**Chapter I**

**The problem and its background**

1. **Introduction**

First person shooter games have been the craze of gamers on recent years. The genre has made a name of itself by attracting gamers of various ages to its competitive and skill based gameplay. Although the competitiveness of the games is derived from its multiplayer oriented function. The charm of the genre is still existent on the single-player part as it delivers a richer experience with its impeccable scenery and story-telling.

In a single-player play-through, players are tasked to accomplish given sets of objectives as the journey throughout the environment of the game. Most of the times, they are accompanied with an NPC (Non-Playable Character) to accompany them and fulfil their missions. Enemies encountered by the players are also considered NPCs

AI (Artificial Intelligence) is used to give actions on these NPCs. It matches its actions based on the course of actions the player uses. Because of this, human players have a certain expectation on their NPC counter parts.

Pathfinding algorithms are used to give the AIs a certain distinction and understanding on their surroundings. It lets them choose a path that will reach the destination with the least distance covered.

A\* is a [computer algorithm](https://en.wikipedia.org/wiki/Computer_algorithm) that is widely used in [pathfinding](https://en.wikipedia.org/wiki/Pathfinding) and [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal), the process of plotting an efficiently traversable path between multiple points, called nodes. Noted for its [performance](https://en.wikipedia.org/wiki/Computer_performance) and accuracy, it enjoys widespread use. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, although other work has found A\* to be superior to other approaches.

**Comparison Table of Pathfinding Algorithms**

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

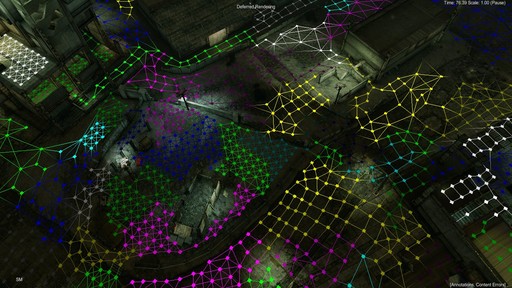
Table 1. Execution time (ms), Traversed Nodes and Length of path with 10%   
blocked node in grid map (Grid size: 64\*64 blocked node: 10%)

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

Table 2. Execution time (ms). Traversed Nodes and Length of path with 50% blocked node in grid map (Grid: 64\*64 blocked node: 50%)

1. **Statement of Problems**

The following problems have been observed in the given and current technology:

1. **Fast Tactical Pathfinding** - In combat situations, it’s important to be able to move around from cover to cover quickly and effectively as a player would. However, in most cases the shortest path isn’t the one you want. The AI controlling the enemies may decide that taking the shortest path will be the best decision, even if that path is right in front of where you are. Various waypoints are used by the AI to identify which path it should follow. Figure 1 displays the numerous paths an AI can traverse on a specific terrain.

*Figure 1 Waypoint Network*

1. **Efficient Combat Reasoning** – To make good decisions in combat, you need good quality information taken from annotations, terrain analysis, pathfinding, line of sight, and many other systems. This can quickly get inefficient though, as the need to evaluate many different options to find combat positions reliably are required. Figure 2 shows the possible positions an AI can take and those that it can’t. The white boxes show the area within the vicinity of the player while the crossed out area displays the area that are out of range.

*Figure 2 Combat Reasoning*

1. **Motion Planning** – Most games today have the AI dictating to the animation what should be done, for example when following a path. While this works most of the time, it can result in low quality movement — for example when taking the shortest path means the animation isn't as smooth as going the longer way around. Having the AI in complete control also causes bugs when there's a mismatch between what the AI wants and what the animation can provide.
2. **Objectives of the Study**

The objective of the study is to help improve the Artificial Intelligence currently used by modern games such as First Person Shooters. The use of pathfinding algorithms will help the AIs to make the best decision that will give the players a better experience in playing the game.

Specific Objectives

1. Pathfinding performance problems are often resolved using hierarchies or better heuristics. Application of A\* algorithm will be necessary to calculate a more appropriate route.
2. In improving the AI systems that reason with large quantities of information, the AI reasoning should be able to deal with this, carefully making requests for information which will prune its option down quickly, but also batching up queries efficiently.
3. Using the A\* algorithm, it will help the AI to distinguish which action the characters must do in following an appropriate path, with this, the movement of the NPCs will not deteriorate in terms of its performance.
4. **Importance of the Study**

The study seeks to benefit the following people:

1. To the ***game developers,*** this study will help them know the current issues of pathfinding Artificial Intelligence on first-person shooters and will help them advance and improve the quality of the AI.
2. To the ***gamers,*** the study will let them know the existing problems found on FPS games which will help them identify the problem.
3. To the ***students,*** that the study will let them be interested to games, as well as game development and help them expand their choices on their career.
4. To the ***other developers,*** that the study will be useful for them on understanding the problems of pathfinding algorithms and knowing the solutions for it.
5. Lastly, to the ***future researchers,*** that the study will help them on their own study and may help on furthering the study through finding out better solutions on recent game AI problems.
6. **Scope and Limitations**

Scope

The study covers the understanding on how an AI works in Call of Duty, a First Person shooter game. To improve the AI’s ability on how it performs, how it acts based on its surroundings, and identifying the path it should take.

