Ciência e Natura, v. 37 Part 2 jun. 2015, p. 230–238

ISSN impressa: 0100-8307 ISSN on-line: 2179-460X

Simulation and Comparison of Efficency in Pathfinding algorithms in Games

Azad Noori [1], Farzad Moradi [2]

[1]Department of Computer at Technical and Vocational University, Tehran, iran
[2] Saghez Branch,Islamic Azad University, Saghez, iran

Abstract

There are several routes to go from point A to point B in many computer games and computer player have to choose the best route. To do this, the pathfinding algorithms is used. Currently, several algorithms have been proposed for routing in games so that the general challenges of them is high consumption of memory and a long Execution time. Due to these problems, the development and introduction of new algorithms will be continued. At the first part of this article, in addition to basic and important used algorithms, the new algorithm BIDDFS is introduced.

In the second part, these algorithms in the various modes, are simulated on 2D-Grid, and compared based on their efficency (memory consumption and execution time) , Simulated algorithms include: Dijkstra, Iddfs, Biddfs, Bfs (Breadth), Greedy Best First Search, Ida, A*, Jump point search, HPA*.*

***Keywords**: BIDDFS, JPS, A*,HPA*, IDA *.*

Recebido: dia/mês/ano Aceito: dia/mês/ano

1 Introduction

One of the important concepts in the game is trying to use artificial intelligence that has been in them [1]. For achieving the aim, several methods and techniques have been proposed .One of the most important methods is the use of pathfinding method, Some of them are discussed in part [2].

Routing algorithms are divided into two groups: informed and uninformed .Uninformed method (blind search) pathfinding is performed in all directions. When this method usually is used that the target location is not specified such as Dijkstra and IDDFS but, in formed method, heuristic function usually is used to locate the target so that routing direction is guided toward the target. In this method, target location is usually determined at first. Such methods A*, IDA*, HPA* and etc . The biggest challenge of these Algorithms is the use of resources such as memory and their execution time. Algorithms such as depth first search, Breadth first search, Dijkstra, iterative and A Star due to games restrictions in the use of resources such as memory and time are not usable.By increasing complexity of new games, game designers are thinking of new ways to solve old problems.

This paper consists of two parts. In the first part, theoretical concepts are presented. Some discussions such as search and display space, informed and uninformed search methods, heuristics and some of the most important functions of pathfinding algorithms and methods have been described. The second part includes the implementation and simulation. Some of the most important algorithms are evaluated and analyzed in experimental implementation. Algorithm A is one of the most important and basic of these algorithms due to excessive memory consumption and computational overhead that is currently not used, Instead of it , some methods such as HPA*[5] and jump point search [22] are used to improve and modify its defects. The development of these algorithms have been studied, finally, simulation is performed in section [6]. These algorithms are implemented on a 2D grid map. They are compared together according to performance and the number of nodes traversed from start to end [2,3].

2 History And Previous Work

In [4], the author mentions to intelligent routing that is required in route selection in games. In an article in [5] new algorithm called HPA* has introduced. By dividing map, several smaller sections cause decreasing the complexity of the routing algorithms used in the Grid maps.

