IMPLEMENTATION OF JUMP POINT SEARCH ALGORITHM TO THE ENEMY NPC FOR THE PURSUIT OF PLAYERS IN THE MAZE GAME

Ilham Ramadhan

Computer Engineering Courses
Faculty of Engineering and Computer Science of the University of Computer of Indonesia
Jl. Dipati Ukur 114 Bandung
E-mail : ilhamyudo27@gmail.com

ABSTRACT

One of the pathfinding algorithms that can be used in the search for and recognize the path from one point to another is a Jump Point Search algorithm (JPS). JPS is a development of algorithm A * developed by Daniel Harabor and Alban Grastien in 2011. This study aims to measure the performance of the algorithms Jump Point Search in finding a pathway with hitch which is dynamic. Research on the algorithm of the Jump Point Search will be applied to a maze game.

The characteristics of the games will be built based on a game that is game Bomberman, where the game has some features that can be used in this research. One such feature is the use of an obstacle to hinder the enemy in pursuit of the player.

The results showed that on a test run the algorithm with the simulation program was obtained by this algorithm can recognize a path change dynamically, it is indicated from finds success percentage rate above 90%. Thus this algorithm is used in the case of the viable game maze.
Keywords: artificial intelligence, games, labyrinth, pathfinding algorithm

1. INTRODUCTION

The maze-themed game is a game that requires the player to find his own path with obstacles in the form of complex, sinuous lines, and has a lot of impasses. In some games the mazes are also added NPC or ¬ a Non-playable Character as an enemy who will chase the player, for example, there is the game Bomberman. NPC enemies in the game Bomberman is required to recognize the surrounding circumstances, such as the existence of a wall, obstacle or find players. In such cases, then the NPC enemy implement an algorithm in finding players, namely the pathfinding algorithm. In the game Bomberman, the pathfinding algorithm is applied to enemies in pursuit of players is algorithm HPA * (star) [1]. But in the implementation, this algorithm does not run perfectly, because if a way were found covered by a dynamic barrier and position the start and goal is found in the cluster that is different then the enemy will stop moving until the roads open.

One of the pathfinding algorithms that can be used in the search for and recognize it, is a Jump Point Search algorithm (JPS). JPS is a development of algorithm algorithmA * developed by Daniel Harabor and Alban Grastien in 2011. This algorithm is said to be competitive and the algorithm is faster than HPA *, simple but effective, maintaining an optimality but it does not require additional memory, and very quickly but do not require preprocessing [2]. Application of algorithm of JPS in-game maze applied to identify the circumstances surrounding of the NPC so as to seek the best path in the pursuit of the player. JPS algorithm is done to speed up the process of the search path and reduce memory usage for the open and closed lists in the Save nodes on the A * algorithm. In the JPS storage nodes will be trimmed to just a few that referred to as jump points.

Therefore, in this study, the Jump Point Search algorithm will be used to implement in NPC as an AI in recognize the circumstances surrounding so as to determine the shortest path that can be traversed in the pursuit of the enemy. This algorithm was chosen due to the characteristics of the problem see on NPC enemies game maze equals problems in pathfinding.

2. ANALYSIS and DESIGN of MODEL IMPLEMENTATION

To see how to work the Jump Point Search algorithm in the maze of games then here is the analysis of the game labyrinth, artificial intelligence and the Jump Point Search algorithm itself.

2.1. The Game Maze

A game is a program designed in such a way where players interact with the artificial conflict which has rules and have the output [3]. The game has various kinds of type, one type of game this is the game maze. Maze game is usually a game that requires the player in looking for a way out of the maze game there are a few that require players complete a goal in the maze. The maze of streets which have undertaken the complicated trails, meandering and have a lot of impasses. An example of a maze game is still known to date is the game Bomberman.



