# Heuristic Path Finding Method for Online Game Environment

By

**Jia-jia Tang**

Bachelor of Educatıon, Natıonal Taınan Teachers College, R O C , 1999

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
MATHEMATICAL, COMPUTER, AND PHYSICAL SCIENCES
(COMPUTER SCIENCE)

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

August 2010

*Your file Votre référence*
*ISBN* 978-0-494-75117-6
*Our file Notre référence*
*ISBN* 978-0-494-75117-6

## Abstract

A Heuristic Path Finding Method is developed that resolves the bottleneck for system performance in the online game industry  In order to obtain a satisfactory performance, the background processing system for pathfinding has to sacrifice either efficiency or accuracy

Under this method, designers analyze the game map structure and build area information first  The online game system will then generate path templates for in-game usage based on the map information  As the templates are being generated, the system's pathfinding Artificial Intelligence (AI) will pick a path from the templates and adjust it accordingly to produce a real path  This method improves pathfinding approaches with higher accuracy, is less time consuming and requires fewer resources from the game system  We have created a testing system for testing and evaluating pathfinding related work  With this testing system, extensive experiments have demonstrated the advantages of our method over a few known algorithms

# Table of Contents

## List of Figures

## List of Tables

## Acknowledgments

I would like to thank my supervisor, Dr Liang Chen, for his continuous guidance and support throughout my studies at the University of Northern British Columbia I also wish to express my thanks to Dr Charles Brown and Dr Jianbing Li for their constant encouragement, valuable advice and patient guidance throughout the years of my graduate studies

I want to thank General Manager Juinn Ying Wu and Department Manager Wenhsiung Lin for giving me a full view and practical aspects about internet service and database system when I worked for NetForce Inc I want to thank Department Manager Jack Jair who helped me build the entire concept of online game hosting service when I worked for Lager network technology, Inc in his department Also, I would like to thank William Lin, the Technical Department Leader at Magellan Interactive, who guided me through projects to show how AI movement is implemented under 2D and 3D engines

As well, I would like to take this opportunity to thank the University of Northern British Columbia for providing every support throughout my studies in Canada In addition, I would like to thank all my friends, Jeremy Hsieh and Vivian Feng for giving me advice and suggestions on my thesis

Last but not least, I would also like to give my sincere thanks to my parents and family members for their selfless love, endless patience and huge support Without them, I could not have reached this point

# Chapter 1 Introduction

## Pathfinding Used in The Modern World

Pathfinding is a topic that has been discussed over and over for generations At the same time, there have been many algorithms developed to solve various types of pathfinding issues In this chapter, we are first going to have a glance at how pathfinding utilities are used in the modern world, and then gradually narrow the scope to focus on the problems we want to resolve

## 1.1 Pathfindings under Different Applications

When developing a pathfinding method, there are different types of requests and accessible information one needs to consider In terms of requests, some methods dedicate to finding the target despite of potential detour, some methods seek the shortest path regardless of the time required, while other methods pursue the shortest time by lowering the importance of path length requirement In terms of accessible information, some methods provide world coordinates, others provide only a response time Simply put, there is no ultimate way to satisfy all kinds of pathfinding requests All methods are limited by the accessible information provided by the environment Here we will list the background information that researchers developing pathfinding techniques would based on

### 1.1.1 Pathfinding without Geographical Information

Cyberspace is the most known example in this category Browsing the internet requires the use of computers, cables and virtual messages With a group of computers connected by cables, every computer generates messages to establish its relative position within the group What this achieves is that the computer group can build its own virtual map specifying the physical relationships of all group members

There is no geographical information such as direction or distance required for this map The metric used to estimate the relation between computers is substituted by the time needed for sending and receiving information The most known algorithm in this category is Dijkstra's algorithm (Tim and Mark 2000)

### 1.1.2 Pathfinding without An Overview of Geographical Information

The typical type is robot pathfinding There are annual robot competitions hosted by organizations such as FIRST Robotics Competition (FIRST Robotics Competition, FIRST) Though there have been no breakthrough for a long time, robot pathfinding is still among competition options Without knowing at first its starting location, a robot needs a pathfinding Artificial Intelligence (AI) to explore its environment and find a path to the destination It tries to build a map according to the information gathered from the environment This method does not guarantee the shortest time or the shortest path to reach a target, the AI may reach the destination without completely exploring the entire map, or it may need to explore the entire map before reaching the target

The main challenges of this pathfinding method are to correctly recognize its environment and to initialize a map scratch Once these can be achieved, the AI can avoid repeating errors and reason out correct paths more efficiently

### 1.1.3 Pathfinding with An Overview of Map Information

One example of this type of pathfinding is the Global Positioning System (GPS) (Global Positioning System, National Executive Committee) Nowadays wireless communication is everywhere as satellites are being sent into space and orbiting our planet Since now we can have a clear view of the earth, using the GPS is definitely a viable solution Users send message to the server to identify their location and request for a path to the target The server processes the messages, builds the best solution according to the current map information, and then guides the users to their destination

Electronic game is another example The designer of the game sets up fantastic virtual environment for players to explore To the game engine, the map is already in existence and fully-learned The AI only read the on and off signals, unlike players who can recognize 2D image in a much quicker and more meaningful way In order to make Non-player Characters (NPC) move more smartly or humanly, the game system plans the path according to its processing maps

These types of pathfinding methods have a clear view of the map, geographic orientation, and the condition of all part of the map All these attributes assist in reaching a better result

## 1.2 Game Types and Pathfinding Requirement

Our method is focused on the online games, but it is important to explain our method further by breaking down the game category into different game types, including online game, and the reason for this will be apparent shortly Here we are going to introduce our catalog of game types and how they affect pathfinding requests In general, graphics are always the most resource intensive functionality in game design Visual effects are always the first priority, resulting in various other functions being pushed aside to be less important, and AI is an example When the rules of a game are simple, designer can place more emphasis on the story telling while simplifying the AI design without affecting the overall experience for the players

Here we will discuss five game types according to the way the AI moves We will also describe the role AI movement plays in each game type and how it is implemented and simplified in consideration of performance The five game types are

1) Pathfinding not required at all,
2) Pre-determined movements,
3) Pathfinding for few requests,
4) Pathfinding for numerous requests,
5) Pathfinding for numerous requests in an online game server

### 1.2.1 Pathfinding not Required At All

Games fall under this category include puzzle games, object-finding games and simple First Player Shooter games (FPS) Basing on common games, puzzle games such as Jigsaw365 (Figure 1-1) and Luxor (Figure 1-2) allow objects to move to destination directly without any hindrance Object-finding games such as The Mirror Mysteries (Figure 1-3) and Big City Adventure have fixed objects that require no movement at all Simple FPS games such as The Bank Of Jasker (Figure 1-4) permit objects to move up and down but in the same spots These limitations restrict players' movements in the game and involve high repetition in user control Some games of this type introduced attributes such as timing or story narration to enrich game content

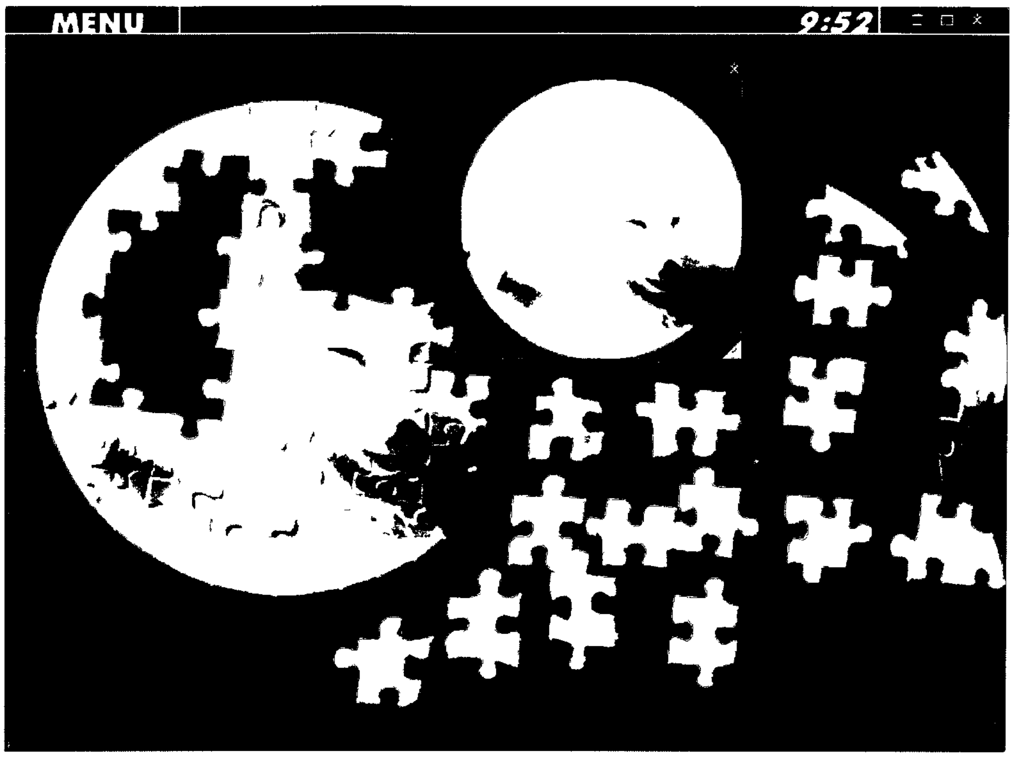


Figure 1- 1 Snapshot of Jigsaw365 (Jigsaw365, Playtonium)

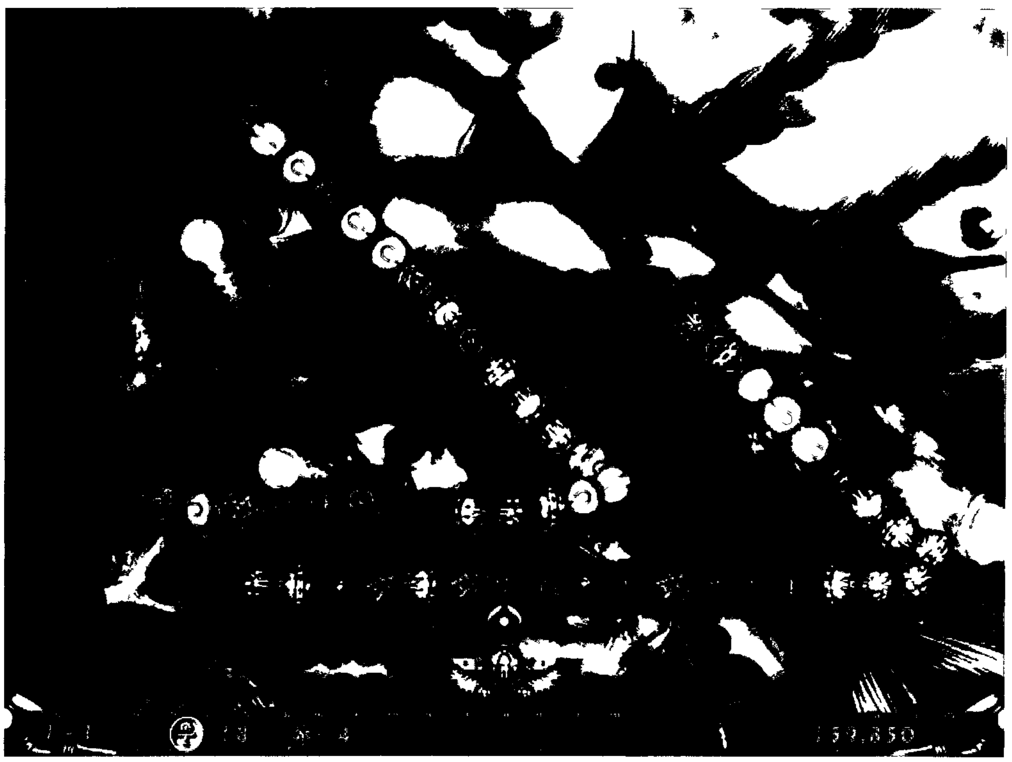


Figure 1- 2 Snapshot of Luxor 3 (Luxor 3, Big Fish Games)

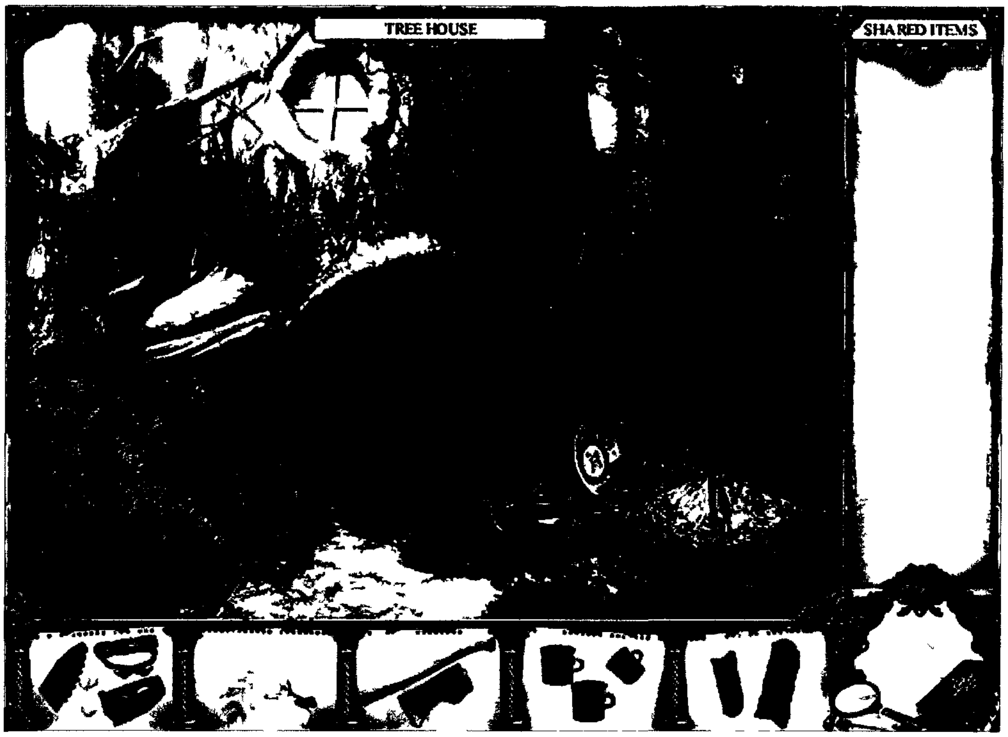


Figure 1- 3 Snapshot of Mirror Mysteries (Mirror Mysteries, Big Fish Games)

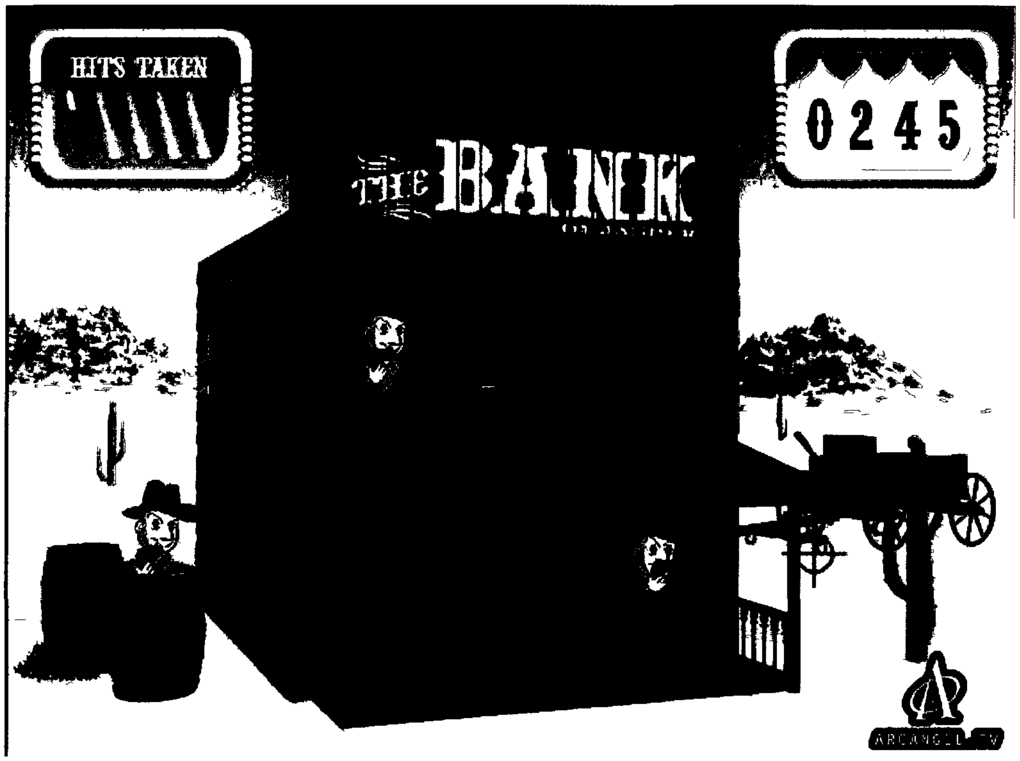


Figure 1- 4 Snapshot of The Bank of Jasper (The Bank of Jasper, Flashgames247 com)

## 1.2.2 Pre-determined Movements

In this game type, game objects automatically move around according to predetermined path This movement does not interact with the environment Some games of this type move objects according to the player's intention, but this does not involve pathfinding, the player simply chooses one of the pre-defined paths and follows it Some examples of this game type are Cake Shop (Figure 1-5) and Go-Go Gourmet (Figure 1-6) Even though players do not control the movements and cannot change the paths, the movements of the objects provide some other attributes to the game design, such as time management We categorize this as a separate game type as it does allow some degrees of variation in object movements, which adds to the richness of the game content However, this variation in movement does not require pathfinding, it is only a variation in a list of selection

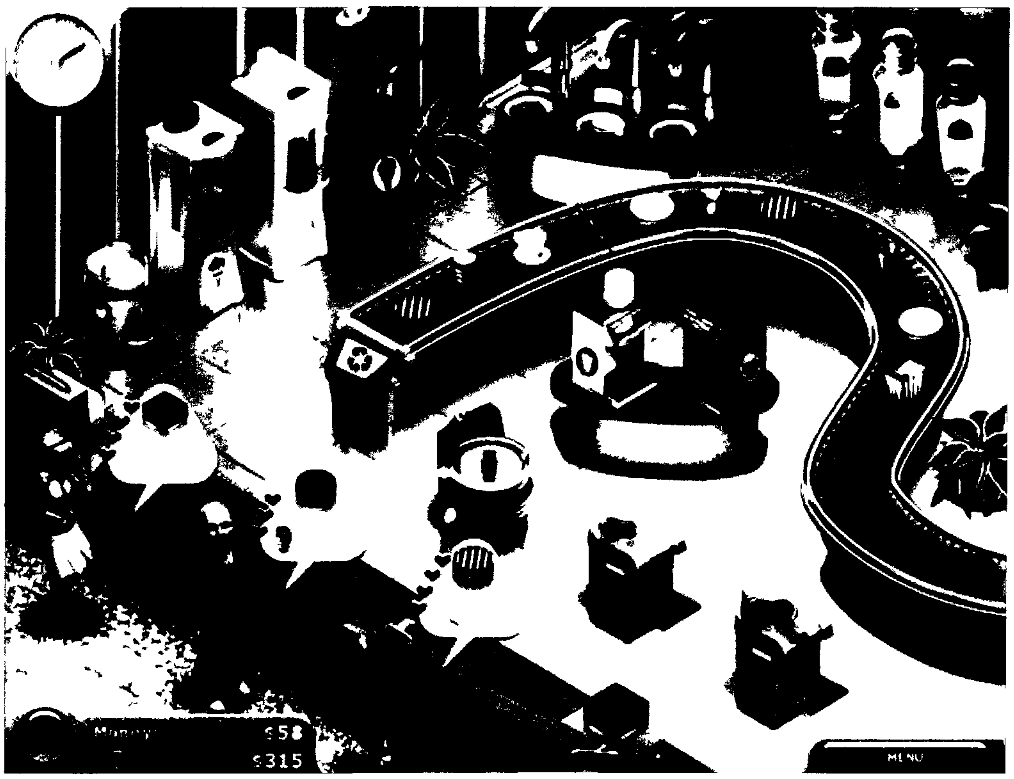


Figure 1- 5 Snapshot of Cake Shop 2 (Cake Shop 2, Big Fish Games)

