to one side of the beginning stage, Greedy Best-First-Search will attempt and spotlight on the way which prompts towards the right. This encourages the calculation to catch the objective in its outskirts rapidly. Whatever is left of the strategy is a similar where it backtraces the pointers to the parent hubs to figure the way voyaged.

Voracious Best-First-Search runs substantially quicker than Dijkstra's calculation since it utilizes the heuristic capacity to appraise the separation to the objective which encourages it channel its ways to spare time and assets. In any case, this calculation, dissimilar to Dijkstra's, does not ensure a most limited way. It will lead the NPC to the goal, dodging all deterrents. However, the way picked may not be the best one.

Transportation Networks

In road networks, it is getting more important to find a path to the final end point. If a person is new to a location, sample time can be wasted in locating the end point.

There exist some products established to overcome this difficulty, providing a map of the region. After entering the starting and the final destination, it is likely to acquire the shortest possible track.

Experimental studies were performed to select the best algorithm for using in the path finding process. For instance study by Jacob et al (1999) from Los Alamos National Laboratory who compared the

shortest path algorithms by doing some experimental analysis on big database (Figure 2.2), assessed the Dijkstra, A* and modified A* algorithms with respect to the calculation time on the real network solution quality, affluence of execution and the extensibility of the algorithm,Found out that A* algorithm has a better time efficiency.

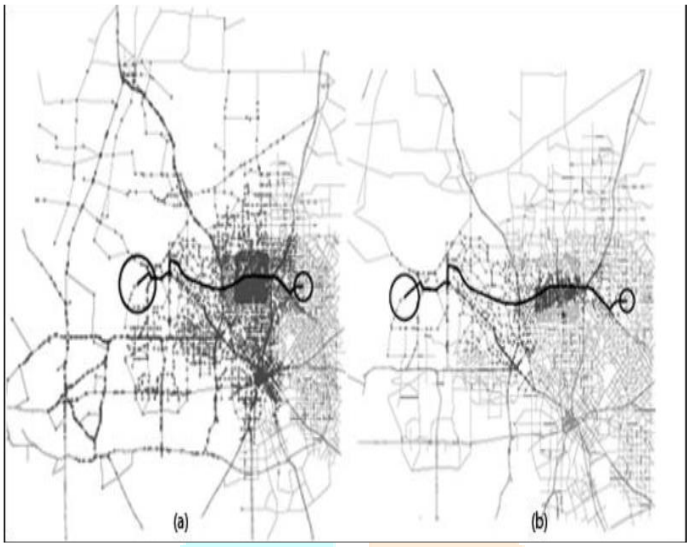

Figure 2.2: (a) Dijkstra's Search (b) A* search on road network of Dallas Ft-Worth urban area

Figures 2.2 showed that A* Search is also much more effectivewith respect to the number of nodes visited. Therefore, Dijkstra's Algorithm searches much more nodes than A* does.

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. SchlossDagstuhl-Leibniz-Zentrumf¨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.
Jacob R., Marathe M. and Nagel K. (1999). A Computational Study of Routing Algorithms for Realistic Transportation Networks, ACM Journal of Experimental Algorithms.
38.