Figure 1. Bomberman Game

2.2. Artificial Intelligence

Artificial Intelligence is often translated as artificial intelligence or artificial intelligence. The definition of AI itself varies depending on the point of view of experts each.According to Sarbjeet Singh and Sukhvinder Singh [4], AI is a field of study that is based on an intelligent thought that can be considered as a form of computation. They say that there are two important issues to be aware of are:

1. Knowledge representation

It has been long known that languages and models used to represent the reality of having a deep impact on a person's understanding of the reality itself. When people think about a particular system, they form a mental model of the system and make them know the reality of the system. This fact caused them to be able to make a prediction or inference from a system. But when a model doesn't quite fit in with the original problem, then the discovery of reality and the ability to predict becomes very difficult.

In Artificial Intelligence, a significant effort has been made in the development of the language used to represent knowledge appropriately. With AI, the engineers and the computer scientists were able to create a machine that can replace humans in carrying out the work hazards. For example, a police robot that can handle an active bomb. With the AI too, they can predict the possibilities that can occur on a system, such as the approximate reach spacecraft to the moon.

2. Knowledge manipulation

Many of the problems experienced by humans are not fully understandable. Understanding that half is reflected in the fact that a rigid algorithmic solutions could not be applied. On the contrary, the concept of search is used to solve the problem. When the search is used to crawl every space solution, it can be said it is very tiring. Therefore, if someone wants to find a solution to a problem, then it should selectively explore the search space of the problem.

One example is the search heuristics. With the heuristic search, the search results from the knowledge obtained from the special issue-universal search. The success of heuristics has caused the application of AI techniques focus commonly used for certain problem domains. This led to the development of an expert system capable of delivering a better reasoning in narrow domains can be defined in fields such as medicine, mathematics, chemistry, robotics, and flight.

Troubleshooting techniques that exist in the AI are divided into four, namely [5]:

a. Searching

Searching is a great technique to resolve the issue in a way represents a problem into the state and problems of space, as well as using the search strategy to find a solution.

b. The Reasoning

The reasoning is the technique to resolve the issue in a way represents a problem into the knowledge base (knowledge base) using logic or formal language.

c. Planning

Planning is a technique to resolve the issue by way of breaking down the problem into the sub-sub problems and resolve sub-sub problems one by one.

d. Learning

Learning is a technique to resolve problems by automating the machine to find the expected rules could apply to data that are not yet known.

The selection of the techniques used in solving problems depending on the characteristics of the problem will be solved. For example the

implementation of artificial intelligence in the game which is often applied in pathfinding. The use of the most frequently used pathfinding on a game is to influence the movement and decision making that is applied on a Non-Playable Character or NPC. While the techniques for solving pathfinding in games using search techniques or searching.

2.3. The Jump Point Search Algorithm

Pathfinding in full uniform-cost grid environment is a problem often encountered in the field of Robotics and video games. There are several algorithms devoted to pathfinding, for example, there is the algorithm HPA *. But this algorithm cannot move toward a goal if the line already obtained closed by hitch's dynamic as well as the start and goal are on a different cluster [1].

The Jump Point Search algorithm or can be abbreviated algorithm JPS pathfinding algorithm is one that is. The algorithm was published in 2011 by d. Harabor and a. Grastien [2]. This algorithm is one of the developments towards A * algorithm that aims to reduce memory usage and make the search path becomes faster.

Notation and terminology used in the algorithm of JPS are as follows

1. p (x) is a start-up or initial node
2. z is a goal or destination node
3. neighbours (x) is the node around the node x
4. y of x can be achieved by passing some of the nodes in the specified direction, ycan be expressed as follows:

$$y = x + k\vec{d} \quad (1)$$

x = active node
k = jumlah *unit* langkah dari x menuju y
$\vec{d}$ = the intended direction
y = node successor of x

At JPS there is a new strategy that could accelerate the search path by means of selectively expanding into a specific folder on the grid nodes is called a jump point.