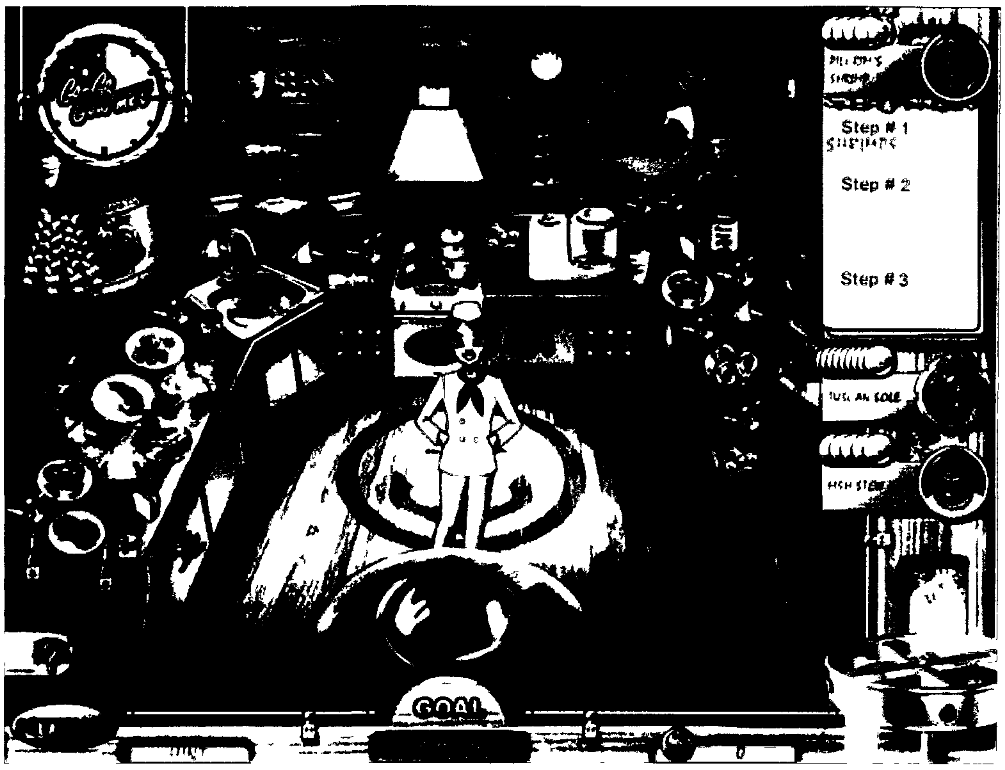


Figure 1- 6 Snapshot of Go-Go Gourmet (Go-Go Gourmet, Big Fish Games)

### 1.2.3 Pathfinding for Few Requests

Movement in this game type is more intelligent than that of the previous two categories Games of this type come with a complete concept of their own virtual world, such as direction, distance, static objects, interactive objects, friendly relationship, hostile relationship etc Based on these information, computer controlled objects interact with the players in a meaningful way, such as moving closer to actively attack, defending for the player, following player as a companion, or even just as an impartial third party Even though interactive AI can also be achieved in simpler games requiring less or no movement in the objects, such as those we mentioned earlier, the AI in this particular game type chooses its response according to the object attributes and various selections while moving freely and intelligently in the game world  This creates the illusion of time and space and promotes players to explore the game world by giving them more control flexibility in movement

Most console games belong to this game type, such as The Sims (Figure 1-7) and Neverwinter Nights (Figure 1-8)  Since a game's performance relies heavily upon the player's hardware, games only activate objects within player's viewport to increase its overall performance

Figure 1- 7 Snapshot of The Sims 2 (The Sims 2, Electronic Arts Inc )

Figure 1- 8 Snapshot of Neverwinter Nights (Neverwinter Nights, BioWare)

### 1.2.4 Pathfinding for Numerous Requests

Strategy games are a type of games that have active AI objects running outside of a player's viewport Efficiency is the first priority for game systems AI plays an important role in the game content of this type Both the computer and the player control a group of game objects The computer directs the objects of its own team to compete with the player Though the movement of the computer's team is not visible to the player, it does perform tasks almost the same way as the player does The AI team gathers game resources, defeats enemies, constructs buildings and builds its troops Even objects belonging to a player rely on AI control The game provides functionality to allow a player to queue tasks on one object, then the object automatically moves around the world to complete a series of assignments

This is a game type of heavy interactivity between AI and the players Interactivity means higher capacity and resource requirement Since the game AI is not smart enough to actually set a strategy to beat the player, there are two ways to lighten the game' s workload one is to allow the AI to cheat in game, the other is to simplify possible activities

If AI cheats in the game by calculating in advance instead of making real movement according to player's concurrent status in runtime, the game could end up being very unfair to the players or too easy to win, both cases would make the game not worth playing Therefore, this game type generally limits the degree of controls in AI's ability so it has expertise in a few specific tasks, which helps to level the abilities of AI and the players Although currently in most games of this type, the AI generally cannot beat the players, it is a very challenging game type and the only one that emphasizes the role of AI

The amount of pathfinding requests in this type is significantly more than the previous types, but the targeted system is still focused on the personal computer To allow acceptable performance, there should be limitations on AI, so as on pathfinding techniques A example of this game type is Age of Empire (Figure 1-9)

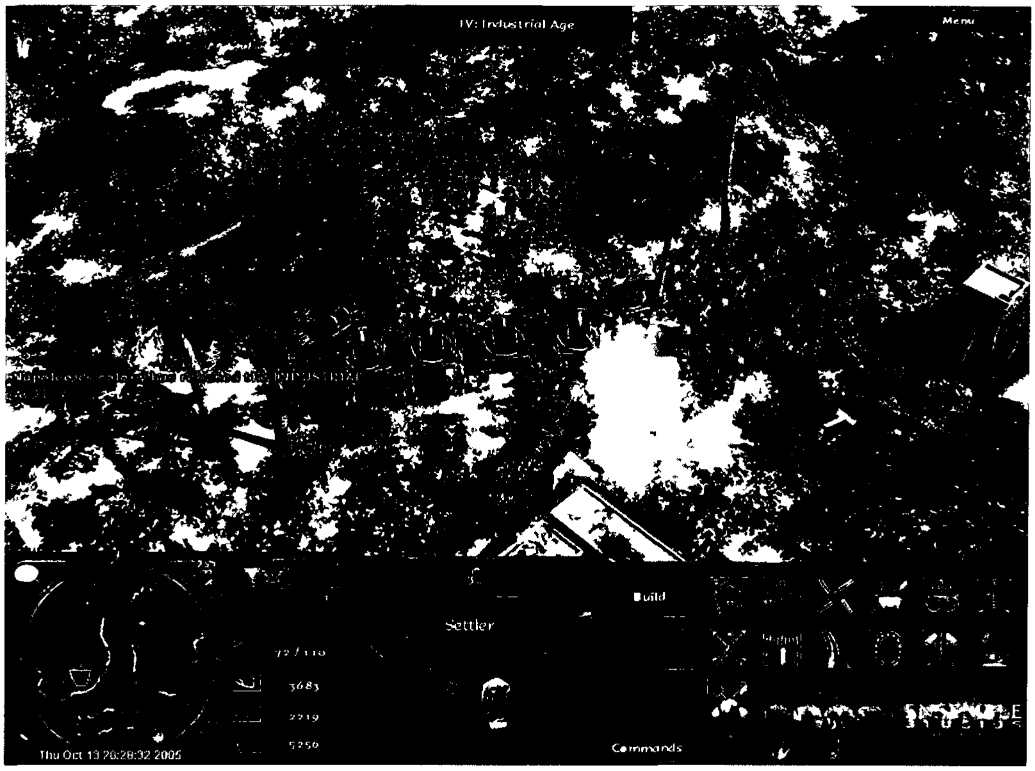


Figure 1- 9 Snapshot of Age of Empire 3 (Age of Empire 3, Microsoft Game Studios)

## 1.2.5 Pathfinding for Numerous Requests in An Online Game Server

Online games are the main topic under this category It is basically constituted by internet, hosting server and clients Game content and interaction between players are mainly located on the server How resource is used is different in online games and in console games A console game has to handle the physics of the movements, virtual effect, AI calculation and game content on its own Virtual presentation usually uses most of the system resources, game content and AI compete for the remaining For online games, on the other hand, the client side is devoted to graphic and the physics of the movements, and the game server is dedicated to content management and AI Theoretically, online game is more likely to have better AI, but, because of its multi-player attribute, management of client interaction adds another heavy load to the system Pathfinding calculation is still at risk of being sacrificed

In thıs envıronment, there are usually many players loggıng on to a host server concurrently, and the objects around each player are actıvated when any of the onlıne players approach There ıs nowhere to secretly save system resource To keep performance wıthın acceptable range, onlıne game servıce provıders can eıther upgrade hardware to ıncrease ıts capabılıty, or to optımıze system structure to meet the large amount of requests Examples of thıs game type are World of Warcraft (Fıgure 1-10) and Lıneage (Fıgure 1-11)

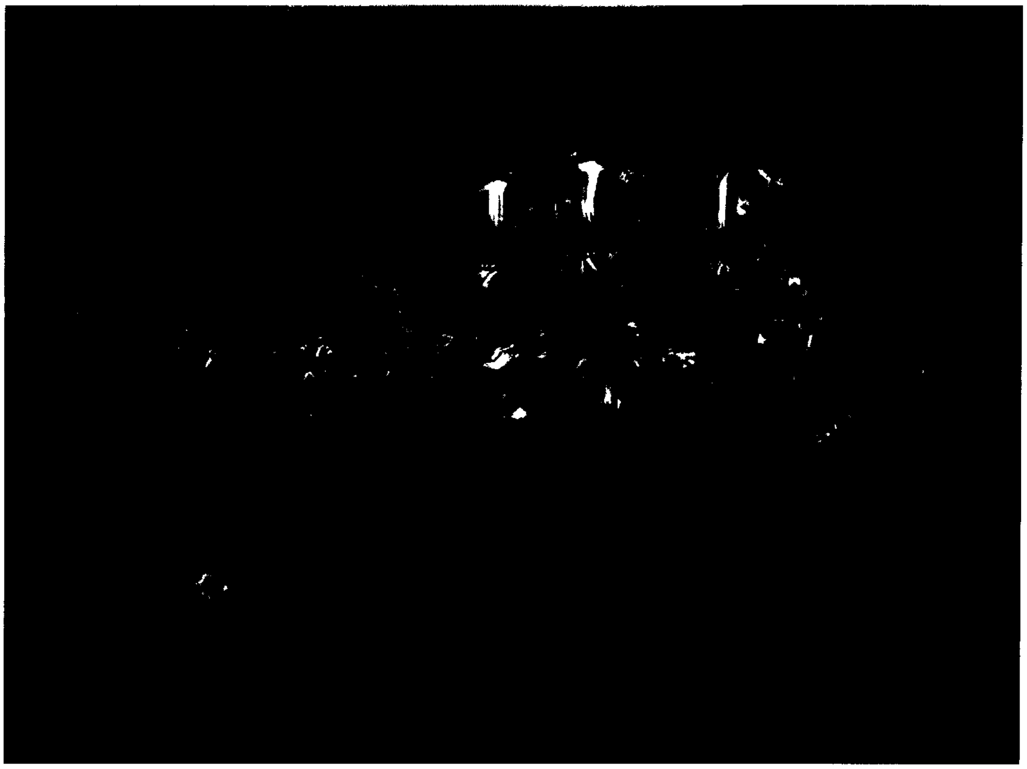


Fıgure 1- 10 Snapshot of World of Warcraft (World of Warcraft, Blızzard Entertaınment)

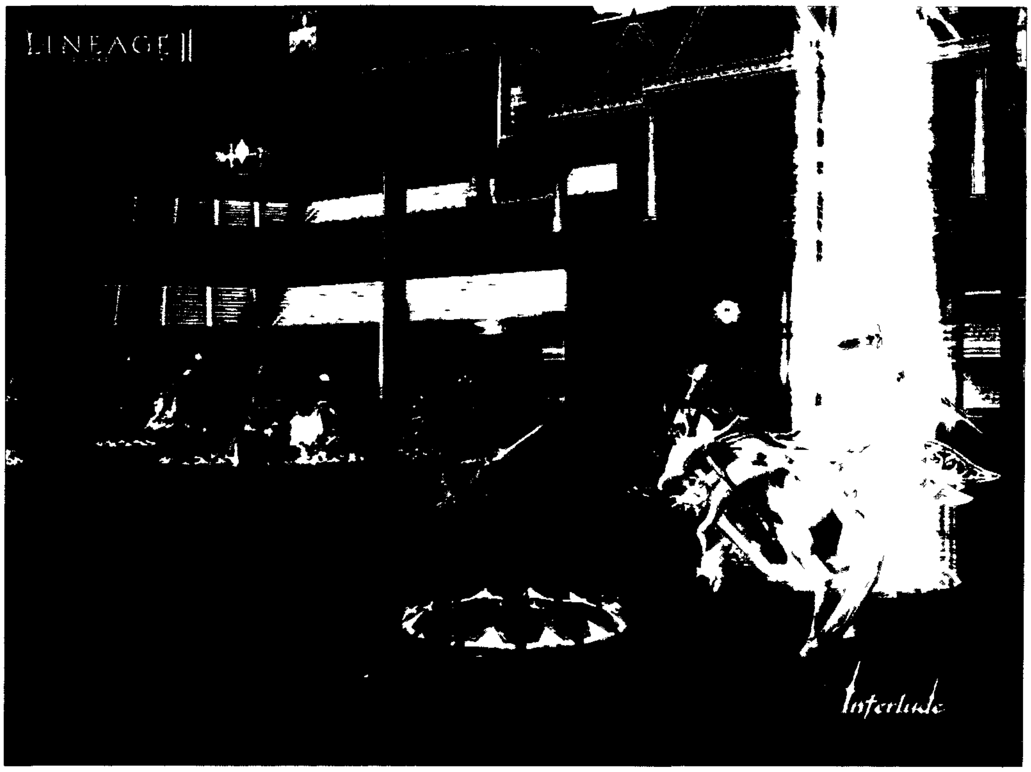


Figure 1- 11 Snapshot of Lineage 2 (Lineage, NCSoft)

## 1.3 Motivation and Contribution of This Thesis

### 1.3.1 Motivation

There are various pathfinding methods already in existence They are devoted to make the AI smarter at every movement decision, and, in order to meet quality requirement, the AI takes time to "think" in order to process all the requests simultaneously But there is no such time for online game With a wide accessibility via the Internet, instant response and massive requests processing are the new demands for game performances

In many well developed online games where players are tempted by fantastic effects and abundant contents, we can still observe problems caused by poor pathfinding approaches, such as detouring, being stuck and going through solid objects In gaming industry, an excuse for adopting such poor pathfinding approaches always rooted from the tradeoff between the good solutions and computing resources There is an apparent dilemma between the good solutions and the computing time for finding the good solutions This thesis attempts to find an approach such that good solution and acceptable online time cost can be achieved simultaneously

### 1.3.2 Contribution

Our method is designed to resolve a massive amount of requests within a limited amount of time Therefore, our theory focuses on a pathfinding method that is equipped with the map information Time is in fact a challenge as computers only processes binary information, which is good for massive simple checking, but not for more advanced processing

Instantaneous response is a very important attribute to game performance With numerous requests, system resource will hardly afford to allow AI making the best decision at all times The game system maintainer can choose to either expand hardware resource or to sacrifice AI accuracy Game producers inevitably ponder between performance and accuracy when considering pushing a product from console to an online platform

Heuristic Path Finding Approach is a solution to massive pathfinding requests that need instant response It makes good use of the preparation phase by analyzing the map structure and creating optimized path templates before the service goes online At runtime, the system knows the possible connection between the start and the ending location as soon as requests are sent in by referring back to the path template, AI can determine that there is no valid path in between immediately if there is no corresponding path template Our method groups together the locations as areas and builds paths in advance, so that it saves a lot of time from re-performing the search for each single request It minimizes uncertainty of possible path, simplifies the work of runtime pathfinding and lowers the usage of system resource

The remaining chapters of this thesis are organized as follows In the next chapter, Overview of Current Pathfinding Approach, we introduce 3 pathfinding methods which are popular in game development Chapter 3 describes how Heuristic Path Finding Method works and its design details In Chapter 4, Implementation Details and Subjects of Measurement describes the design of the testing system we created to test the various pathfinding methods, and discusses the major indicators used in the experiment to compare the methods Chapter 5 demonstrates our experiments and presents the results and the accompanying analysis Finally, the conclusions and discussions of future work are provided in Chapter 6

# Chapter 2 Overview of current pathfinding approach

A game map design highly affects pathfinding performance The reason is that generally, game designers create virtual world by decorating it with many objects such as plants, swamps, fields, buildings etc , and would expect players to get immersed into this fantasy Technical designers tend to unify all objects and set them into a certain kind of order Similar to a driver who needs a trip plan to reach his destination, the AI of a game needs a game map in order to analyze its surroundings to build a path plan A coordinate system is the most straightforward reference to AI movement in that it provides direction and measurement, and is implemented as the foundation of the game environment

Various pathfinding methods and theories exist, and some of them are very comprehensive Comprehensive method naturally demands more system resource Taking this into consideration, one needs to realize that in a virtual game world, there are so many details to plan and manage, and pathfinding is only a minor one Without a doubt, all game designers want to create a vivid game world by using good pathfinding technique, but there is only limited resource on hand Therefore, instead of using the ultimate solution, game designers usually choose less complicated methods but are just sufficient enough to meet the requirement The following discussion introduces three of the most popular methods A* pathfinding algorithm, waypoint and navigation mesh

A game map is generally designed in coordinate system with integer values The coordinate system is straightforward to human and at the same time fits the computer's functionality The reasons to develop game maps with integer values are 1) floating-point arithmetic is much slower than integer computation, and 2) monitors used to present game content could not present the partial of one pixel Under this guild line, there are only eight directions in maximum for one object to move in a two-dimensional game world

## 2.1 A* Algorithm

A* algorithm was proposed in 1968 (Hart, Nilsson, 2003), as a type of heuristic pathfinding method It develops a path by setting current position on the surrounded position which carries lowest evaluating value until reaching the destination The evaluating value is generated by the following function

$$f(n) = g(n) + h(n)$$

g(n) is a path-cost function, which is the cost from the starting position to the current position In game development, the cost represents not only the distance in between, but also some game environmental attributes, such as effort required for climbing a hill, hostile rate, friction of the ground, etc In this thesis, only distance is taken into consider

h(n) is an admissible "heuristic estimate" of the distance from current position to the goal This function provides directional information to guide path toward the goal There are several heuristic methods introduced on website Amit's A* Page (A-star-trap, Amit's A* Pages) In game development, this function is developed in simple way because avoiding complicate computation without affecting presentation is the major concept to save resources A game system won't need any pathfinding method if its game world contains no obstacles If obstacles do exists in a game world, the heuristic estimate applied with fixed tuning value could never fit all game maps In this thesis, heuristic estimate is the sum up of the vertical distance and the horizontal distance between current position and the goal