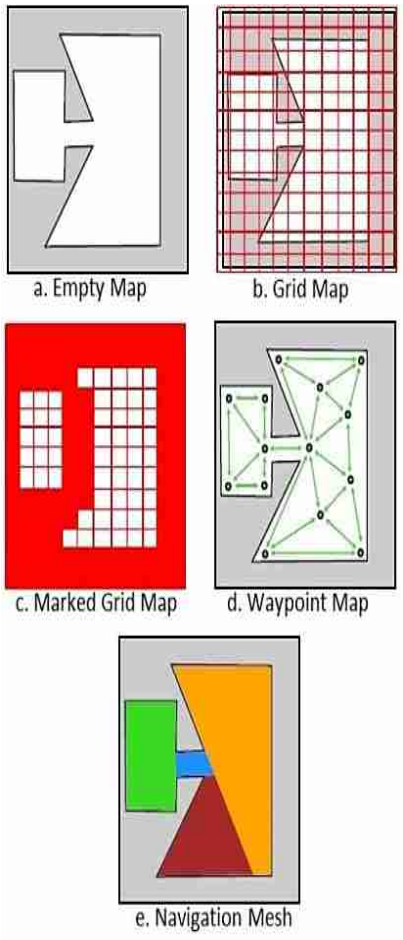
In [6] the author introduces a new algorithm called DHPA * and SHPA * algorithm for static and dynamic environments based HPA * and compared them according to their speed and efficiency with HPA *. Also in [7] some method such as bidirectional search and path smoothing are proposed on HPA*. in[8]While implementation of algorithm A* with C++ are checking , 2-dimension games based networking is presented. In [9] Routing algorithm role is considered in cognitive maps and Network modeling method is proposed .New algorithm called BIDDFS based DFS is introduced in [10] so that it is useful for tree search space and it is reduce iterative search problem in IDDFS.

3 Display The Search Space

The first step to examine different methods of routing, is finding a good way to display the search space. At this step, game space should be converted into the right structure to display the search space until we can implement and test routing algorithms. In other words, selection of an appropriate representation for the actual direction of motion is necessary. It has highly effective on the effectiveness and acceptability of the resulting path. For example simpler method can cause the reducing the number of search operations in the Algorithm A *, therefore, algorithm runs with higher speed. some algorithms, like HPA * by dividing the map into several clusters, or by displaying a map nav-mesh method as convex polygons, and by reducing the search space, all of them lead to increase A * algorithm speed. To implement and

Recebido: dia/mês/ano Aceito: dia/mês/ano

test the routing algorithms, route maps should become preferably weighted graph maps, some conversion of the state space to graph is located at the source [2] such as waypoint method (Figure 1.d) and nav-mesh approach(Figure 1.e) that put in the source [20] .Also, displaying map as grid has been described and used in [10]. Display of the search space can be displayed in 1 to 5 modes. [15-16].



Figur 1: Map representation [16]

4 Pathfinding in games

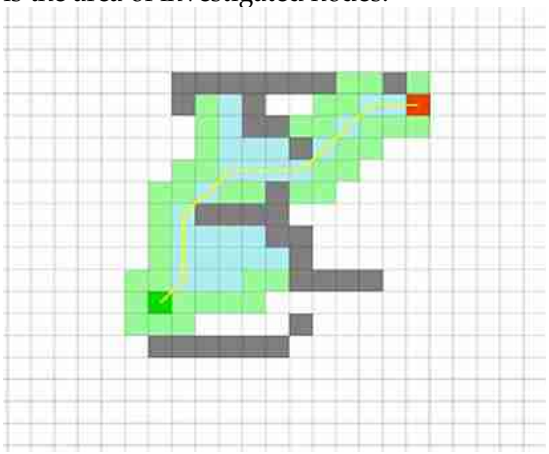
In many games, there are multiple choice and different path to the same destination The computer player needs to make the best decision and estimate shortest path from source to destination.

4.1 Uninformed search algorithms (blind search)

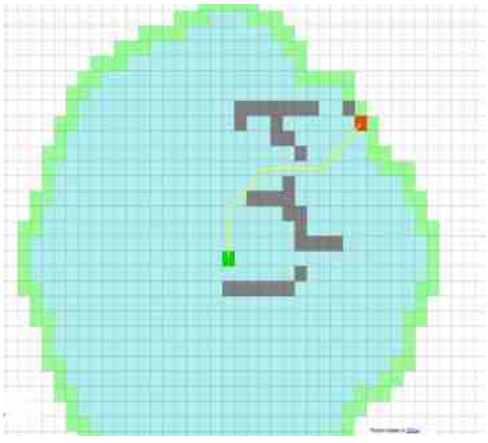
In this way, the search is performed in all directions and so many nodes are examined. This method is usually used when the first target node location is unknown. All of the algorithms that are located in this group guarantee to find the shortest path (if such path can be existed), however, it is time – consuming and a lot of space are searched. In this group are the most important algorithms that can be used include: BFS (Breadth-first search)،DFS ،IDDFS ،BIDDFS, Dijkstra,etc.(fig 2.b)

