Applying Theta* in Modern Game

Phuc Tran Huu Le*, Nguyen Tam Nguyen Truong, MinSu Kim, Wonshoup So, Jae Hak Jung
Yeungnam University, Gyeongsan-si, South Korea.

*Corresponding author. Tel: +821030252106; email: phucleth@ynu.ac.kr
Manuscript submitted May 2, 2017; accepted July 14, 2017.
doi: 10.17706/jcp.13.5.527-536

Abstract: There are many commercial games using path-finding algorithm for making movements from their characters. There more development games are, the more requirements the customers demand to the game manufacturer. Therefore, in this paper, there is a comparison between A* and the post-smooth variant of A* - Theta*. Besides that, there is a demonstration among these movements of the characters made by different maps, which are taken from Dota 2 and League of Legend – the most popular games. At the end, there is a positive conclusion on the path-shape made by Theta* in order to make these commercial games meet the request of their customers.

Key words: A*, MOBA games, RTS games, Theta*.

1. Introduction

Searching for a path between two points in the certain type of map is a process of giving a series of movements for an object from one point to another, without having any obstacles/block in its path as in [1], [2]. This scientific process is applied in many fields such as: game developing, artificial intelligence, robot controlling and network. And this topic is a basic AI task in a commercial game. There are many applied path searching algorithm in games, such as, Deep-First search as in [2], Breadth-First search as in [2], Greedy search as in [2], Dijkstra, A*, variants of A*as it mentioned in as in [1].

In modern games, there are many games using these algorithms to make the characters move in the certain area as similar as the real-life character as in [3]. The movement of character becomes smoother than before and as same as the one from a real human in real life as in [4]. An optimal path must contain two points: the validity (collision free) and a path-length or the process time required for completes the search as in [5]. In main problem of path-finding in game developing, especially the role play and strategy online games, is to find the way to pass the obstacles in the certain map with the less use of computer's resource. To the current, A* has been applied very wise in many games as in [6], such as Age of Empire series, World of Warcraft series, and Dota series. Besides A*, there are many games used the different variant of A* such as Hierarchical Path-finding A*, Navigation Mesh and IDA* as in [6]. In 2007, Theta* made from A* as in [7]-[9]. The main purpose of this paper is to check how the theta*'s path looks like when it calculate on the same map as the A* applied in some games. And how the movements of characters in games are improved

In this paper, there are three sections: demonstration of A* path in certain maps of some strategy games, demonstration of Theta* path in these similar maps, and the comparison between A* and Theta* in path-length and processing time. Applying this Theta* algorithm brings many outcomes to the game developing industry as in [7]-[9].

2. Current Research

At the early time, computer game developing industry focuses on solving the pathfinding problem by using early algorithm such as depth-first search, iterative deepening, breadth-first search, Dijkstra, best-first search, A* and iterative deepening A*as in [6]. But nowadays, these games become more complexity in map solution and higher demand on characters' movement. Therefore, these new games need more efficient in time, resources using and path shape as in [6].

2.1. A* Pathfinding

A* is a generic search algorithm which is expanded from Dijkstra by applying a heuristic value as in [1]. A* explores to the closest cell and exams that to find the best optimal value and add it on the remaining list. But the path found by this search process looks "unnatural" as in [7]-[9].The found path seems to be made by the non-human object as in [8].The formula of A* is the formula (1)

$$F = G + H \tag{1}$$

where G is the distance from the start cell to the current cell and H is the estimated distance from the current cell to the goal cell.

During the process, A* algorithm firstly starts with the start node/point, then over and over check the most node/point in its promising list until the goal is found. A* algorithm follows the below pseudo code as in Fig. 1.

```
1.  Add the starting node to the open list.
2.  Repeat the following steps:
    a.  Look for the node which has the lowest
        f on the open list. Refer to this node
        as the current node.
    b.  Switch it to the closed list.
    c.  For each reachable node from the current
        node
         i.  If it is on the closed list, ignore
             it.
        ii.  If it isn't on the open list, add it
             to the open list. Make the current
             node the parent of this node. Record
             the f, g, and h value of this node.
       iii.  If it is on the open list already,
             check to see if this is a better
             path. If so, change its parent to the
             current node, and recalculate the f
             and g value.
    d.  Stop when
         i.  Add the target node to the closed
             list.
        ii.  Fail to find the target node, and the
             open list is empty.
3.  Tracing backwards from the target node to the
    starting node. That is your path.
```

Fig. 1. Pseudocode of A* as in [10].