In addition to evaluating function, A* algorithm carries two lists  open list and close list  Open list contains all visited positions which are assigned with a cost value, a heuristic estimate and its previous step  One position would be moved out of the open list into the close list when all of its surrounding positions are assigned with evaluating values

Pathfinding function sets current position on starting point at the beginning and puts all surrounded position into open list with their cost, heuristic estimate and previous step generated according to current position  After all surrounded positions being evaluated, pathfinding function sets the current position on the position stored in open list which carries lowest evaluating value  This process repeats until the current position reaches the goal  The final path is built by tracing back previous steps from the goal toward the starting position

Two-dimensional game maps developed in coordinate system with integer values look like grid-maps  We will use the term “A* grid” to represent the pathfinding method implemented in an A* algorithm based on a pure coordinate system

## Feature

Although the A* grid may not be the ultimate solution for game pathfinding, it is definitely one of the more popular ones  Dating back to older games, game maps were small and assembled by tiles  The viewport of the game was similar to looking at the world from the sky (Fig  2-1)  While restricted by RAM memory and the speed of the computer, the map structure was expanded to contain game object information  Therefore pathfinding AI was allowed to plan an accessible path by referencing map data

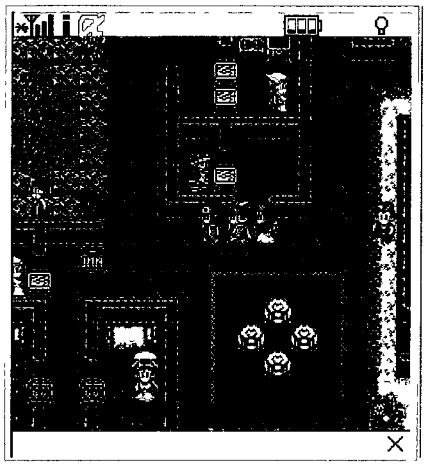


Figure 2- 1 Snapshot of Dragon Warrior III (Dragon Warrior III, Enix Corporation)

In theory, vertexes are what A* grid is based on to evaluate pathfinding details Starting with a vertex on a simple two dimensional map, movement in all directions should be allowed However, when it is implemented on a coordinate system, each step has only 8 possible directions, moving upward, downward, rightward, leftward, and towards the four angles diagonally It would take the object the same amount of time to move horizontally and vertically, and the same amount of time to move diagonally

The way an A* grid works is very similar to a man walking in the dark Eyesight is no longer enough to lead his direction, he has to move around with both hands stretched out to make sure that next step is clear of obstacles and is safe Pathfinding AI implemented in an A* grid selects one optimal step from surrounding grids A path is built by tireless grid-checking from the starting point to the destination

## Advantage

An A* grid method uses a distance and moving cost to evaluate the next step (A* algorithm, Wikipedia) In order to achieve the best next step, the AI sums up the cost of moving from the starting point to the current location and adds it to the distance between the current location and the destination The distance estimation highly depends on the coordinate system which not only contains the distance measurement but also the direction Compared to Dijkstra's algorithm (Dijkstra's algorithm, Wikipedia), which is generally used in Internet routing services, the A* algorithm avoids most unnecessary checking and saves time

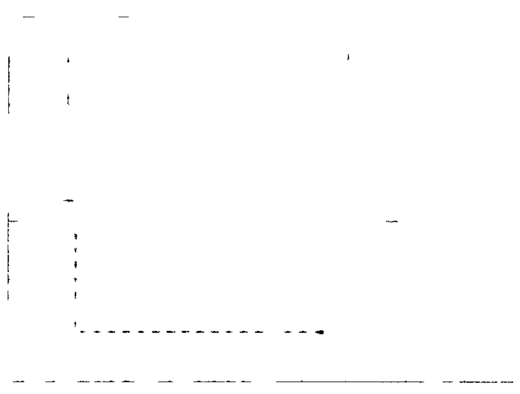


Figure 2- 2 A* algorithm used on simple map

This image shows the path built by A* algorithm and the grids that were checked during pathfinding process (A-star-trap, Amit's A* Pages)

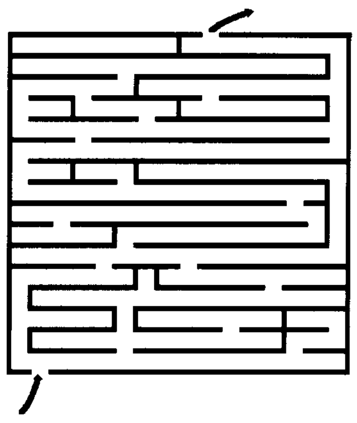


Figure 2- 3 A maze like map (Maze, Wesg ca)

On a simple map (Figure 2-2), directional movement helps greatly with the pathfinding On the other hand, with a more complicated map (Figure 2-3), it inevitably takes the AI more time to resolve a task Since responsiveness is the highest priority to game system, the designer usually sets a limit on the length of time for processing before aborting the pathfinding task to prevent the task from occupying the system resources too long When pathfinding takes excessive amount of time which might force the game system to abort the process, the NPC might appear as silly by not reacting much unless the player does something to it Other than the aborting technique, the game designer can also solve this over-time issue fundamentally by simplifying the game map After all, the game world is an imagination, its content is definitely negotiable, unlike the paths specified by the GPSs that cannot be changed freely since they have to be based on actual roads and highways

The A* grid methods is still popular in the game industry As game content is getting more complicated and as various pathfinding methods are being developed, the A* algorithm is usually where most pathfinding methods are derived from

## Disadvantage

A disadvantage is that there are only eight directions allowed for each step in an A* grid method If one game object has to move over a long distance at a direction other than those allowed, it usually ends up moving in a zigzag motion and wastes time when turning This problem is not obvious if a map is small and less complicated, and when each grid can represent a physical object or a relatively large region

As hardware continues to improve, the game system is now designed to show a more refined visual presentation with each grid representing a smaller area This way object should appear to not move grid by grid but point-to-point As a result, each grid, while covering a much smaller area, is no longer regarded as representing an object or location, but used more as a reference point to calculate distance

Furthermore, the measuring unit of each grid would also have decreased since each grid now covers a smaller area To produce a map containing the same amount of objects and space as the older hardware, a designer using the newer hardware would require more grid and therefore more resources

More grid and more resources mean bigger game map, which is exactly what the A*grid method is not suitable for

## 2.2 Waypoint Method

When movements are without a grid, the moving direction becomes more important for the object at each step The Waypoint Method (Waypoint Method, Wikipedia) uses links and nodes as additional information to accelerate the A* pathfinding work

Waypoint Method is very similar to the A* algorithm on a high level The object starts its movement from the closest node, goes through nodes and links toward the node closest to the destination, and then moves from there to the target location There are two phases to validate Waypoint method (1) the design of waypoint net, and (2) the planning of a route by the pathfinding AI using the waypoint net

Waypoint Method is primarily based on the waypoint net A waypoint net is composed of the selected accessible points, which are called the waypoints, and the links between those points to form a set of paths It usually looks like a net and guides the object moving along the links to deliberate waypoints Waypoint net is created to reduce the number of possible moving direction and the amount of time to make direction decisions It is unlikely for the pathfinding request to be always located exactly on the waypoint nodes Therefore, during runtime, it is AI's responsibility to identify the nearest nodes around the starting point and the destination

After the tuned locations are identified, the pathfinding AI tries to build a path based on the links that connect two arbitrary nodes The link contains distance information between two nodes In other words, the information provided by a waypoint net include the points, the direction and the distance This is sufficient for A* algorithm to perform pathfinding Therefore, Waypoint Method is essentially a method using the A* algorithm, assisted by coordinate system and waypoint net

## Feature

Waypoint Method uses waypoint net to assist the pathfinding process How a waypoint net is designed is an important issue Unlike the A* grid method which is able to identify its grids quickly by referencing to a coordinate system, there is no simple conversion for the Waypoint Method to convert nodes from a coordinate system to a waypoint net The pathfinding AI has to scan its surrounding area for both the start and end points to identify the closest nodes

Furthermore, the A* grid uses measurement to estimate the moving cost, whereas the waypoint retrieves the distance information from links If a waypoint net is designed intensively, it may require longer processing time and occupy more memory resource than an A* Grid Method when resolve a task If the waypoint nodes are designed loosely (Figure 2-4), the waypoint may detour round an object for a longer path

The A* algorithm is implemented so that the object stops at the first occurrence it finds the destination, making it unnecessary to scan all irrelevant nodes Under the Waypoint Method, if the player is in the middle of two waypoints, the order of checking the direction may affect the outcome (Figure 2-5) If an obstacle is allowed to be dynamic, it may block the path by being on the links This can happen even ıf there ıs stıll room to bypass the obstacle (Fıgure 2-6) The qualıtıes of the waypoınt net desıgn hıghly affect the performance of the Waypoınt Method

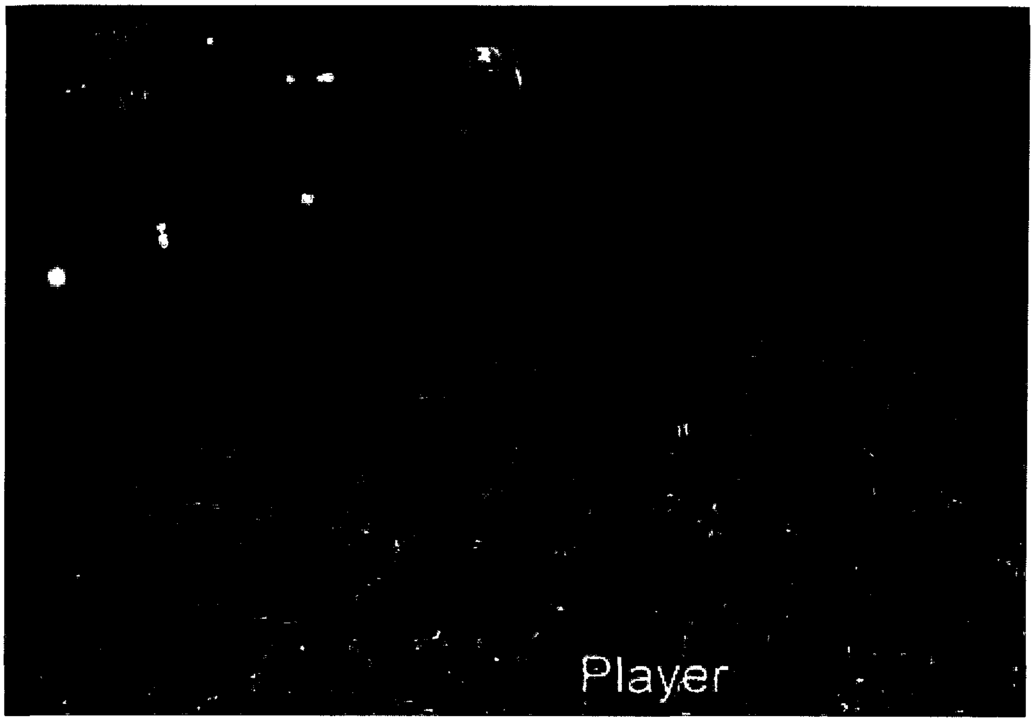


Fıgure 2- 4 An example of how Waypoınt net detours NPC

Mob's path would be dısturbed by Waypoınt nodes (, red path), as the green path ıs the best solutıon Orıgınal ımage comes from Neverwınter Nıghts ( Neverwınter Nıghts, BıoWare )

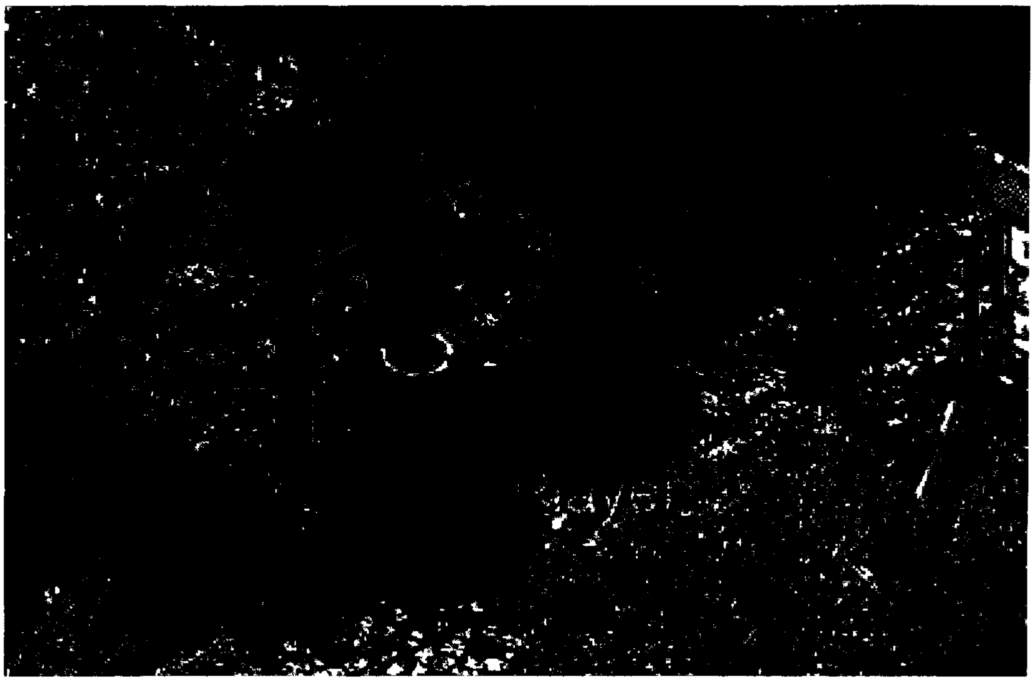


Figure 2- 5 An example of waypoint evaluation affected by player's location

AI uses the waypoint behind the NPC as start point The distance from start point to both waypoints besides the player seems the same to AI AI would choose the waypoint that was evaluated first while the waypoint on player's right side is more optimal Original image comes from Prince of Qin ( Prince of Qin, Beijing Object Online Technology Corporation )

Figure 2- 6 AI blocked at the bottleneck of Waypoint

If a "random obstacle" is possible, an obstacle which falls on the node at the center of bridge would block the path of the Waypoint Original image comes from Baldur's Gate II ( Baldur's Gate II, BioWare )

### Advantage

Waypoint Method is very efficient on a map with less passable area, like the streets in a metropolitan area, or a complicated map (Figure 2-3) The AI only makes the decision when the object reaches a node, instead of at every step If nodes are properly designed and tuned, this method is much quicker than the A* grid in runtime and generates human-like movement The pathfinding AI can find all required data from the waypoint net Other than identifying the starting and ending nodes, it is not necessary to scan an entire map to build a path

### Disadvantage

There is an essential difference from a game map to real world map The major composition of a game map is a large passable field whereas in the real world map it is mostly comprised of obstacles When we reach a large passable area, such as a forest, we tend to follow the path built by others to ensure that we will not get lost But when we play a game and our characters confront monsters in virtual forest, we expect monsters to move straight toward us instead of moving to the closest road first and then following the path toward us The game map may contain much more accessible area, not only for the convenience of design, but also for higher flexibility and easier control for users As the game map expands even larger nowadays, keeping the nodes dense enough without spending too much time looking for required nodes and links in runtime would be a challenge

## 2.3 Navigation Mesh

Navigation Mesh Method expands the concept of a step from strictly refined location to loosely defined area This concept was evolved around the times of the dot-com bubble (Dot-com bubble, Wikipedia) when Internet accessibility was being heavily invested in Multi-user online services became popular, and capacity was another critical issue besides visual presentation

As massively multiplayer online games became possible, game maps were also expanded for entertaining purpose Under multiplayer platform, it is impossible for the game to secretly sacrifice lower priority AI tasks to improve performance without the users noticing For pathfinding tasks on a large map, the A* grid is very slow because a large map contains larger search space Waypoint Method has similar problems on large maps as well However, as the A* algorithm has proved its reliability for years in game industry, game designers wanted to use the same algorithm As a result, instead of using another more accurate but complicated method, designers would tend to increase the system performance by reducing AI selection Navigation Mesh Method (Navigation Mesh Method, Wikipedia) is one of the newer methods invented under this concept

Navigation Mesh Method is more of a game specific pathfinding method for AI Waypoint Method is sufficient for humans as it is easy for us to bypass obstacles with a pathfinding result on hand For the AI it is a different story AI does not "see" things It never chooses shortcuts other than the ones given Therefore, the Navigation Mesh is designed to give the AI freedom to choose any location within a chosen area, and will not make the path different even if the AI chooses another location in the same mesh

## Feature

Navigation Mesh Method uses a pre-processed map A designer defines convex polygons to cover entire passable areas Unlike pathfinding methods introduced previously, this method shifts most of the address checking work to boundary calculations This change would be valuable only if the game map is large enough to make the calculation cost less than address checking The game system also needs the relationship between conjunct areas At runtime, when pathfinding tasks are triggered, the game system looks for convex polygons, so called meshes, which the starting and ending points belong to It sets a start area as a checking base, picks the best candidate from all surrounding neighbors, and then sets it as a new starting area This process is repeated until reaching the ending area After a mesh path is confirmed, the game system should polish it into final movement path as the final result (Figure 2-7)

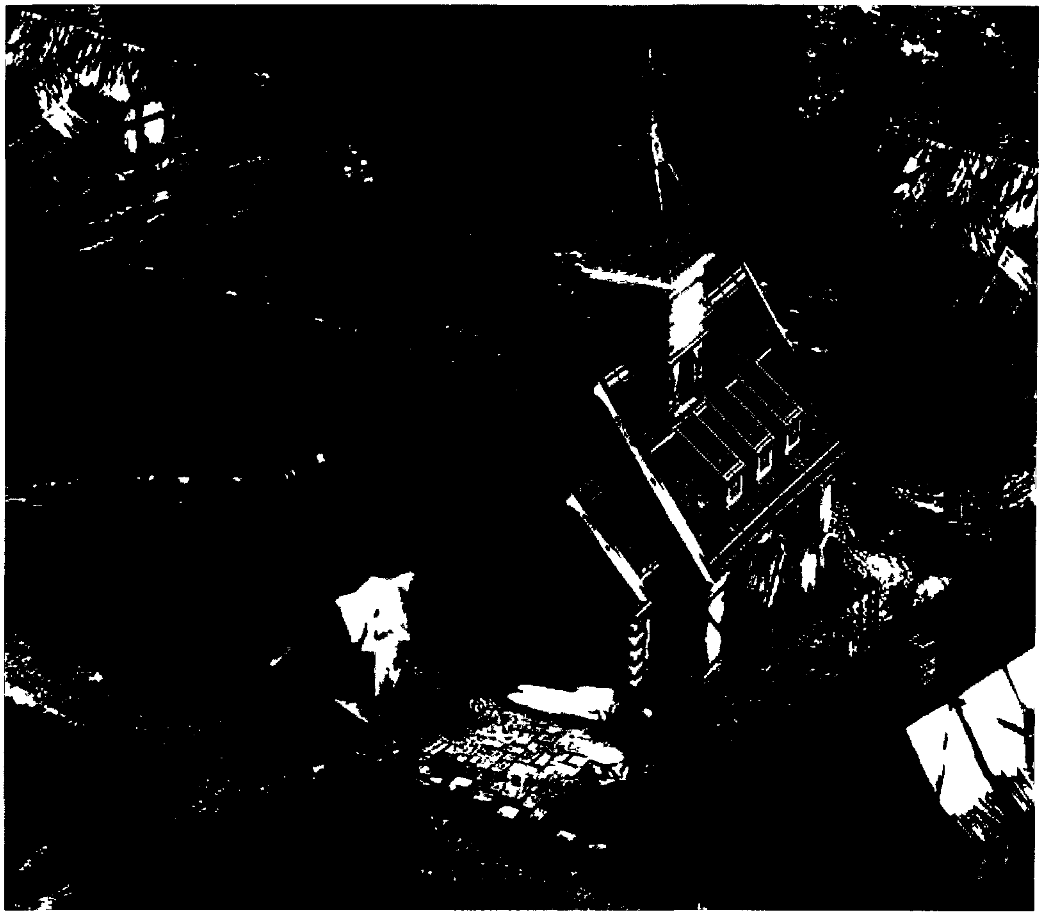


Figure 2- 7 The progress of path-finding in Navigation Mesh Original image comes from 1701 A D ( 1701 A D , Aspyr )