4.2 Informed search algorithms

In this method, heuristics function is usually used to locate target so that routing will be guided toward target .In this method, target location is usually determined from start point. Heuristic function work is the estimating of distance between the current node and the efficiency of this method depends on the intended heuristics function work. A * algorithm and its derivative algorithms used informed search method.(fig 2.a) Finding the difference between informed and uninformed method can be observed in following Figures. The blue color is the area of investigated nodes.



Figur 2.a : informed search (A*)



Figur 2.b : blind search(Dijkstra) (Xu, 2013)[24]

Recebido: dia/mês/ano Aceito: dia/mês/ano

4.3 Grouping search algorithms

In Table 1. Some of the most important routing methods and algorithms that have been proposed for the computer gaming environment (they are sorted by year).

5 Introduce Some Basic Algorithms In Each Group

5.1 Uninformed Algorithms

5.1.1 Dijkstra

This algorithm is one of the graph traversal algorithms that solves the shortest path problem from start point for weighted graphs that does not have any edge with negative weights, finally, by creating the shortest path tree , shortest path from the source to the all of destination vertex of graph is thus obtained Also ,you can use this algorithm to find the shortest path from source to destination vertex of algorithm so that when this algorithm is running , as soon as shortest path from source to destination, the algorithm can be stopped. Dijkstra algorithm trend is available in source [1]. Implementation sample is tested in two-dimensional Grid that it is visible in figure 2.b. As you can see, search is done in all directions and many nodes have been checked Finally, the shortest route is selected, The main characteristic of Dijkstra algorithm is that guarantees finding the shortest path in return, it will take time to implement because all paths must be tested and the shortest path should be selected .The nearest node to the source node should be found in this algorithm at first, and then the search continues in all directions (Blind search). The Dijkstra

Table 1. Pathfinding algorithms

a. Informed Search

Name	Year	Author(s)
A*	1968	Peter Hart,Nils Nilsson,Bertram Raphael
BFS(best first search)	1984	Judea Pearl
IDA*(iterative deepening A*)	1985	Korf, Richard
LRTA* (Learning Realtime A*)	1990	Korf
SMA* (SimplifiedMemory Bounded A*)	1992	Russell, S
D*(dynamic a*)	1994	Anthony Stentz
HA*(Hierarchical A*)	1996	Holte, Perez,Zimmer
LPA*(Lifelong Planning A*)	2004	S. Koenig, M. Likhachev, D. Furcy
HPA* (Near Optimal hierarchical Path-finding)	2004	Adi Botea, and Martin,Muller and Jonathan Schaeffer
PRA*(Partial Refinement A*)	2005	Sturtevant and Buro
TRA*(Triangulation-Based Pathfinding)	2006	Douglas Demyen and Michael Buro
Theta*	2007	A. Nash, K. Daniel, S. Koenig and A. Felner
HPA* Enhancements	2007	M. Renee Jansen and Michael Buro
DHPA* AND SHPA*	2010	Alex Kring, Alex J. Champandard, and Nick Samarin
Jump point search	2011	D. Harabor; A. Grastien
GHPA*(grid-specific feauture of hpa*)	2011	Uwe koch
JPS (jump point search)	2013	Patel
PHI*	2013	Nash, A.Koenig, S
b. Uninformed Search		
BFS(Breadth-first Search)	1950	E. F. Moore
DIJKSTRA	1959	Edsger Dijkstra
DFS(depth-first search)	1882	Charles Pierre Trémaux
IDDFS(Iterative deepening depth-first search)	1985	Korf, Richard
BIDDFS(boundary iterative-deepening depth-first search)	2015	Kai Li Lim and others

Recebido: dia/mês/ano Aceito: dia/mês/ano

algorithm works in static environments well but, it will be changed in dynamic environment, and it will be inefficient.