2.2. Navigation Mesh in Games

Navigation Mesh (NavMesh) is applied as the shortest pathfinding method in 3D games because they

required polygon structure in their maps as in Fig. 2. In NavMesh, the properties of polygon object or terrain can guarantee a free-walk or a game character as long as the character which stays in the same polygon is a set of complex polygon as in [5]. NavMesh combined with A* is the best solution for the current commercial games.

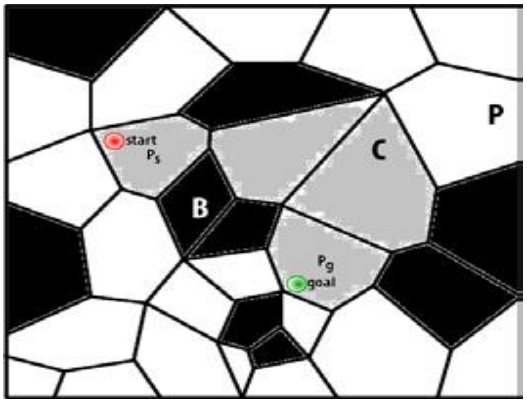


Fig. 2. Construct a NavMeshas in [5].

2.3. Theta* Pathfinding

Theta* was introduced by Alex Nash in 2007. This method is a post-smoothing process of the A* algorithm as in [7]-[9]. Basic Theta* is a variant of A* that propagate information along grid edges (to achieve a short runtime) without constraining paths to grid edges (to find any-angle paths)as in [9]. Unlike A* on visibility graphs, is not guaranteed to find true shortest paths. Basic Theta* is simple to understand and implement, fast and finds short paths as in [9]. Theta* gave a path which is shorter than A* with post-smoothed paths and Field D* (the only other version of A* we know of that propagates information along grid edges without constraining paths to grid edges) with a runtime comparable to that of A* on grids as in [9]. The pseudo code of Theta* is described as in Fig. 3.

```
1  Main()
2          g(sstart) := 0;
3      parent(sstart) := sstart;
4      open := ∅;
5      open.Insert(sstart, g(sstart) + h(sstart));
6      closed := ∅;
7      while open 6= ∅ do
8          s := open.Pop();
9          if s = sgoal then
10             return "path found";
11         closed := closed ∪ {s};
12         /* The following line is executed only by AP Theta*. */;
13         [UpdateBounds(s)];
14         foreach s'∈ nghbrsvis(s) do
15             if s'∈ closed then
16                 if s' 16 6∈ open then
17                     g(s') := ∞;
18                     parent(s') := NULL;
19                 UpdateVertex(s, s');
20     return "no path found";
21 end
22 UpdateVertex(s,s')
23     if LineOfSight(parent(s), s') then
24         /* Path 2 */
25         if g(parent(s)) + c(parent(s), s') < g(s') then
26             g(s') := g(parent(s)) + c(parent(s), s');
27             parent(s') := parent(s);
28             if s' ∈ open then
29                 open.Remove(s');
30             open.Insert(s', g(s') + h(s'));
31 else
32     /* Path 1 */
33     if g(s) + c(s, s') < g(s') then
34         g(s') := g(s) + c(s, s');
35         parent(s') := s;
36         if s'∈ open then
37             open.Remove(s');
38         open.Insert(s', g(s') + h(s'));
39 end
```

Fig. 3. Pseudocode of Theta* as in [9].

2.4. Pathfinding Algorithm in Games

A* has been applying popularly in a lot of games as in [6]. Age of Empires (AOE) series are these RTS classical real-time strategy. These games use grid map to present their maps. A* algorithm is applied to Age of Empires. Although it looks perfect theoretically, many Age of Empires players are annoyed by the terrible pathfinding as in [6]. Besides AOE, Civilization series used polygon structure for making their maps. A* is also applied to this game series. Moreover, A* is also applied to the first-person combat/ shooter games like Counter-Strike which only involves a few unit moving around at the same time.

In this paper, in order to compare the path found by A* and Theta*, Dota 2 and League of Legend as in [11], [12], two MOBA and RTS games with single character controlled by player, are used to create the map condition for these two algorithms.

3. Applying Theta* in Most Recent Games

3.1. Map Transformation

In order to demonstrate the advance of the path found by Theta*, some scenes from different games are captured. These scenes contain the game interfaces of the picked characters with the chosen start and goal points. Moreover, to have a better demonstration, maps in the game were transform from theirs types of map into the 2D grid map type. The converted information includes obstacle, position of the character (start point) and the position of the goal (destination point) as in Fig. 4.