There are commercial products that help designers to place meshes, such as Source SDK (Source SDK, Valve Developer Community), which is developed by Valve Developer Community This tool predefines basic mesh shapes so that a designer can fill them in on a game map After the game map is completed, this tool processes the game map to merge all shapes into fewer convex polygons The designer reviews the result and can attach or detach merged polygons to assist the game AI Though the mesh generation tool hardly creates perfect maps, it does simplify the designer's work significantly

## Advantage

This method simplifies the choice of pathfinding and indirectly increases pathfinding performance All resource saved by less selection is invested into geometry calculation It is excellent for pathfinding on large maps

## Disadvantage

Navigation Mesh Method still uses the A* algorithm As an attribute of this algorithm, the AI evaluates the next step by analyzing surrounding nodes For the navigation mesh, there is no fixed node within a mesh to be the basis of evaluation The game system generates one reference node for each mesh to help the evaluation process No matter how much effort the AI spent in generating an ideal reference node, there would be at least a few detours caused by unbalanced mesh design

Take Figure 2-7 as an example, the meshes on the left are quite large while meshes on the right are relatively small The AI sets reference nodes on the center of each mesh The path built by reference nodes on the left side (green path) is much longer than the path on the right (blue path) But the final path, the red line, built from the blue path, is much longer than the path built from the green line (Figure 2-8) This could be avoided by reconfirming the built path Nevertheless, reconfirmation and redoing the path doubles the AI loading The path length built during the pathfinding process of the Navigation Mesh Method is not as straightforward as that built by Waypoint Method

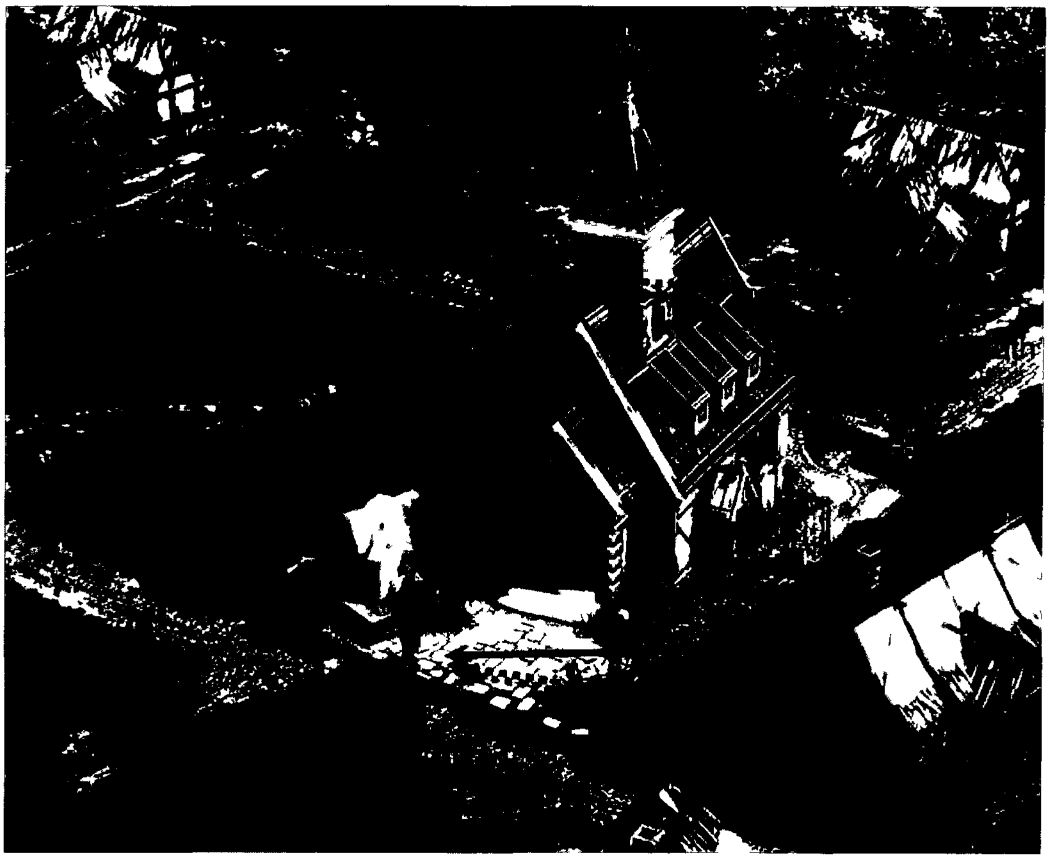


Fıgure 2- 8 Qualıty of path ıs affected by the desıgn of mesh Orıgınal ımage comes from 1701 A D ( 1701 A D , Aspyr )

Addıtıonally, the way meshes are defined greatly affect the performance of pathfindıng If the map ıs too complıcated or meshes are too small, the geometry calculatıon may consume even more system resource than address scannıng

After goıng through these popular pathfindıng methods, we have developed a better concept of what the game ındustry really needs ın pathfindıng, how ıt ıs practıced ın current avaılable products, and why they are not compatıble ın certaın condıtıons We wıll now focus on our Heurıstıc Path Fındıng Method

# Chapter 3 Heuristic Path Finding Method Approach

## 3.1 Overview of Heuristic Path Finding Method Approach

Our method targets the online hosting service Specifically, Heuristic Path Finding Method is a method that should be implemented on a server By breaking down the structure of the system and the components of a request, the Heuristic Path Finding method efficiently moves a part of the tasks out of runtime, thereby reducing processing time and increasing efficiency Many other pathfinding AI, such as those used in GPS, emphasize accuracy, however, they are highly inefficient to be applied in a game setting On the other hand, we have designed our method with efficiency as the highest priority, while minimizing any possibilities to sacrifice accuracy Under our structure, the system is very capable of dealing with massive pathfinding requests and still works effectively This is apparent especially when comparing with other methods mentioned previously over a long term basis

The process of the pathfinding work can be divided into two parts The first is preparation Heuristic Path Finding Method uses processed map information which contains ordered vertex sets and relationships of all passable areas The map data is supposed to provide minimum, well-structured and sufficient information that can be understood and processed efficiently by the game system Before the main system provides service, it loads the processed map information, analyzes and rebuilds the relationship of the passable areas According to the map information, our method builds a path prototype for each pair of two different passable areas based on A* algorithm, and stores the path template in a database system The database system is an independent system located on server side

The second phase ıs runtıme executıon After ınıtıalızatıon, the server goes onlıne Requests would be sent as paırs of startıng and endıng poınts The game system ıdentıfies correspondıng areas for each pathfindıng request and then sends a query to the database to retrıeve a matchıng path The database wıll perform a data search, and then sends a set of path template back to the game system In the end, the game system optımızes the path template to fit the request as the final path

In the followıng content, we use "database" as the abbrevıatıon of "database system" And we wıll use "maın system" to represent the major purpose of the system on server sıde If ıt ıs a game server, "maın system" represents the game system, ıf ıt ıs out testıng system, thıs term ıs an alıas of the testıng system

## 3.2 Heuristic Path Finding Approach

In thıs sectıon, we are goıng to demonstrate step by step how Heurıstıc Path Fındıng works Let's take Fıgure 3-1 as a sample game map for ıllustratıon

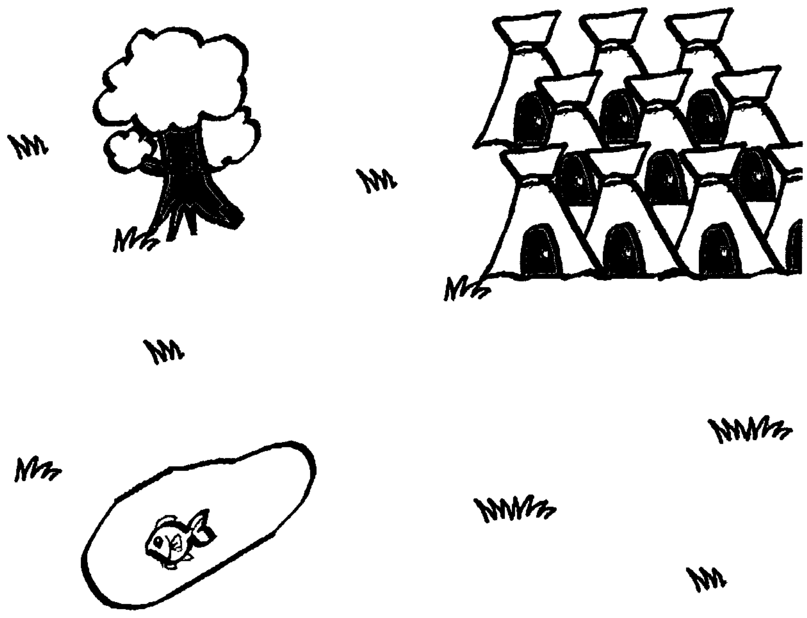


Fıgure 3- 1 Sample game map

### 3.2.1 Pre-processing

1 When we receive an online game map shown as Figure 3-1, we would rule out the area that has obstacles in it (Figure 3-2) and then create convex polygons to hover over the entire passable area

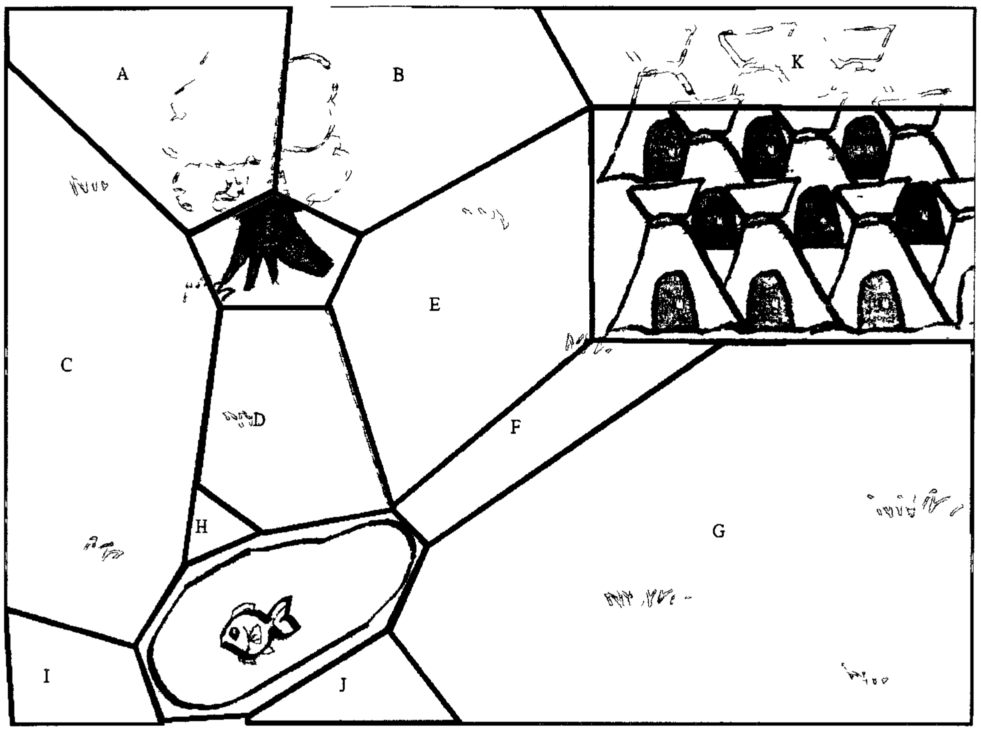


Figure 3- 2 Processed map

2 We would identify vertexes of all the polygons and denote their relationship After the data recognition, we would deposit everything into a database (Table 3-1)

| ID | Vertexes | Relation |
|---|---|---|
| A | (0,0), (300, 0), (284, 197), (194, 243), (0, 58) | B, C |
| B | (300, 0), (560, 0), (618, 107), (377, 244), (284, 197) | K, E, A |
| C | (194, 243), (195, 243), (228, 322), (205, 511), (188, 597), (137, 682), (0, 641) (0, 58) | A, D, H, I |
|  |  |  |
| K | (560, 0), (1023, 0), (1023, 108), (618, 107) | B |

Table 3- 1 Polygon information

3 We would then execute the pathfinding program and obtain a list of path templates (Table 3-2)

| ID | Path | Detail |
|---|---|---|
| 1 | AB | AB |
| 2 | AC | AC |
| 3 | AD | ACD |
| | | |
| 66 | JK | JGFEBK |

Table 3- 2 Area paths

### 3.2.2 Online Path Finding

1 After all the preparation is complete, the game server is ready to provide online service to satisfy requests Let's assume that we receive an AI request for a path from (380, 730) to (740, 60)

2 The pathfinding AI would first identify where these 2 points belong They are area-J and area-K

3 The AI would retrieve the path template from the database The path is J-G-F-E-B-K The path built by the reference nodes is shows in Figure 3-3

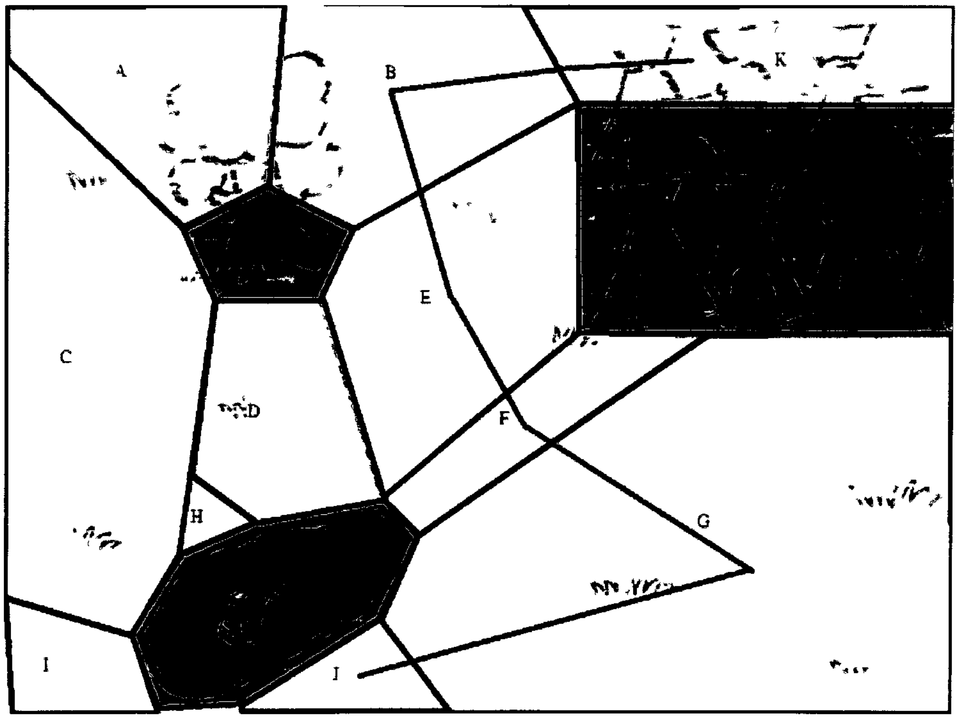


Figure 3- 3 Path built by reference nodes

4 The AI would optimize the path template by adjusting the intersections on the edges The resulting path is (380, 730), (414, 677), (452, 547), (448, 502), (618, 106), (740, 60) It is shown in Figure 3-4

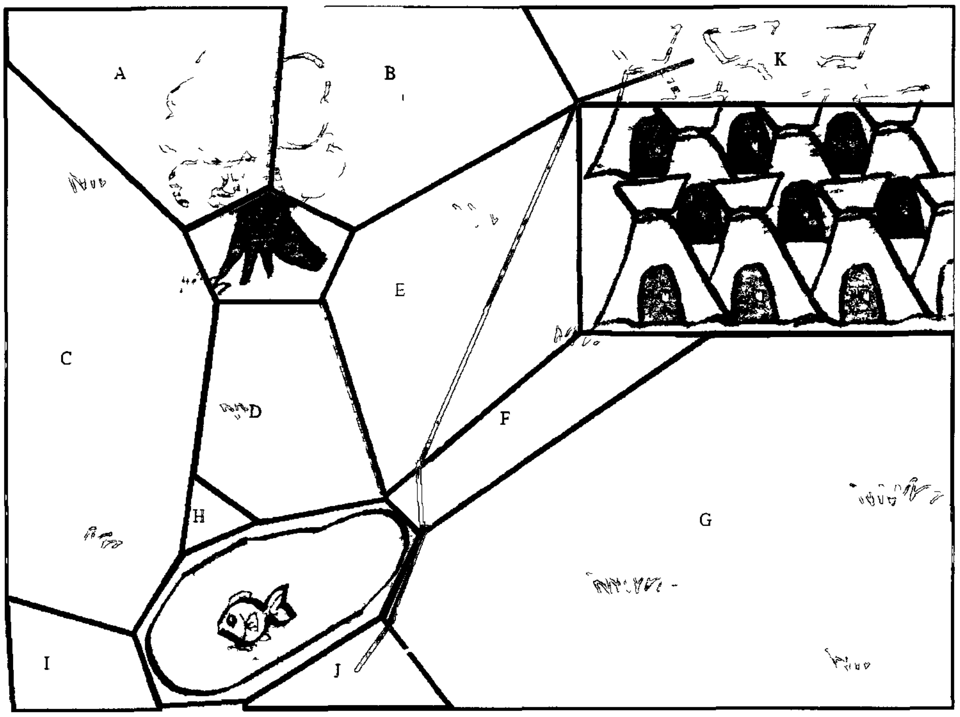


Figure 3- 4 Optimized path

## 3.3 Design Details

As mentioned, Heuristic Path Finding Method is a method for online service It reduces the runtime workload by using a distributed structure We will describe how it is designed, the role each part plays, and the contribution made by each part to the overall performance

### 3.3.1 Creating Map Data

Heuristic Path Finding Method is a pathfinding method with an overview of graphical information The format of the map information is essential to our method Heuristic Path Finding Method creates a path prototype before the game system is ready to provide service Efficiency is the first priority for a game system Even if longer initialization time is allowed in the online game business model, the game system maintainer would always want to make the loading time as short as possible Therefore we need to make the map data used by Heuristic Path Finding Method as compact as possible with sufficient information

## Convex Set

The first decision that should be made is how detailed the map information should be Take Figure 3-5 as an example, it is a simple map, 4x4 units in size One unit represents one step, moving across this map takes 4 steps at the most If there is a request to build a path from A1 to D4, there would be 22 checks for step-checking among candidate steps under the A* grid method (Table 3-3)

If we use Heuristic Path Finding Method, the pathfinding AI would create 120 path templates to cover all possibilities The A* Grid Method only creates storage memory as large as the size of map and each grid is assigned with a number, it requires a space for only 16 integers in this example Heuristic Path Finding Method takes at least 240 integers in storage space to denote the starting and ending location, excluding path details (Table 3-4)

For execution time, the A* grid method takes 22 times of step checking for path a1 to d4 For Heuristic Path Finding Method, if the path templates are stored in a binary tree, it takes O(log 120) for an average case to search for a path, or O(120) in the worst case (Binary search tree, Wikipedia)

| | | | |
|---|---|---|---|
| a1 | a2 | a3 | a4 |
| b1 | b2 | b3 | b4 |
| c1 | c2 | c3 | c4 |
| d1 | d2 | d3 | d4 |