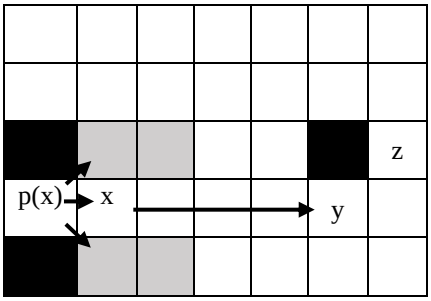


Figure 2 Jumping Horizontally in JPS

In Figure 2 it can be seen that node x expanding with the parent p (x). The intendeddirection of expansion pack node x is horizontal to the right or straight to the right.Unlike A * algorithm that incorporates all the existing nodes of x on y, then headed for JPS it is not done. JPS will immediately conduct a search to the right to find the nodes like y. If it finds the node such as y, then mark the node as a suksesor from x and gives the value of g (y) on the node.

$$g(y) = g(x) + dist(x,y) \quad (2)$$

g (x) = cost of node x
Dist (x, y) = distance from node x towards y
g (y) = cost of initial node towards y

The calculation of the value of the heuristics on each node in the JPS are also required to calculate the distance of the node with the destination node. The calculation of the value of the heuristics for the nth node in the jump point can use calculation Eucliden Distance, Manhattan Distance or Chebyshev Distance that the formula is as follows:

a. The Manhattan Distance

$$h(y) = (targetX - currentX\ abs\ |\ |\ +\ abs\ |\ currentY - targetY\ |) \quad (3)$$

h (y) = the minimum cost estimate of node y to the destination node
currentX = X position of the initial node
currentY = Y position of the initial node
targetX = X position of the destination node
targetY = Y position of the destination node

b. The Euclidean Distance

$$h(y) = \sqrt{(currentX - targetX)^2 + (currentY - targetY)^2} \quad (4)$$

h (y) = the minimum cost estimate of node y to the destination node
currentX = X position of the initial node
currentY = Y position of the initial node
targetX = X position of the destination node
targetY = Y position of the destination node

c. The Chebyshev Distance

$$h(y) = \max(|\text{-}currentX\ targetX, targetY\ |\ |\text{-}currentY\ |) \quad (5)$$

h (y) = the minimum cost estimate of node y to the destination node
currentX = X position of the initial node
currentY = Y position of the initial node
targetX = X position of the destination node
targetY = Y position of the destination node

Once found the value of the cost of a node that is from the beginning towards y in the form g (y) and the value of the heuristics of the y toward the goal, then it's necessary equations that can

determine the true value of the nodes that are in the jump point, therefore the equation in A * f (x) = g (x) + h (x) is still used in the JPS. Similarities in the JPS are as follows:

$$f(y) = g(y) + h(y) \quad (6)$$

f (y) = true value of node y
g (y) = cost of initial node towards y
h (y) = the minimum cost estimate of node y to destination node

If the search direction is determined to find a deterrent, then it is assumed the direction of the searchdoes not yield anything – anything. During the expansion of x, it can be seen that there is another node evaluation of grey – grey. On the process of jumping is also required an order that can recognize whether a nodes need to be viewed or can be ignored. This rule is known as pruning.

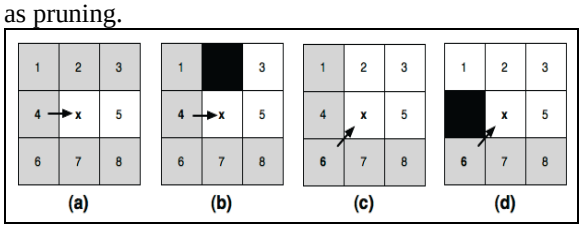


Figure 3 Pruning Process

Pruning has the sense i.e. pruning which in this case means that nodes that are not required will be ignored. Node – nodes can be ignored by way of looking at the directions used in the expansion. In pruning, there are two terms used, namely natural and forced neighbours. Natural neighbours is a node which can not be ignored, while the neighbours is forced node is forced to not ignore.