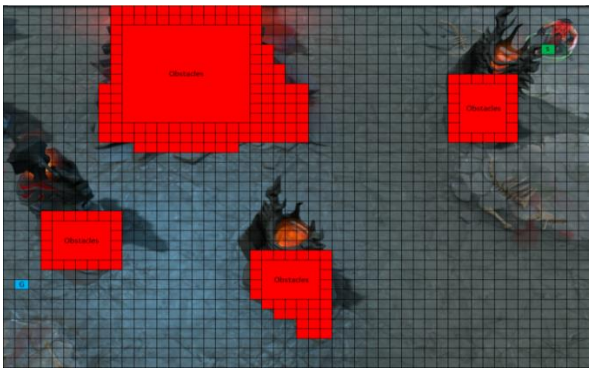


Fig. 4. Example map converted from the game screen.

The character's movements made in game are recorded and compare with the path found by A* in our display system. Seven screens are captured in the Dota 2 and League of Legend under player's view. As in Fig. 4, the light blue cell has a character "G" with the word "goal/ destination cell" is a destination cell. The green cell has a mark "S" is a start cell. There red cells in the map with the mark "obstacles" are these obstacles that can't be passed. Our converted map is a cell grid map type. There are seven converted map from the game itself to our type of map. From these maps, each map are used twice to pick two different set of goal and start position for demonstrating the path found by A* and Theta*.

3.2. Comparison between A* and Theta*'s Path

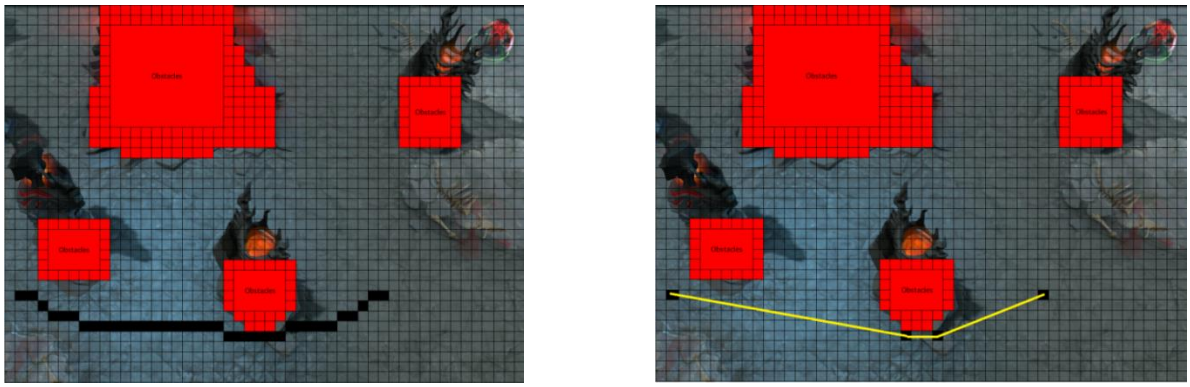
In order to compare the A*, Navigation Mesh with A* and Theta*, these found map found by section 3.1 are used to demonstrate the path shape found by A* and Theta*. As section 3.1, there are fourteen cases are used to compare the path found by two different algorithms. In this paper, these algorithms are programmed in Java and tested on Intel Core I7 with 4GB Ram.

4. Discussion

As in Fig. 5 to Fig. 18, with these visible found paths, the Theta*'s results are better than A*'s ones with or

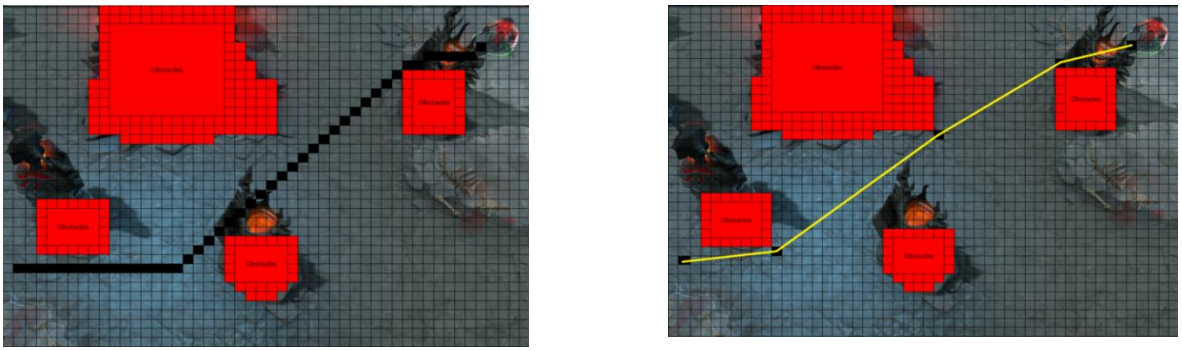
without the Navigation Mesh when the comparison comes to path-length and path shape. As Mr. Nash said path found by Theta* seems to close to the true path found by a real human (true path). These characters follow the path found by Theta* will have more natural movement, and the game is definitely considered highly in graphic. Even though, the process outputs better in path shape but there is some drawback in computation cost. As in Fig. 19 and Fig. 20**,** in all cases, the time taken by theta* are longer than A* (about 70%), while the reduction of path length is 10 times shorter.