Figure 3- 5 Sample map

| Confirmed path | Candidates in open list |
|---|---|
| a1 | a2, b1, b2 |
| a1,b2 | a2, b1, a3, b3, a3, b3, c1, c2, c3 |
| a1,b2,c3 | a2, b1, a3, b3, c1, c2, d2, d3, d4, c4 |
| a1, b2, c3, d4 | |

Table 3- 3 Path finding for the map on Figure 3-5

| Start | End | Middle steps |
|---|---|---|
| a1 | a2 | |
| a1 | a3 | a2 |
| a1 | a4 | a2, a3 |
| a1 | b1 | |
| | | |
| d2 | d4 | d3 |
| d3 | d4 | |

Table 3- 4 Path template generated by Heuristic Path Finding Method for Figure 3-5

If we divide this map into 4 groups (Figure 3-6) and apply it to Heuristic Path Finding Method, the pathfinding AI would create 6 path templates for it (Table 3-5) We expand steps into areas in order to reduce the amount of path templates to be created, nevertheless, we do not want this expansion to break a pathfinding task into several minor and cross-referencing tasks Passable areas should not overlap each other Each area should guarantee that one object within its range has direct access to another point in the same area The area should be a convex set (Convex set, Wikipedia)

With processed map, Heuristic Path Finding Method uses 12 integers to store path templates The maximum searching time is O(6) A* grid method does not use a processed map Thus, its storage space and execution time remains the same For this reason, Heuristic Path Finding Method becomes attractive

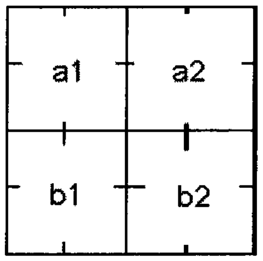


Figure 3- 6 processed map

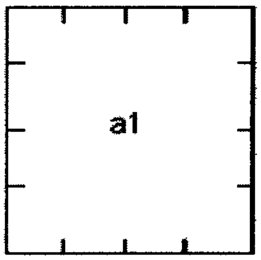


Figure 3- 7 Simplified map

| Start | End | Middle steps |
|---|---|---|
| a1 | a2 | |
| a1 | b1 | |
| a1 | b2 | |
| a2 | b1 | |
| a2 | b2 | |
| b1 | b2 | |

Table 3- 5 Path-template for Figure 3-6

Since the sample map is a square without any obstacles, it could be taken as one single passable area (Figure 3-7) Within one passable area, our method will not require any path to search With direction and destination provided by pathfinding AI, game system could correctly move game object to wherever it should be The A* grid method is uncompetitive at all in this case

## Polygon Shape

The second decision is the description of boundary Boundary helps to identify the area where the specific location belongs to Heuristic Path Finding Method uses a processed map When the pathfinding AI receives a pair of starting and ending points, it has to identify their location by checking through convex sets If a convex set is formed by random curve (Figure 3-8), the system should go through each step to confirm that the requested point is within the boundary

Take Figure 3-9 as an example, it takes 20 units to surround an area of 4x4 units A larger area will require a longer boundary In the worst case scenario, requested point will be located on the last checked area The time saved during the pathfinding process by using convex set may be consumed by area identification Therefore, we need a boundary description, which is simpler than recording every passable location around an area

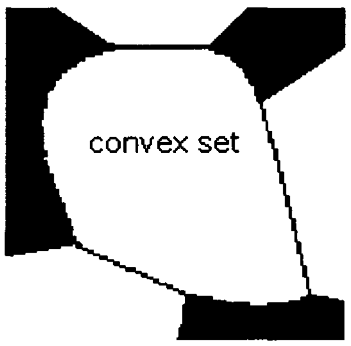


Figure 3- 8 Sample of convex set

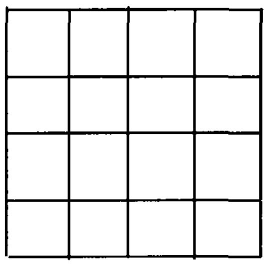


Figure 3- 9 Map boundary

The simplest way to describe a passable area is to use it as a convex polygon When dealing with a graphical calculation, a straight line is always easier than a curve Map designers only need to call out vertexes of convex polygons, which the main system is based on in order to reverse back edges and relation of those polygons

## Data Structure in Runtime

To meet the efficiency requirement, the map is simplified into vertex polygons to describe the passable areas The system would load the polygon vertexes, rebuild the graphic structure, restore the relationship and keep the information in memory Checking the boundary of a passable area should be done in a routine manner, rather than allowing the AI to frequently allocate and release the memory then retrieve the data from the database, keeping the required information in memory of a gaming system will shorten the checking time

We have vertexes used to form the polygon boundary, these vertexes can also be used to ıdentıfy the relatıonshıp of the polygons The map used by Heurıstıc Path Fındıng Method ıs partıtıoned ınto convex polygons (Fıgure 3-10) Adjacent areas share the same edge, such as edge Va1-Va2 of polygon A Edges that belong to only one polygon are taken as adjacent to non-passable area, lıke edge Vb1-Va1 of polygon B

Accordıng to thıs rule, the system can ıdentıfy adjacent passable areas by scannıng though polygon vertexes Sınce the vertexes are checked so frequently, we can keep the relatıonshıp of the vertexes ın memory whıle trackıng the adjacent areas and correspondıng edges durıng map loadıng process

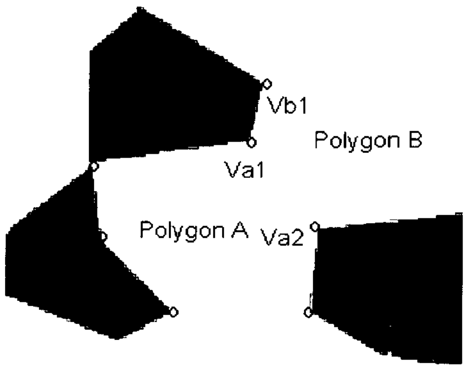


Fıgure 3- 10 Sample may ın convex polygon

```
Struct Map
{
  Polygon1
    Vertex 1  {x1, y1}
    Vertex 2  {x1, y1}

    Vertex 3  {x1, y1}
    Neıghbor 1
      Neıghbor ID  Polygon 3
      Edge  Vertex 1 & Vertex2
      Neighbor 2

   Polygon 2

}
```
Table 3- 6 data structure

After the map is completely loaded, the map information would be kept in memory as the structure illustrated in Table 3-6

### 3.3.2 Preparing Path Prototype

With the loaded map information, the system is capable of building a path prototype In order to increase pathfinding efficiency at runtime, Heuristic Path Finding Method prepares a path prototype for every pair of two different passable areas that are accessible to each other, and stores them in the database When a pathfinding request is received, the system retrieves the path within the corresponding areas based on the starting and ending points, finally, it adjusts the path to fit the current conditions When the passable area is properly divided, this process helps the system cut down on task processing time without losing accuracy

Heuristic Path Finding Method uses the A* algorithm to build a path prototype One important attribute of the A* algorithm is that it evaluates the candidate points based on the cost accumulated from the starting point and distance estimation from this point to the destination When we said that "we are going to build path from area-A to area-D", we do not have any specific location that could be called as "step", we do not even have starting and ending points Without a specific location, we cannot provide the pathfinding AI distance information, which the A* algorithm bases its evaluation on

In order to make the A* algorithm work under ambiguous map information, such as the passable areas used by our method, we have to create temporary reference points and adjust it gradually to a better location, which allows the following evaluation to benefit from (Figure 3-11 and 3-12)

We call a gradually changing reference point as "dynamic reference point" Since the reference point would be changed anyway, we set the reference point as an average vertex point which only needs a simple calculation (Table 3-7) and is always within a passable area due to the attribute of convex polygon

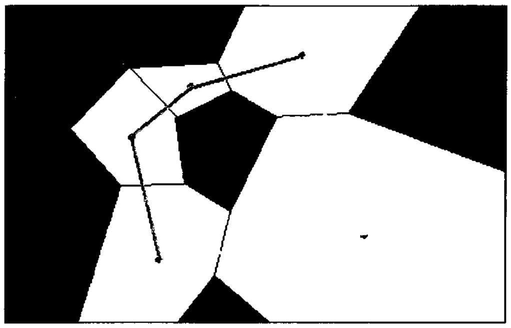


Figure 3- 11 Path built from fixed reference points

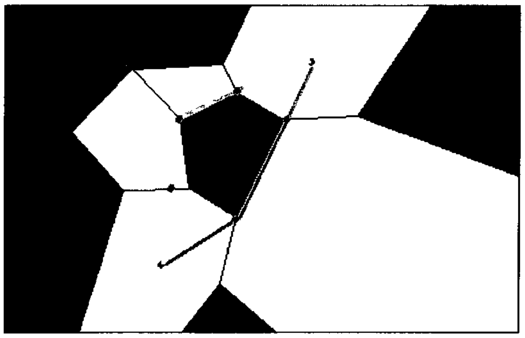


Figure 3- 12 Path built from dynamic reference points

```
struct pomt {
   mt pX,
   mt pY,
},

pomt   pVertex1, pVertex2, pVertex3,     pVertexN,
PtTemp = (pVertex1 + pVertex2 + pVertex3 +     +pVertexN ) / N,
```

Table 3- 7 Rule of average vertex locatıon

Havıng a processed map and rule of reference pomt generatıon, Heurıstıc Path Fındıng Method ıs ready to buıld a path template We now use Fıgure 3-13 as an example to ıllustrate how a dynamıc reference pomt helps our method generate a path template

Fıgure 3-13 ıs a map wıth 4 passable areas C, D, E and F Heurıstıc Path Fındıng Method ıs supposed to buıld area path CD, CE, CF, DE, DF and EF The followıng steps show how our method buılds the path template for CF

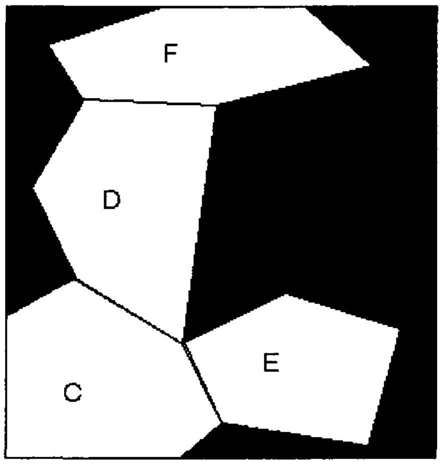


Fıgure 3- 13 Sample game map

Step 1 The pathfındıng AI generates reference pomts accordıng to the rule ın Table 3-7 as the startıng pomt ın area C, denoted as P_start, and the endıng pomt ın area F, denoted as P_end (Fıgure 3-14 (a))

Step 2 By looking at the relation list, the pathfinding AI establishes that area C has 2 neighbor areas area D and E The AI puts both areas into an open list according to the rule of the A* algorithm, and generates a reference point for each area, denoted as P_dTmp and P_eTmp (Figure 3-14 (b))

Step 3 Heuristic Path Finding Method evaluates paths from P_start to P_dTmp and P_eTmp by A* algorithm The evaluation would suggest that P_dTmp is closer to P_end The AI sets area D as the next step and creates new reference point, dTmp2, on the intersection of the path from P_start to current P_dTmp and the line Edge_CD of polygon D (Figure 3-14 (c))

Step 4 The AI puts area C to close list according to the rule of the A* algorithm By looking into the relation list, the AI establishes that area F is the only neighbor to area D, which is still out of close list AI puts area F into open list

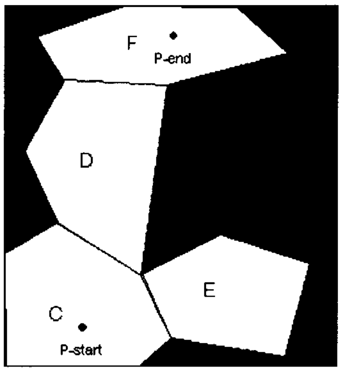
Step 5 Knowing that the ending point is located in area F, the AI takes ending point P_end as the reference point The AI has referenced a point before the path has entered area D, P_start in this case, and a reference point after the path has gone out of area D, P_end in this case At last, the AI adjusts the reference point for area D to its final location

Step 6 The AI builds a line from P_start to P_end as LineCE and extends the edge D_Left to D_Right to infinite line, LineD The AI finds the intersection of LineD and LineCE on P_dTmpF (Figure 3-14 (d)) If P_DTmpF is out of section D_Left and L_Right, the AI can use the closest vertex to replace it

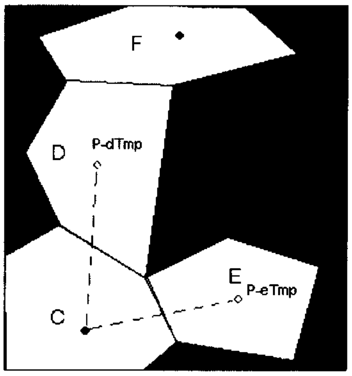
Step 7 With reference point P_dTmpF and P_end, the AI modifies the reference point of area F to P_fTmp2 (Figure 3-14 (e))

Step 8 Area F is the final area The AI has routed a real path which starts from P_start, goes across P_dTmpF, PfTmp2 to the ending point, P_end The path template is CDF The AI now stores the path template into database (Figure 3-14 (f))

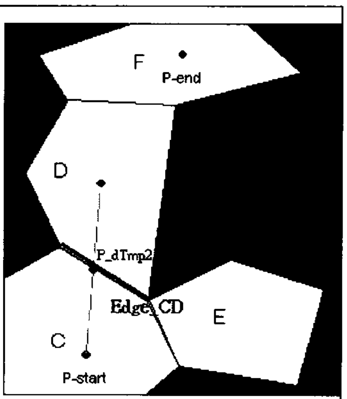
Step 9 Path template CE was completed



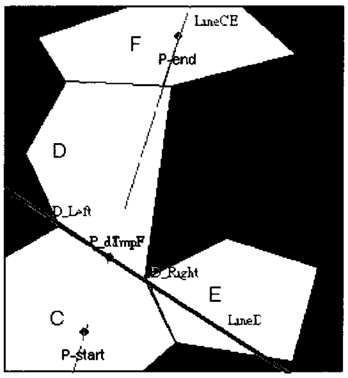
(a) AI generates start and ending points



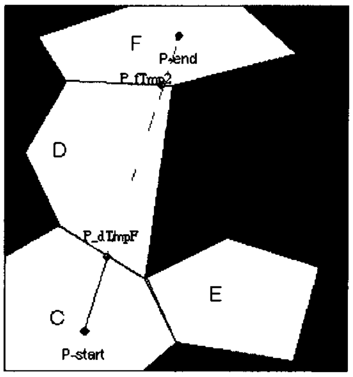
(b)AI generates reference points for area D and E



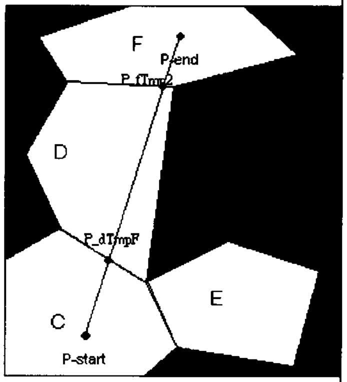
(c)Reference point on area D is modified to the intersection on edge



(d)The final reference point of area D



(e)AI modifies reference point of area F to P_fTmp2



(f) Path CF is completed

Figure 3- 14 path-finding steps

From this illustration we can see that reference point has been modified twice and the final version is very close to the shortest path Correct reference point is crucial for A* algorithm to build the information on the cost of an arbitrary point for path evaluation At the phase of building path template, only area information is stored in database Other information such as P_start or P_dTmpF, in this case, is discarded

The area path is stored in the database under the schema illustrated on Table 3-8 and Table 3-9 Splitting data into 2 parts shifts the sorting task from the game system to the database system, which also prevents the game system from reversing the order Comparing with the other methods described earlier, our method takes more time for preparation This would actually be intolerable if it were for one-time usage However, since our method is aimed for a massive request processing, the time used for initialization will be redeemed gradually by the time saved during the runtime phase in the overall time consumption

| Id | Identity |
|---|---|
| idStart | ID of start shape |
| idEnd | ID of ending shape |
| idPath | Path identity |

Table 3- 8 Map path description

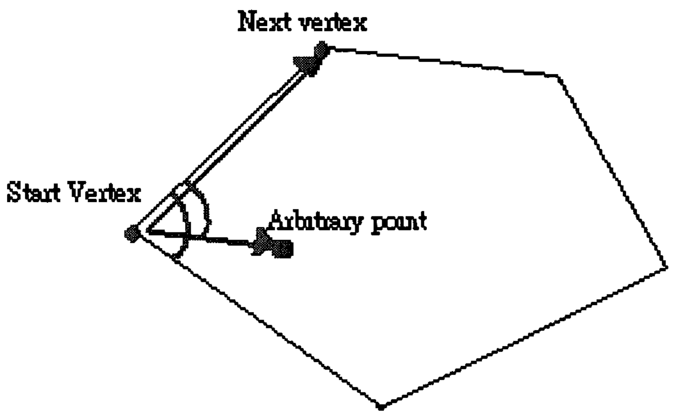
| Id | Identity |
|---|---|
| idFind | Order of path |
| idPath | Path identity |
| IdMesh | Shape identity |

Table 3- 9 Single Path description

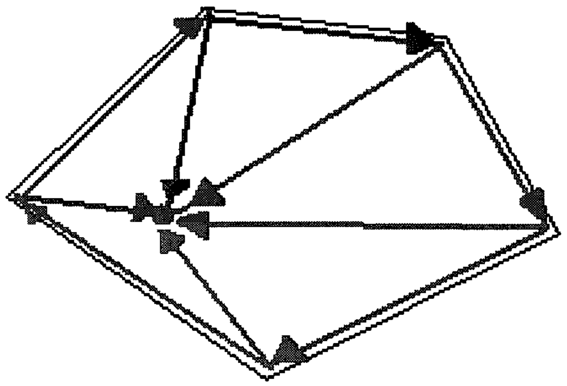
### 3.3.3 Processing the Pathfinding Request

When the game system starts to provide the hosting service, the pathfinding request would be passed as a pair of starting and ending points The game system identifies the corresponding areas and then queries a path from the database We define a map as a list of vertex of a convex polygon This approach not only guarantees that two different arbitrary points within the same passable area have direct access to each other, but also simplifies the way to identify a point

Being a convex set, any of its interior angles must be less than 180 degree Therefore, for any point within it, the angle between the vector of one vertex to its neighboring vertex and a vector of same starting vertex to an arbitrary point within the convex polygon must be less than the interior angle of the start vertex, illustrated on Figure 3-15 (a) If one of these points meets this rule after checking through all the vertexes of one convex polygon, its area location can be confirmed (Figure 3-15 (b))



(a) Angle of inner point to edge is smaller than vertex angle



(b) AI checking through all vertex angles to identify a inner point

Figure 3- 15 Inner point identification

The database selects one path template from the prepared data according to its starting and ending areas, and then it sends the path back to the game system in the correct order Consequently, the game system will convert the path template into a real path by gradually modifying the reference points in the same way it builds a path template without a pathfinding evaluation (Figure 3-16) If any 2 different areas have no accessible path to each other, there would be no record in the database and the game system is notified without wasting any time on the pathfinding process (Figure 3-17 and Table 3-10)