5.1.2 IDDFS(Iterative deepening depth-first search)

This algorithms works like Dijkstra and it is suitable for tree searching space (korf ,1985), Actually, the algorithm is derived from DFS and a certain threshold is used to prevent iterative search in this algorithm(the problem that was in DFS)the first threshold is set to 1, and the search is performed to a depth of 1, If target is not found, then one unit is added to threshold, and search is done up to a depth of 2 and continues to do the same routine until the goal to be found.[23]

5.1.3 BIDDFS (boundary Iterative deepening depth- first search)

BIDDFS is an uninformed boundary search algorithm. The BIDDFS is a newly proposed algorithm aiming to address the memory drawback of the conventional Dijkstra's algorithm (Dijkstra) and the searching redundancy of the IDDFS. Its main concept utilizes a boundary search. The BIDDFS explores the compromise between the IDDFS redundancy and the Dijkstra's algorithm's increasing memory requirements on larger maps [11].

This algorithm[Lim and other ,2015] is created because of solving IDDFS problems ,the problem of IDDFS is iterative search means that each time that search starts from the roots and develops in depth and goal is not found, search starts from the root again and search will be continued in another path. But in BIDDFS method, this problem was solved by storing the last parent node that was traversed in the previous search. As a result, next search are not starting from root again. But, search will be continued from the node which is stored.

5.2 Informed algorithms

5.2.1 Greedy Best First Search

Greedy search is one of the best first ways to search ,the aim of this method is to minimize the cost of achieving the goal by using the estimate function (heuristics)In this strategy ,node which is closer to goal, It is Initially developed. This means that instead of the closest vertex is selected from the starting node, it selects nearest node to the destination node. In the greedy search, we have

$$f(n)=h(n) \quad (1)$$

The main feature of this algorithm is that it does not guarantee to find the shortest path But it has a higher Execution speed. In this method, finding a path from source to destination in the fastest time is so important Therefore, the obtained path is not necessarily the shortest path. The problem of simulation is visible in section 6. In this method, the nearest node to the target node is searched at first. In fact, unlike Dijkstra, Search is not in all directions but It is only toward the purpose.

5.2.2 A*

The solution to the above problems is to combine features of both methods (Dijkstra and Bfs) and devised a new method called A *. In this method, revelation of standard methods and conventional techniques have been combined so that advantages of both techniques can be used. This algorithm uses Greedy Best First Search and operates as follows:

- Path cost function g (n): the cost of the path from the initial node to n node
- Heuristic function h (n): The estimated cost of the cheapest path from n node to target node
- The evaluation function f (n): The estimated cost of the cheapest path through n.

$$f(n)=g(n)+h(n) \quad (2)$$

According to find short path optimization, A* works much better than the greedy bfs. And in terms of run speed, A* acts faster than Dijkstra . In other words, it finds the shortest path and its running is faster than Dijkstra. However, in the worst case, this algorithm may act as Dijkstra means that the performance of it is depended on used heuristic function one example of its implementation is 2D grid that it can be observed in Figure2.a. The source [4], you can find a sample implementation of this algorithm.

5.2.3 Hierarchical Pathfinding A * (HPA*)

Recebido: dia/mês/ano Aceito: dia/mês/ano

Hierarchical pathfinding A* was developed by Adi Botea and his colleagues in 2004 [5]. It is combination of pathfinding and clustering algorithms, which works by creating an abstract graph on the basis of two dimensional grids. The main HPA* principle is based on dividing search problem into several smaller sub problems, and caching results for every path segment [7]. HPA* pathfinding phase consists of two parts called preprocessing and online search. During preprocessing start and goal nodes are inserted into abstract graph, and inter-cluster edges are added. Then A* is used on abstract graph to find the shortest route. During online search the shortest route found in abstract graph is refined to full path in initial graph using A*. To find full path from start to goal node A* is used in every cluster on nodes that connect clusters. Finally partial results from every cluster are combined into full path [20].