A* have been using for many games until now as in [6], [8]. Even the most popular and famous games such as Dota 2 and League of Legends still used the technique such as Waypoint or Navigation Mesh for the game characters. The path found by these techniques are very fast but the path shapes look unnatural like A*. As the smooth-post algorithm of A*, Theta* showed the smooth and close to true path.



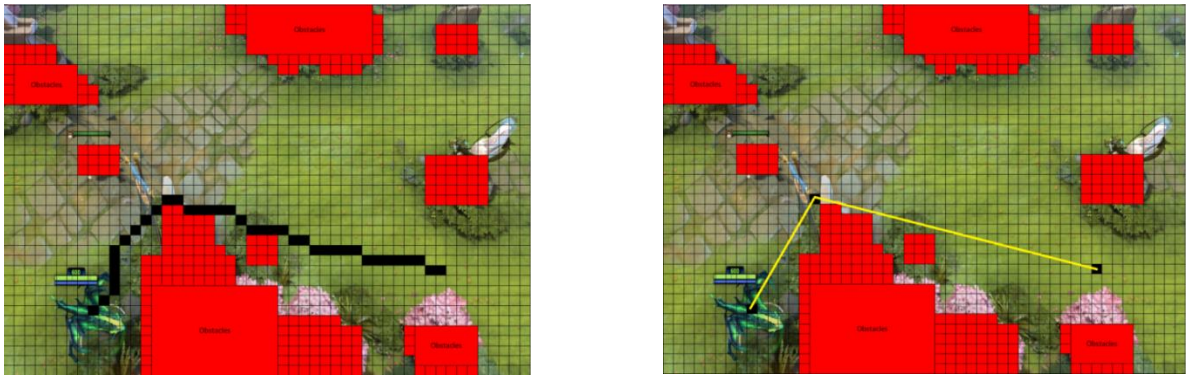
(a) (b)

Fig. 5. Path found by A*(a) and Theta* (b) in case 1.



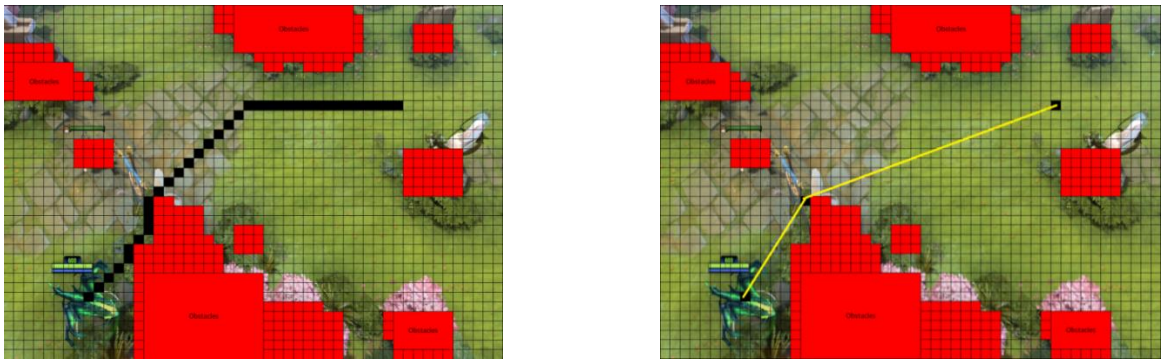
(a) (b)

Fig. 6. Path found by A*(a) and Theta* (b) in case 2.