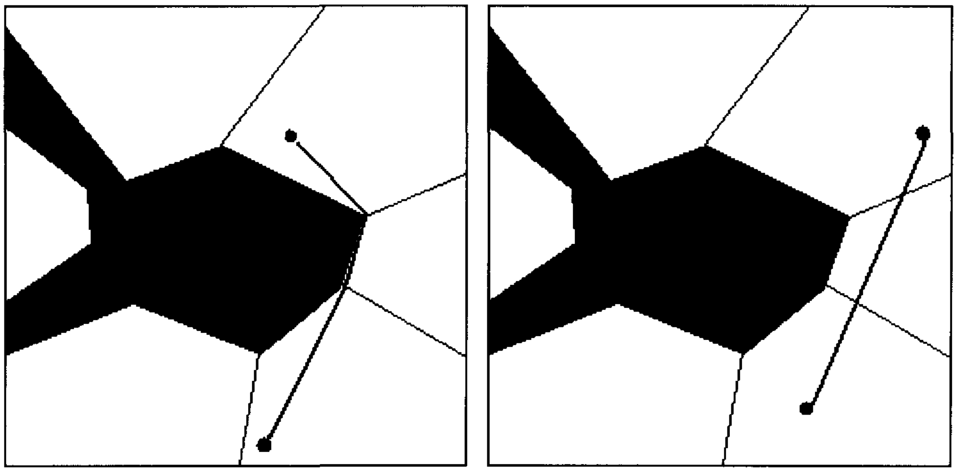


Figure 3- 16 Two final path generated from same area path

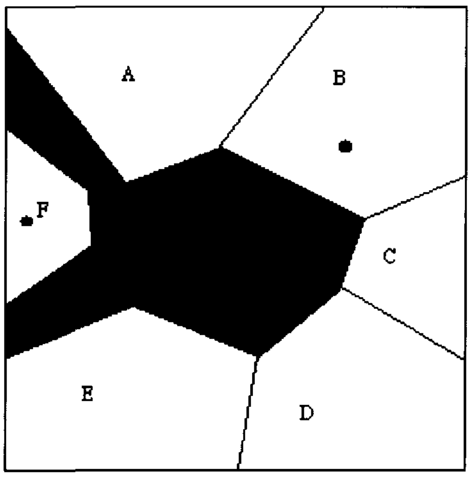


Figure 3- 17 Request that is not able to get a path template from the database

| Start | End | Mıddle steps |
|---|---|---|
| A | B | |
| A | C | B |
| A | D | BC |
| A | E | BCD |
| B | C | |
| B | D | C |
| B | E | CD |
| C | D | |
| C | E | D |
| D | E | |

Table 3- 10 Path templates for Fıgure 3-16, 17

From the entıre pathfındıng workflow, we can see that Heurıstıc Path Fındıng Method greatly sımplıfies AI's work by shıftıng a major portıon of the pathfındıng process from runtıme to the preparatıon phase Effort spent at searchıng for a path and maıntaınıng the path ınformatıon ıs not carrıed out by the game system, ınstead, ıt ıs accomplıshed by the database system, whıch ıs an expert for thıs type of work

We can see that, Waypoınt Method accelerates the A* grıd method by swıtchıng the focus from estımate calculatıon to map sımplıficatıon, Navıgatıon Mesh Method for further sımplıfyıng the map by ıntroducıng the area concepts ın passable steps We further ımproves onlıne pathfındıng progress by convertıng the solo pathfındıng progress ınto a dıstrıbuted structure such that a resource costly portıon of computatıons can be done off-lıne, the repetıtıve calculatıon of these computatıons can be avoıded Thereafter, by adoptıng our algorıthm, not only the pathfındıng efficıency ıs ımproved but also the entıre gamıng system achıeves a better performance sınce the resources (tıme and memory) become predıctable

The system resources used for ınıtıalızatıon are relatıvely heavıer than other methods, but the workload ıs lıghter at runtıme Heurıstıc Path Fındıng Method ıs free from complıcated pathfındıng wıth database support, ıts area of ıdentıficatıon ıs simplified by the map definition Heuristic Path Finding Method takes advantage of a distributed structure, general functionality of operation system and characteristics of data types to meet massive pathfinding requests instantly

# Chapter 4 Implementation details and subjects of measurement

Pathfinding is not the most important AI functionality, but it helps computer controlled objects to move similar to humans and make virtual world more vivid The first requirement for a game system is being responsive to user input Accuracy of AI is not a major issue but it still needs to respond correctly when the computer controlled character faces the player

Heuristic Path Finding Method targets the online game pathfinding functionality As we mentioned in Chapter 1, online game provides service via internet Comparing to console game, the number of concurrent users of online game is hundreds if not thousand times more than that of console game, and the demands for game functionality also expand greatly Having many players logging on to the same virtual world, fairness and concurrency become important issues The virtual world is not stationary, a game system does not have an infinite amount of time to calculate for the best solution In order to resolve massive requests in a limited time, game system can choose either adapting more well-designed algorithm or managing system resource more efficiently Our method is the second category It breaks down the workflow of pathfinding process, and introduces database to assist management

In order to prove that our Heuristic Path Finding Method has better performance on massive pathfinding environment, we have built a testing system which is able to perform all pathfinding methods mentioned previously and generate reports for performance analysis In this chapter, we will describe the implementation of the testing system and subjects of measurement it focuses on

## 4.1 Introducing Database to Pathfinding Functionality

Database is introduced to the world by business model It is an excellent tool in dealing with a huge amount of data Some of the main characteristics of a database include the ability to analyze data by specified rules, create the best structure to hold information, and specialize in selecting data efficiently

Database is constituted by four basic functionalities insert, update, select and delete When a user inserts a row of data into a database, the database would first try to find a proper location to store the data If no proper location is found, database could expand current structure based on an efficient selecting process, build space to allow further incoming data, and adjust the maximum size limitation accordingly Functionality of deleting data shares the same basic rules as insertion The only difference lies in that database shrinks it structure according to the rules instead of expanding it Usually updating data will not cause changes to structure, since data size and format are already defined when the table is created

The most useful functionality of database to pathfinding in a game setting is selecting data Database selects the required data according to query statements The rules specified by the statements may be very complicated The need to perform complex selection in a short time is the reason why a database should alter its structure during the insertion and the deletion processes Database takes time to optimize data structure whenever a change is required, which helps database to select as efficiently as possible Heuristic Path Finding Method adapts to this attribute by building all paths in the initialization phase and leaving path selection to runtime By shifting time consuming work to the initialization phase, our method changes pathfinding work into data selection in the runtime phase

Introducing new factor into existing mechanism always requires additional resources If using Heuristic Path Finding Method comes with a cost, we need to know clearly what the cost is for using this new feature, and how it can be mitigated

Online game host usually has at least one database to maintain user accounts A game company always wants to gather user information as detailed as possible for business analysis On the other hand, game map and path template are mostly defined in a compact format for reasons we mentioned earlier in chapter 3 Comparing to user information, pathfinding data occupies relatively less space but provides essential support for the game system Schema of path template is designed in a straightforward manner for the database Selecting one path from a path template table does not require complex functionality such as pattern recognition or data mining, which are used in business analysis The workload that Heuristic Path Finding Method adds to database is nothing when comparing to what it can achieve for game hosting, such as user account management

In the game world, player movement triggers AI pathfinding requests constantly Game system is responsible of processing the requests and adjusting path template to each path used in the game Therefore the game system decides the number of connection to spare for database connection This means that our method will not be allowed to carry out any unexpected connection or to impair the workflow of game system As a result, while introducing database into pathfinding functionality, our method will not affect the original budget and performance It simply hands out part of its work to database, and relies on the database's specialty to increase total performance

## 4.2 Map Information Storage

Our testing system contains four pathfinding methods A*grid method, Waypoint method, Navigation Mesh Method and Heuristic Path Finding Method All methods are implemented under the same A* algorithm and storage preference By unifying pathfinding functionality, the testing system ensures that any performance difference is caused only by the difference in the structure of the methods rather than by any potential self-optimization of the functionality of each method Each method has its own data structure to hold map information Here we will describe the attributes of these data structure for each method and illustrate how it is implemented in our testing system

When game application starts, player sees a virtual world illustrated or described on screen In the virtual environment, there might be grass, tiles, trees, walls, buildings, etc Objects are not randomly put on the map, but by designers' will to give soul to the virtual world However, to the pathfinding AI, every object only means one of the two things passable location or non-passable location

Using A* grid method, one grid represents one step of movement Pathfinding map could be set as a simple matrix to mimic coordinate system Objects which occupy one or more grids on the map set flag on corresponding grids Game system determines passable grid according to the flag on grid Table 4-1 illustrates how map information is used The final path built by A* grid method is continuous points linking the starting and the ending points

```
enum _e_obȷect_ıd_
{
   eObȷectGrass = 0,
   eObȷectTıle,
eDecorateStnp,
eDecorateCurtaın,
   eObstacleTree,
eObstacleWall,

},
unsıgned ınt uıMapInfo[50][50],
uıMap[25][25] = eDecorateCurtaın,
uıMap[23][25] = eObstacleTree,

locatıon getNextPoınt(locatıon poGnd, locatıon EndGnd)
{
location poCheck = poGnd,
locatıon poNext = poGnd,
unsıgned ınt ıCost = MAX_UNSIGNED_INT,
unsıgned ınt ıTempCost,

ıf (ınt ıdDırectıon = 0, ıdDırectıon < 8, ıdDırectıon++)
{
   ıTempCost = getCost(poCheck, EndGnd),
   If ( ıCost > ıTempCost )
   {
    poNext = poCheck,
    ıCost = ıTempCost,
   }
poCheck = poCheck + locatıonShıft[ıdDırectıon],
 }
return poNext,
}
```

Table 4- 1 Map data used by A* Grıd Method

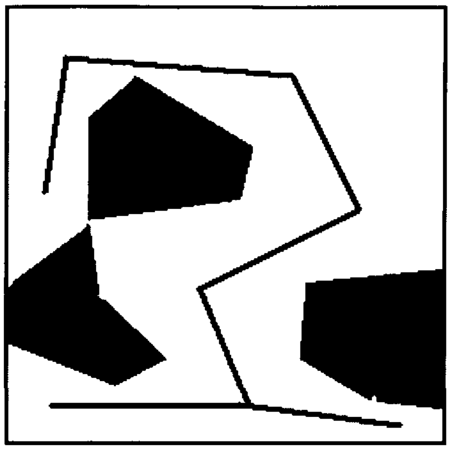


Figure 4-1 Paths built by waypoints

Waypoint method uses processed map information Map is described as a series of waypoints with links to other waypoints (Figure 4-1) Every waypoint represents passable location and every link represents accessible path Whenever a request is sent in as a pair of starting and ending points, testing system finds the closest waypoints to either point, and then Waypoint Method tries to find a path between these two waypoints Table 4-2 illustrates the structure of map information and how it is used The final path generated by Waypoint Method is a series of waypoints, corresponding links, and straight path between waypoints and the starting and ending points

```
class wpmap
{
    Public $pLink, //neighbor array
}

class waypoint
{
    protected $mapdata, // wpmap array

    protected function getNeighborList ( aWpData $poItem, array &$lst = array() )
    {
        $mapPoint = mapdata[$poItem],
        while ( list($nbKey, $nbVal) = each($mapPoint->dLink) )
        {
            $pt = $this->stridToPoint($nbKey),
            $wpfind = new aWpData($pt),
            $lst[] = $wpfind,
        }
        return count($lst),
    }
}
```

Table 4- 2 Map data used by Waypoint Method

Navigation Mesh Method uses processed map as well Its map information contains sets of convexes for passable areas to be described as convex polygons In this regard, Heuristic Path Finding Method is similar to the Navigation Mesh Method in that it also uses area-based pathfinding estimation To avoid testing system being in favor of either the Navigation Mesh Method or our method, we use the same structure as we described in chapter 3 for both methods Table 4-3 shows how map structure is defined and used in the testing system Whenever a request is sent to AI, the AI identifies areas where the starting and ending points belong to Then the AI builds a path according to roughly created reference points and gradually modifies reference points to a final path The process is similar to what is performed by Heuristic Path Finding Method during the preparation phrase

```
class nmmapdata
{
    public $lstVertex = array(),
    public $lstNeighbor = array(),
}

class NaviMesh
{
    protected $mapdata, // array of nmmapdata

    protected function getNeighborList ( $meshId, array &$lst = array() )
    {
        $meshData = $this->mapdata[$meshId] ,
        while ( list($key, $val) = each($meshData->lstNeighbor))
        {
            if (  isset($this->lstClose)
               &&     array_key_exists($key, $this->lstClose))
                 continue,

            $lst[] = $key,
        }
        return count($lst),
    }
}
```

Table 4- 3 Map data used by Navigation Mesh Method

Last but not least, our Heuristic Path Finding Method is a method that keeps game map during runtime Its map is not used for pathfinding at all, but for area identification and path modification Details of our method are described in chapter 3

Heuristic Path Finding Method does not use brand new map structure, but it chooses the one that helps the AI to build better path in 2 phase process

## 4.3 Territory Type

Our testing system implemented four pathfinding methods Only the A* Grid method uses unprocessed map data It honestly reflects the shortcomings of a map, such as redundant data stored on the map, which definitely affects performance of the pathfinding Every type of processed data is altered based on a set of specific problems that the process is designed to resolve These solutions in general are to simplify complicated conditions or to skip repetition Waypoint method focuses on accessible direction, Navigation Mesh method focuses on decreasing the number of choices, Heuristic Path Finding Method focuses on saving system resource for massive tasks processing No matter how a method tries to simplify source data, the resulting path would inevitably be affected by the style of the map content Every method has its pros and cons on different map types and this is what our testing system trying to reveal Here we will discuss attributes of each map type which would be used in the experiments

We define map type according to how obstacles are scattered around on a map Taking a map with ten obstacles as an example, if obstacles cluster in the center, which could be taken as one single obstacle, the map could be described as less broken map (Figure 4-2) If obstacles spread evenly on the map, it is defined as highly broken map (Figure 4-3)

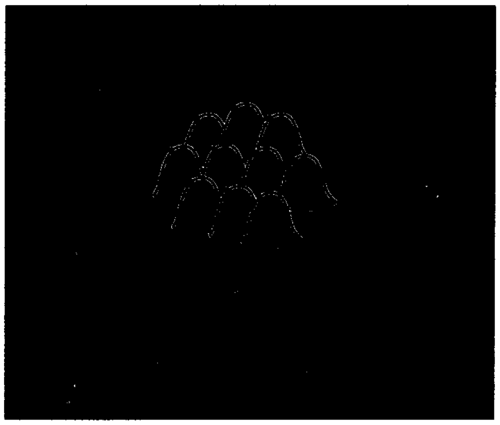


Figure 4- 2 Less broken map

Figure 4- 3 Highly broken map

Upon this concept, we can define maps into four types

## Spacious map

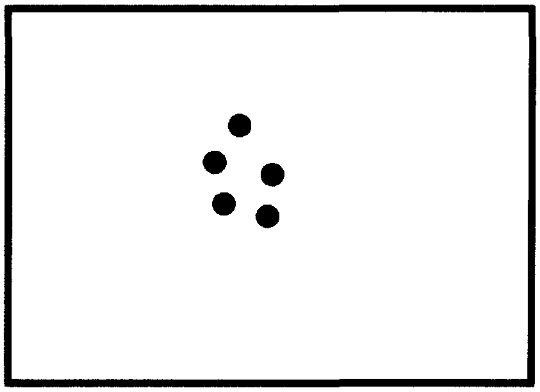


Figure 4- 4 Spacious map

If one map contains only a few number of obstacles or obstacles crowded together, there is higher possibility for one object traveling through the map without hitting an obstacle This kind of map is identified as spacious map (Figure 4-4) Pathfinding method is not really necessary on this map type

## Lightly obstructed map

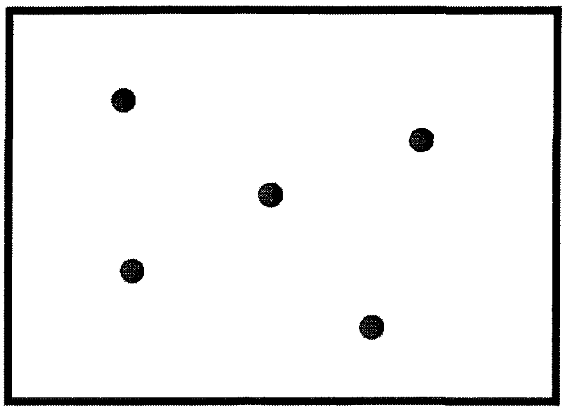


Figure 4- 5 Lightly obstructed map

A map has obstacles spreading around evenly with a potential to break accessible area into several small pieces of convex polygons, we define it as lightly obstructed map (Figure 4-5) Accessible portion of this map type is still large, but NPC need more indication from AI to avoid hitting obstacles

## Complicated map

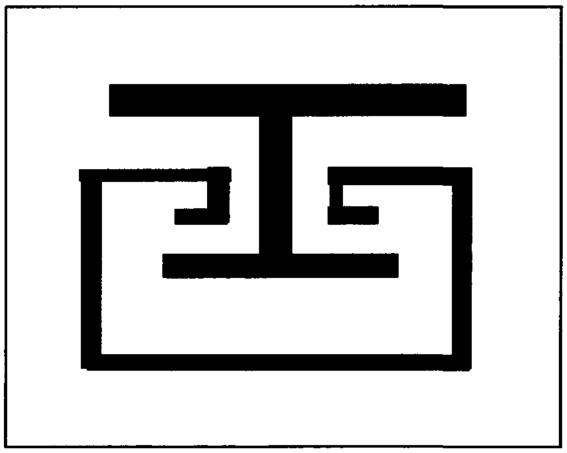


Figure 4- 6 Complicated map

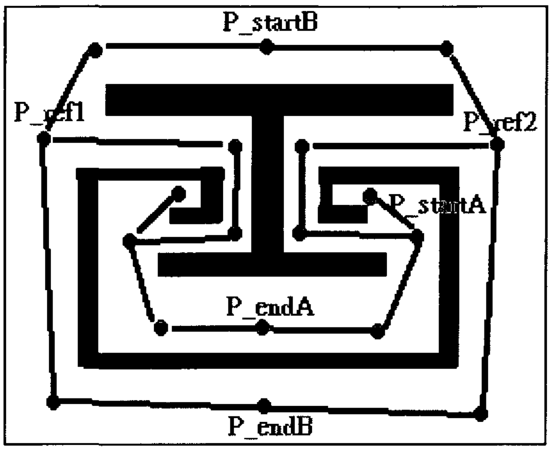


Figure 4- 7 Waypoint map for complicate map

If a map has many obstacles which purposefully form paths that allow travelers to pass through, we define it as complicated map (Figure 4-6) Though it seems the obstacle here makes pathfinding work difficult, this map type has in fact simplified the work if the map data is processed For example, if we process the map to waypoint map, illustrated as Figure 4-7, we could see that there is only one path to move object from P_startA to P_endA, and there are only two directions for object on P_startB to move to P_endB by passing though either P_ref1 or P_ref2 Pathfinding AI is necessary and beneficial for this map type because even though this type of map is more complicated, it offers fewer choices in the available routes Hence, as soon as the AI is able to determine the turning corners at each point, it will easily figure out the best route

**Broken area**

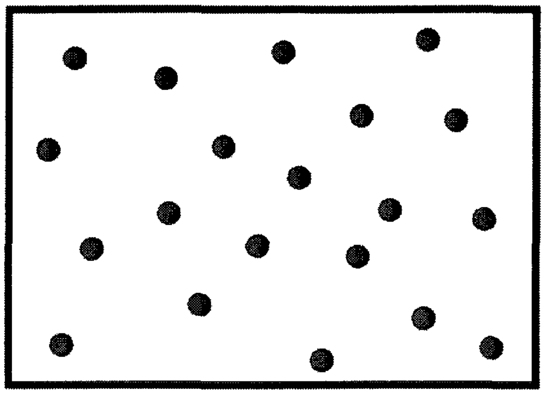


Figure 4- 8 Broken area

Maps that match none of previous three conditions could be defined as broken area (Figure 4-8) Processed map data of this type only reduce redundant checking, but is of no help on simplifying pathfinding work