5.2.4 JPS (Jump point search)

This algorithm is the kind of algorithm A *so that factor can jump over square Grid. Namely to reduce the number of examined nodes that are in open list, you can jump on some nodes, do not examine all the nodes individually. Some specific points on the grid will get special attention in this method for example, the point of corners Grid. JPS can be implemented as an optimization to the A* algorithm with minor changes. JPS excels in large, open areas of a map. It is in these open areas that JPS can skip, or *jump*, over a large number of intermediate nodes that would otherwise be expanded using a traditional A* algorithm [21],[22].

5.2.5 IDA *

This algorithm is similar to the algorithm IDDFS but it use heuristics. In this method, the cost threshold is used instead of deep threshold. In this method, the cost f-Limit is conducted thorough depth search. If the target node is found, the final route is achieved. Otherwise, one unit of f-Limit is added and search will be iterated again. If the number of iterations are not lot , the efficiency of this algorithm is similar to the A*. This algorithm was presented in 1985 by Korf [23].Like IDDFS method, a disadvantage of this method is the searching redundancy.

6 Simulation And Comparison of Above Algorithms

All experiments were done on a PC with intel (2.50GHz) cpu core i5 and Ram 4G, Windows 7. Algorithms are implemented on a2-dimension square Grid. For Heuristic function, Manhatan distance was used so that it is the best choice in square Grid. | y2 - y1 |+ | x2-x1 | movement is allowed in 8 directions. And route cost from each node to its neighbor nodes is 1, and cost diagonally movement line is taken $\sqrt{2}$, movement is allowed as diagonal movement. And test for each algorithm in is repeated 100 times until possible errors will be minimized. The grid size is 64*64. The first map, the number of blocked nodes is 10%.But in second map, the number of blocked nodes is 50%.

In Tables 2 and 3 . The results of the simulation of routing algorithms in 2D Grid environment are considerable. In these tables, you can see run time of algorithms, the number of traversed nodes and length of founded paths. As you can see informed logarithms have less execution time than uninformed logarithms and they traverse the less number of nodes. Best performance belongs to the HPA* algorithm and in the second place we can put JPS. Second block BIDDFS algorithm performance has considerably improved by increasing the number of nodes. Even Unlike mode 1, execution time is less than IDA *.

Recebido: dia/mês/ano Aceito: dia/mês/ano

Table 2: Execution time (ms), Traversed Nodes and Length of path with 10% blocked node in grid map (Grid size : 64*64 blocked node : 10%)

Type	Algorithm	Execution time(ms)	Traversd Nodes	Length
Uninformed Search	Dijkstra	1.89	496	23.36
	IDDFS	9.64	423	23.36
	BIDDFS	3.67	231	23.36
	BFS(Breadth)	7.33	993	23.36
Informed Search	Greedy Best First Search	2.2	53	29.31
	Ida*	5.232	312	28.54
	A*	1.96	46	23.36
	Jump point search	1.54	312	23.36
	HPA*	1.11	36	23.36

Table 3: Execution time (ms), Traversed Nodes and Length of path with 50% blocked node in grid map (Grid size : 64*64 blocked node : 50%)

Type	Algorithm	Execution time(ms)	Traversd Nodes	Length
Uninformed Search	Dijkstra	5.808	1535	16.49
	IDDFS	56.6	1631	16.49
	BIDDFS	35.41	971	16.49
	BFS(Breadth)	13.335	1521	16.49
Informed Search	Greedy Best First Search	4.205	86	21.31
	Ida*	10.632	734	20
	A*	4.016	98	16.49
	Jump point search	2.554	832	16.49
	HPA*	2.170	82	16.49