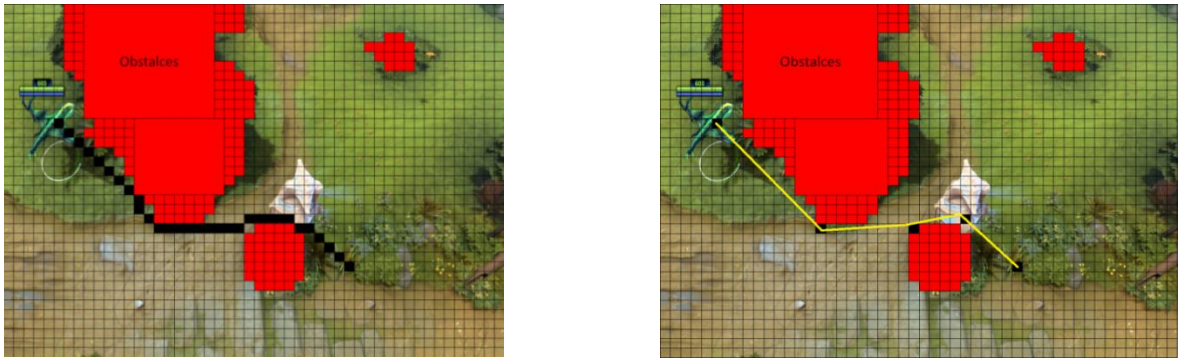
(a) (b)

Fig. 7. Path found by A*(a) and Theta* (b) in case 3.



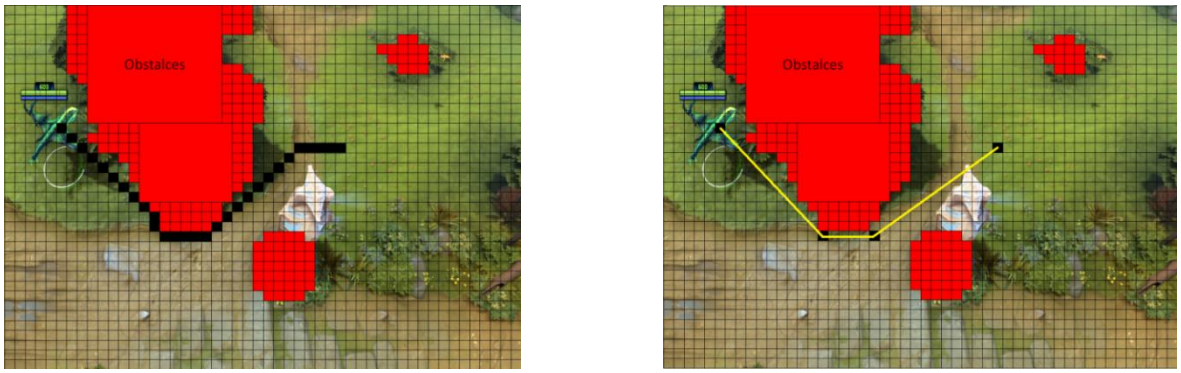
(a) (b)

Fig. 8. Path found by A*(a) and Theta* (b) in case 4.



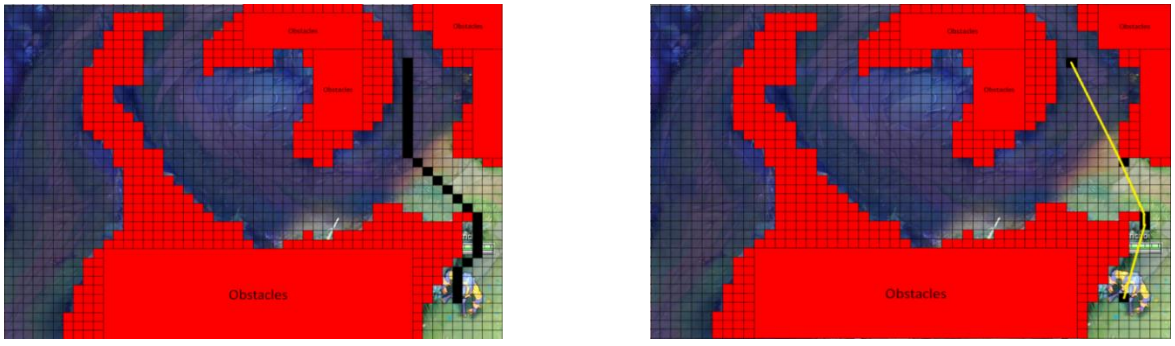
(a) (b)

Fig. 9. Path found by A*(a) and Theta* (b) in case 5.