There are several conditions that must be observed, depending on how the expansion of p (x) to x is done, is done in a straight vertical or horizontal and it is done diagonally.

a. Pruning straight:

Prune all nodes n in neighbours (x), in addition to the nodes that can betargeted by using the directions of expansion which is used horizontally or vertically. Can be seen in Figure 3 (a) that a node is grayed out is ignored becausenodes can not be achieved using direction of expansion are used i.e. horizontally to the right. In Figure 3 (a) only nodes with just 5 numbers drawn, as is the direction ofexpansion can be used.

b. Pruning diagonal:

Prune all nodes n in neighbours (x), in addition to the nodes that can be targeted by using the directions of expansion used diagonally. Can be seen in Figure 3 (c) that the node is grayed out is ignored because nodes can not be achieved using direction of expansion that is used that is diagonal to the top right, while nodes with number 2 and 5 are not grayed out because the nodes are considered cannot be reached from p (x). Hence nodes that cannot be ignored in Figure 3 (c) only nodes with numbers 2, 3 and 5 only.

Node – nodes result from pruning has been done, either straight or diagonal is called the natural neighbours while the node is grayed out in Figure 3 are called as non-natural neighbours. When neighbours(x) has a node that is an obstacle, then the result of pruning will produce a forced neighbours. Forced neighbours can be recognized if n is not a natural neighbours of x which can be traversed and x has the neighbours nodes which cannot be traversed the distance is close to n.

In Figure 3 (b) it can be seen that the node with thenumber 3 is forced neighbouron pruning straightbecause there are nodes that cannot be undertaken near the black node number 3, while in Figure 3 (d) n = 1 is forced neighbour on pruning the diagonal.

JPS algorithms have in common against A* algorithm, where the nodes that will be used in determining the path stored in the memory. But the difference is the algorithm does not store all JPS nodes that become pathway toward a goal. On the explanation of the algorithm A * has previously been explained about how the storage nodes in the A* algorithm in the form of the open and closed lists. On the storage nodes on JPS closed list was removed. This algorithm will only retain a few nodes to be included in the open list.

Open list inthe JPS is nodes suksesor or y node of node x is called a jump point stored into memory. A node y be put into open list and called a jump point if it meets the following requirements:

1. The Node y is the goal node.
2. The Node y has at least one forced neighbours .
3. on diagonal, search search horizontally or its vertical bar has a node has a condition 1 or condition 2.

For more details on the conditions of 3 can be seen in Figure 4, where x expanding diagonally, in the expansion found node y entered into the open list because on expansion horizontally to the right on node y there is a goal.

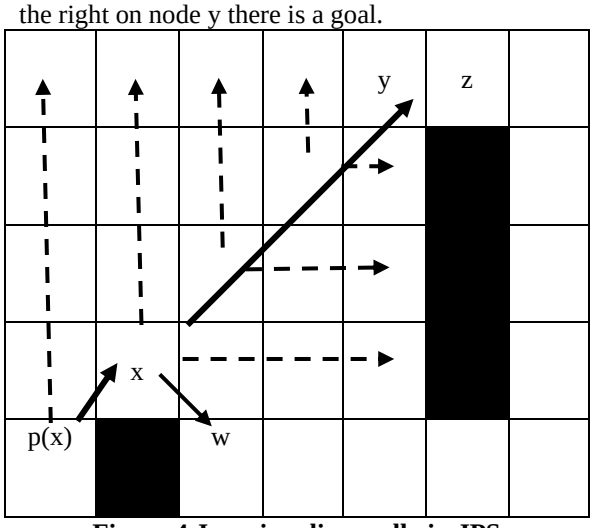


Figure 4 Jumping diagonally in JPS

In Figure 4 it can be seen that x expanding diagonally. At the beginning of the expansion done vertically or horizontally against the node x where w is forced because the neighbours cannot be achieved by the expansion of p (x). On any vertical or horizontally expansion every node visited pruning rules will be applied. If expansion vertically or horizontally is not found the node that can be inserted into the open list orencountered a barrier, then continue diagonally expansion. Can be seen in the pictures that the expansion is stopped on node y due to expansion horizontally found node z or the destination node.