Recebido: dia/mês/ano Aceito: dia/mês/ano

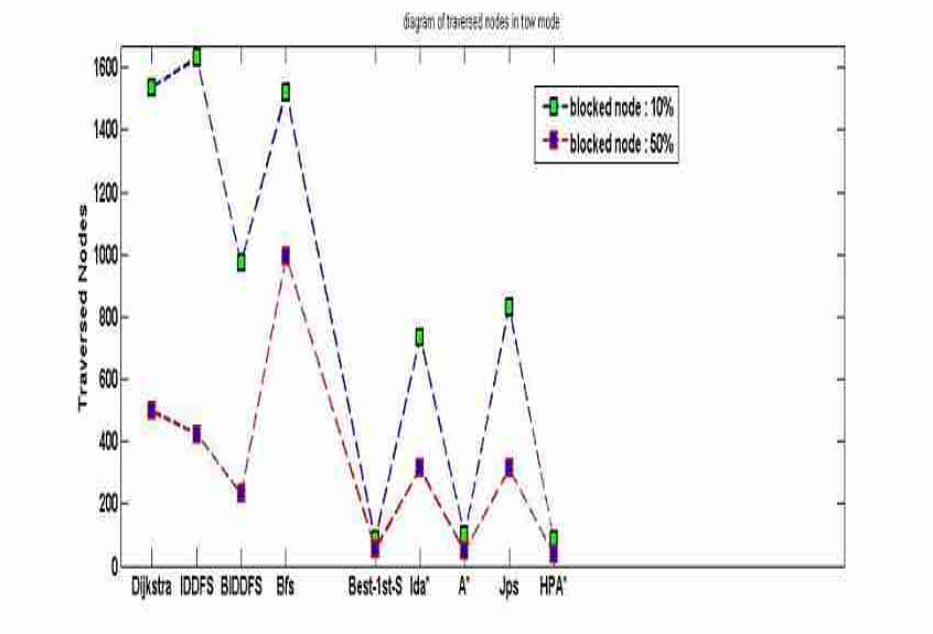


Figure 3: compare Traversed node in 2 mode

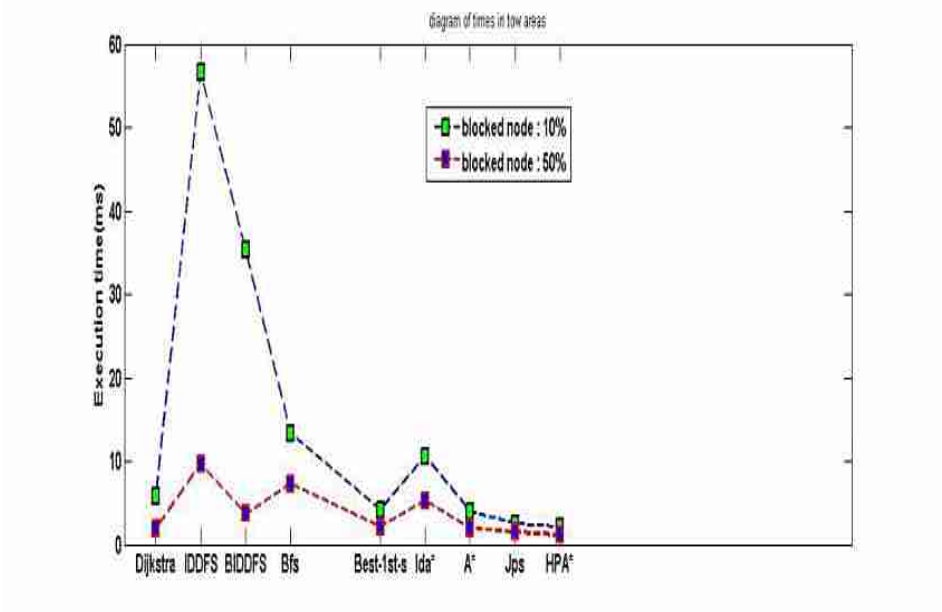


Figure 4: compare execution time in 2 mode

7 Conclusions

Using informed algorithms A* requires prior knowledge of the target location and their performance depends on used heuristics function (heuristics), there are various versions of A * Some of them cause to reduce the use of memory, or reduce the search space such as JPS, HPA*, IDA*and…. as we saw that HPA * had better performance among them but efforts to develop to be continued. However, uninformed algorithms such as BIDDFS do not prior knowledge about the target location but they have higher running time. However, when there are large number of obstacles, they have good performance in multi-objective problems, by using parallel processing algorithms, you can reduce the running time and In games where there is no prior knowledge about target location, they can be used.