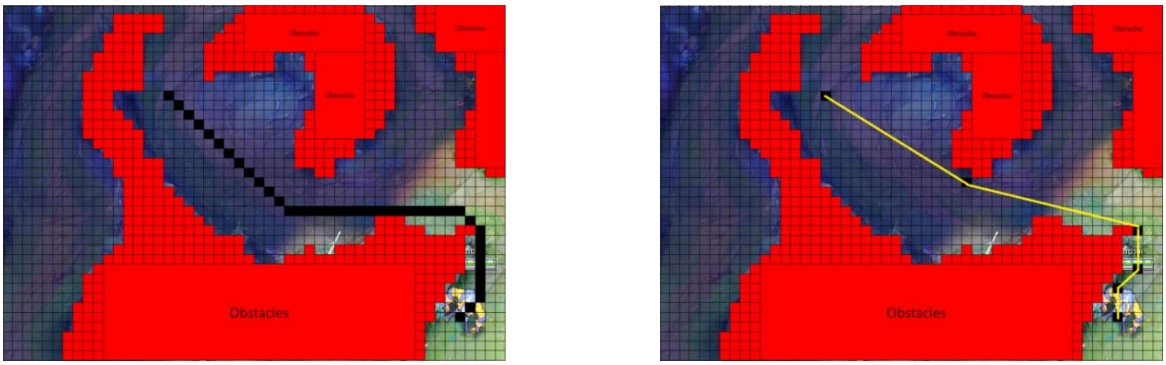
(a) (b)

Fig. 10. Path found by A*(a) and Theta* (b) in case 6.



(a) (b)

Fig. 11. Path found by A*(a) and Theta* (b) in case 7.



(a) (b)

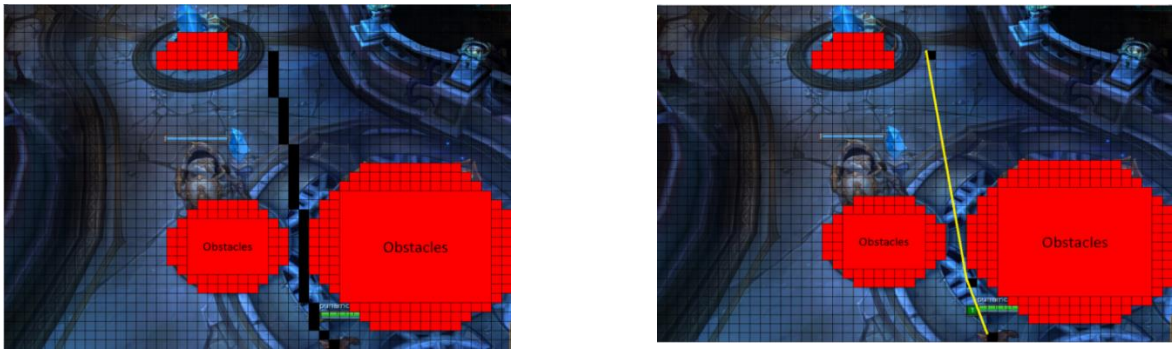
Fig. 12. Path found by A*(a) and Theta* (b) in case 8.

(a) (b)

Fig. 13. Path found by A*(a) and Theta* (b) in case 9.

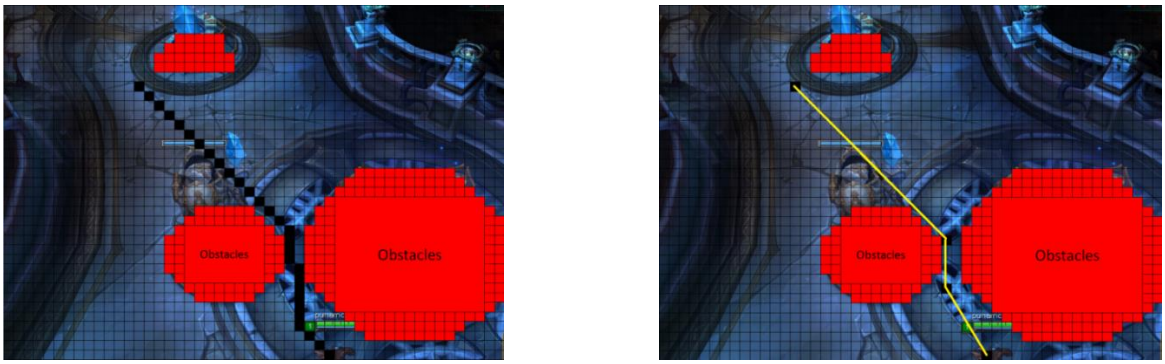
(a) (b)

Fig. 14. Path found by A*(a) and Theta* (b) in case 10.