Limitations

The study will not cover on how an AI works nor what an AI is. The study will only focus on how AI is used on Call of Duty games. Other First person shooter games will not be a part of the study, such as: “Medal of Honor”, “Battlefield”, “Halo”, “Half-Life”, etc. Though the idea or logic on how pathfinding algorithms are used on said games will be used by the researchers for reference purposes only. The study will not include how pathfinding is used on outside systems such as robotics.

1. **Definition of Terms**

**A\*.** A [computer algorithm](https://en.wikipedia.org/wiki/Computer_algorithm) that is widely used in [pathfinding](https://en.wikipedia.org/wiki/Pathfinding) and [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal), the process of plotting an efficiently traversable path between multiple points.

**Algorithm.** A self-contained step-by-step set of operations to be performed.

**Artificial Intelligence.** Used to generate [intelligent](https://en.wikipedia.org/wiki/Intelligence_(trait)) behaviors primarily in [non-player characters](https://en.wikipedia.org/wiki/Non-player_character) (NPCs), often [simulating](https://en.wikipedia.org/wiki/Simulating) human-like intelligence.

**Bugs.** An error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

**Combat Reasoning.** The ability of an AI to identify the most suitable position it can take during an encounter with the player.

**First-person shooter (games).** A [video game genre](https://en.wikipedia.org/wiki/Video_game_genre) centered on gun and projectile weapon-based combat through a [first-person perspective](https://en.wikipedia.org/wiki/First_person_(video_games)).

**Nodes.** Multiple points used to traverse between paths.

**Non-playable characters.** Characters in a game that cannot be controlled by a player.

**Pathfinding.** The plotting, by a computer application, of the shortest route between two points.

**Player.** The person the controls the character in a game.

**CHAPTER 2**

**REVIEW OF RELATED LITERATURE AND STUDIES**

In this chapter lies studies related to and supports our research*.*

**Related Systems**

**1. Foreign Systems**

In the given study, the researcher used the A\* Algorithm in order to find the shortest path to an objective through obstacles. The researcher takes into consideration four factors that can affect the AI path decision: (1) the AI starting point, (2) the destination point, (3) the obstacles, and (4) the “cost score” for each possible path. This cost score is calculated based on the series of movements that the algorithm predicts. For instance, the algorithm finds that two routes A and B have the same travel distance of 15m. Route A requires a change in elevation and 5 directional changes. Route B requires only two directional changes. From the algorithm, we can conclude that route A would yield a lower “cost score” and will be marked more desirable by the AI.

In 3D computer graphics, **hidden surface determination** (also known as **hidden surface removal** (**HSR**), **occlusion culling** (**OC**) or **visible surface determination** (**VSD**)) is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint.An Occlusion Culling method used here involves dividing the entire map area into sections. Sections are rendered individually. The sections to be rendered are determined by which sections are visible to the player camera. Although this works a bit less efficiently since sections are large and that entire sections may not even be visible. This means that even if just 1/8th of a section is visible, the entire section will still be rendered. We’d suggest that the researcher change his/her OC method (or maybe include this in our research scope.)

**2. Local Systems**

This research uses the A\* Algorithm the same way as the previous study, this time making use of it in an interface to help the visually impaired by applying “shortest path”. The user (a visually-impaired individual) has a specialized crane that contains a microprocessor, sensors, and a transmitter. They will also possess a headset that will relay instructions on how to proceed from a remote server. This server is what contains the application logic programmed in MATLAB and Microsoft’s C#.

A brief explanation of its working process is that the user’s crane will help pinpoint the current location of the user. The sensors embedded in it will detect the location of obstacles and clear waypoints. The crane then transmits these data to the server, who then uses the A\* algorithm to find quickest way to the user-defined destination (via the headset), taking into account the user’s current location, the defined destination, the waypoints, and the obstacles.

This study, however, is limited to indoor use, as per the researchers.

The way pathfinding works is that the system scans the environment. Using the sensors on the crane, the system finds all the obstacles, waypoints, and the user’s current location. The user’s desired destination is fed to the server using the headset. Once all of the data is gathered, the server gathers the data and assigns the user location and destination, and all of the waypoints and obstacles to their own individual nodes. The nodes are then mapped as a matrix and their cell locations are determined by their real world locations. Once the matrix is built, the A\* Algorithm may now come into play, finding all possible paths defined by the waypoints and avoiding obstacles. Each successful path is weighed by their “movement cost” (“F” cost, as defined by the researchers). The path with the smallest movement cost is considered the shortest path is then converted into a vocal message to be transmitted to the user’s headset.