8 References

1.Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001).

2.Ian Milington and Jon Fung ., 2009,"artificial intelligence for games",2nd edition,,morgan kaufman publishers,Elsevier.

Recebido: dia/mês/ano Aceito: dia/mês/ano

3.Brian Schwab., 2009. "AI Game Engine Programming ", 2e,Course Technology, a part of Cengage Learning.

4.Cui Xiao, Shi Hao., 2011,"A*-based Pathfinding in Modern ComputerGames", (IJCSNS) International Journal of Computer Science andNetwork Security, 11(1).

5. Botea A, Müller M, Schaeffer J,. 2004. "Near Optimal Hierarchical Path-Finding", Journal of Game Development, 1(1), .pp 7-28.

6. Alex Kring, Alex J. Champandard, and Nick Samarin., 2010. "DHPA* and SHPA*: Efficient HierarchicalPathfinding in Dynamic and Static Game Worlds".Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.

7. M. R. Jansen, M. Buro,. 2007." HPA* Enhancements". Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, USA.

8. Shane T. Mueller, Brandon S. Perelman, Benjamin G. Simpkins,. 2013. "Pathfinding in the cognitive map:Network models of mechanisms for searchand planning",Elsevier.

9.Amit'sThoughts,.2009."MapRepresentations on pathfindig", http://theory.stanford.edu /~amitp/GameProgramming/MapRepresentations.html.

10. Kai Li Lim, Kah Phooi Seng,. 2015." Uninformed pathfinding: A new approach ", Elsevier.

11. Khammapun Khantanapoka, Krisana Chinnasarn,. 2009. "Pathfinding of2D & 3D Game Real-Time Strategy with Depth Direction", Eighth International Symposium on Natural Language Processing.

12.Junfeng YAO, Binbin ZHANG, Qingda ZH. ,2009. "The Optimizationof A* Algorithm in the Practical Path Finding Application".WorldCongress on Software Engineering.

13.DewanTanvir Ahmed, Shirmohammadi Shervin, .2009. "Intelligent PathFinding for Avatars in Massively Multiplayer Online Games", IEEE Workshop on Computational Intelligence In Virtual Enviroments.

14. Russel, S., Norvig, P,.1995. "Artificial Intelligence: A Modern Approach" ,Prentice-Hall, Inc.

15. Gregory,Jason ., 2009. Game Engine Architecture , A K Peters.

16. Uwe Koch., 2011,"Applying graph partitioning to hierarchical pathfinding in computer games", matriculation number 9444829,Universit at Leipzig,Institut f˙ur Mathematik und Informatik .

17.Imants Zarembo,. 2013."pathfinding algorithm efficiency analysis in 2d grid",proceedings of the 9thinternational scientific and practical conference.volume 1I.

18.John.Wiley.and.Sons,, 2006.Algorithms.and.Networking.for.Computer.Games.

19.Game AI ,.2010. "Fixing pathfinding once and for all", http://www.ai-log.net/archives/ 000152 .html.

20. R. C. Holte, M. B. Holte, R. M. Zimmer, A. J. MacDonald,.1996." Hierarchical A*: Searching Abstraction Hierarchies Efficiently". AAAI 96 Proceedings of the thirteenth national conference on Artificial intelligence, vol. 1. pp. 530 – 535.

21. Bryan Tanner .,2014,"Jump Point Search Analysis".Florida State University.fsu.edu.

22. Patel, A. 2013. "Variants of A*. Retrieved from,http://theory.stanford.edu/~amitp/Game Programming/Variations.html, (2013, Jul 18).

23.Korf,Richard, 1985. "Depth-first Iterative-Deepening:An Optimal Admissible Tree Search".

24.Xu,X.(2013).Pathfinding Visual. Retrieved fromwww.github.com:http://qiao.github.io/PathFinding.js /visual/

Recebido: dia/mês/ano Aceito: dia/mês/ano