With the concept of map types, we now have a glimpse of what pathfinding AI confronts In pursuit of large game environment and performance optimization, how to maneuver pathfinding AI with scarce resource is what our method wants to achieve

## 4.4 Memory Usage

Performance of game system does not rely on the speed of Central Processing Unit (CPU) only, but also on data structure that stores constantly referenced static data Memory is an important resource for game system During runtime, memory cache is more convenient than disk storage If game data is too large to be kept in cache, game system has to save it in storage device Swapping data from storage device to memory cache is a time consuming technique and would obviously slow down system process This is what game system designers desperately want to avoid Pathfinding work consumes memory, too There are three types of memory usage that pathfinding method needs map data storage, runtime allocation and prepared information

**1) Map data storage**

In the very first version, a map was defined almost identical to a coordinate system, such as the map used by A* grid method The bigger the map size, the larger memory usage required With the intention to enhance performance, developers not only try to improve algorithm of pathfinding method, but also simplify the map it uses Map information is constantly referenced to, and must be kept in RAM (Random-accessed Memory)

**2) Runtime allocation**

In runtime, most methods need temporary allocated memory to store all potential selections, so called “open list” in A* algorithm, and fully checked selections, so called “close list”, before the final decision is made The temporary memory may become too large when consumed by temporary data, which may cause memory swapping in runtime Open list is scanned repeatedly whenever evaluation for the next step is needed This kind of memory usage is crucial to performance but highly unpredictable

**3) Prepared data**

Only Heuristic Path Finding Method uses prepared data Prepared data is used for assisting pathfinding AI Prepared data consumes system memory in keeping its accessibility Referencing to the prepared data takes additional processing time in runtime which is not necessary in other pathfinding methods Preventing the prepared data from holding back system performance is therefore an important task Designers should take this into account before employing prepared data

Minimum usage of both runtime memory and static data is an ideal that pathfinding methods should pursue If large-sized static data is inevitable, minimizing memory usage for runtime allocation should have higher priority to guarantee better performance

## 4.5 System Structure

In order to perform pathfinding task, there should be a map, a pair of starting point and ending point and specific pathfinding methods Heuristic Path Finding Method requires additional database connection There are 3 other pathfinding methods, A* grid method, Waypoint method and Navigation Mesh method, to be compared with our method The testing system we use also implements tools such as image decoder for A* grid method and waypoint generator to simplify map generation It also has record parser to translate result into analysis data

We created our system under the server-client structure Hypertext PreProcessor (PHP) provides comprehensive basic utility Under Apache web host, PHP core functionality is good at memory management, technique sorting and database communication, which is good for us to concentrate on developing pathfinding method We use BMP file as the source of map data Instead of using PHP's external library to decode image files when executing pathfinding task, we convert image files by our simplified external executable file to text file, which is the format PHP excels at

## 4.6 Testing Structure

Heuristic Path Finding Method is a method derived from A* algorithm, and so as the other three methods implemented in the testing system Our experiments focus on the difference in output between each method By avoiding difference in characteristics that all methods share in common, we wanted this testing system to reveal the difference in performance affected by designed structure Here are guidelines we followed when implementing the testing system

### 4.6.1 Hardware with medium capability

Hardware is an important element, together with operating system and pathfinding methods, to affect the overall performance Also, the hardware determines the level of complexity in the available mathematical calculation for pathfinding work A* grid method uses unprocessed map data, together with simple math calculation, to resolve a pathfinding request, whereas the other three methods use the processed map data with elaborate mathematical calculation One pathfinding task could be resolved by either numerous simple calculations or fewer elaborate calculations When using low-end hardware, difference in performance would be obvious especially when map data is changing On the other hand, when using high-end hardware, changes in map content may affect performance only slightly, or even unnoticeably Therefore, we built our testing system based on medium-leveled hardware to make difference significant

### 4.6.2 United version of A* algorithm

All methods, including Heuristic Path Finding Method, are based on A* algorithm It is important to develop all methods under the same version of A* algorithm There are existing open source codes for pathfinding methods available, but each of them specializes in different aspects, such as data management, cache technology or order evaluation To minimize the variation in core functionality, our testing system applies same technique at common functionality, such as sort, stack and data dump, on all methods

### 4.6.3 Same pair of starting and ending points

Comparing pathfinding performance between two paths of different complexities makes no sense The testing system generates and stores the same set of testing data for each map to be applied to all four methods Every method completes all tasks and saves execution-related data as text file for performance analysis

### 4.6.4 Amount matters

We want a pathfinding method to be capable of fitting into the modern game environment, which asks for immediate response in a highly interactive virtual world It is an environment that is prone to trigger massive request in peak time Massive amount of testing is requested for our experiment In order to make difference in performance significant, each method processes and completes all tasks in one setting, and then another method does the same from beginning to end, rather than having each method processing one task only, then alternates until all methods complete all tasks

### 4.6.5 Diversity in map types

We expect each pathfinding method to have its own specialty in some map types We provide maps of all types to ensure our experiments were not in favor of any particular method A variety of map types ensures result of our experiments to be neutral

## 4.7 Disadvantage of testing system

During the development of our testing system, we have encountered an issue on Waypoint method, which was not able to be resolved fundamentally due to the shortage of manpower at that time We have found a way to get around it, but the solution has affected the experiment result in a certain way

When we were evaluating the testing result during development, we found that the Waypoint method relies too much on the design of the pre-processed map It needs different plotting strategies for different maps For example, waypoints cannot be too loose in the spacious map, and Waypoint method would sometimes detour to a longer path Waypoints cannot be too intensive, either Intensive waypoints cause redundant checking The more links one waypoint has, the slower this method would be

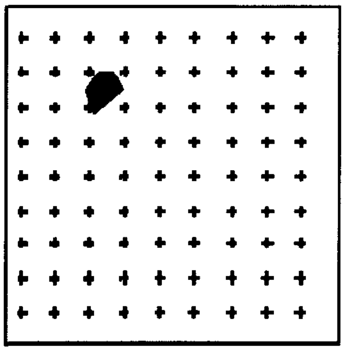


Figure 4- 9 Generated waypoints

Waypoint is not only for eliminating unnecessary checking, but also for ruling out unnecessary selection At that time, we were in the process of deciding map content We wanted each map used in the experiment to correctly represent the map type it was supposed to be Whenever we made a change to map content, waypoints would be reprocessed After several times of resetting, we finally used a tool to generate grid like waypoint-set (Figure 4-9) This tool saved us a lot of tuning time for plotting Waypoints, but was not smart enough to solve the design problem Here we list the issues that we took into consideration when reviewing the output of the Waypoint method

## 1) Pre-checking for starting and ending points

In our design, the system tries to identify the closest waypoints to the starting and the ending points before pathfinding work begins There are chances that the closest waypoint is not accessible to the requested points (Figure 4-10) Instead of spending time on tuning reference nodes, the testing system used unprocessed map to assist identification During runtime, the testing system checked each step along the straight path between waypoints and the requested points to ensure that there are no blocking areas in between This is a content design issue Loading time and memory usage for path-checking should not be included as Waypoint Method's performance, but there is no proper way to split them out of the performance result of Waypoint Method Accessibility checking in runtime may slow the performance a little In our analysis, we named each result of Waypoint with postfix _No_Check to denote the accessibility checking was turned off

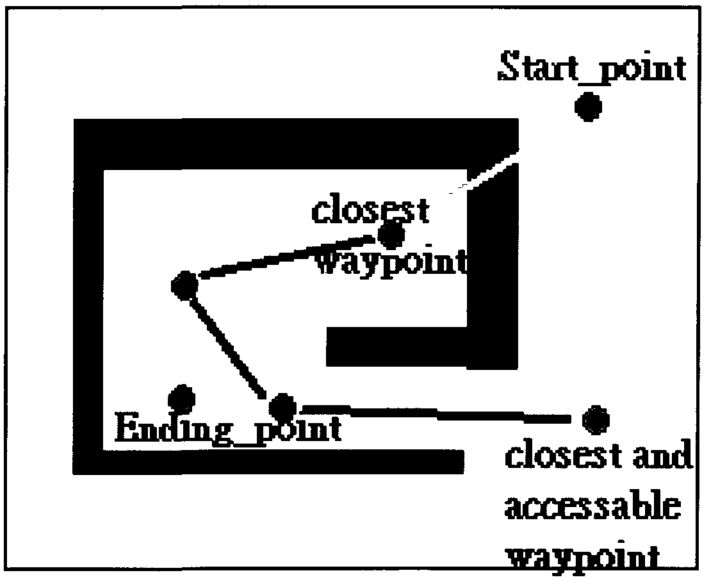


Figure 4- 10 Closest node and valid node

## 2) Rounding total length

When we implemented the waypoint generating tool, we tried to minimize the number of links for one point to reach others, because the processed map information is supposed to reduce unnecessary checking But it ended up giving improper selections to the pathfinding AI A waypoint can only choose points at the top, the bottom, the left and the right of itself One of the advantages Waypoint Method has over A* Grid Method is that linking direction for each waypoint is pre-designed, and it help to make better decision from fewer selections Under current structure, moving diagonally takes two steps to complete (Figure 4-11) In order to make the final path length closer to human design, we assumed that there were half chances for one step to move diagonally, therefore we estimated a realistic path length to be 80% of the final length (Figure 4-12) The estimated value is denoted as "Rounded" in experiment result

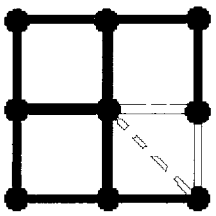
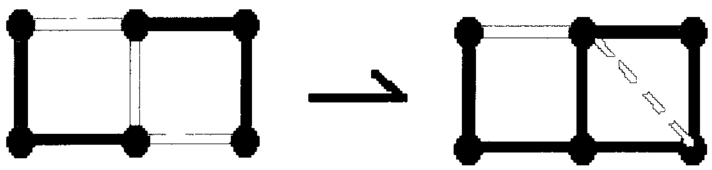


Figure 4- 11 Diagonal movement


Rounded Percentage

$((0\ 5 * 1) + (0\ 5 * \quad)) / ((0\ 5 * 1) + (0\ 5 * 2)) = 80\%$

Figure 4- 12 Function of Rounded percentage

Although rounded value and the assumption it based upon may not seem accurate, we do not consider Waypoint Method experiment as failed When trying to plot suitable waypoints, we found that density is a major issue for this method If the density of waypoints is too low, path length would be doubled In contrast, time spent for pathfinding may extend if density is too high (Figure 4-13, 4-14) Our waypoint generation tool was able to cope with density issue Comparing to density, disadvantage in direction only affects performance in a minor way and the rounded value could help to adjust it The length of the final path that could be rounded to a more realistic value by the rule we made is shown in Figure 4-12

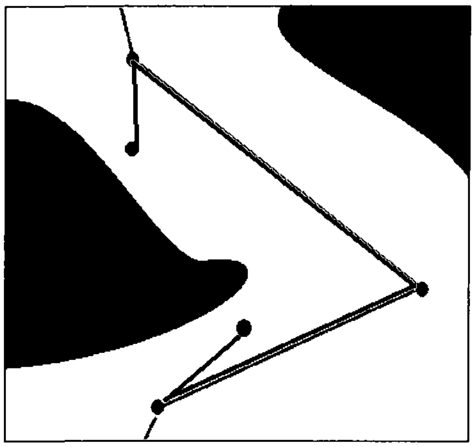


Figure 4- 13 Too few waypoints

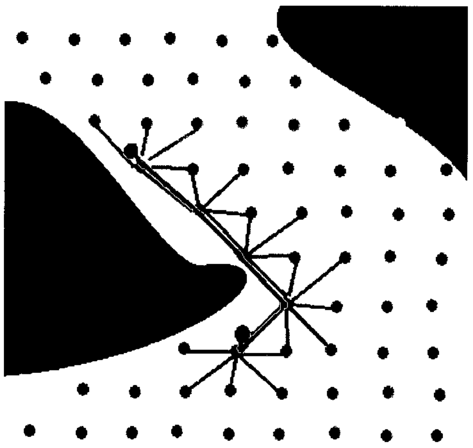


Figure 4- 14 Too many selections

# Chapter 5 The Experiments

This chapter describes the experiments we designed, discusses the testing results and analyzes the outcomes For each experiment, we will explain the reason of the design, as well as the assumption and the expectations we have If the result is different from our expectation, further analysis will be specified A pathfinding report is also constructed containing three major columns method name, length in total and time spent Experiments that used maps other than spacious map, specifically the prepared maps, will have a fourth column showing the loading time

The first column describes which method generated this record "A-star grid" represents A* Grid Method "Waypoint" represents Waypoint Method Postfix of waypoint title in numeric format denotes the horizontal and vertical space between two waypoints The postfix "_No_Check" denotes this execution skipped checking of accessibility which is described in Chapter 4-7 "Navigation Mesh" represents Navigation Mesh Method And "Heuristic" represents our method Heuristic Path Finding Method

The second column, "length in total", shows the sum of the path lengths for all resolved tasks For Waypoint Method, there is a second record noted as "rounded" which is the adjusted value due to the defect of Waypoint generating tool This issue was described in Chapter 4-7 The third column, "time spent", shows the length of time each method took to resolve all tasks The fourth column, "loading time", represents the length of time the game system took to load prepared data We do not have this record for spacious map because the time is immaterial to have meaning on its difference

## 5.1 Performance on Simple Map

The sımplest map ıs the spacıous map type Object travellıng across thıs map type rarely runs ınto an obstacle Almost all dırectıon ıs correct, every step ıs valıd Pathfindıng method has very few selectıons to choose from, ıf any Result of thıs experıment should reveal how long a method needs to buıld a path when every decısıon ıs correct We expect thıs experıment to gıve us a prevıew of all ımplemented methods Wıth thıs basıc concept, we can adjust upcomıng tests and correspondıng assumptıon to be more realıstıc

Expectatıon of thıs experıment ıs that A* Grıd method would get the shortest final path, but longest processıon tıme, whereas Heurıstıc Path Fındıng Method would get the shortest processıng tıme

### Preparation

We have desıgned a map of sıze 250 * 250 wıth one blockıng area on the map (Fıgure 5-1) We dıvıded the map ınto five passable areas for area-based methods (Fıgure 5-2) For Waypoınt Method, reference nodes are generated based on dıfferent densıtıes We generated 2000 sets of testıng data for thıs experıment

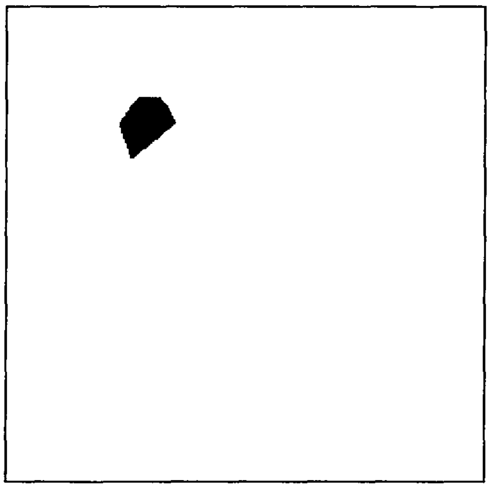


Fıgure 5- 1 Testıng map

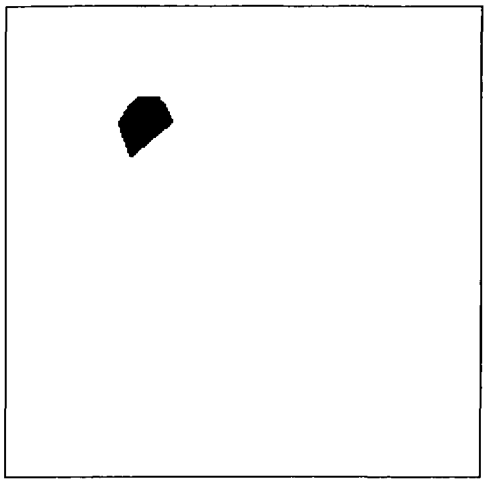


Fıgure 5- 2 Area map for 1st experıment

## Result

| Method name | Length ın total (pıxels) | Tıme spent (sec ) |
|---|---|---|
| A-Star Grıd | 279097 87 | 1702 92 |
| Waypoınt_50 | 395471 513<br>(Rounded 316377 210) | 16 32 |
| Waypoınt_40 | 383775 036<br>(Rounded 307020 029) | 21 85 |
| Waypoınt_30 | 373674 116<br>(Rounded 298939 293) | 34 53 |
| Waypoınt_20 | 361022 793<br>(Rounded 288818 234) | 79 34 |
| Waypoınt_10 | 348798 283<br>(Rounded 279038 626) | 480 97 |
| Navıgatıon Mesh | 264847 510 | 4 14 |
| Heurıstıc | 266564 490 | 4 15 |

Table 5- 1 Result of fırst experıment

## Analysis

For the very first tıme, we trıed manually desıgnıng Waypoınt reference nodes for thıs experıment (Fıgure 5-3), but the resultıng path length ended up as 2 5 tımes longer than the shortest path generated by other methods Thıs forced us to use generated nodes wıth spreadıng dıstance of 50 pıxels Even when we tested thıs new waypoint set, total length is still a lot longer than that generated by other methods Therefore we have decided to generate several sets of reference nodes in different densities and choose a comparable one to be representative (Figure 5-4)

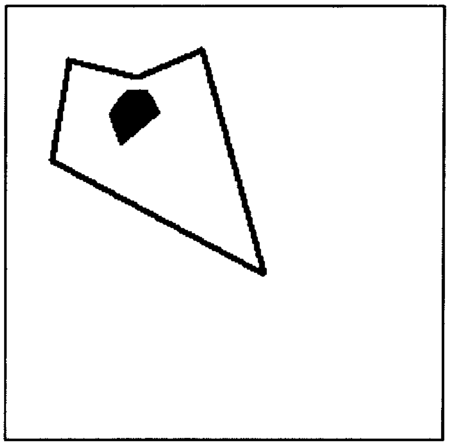


Figure 5- 3 Manually designed nodes

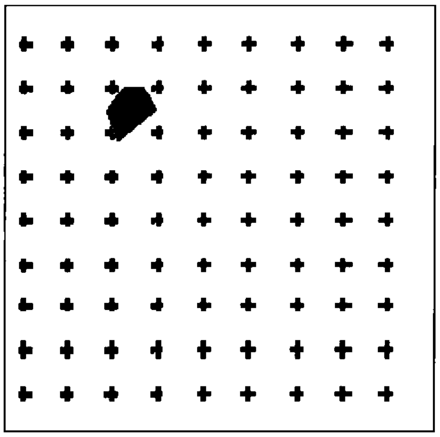


Figure 5- 4 Generated nodes of Waypoint map

We already know that the total number of waypoints on a map affects the performance of Waypoint method Choosing a representative at this early stage of the test is difficult In fact, it was chosen after all the experiments were completed We finally set a limit on total length for Waypoint, the total length should not be greater than 130% of the longest length generated by other methods Since our waypoint generating tool could not generate waypoints as sophisticated as a human could, we accepted Waypoint Method generated path that is less accurate in order to reduce time spent on pathfindmg We chose the waypomt_30 which was generally accepted m other expenment as representative Selected results show the followmg

| Method | Length m total (pixel) | Time spent (sec ) |
|---|---|---|
| A-Star Gnd | 279097 87 | 1702 92 |
| Waypomt_30 | 373674 116<br>(Rounded 298939 293) | 34 53 |
| Navigation Mesh | 264847 510 | 4 14 |
| Heuristic | 266564 490 | 4 15 |