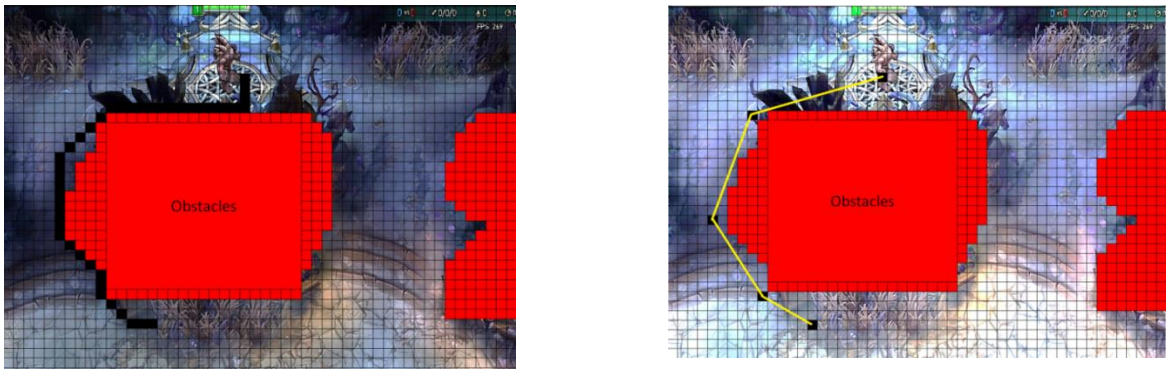
(a) (b)

Fig. 15. Path found by A*(a) and Theta* (b) in case 11.



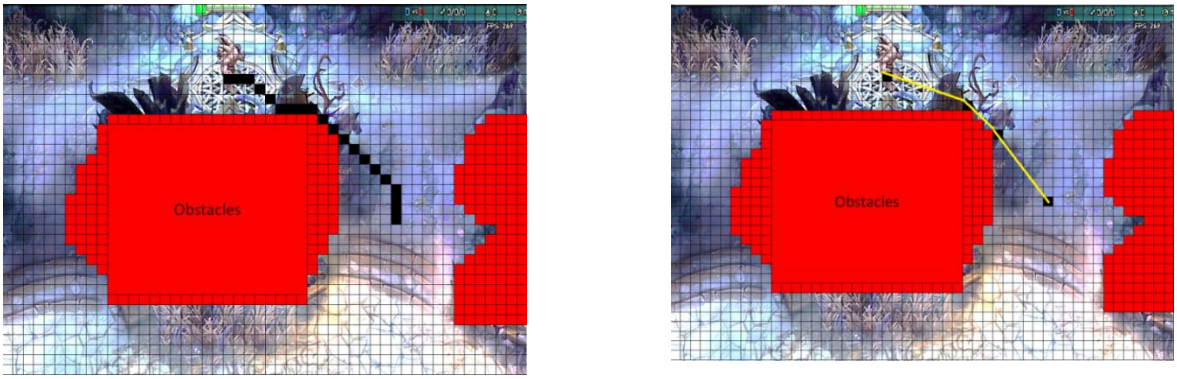
(a) (b)

Fig. 16. Path found by A*(a) and Theta* (b) in case 12.



(a) (b)

Fig. 17. Path found by A*(a) and Theta* (b) in case 13.



(a) (b)

Fig. 18. Path found by A*(a) and Theta* (b) in case 14.

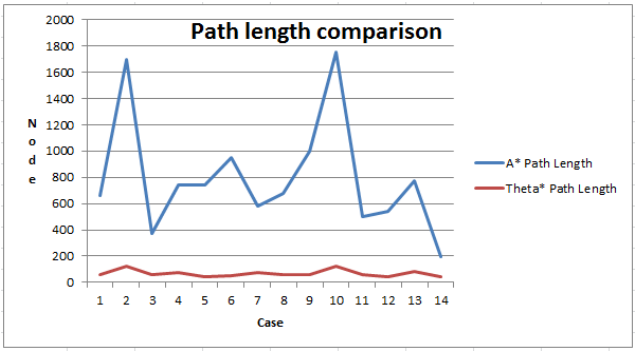


Fig. 19. Found path length comparison between A* and Theta* in 14 cases.

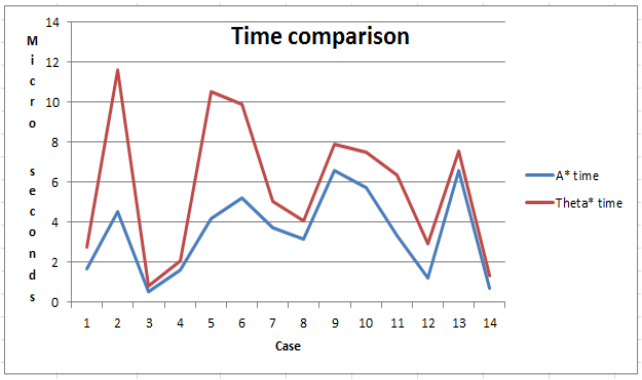


Fig. 20. Process time comparison between A* and Theta* in 14 cases.