3. TESTING

The following is the algorithm testing Jump Point Search conducted using heuristiccalculation of Euclidean Distance, Manhattan Distance and Chebyshev Distance. Testing conducted as many as 30 times in the shape of the maze, the position of enemies and different players.

This test is applied in 3 cases where the case can be seen in table 1.

Table 1 Table Cases

Case	Description
1	Maze size 14 grid of horizontal and vertical grid 11
2	Maze size 18 grid of horizontal and vertical grid 15
3	Maze size 24 grid of horizontal and vertical grid 21

3.1 Test case 1

After the test is done by using the rules case 1 with Jump Point Search algorithm with testing as much as 30 times, then obtained the following test results recap on table 2.

Table 2 table of results case 1

Calculation	Description	Average memory use (kb)	The number of path found	Accuracy (%)
Manhattan	Tanpa Penghalang	71.8	30	100
	Dengan Penghalang	88.1	30	100
Euclidean	Tanpa Penghalang	80.4	30	100
	Dengan Penghalang	98.4	30	100
Chebyshev	Tanpa Penghalang	93.6	30	100
	Dengan Penghalang	105.6	30	100
Average accuracy (%)				100

3.2 Test Cases 2

After the test is performed with the use case 2 rule with the Jump Point Search algorithm with testing as much as 30 times, then obtained the following test results recap on table 3.

Table 3 table of results case 2

Calculation	Description	Average memory use (kb)	The number of path found	Accuracy (%)
Manhattan	Tanpa Penghalang	145.7	30	100
	Dengan Penghalang	180.8	27	90
Euclidean	Tanpa Penghalang	247.8	30	100
	Dengan Penghalang	260.2	28	93.3
Chebyshev	Tanpa Penghalang	259.1	30	100
	Dengan Penghalang	491.7	25	83.3
Average accuracy (%)				94.4

3.3 Testing case 3

After the test is done by using the rules case 3 with Jump Point Search algorithm with testing as much as 30 times, then obtained the following test results recap on table 4.

Table 4 table of results case 3

Calculation	Description	Average memory use (kb)	The number of path found	Accuracy (%)
Manhattan Distance	Tanpa Penghalang	188.8	30	100
	Dengan Penghalang	402.2	29	96.7
Euclidean Distance	Tanpa Penghalang	274.3	30	100
	Dengan Penghalang	507.9	28	93.3
Chebyshev Distance	Tanpa Penghalang	584.3	30	100
	Dengan Penghalang	692.8	27	90
Average accuracy (%)				96.7

4. CONCLUSIONS and SUGGESTIONS

4.1. Conclusion

Based on the results of the testing and implementation of algorithm of JPS on gamelabyrinth, then it can be drawn the conclusion that the JPS algorithm (Jump Point Search) can be implemented on a maze game in recognizing environmental change that dynamic, it can be seen from the level of success that the average achieved over 90%.

4.2. Suggestions

Use the Jump Point Search algorithm uses a heuristic calculation suggested Manhattan Distance where memory usage at least between Manhattan, Euclidean and Chebyshev. The suggestions can be put forward to be the material input and consideration is to try to apply two or more enemies in the maze.

5. BIBLIOGRAPHY

[1] Dewi, K.A.E. 2014. “Implementasi HPA Star Pada Permainan bomberman (Skripsi)”. Yogyakarta: Duta Wacana Christian University.

[2] Harabor, D., and Grastien, Al. 2011.”Online Graph Pruning for Pathfinding on Grid Maps”, Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence

[3] Salen, K., and E. Zimmerman. 2003. “Rules of Play: Game Design Fundamentals”. Massachusetts London, England: The MIT Press Cambridge.

[4] Sarbjeet. S., and Sukhvinder. S. 2010. “Artificial Intelligence”. International Journal of Computer Application (0975 - 8887), Vol. 6 No. 6.

[5] Suyanto. 2007. “Artificial Intelligence”. Bandung: Informatika.