5. Conclusion and Future Works

Nowadays, there are many developed path-finding algorithms and many optimization method. The A* is one of the most used one in many games. In this paper, there is a comparison between A*, A* with Navigation Mesh and Theta*. After these cases, Theta* performs better than A* in path shape and path length. In the other hand, A* algorithm gives the best performance in time processing. In the game industry, players over and over demand in game graphic for characters' movement and action. With the Theta*, the game design company can follow this economic demand. Nowadays, the realistic game is the development trend in this industry. Therefore applying theta* is just the matter of time, and optimizing this algorithm carry many benefits from saving resources to smoothing the path search. Even though, the drawback of Theta* is the time processing, but with some methods, the optimized Theta* can over-perform A* as in [7] – [9].

The further developing idea of this paper is to optimizing Theta* with the same cases and by different methods such as dynamic weight.

Acknowledgement

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and Ministry of Trade, Industry and Energy (MOTIE) of Republic of Korea (No.20133010011760).

References

[1] Schrijver, A. (2012). On the history of the shortest path problem. *Documenta Mathematica,* 155–167.

[2] Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (2009). *Introduction to Algorithm.* McGraw-Hill Companies, Inc.

[3] Zikky, M. (2016). Review of A* as in [A Star] navigation mesh pathfinding as the alternative of artificial intelligent for ghost agent on pacman game. *Emitter International Hournal of Engineering Technology, 4(1)*.

[4] Hu, J., Wan, W. G., & Yu, X. (2012). A pathfinding algorithm in real-time strategy game based on unity3d, Audio. *Proceedings of ICALIP on Language and Image.*

[5] Cui, X., & Shi, H. (2012). An overview of pathfinding in navigation mesh. *International Journal of Computer Science and Network Security, 12(12)*, 48-51.

[6] Cui, X., & Shi, H. (2011). A*-based path finding in modern computer games. *International Journal of Computer Science and Network Security, 11(1)*, 125-130.

[7] Nash, A., Koenig, S., & Felner, A. (2007). Theta*: Any-Angle path planning on grids. *Proceedings of the AAAI Conference on Artificial Intelligent* (pp. 1177-1183).

[8] Daniel, K., Nash, A., Koenig, S., & Felner, A. (2010). Theta*: Any-Angle path planning on grids. *Journal of Artificial Intelligence Research, 39*, 533-579.
[9] Nash, A. (2012). *Any-Angle Path Planning.* Unpublished doctor dissertation, Computer Science at the Faculty of the USC Graduate School University of Southern California.
[10] Nilsson, N. (1998). *Artificial Intelligence: A New Synthesis.* San Francisco: Morgan Kaufmann Publishers.
[11] Valve Corporation. Retrieved from Dota 2,http://www.dota2.com/international/overview/
[12] Riot Games. Retrieved from League of Legends, http://na.leagueoflegends.com/en/

Le Tran Huu Phuc was born in hochiminh city, Vietnam in 1989. In 2009, He received the bachelor degree in University of Missouri – St.Louis, in USA. His major was management in information system. From 2014 to 2016, he graduated in the master program in Yeungnam University, South Korea, majoring in computer engineering with focus fields as artificial intelligent, path-finding algorithm and computer vision. Currently, He is taking the Ph.D program in Yeungnam University.

Nguyen Tam Nguyen Truong was born in tayninh city, Vietnam in 1981. He received his education in physic from HCM City University of Pedagogy in 2005, he received the M.S. and Ph.D degree in 2011 at Yeungnam University, and became an assistant professor. His research includes growth and synthesis of semiconductor nanomaterials. His focus is on cells materials and the operation of solar cells devices using all of the common materials nanochemical and nanostructural analysis techniques. He has published over 20 papers and more than 10 conference papers.

Min Su Kim was born in Gyeosang-si, South Korea in 1980. In 2003, he received the bachelor degree in Yeungnam University. His major was chemical engineering. From 2001 to 2003, he graduated in the master program in Yeungnam University, South Korea, majoring in chemical engineering. He is taking the Ph.D. program in Yeungnam University.

Wonshoup So was born in Gyeosang-si, South Korea in 1979. In 2002, he received the bachelor degree in Yeungnam University in South Korea. He was majored in chemical engineering from 2000 to 2002, he graduated in the master's course in Yeungnam University, South Korea, majoring in chemical engineering, He is taking the Ph.D. program in Yeungnam University.

Jae Hak Jung was born in South Korea. He received his education in Yonsei University in 1988 in South Korea. His major was chemical engineering, he received the master's degree and doctorate in postech, majoring in chemical engineering. In 2006 he became a professor in Yeungnam University.