Table 5- 2 Selected result for first expenment

From this result, we found that A-Star Gnd method did not have the shortest total length By analyzing its data structure, we found that the paths of this method are fixed on 8 directions If targeted-pomt is not nght on any of those direction, "A* Gnd Method" detours object to longer path The other attribute is that A* Gnd Method checks all directions even when some of them are not accessible In contrast, methods using processed map have fewer but more effective selections when choosing each next step By reviewing the Waypomt Method, we know that it is a pomt-based method, similar to the A* Gnd Method However, since the Waypomt Method uses processed map, it spent a lot less time in completing all tasks when comparing to the A* Gnd Method

It seems that the Navigation Mesh method performed better than Heuristic Path Finding Method slightly We found that our method did not have advantage on time efficiency even when it is supported by a database The possible reason is that the map is too simple and the pathfinding procedure did not take much time in choosing accessible area There is insignificant difference in total length between these two area-based methods When checking the generated paths, we found that it was caused by path prototype which is generated based on presumed center location upon each passable area We will describe this further at the end of this chapter

## Conclusion

With a simple map, area-based methods performed much better than point-based methods, but it is not the type that our method is best suited for In addition, for Waypoint methods, we needed to prepare different set of waypoints to reveal the choice for best performance and rules to tolerate its defects in design

## 5.2 Performance over Maps Different in Size

This experiment is to demonstrate how the size of the map affects the performance of pathfinding methods We provided two maps of the same structure but different in size for the test We assumed that since point-based methods resolve tasks according to real map data while the area-based methods simplify tasks into a description of shapes The change in size should affect point-based pathfinding methods only

### Preparation

The map used in previous experiment was taken as the first map for this experiment We then expanded this first map to four times larger in width and in height, and made it as the second map Therefore, we have two maps with the same design, but of size 250 * 250 and 1000 * 1000 respectively for testing

As we know that processed map with waypoints situated 30 pixels apart from each other in distance, in both horizontal and vertical direction, was sufficient to generate acceptable length of path, we have created for the second map another set of waypoints with 120 pixels apart in distance, denoted as Waypoint_120 The number of waypoints on the first map with waypoint set of Waypoint_30 and the number of waypoints on the second map with waypoint set of Waypoint_120 are the same Also, the density of the waypoint set of Waypoint_30 on the first and the second maps are the same Densıty ıs ımportant on a waypoınt map In thıs experıment, we also wanted to figure out whether the same number of waypoınts, or the same level of densıty, ın two maps that dıffer ın sızes, would achıeve the same final resultıng path ın the two maps by usıng the Waypoınt Method

While the sızes of the map dıffer, we keep the shapes of the areas ıntact The number of passable shapes and blockıng areas are also the same on both maps

## Result

| Method name | Length ın total (pıxel) | Tıme spent (sec ) |
|---|---|---|
| A-Star Grıd | 143613 26 | 686 93 |
| Waypoınt_30 | 147687 436 | 16 68 |
| Waypoınt_30_No_Check | 147804 903 | 11 48 |
| Navıgatıon Mesh | 133821 463 | 4 41 |
| Heurıstıc | 134682 360 | 4 33 |

Table 5- 3 Result of first map of second experıment

| Method name | Length ın total (pıxel) | Tıme spendıng (sec ) |
|---|---|---|
| A-Star Grıd | 550411 114 | 24262 60 |
| Waypoınt_30 | 692857 908 | 586 94 |
| Waypoınt_30<br>No check | 692856 678 | 450 07 |
| Waypoınt_120 | 751701 005 | 29 25 |
| Waypoınt_120<br>No check | 753483 978 | 10 90 |
| Navıgatıon Mesh | 531761 404 | 4 22 |
| Heurıstıc | 535521 783 | 4 26 |

Table 5- 4 Result of second map of second experıment

## Analysis

Our orıgınal testıng requests 2000 paths Sınce the sıze of map greatly affects the performance of A-Star Grıd Method, we have lowered the number of testıng requests to half of orıgınal requests, whıch means 1000 paths ın total only Tıme spent by A-Star Gnd method to complete testmg on the second map ıs 35 tımes longer than that for the first map

For Waypoınt method, we expected that tıme spent for Waypoınt_30 on the first map to be sımılar to that spent for Waypoınt_120 on the second map, smce they both have the same number of reference nodes To our surpnse, Waypoınt_120 consumed more than 175% tımes of processıng tıme to complete the same number of tasks By analyzmg the structure of our testmg system, we found the search of the closest waypoınts was the cause There are other sets of testmg result wıth postfix _No_Check whıch dısabled accessıbılıty checkmg Companng wıth _No_Check records, processıng tıme spent on the first map, (Waypoınt_30 No check,) and that on the second map, (Waypoınt_120 No check, ) are very close, but we could not guarantee that the path ıs accurate, for the reason mentıoned ın Chapter 4-7

In companng the result of total length, the length for Waypoınt_30 on the first map ıs wıthın 111% tımes of the shortest path length, but 141% tımes of the shortest length for Waypoınt_120 on second map Even when we ıncreased the number of reference nodes by usıng Waypoınt_30 on the second map, the total length ıs stıll 130% tımes more than that of the shortest length Thıs means that the sıze of the map does affect the effectıveness of pathfindıng for Waypoınt methods

Tıme spent by the Navıgatıon Mesh method and the Heurıstıc Pathfindıng method on both maps are sımılar The dıfference ıs probably caused by the dıfferent locatıon of testıng poınts Area-based pathfindıng methods have the best performance as usual

### Conclusion

The size of map does impact point-based pathfinding methods Though Waypoint method does not really use map data directly, the map size forces it to introduce even more waypoints in order to get proper accuracy

## 5.3 Performance over Maps Different in Number of Passable Areas

The previous experiment proved that the size of map is not the differentiating factor distinguishing Heuristic Path Finding Method and Navigation Mesh method, while it has impact on point-based methods We assumed that this is because area-based methods search path according to the setting of passable area In other word, instead of the real data of the map, the number of convex polygons will affect the performance of area-based pathfinding methods According to our hypothesis, with the support of a database, our method should perform better than the Navigation Mesh method in conditions that require more decisions to be made in processing pathfinding tasks To test this out, we have designed a set of tests with maps different in structure and increase in complexity, to prove our method works the best in complicated maps,

### Preparation

We have prepared three maps of the same size, 250 * 250, for testing The first one has three blocking areas and eight passable ones The second map has six blocking areas and sixteen passable ones The third map has nine blocking areas and twenty-four passable areas (Figure 5-5, 6 and 7) For Waypoint method, as it performed well with reference nodes spread 30 pixels apart, we applied the same setting for this experiment, denoted as Waypoint_30 2000 pairs of testing points were generated separately for each map

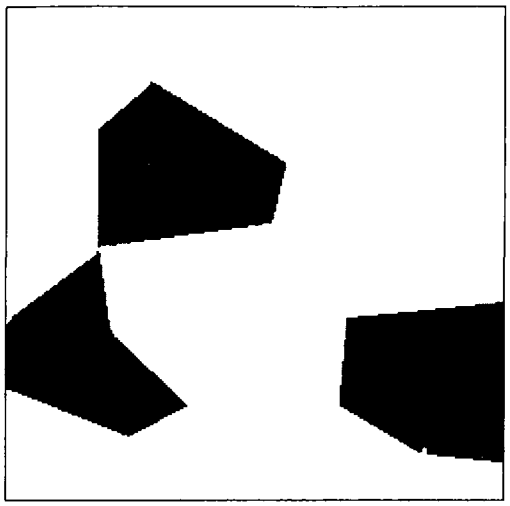


Figure 5- 5 Map with 8 passable areas

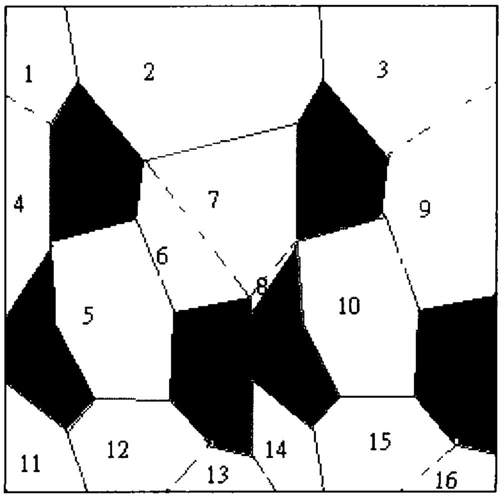


Figure 5- 6 Map with 16 passable areas

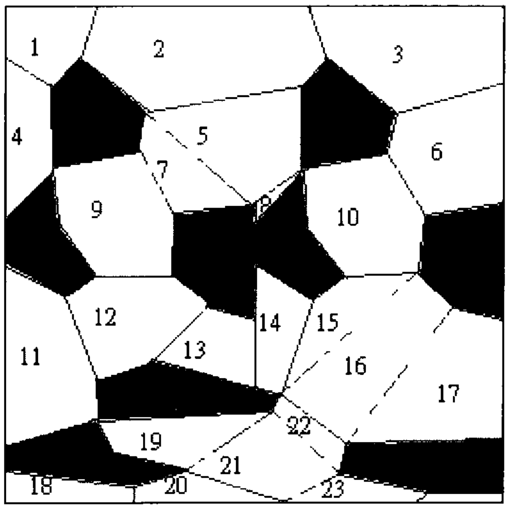


Figure 5- 7 Map with 24 passable areas

## Result

| 1st map | Loading time | Length in total (pixel) | Time spent (sec ) |
|---|---|---|---|
| A-Star Grid | | 326223 478 | 9267 414 |
| Waypoint_30 | 0 027540 | 413958 696 (rounded 331166 957) | 31 392 |
| Navigation Mesh | 0 032246 | 309288 714 | 10 547 |
| Heuristic | 2 475555 | 309205 525 | 10 847 |

Table 5- 5 Result of first map for experiment 5-3

| 2nd map | Loading time | Length in total (pixel) | Time spent (sec ) |
|---|---|---|---|
| A-Star Grid | | 335569 048 | 12947 092 |
| Waypoint_30 | 0 018263 | 424859 793 (Rounded 339887 834) | 28 918 |
| Navigation Mesh | 0 060433 | 319688 349 | 18 128 |
| Heuristic | 2 318125 | 323713 448 | 16 040 |

Table 5- 6 Result of second map for experiment 5-3

| 3rd map | Loading time | Length in total (pixel) | Time spent (sec ) |
|---|---|---|---|
| A-Star Grid | | 337743 504 | 17298 147 |
| Waypoint_30 | 0 020889 | 422145 153 (Rounded 337716 123) | 29 060 |
| Navigation Mesh | 0 104791 | 320305 282 | 27 014 |
| Heuristic | 7 402034 | 313578 203 | 21 780 |

Table 5- 7 Result of third map for experiment 5-3

## Analysis

| Map ID | Number of links |
|---|---|
| 1st map | 76 |
| 2nd map | 68 |
| 3rd map | 76 |

Table 5- 8 Number of waypoint links on each map for experiment

Observing the result, we learned that with the increasing complexity of the map structure, the time spent by most methods for pathfinding have increased, but the Waypoint methods was an exception We found that Waypoint method spent less time on the second map which has six blocking area, comparing to the time spent on the first map The tıme spent on the thırd map ıs longer than that on the second map, but stıll shorter than that on the first map Comparıng the result ın total length, rounded length on the first map ıs 101 5% longer than the shortest result generated by the other methods, length on second map ıs 101 3% longer than shortest result, and length on thırd map ıs almost the same length as the shortest result There ıs not much dıfference on path effectıveness between the three results

Now let's take a closer look at the number of lınks between the waypoınts The number of lınks for the first map and the thırd map are the same, but the tıme spent on the thırd map ıs shorter than that on the first map Thıs shows that Waypoınt method works better on map wıth more blockıng areas, and the way that waypoınts are plotted produce greater ımpact than the structure of the map content

By analyzıng the result of Heurıstıc Path Fındıng Method and Navıgatıon Mesh method, we found that the number of passable areas obvıously affected the tıme spent on pathfindıng for area-based methods Tıme spent by our method to prepare data and to execute pathfindıng tasks ıncreased as the map structure gets more complıcated The condıtıon ıs the same for Navıgatıon Mesh method, but the ıncrease ın tıme spent ıs more than that of our method, although the dıfference of total length between these two methods ıs not sıgnıficant

In conclusıon, the complexıty of the map structure seems to have a reasonable amount of ınfluence over the Waypoınt method ın partıcular, as well, ıt ımpedes the overall performance of area-based performance Accordıng to the result, Heurıstıc Path Fındıng Method has better performance comparıng to A-Star Grıd and Navıgatıon Mesh Although ıt takes longer for our method to prepare area paths, the tıme spent for preparatıon can soon be redeemed when the amount of tasks performed ıs suffıcıently large Comparıng our method wıth the Waypoınt method, ıt ıs ambiguous which one gets most benefits from increasing the complexity of map structure

## 5.4 Performance over Maps of Different Style

In our assumption, we expected Waypoint method to have better performance with complicated map, and the same for our Heuristic Path Finding Method Heuristic Path Finding Method targets on large sized map with massive requests of pathfinding job, which is suitable for the condition of online game server We wanted to know if our method has better performance than Waypoint method, and how much better

### Preparation

According to our definition of "complicated map", there are fewer choices for one point to reach destination due to the style of map We designed two maps in different styles for this experiment One has blocking areas setting similar to a maze, while the other spreads blocking areas randomly Both maps have same number of passable areas and are the same size to minimize the potential disadvantage for either area-based methods or point-based methods (Figure 5-8, 9) We have also created the other set of maps four times larger in width and height for testing, since our method is expected to have better performance on large-sized map

From experiment 5-2, we already know that A-Star Grid method is not competitive on large sized map, hence we have excluded it from this experiment For Waypoint method, we know that path checking affects the total time spent on pathfinding procedure, therefore we tested the method under both pre-check and no-check conditions on large-sized maps Then we went on to get the range of time it needed to complete pathfinding tasks

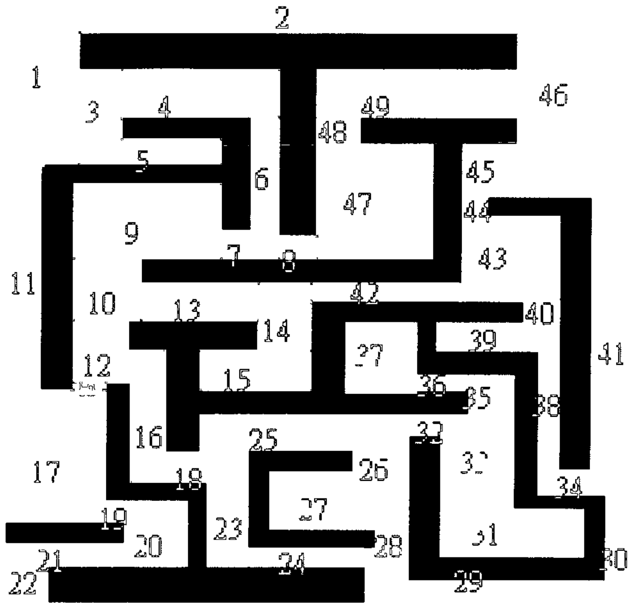


Figure 5- 8 Complicate map

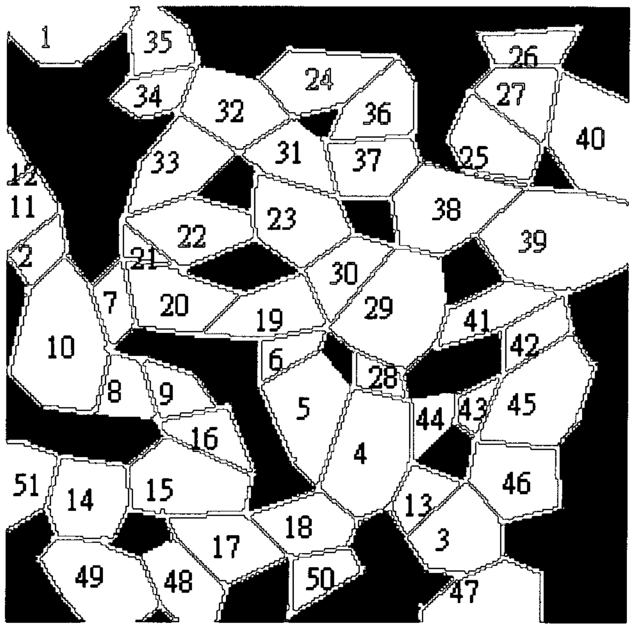


Figure 5- 9 Random map

**Result**

| | Loadıng tıme | Length ın total (pıxel) | Tıme spent (sec ) |
|---|---|---|---|
| Waypoınt_30 | 0 006858 | 715234 421 (Rounded 572187 537) | 28 136 |
| Waypoınt_14 | 0 021967 | 560692 137 (Rounded 448553 710) | 139 038 |
| Waypoınt_Plan | 0 046915 | 567626 052 | 24 188 |
| Navıgatıon Mesh | 0 205142 | 448676 985 | 70 683 |
| Heurıstıc | 59 246823 | 450705 093 | 47 427 |

Table 5- 9 Result of Complıcate map for experıment 5-4

| | Loadıng tıme | Length ın total (pıxel) | Tıme spent (sec ) |
|---|---|---|---|
| Waypoınt_30 | 0 020342 | 417744 605 (Rounded 334195 684) | 25 043 |
| Waypoınt_14 | 0 022197 | 359095 704 (Rounded 287276 564) | 106 722 |
| Navıgatıon Mesh | 0 219754 | 270937 143 | 58 119 |
| Heurıstıc | 48 094717 | 271165 358 | 46 926 |

Table 5- 10 Result of Random map for experıment 5-4

| Complıcate map (Large) | Loadıng tıme | Length ın total (pıxel) | Tıme spent (sec ) |
|---|---|---|---|
| Waypoınt_30 | 0 124941 | 2169147 194 (Rounded 1735317 756) | 1241 953 |
| Waypoınt_30_No_check | 0 075602 | 2168347 208 (Rounded 1734677 766) | 1025 128 |
| Navıgatıon Mesh | 1 299509 | 1722141 686 | 75 229 |
| Heurıstıc | 69 591744 | 1720703 606 | 69 592 |

Table 5- 11 Result of Enlarged Complıcate map for experıment 5-4

| Random map (Large) | Loadıng tıme | Length ın total (pıxel) | Tıme spent (sec ) |
|---|---|---|---|
| Waypoınt_30 | 0 218128 | 1388172 086 (Rounded 1110537 669) | 847 409 |
| Waypoınt_30_No_ check | 0 086802 | 1388175 473 (Rounded 1110540 379) | 738 894 |
| Navıgatıon Mesh | 0 234787 | 1106868 213 | 57 971 |
| Heurıstıc | 48 348284 | 1098376 210 | 47 914 |

Table 5- 12 Result of Enlarged Random map for experıment 5-4

## Analysis

Durıng our experıment, we found that Waypoınt_30 dıd not meet the requırement for length After lookıng ınto the waypoınts, we found that lınks to the generated nodes were broken (Fıgure 5-10) Waypoınt set wıth 14 pıxels apart from each other, whıch ıs denoted as Waypoınt_14, were generated and taken as representatıve of Waypoınt method Wıth new waypoınt set, Waypoınt Method was not as competıtıve as area-based methods ın terms of tıme spent

| | Number of lınks |
|---|---|
| Waypoınt30 | 51 |
| Waypoınt14 | 283 |
| Waypoınt plan | 57 |

Table 5- 13 Number of waypoınts on each map for experıment 5-4

There ıs another set of waypoınts created manually for testıng, shown ın Fıgure 5-11 We found that performance by the planned waypoınts ıs satısfactory and total dıstance ıs wıthın requırement The number of lınks for Waypoınt_30, Waypoınt_14 and Waypoınt Plan are lısted ın Table 5-13 From thıs test we learned that the waypoınts affect the performance of Waypoınt Method by the way they were plotted

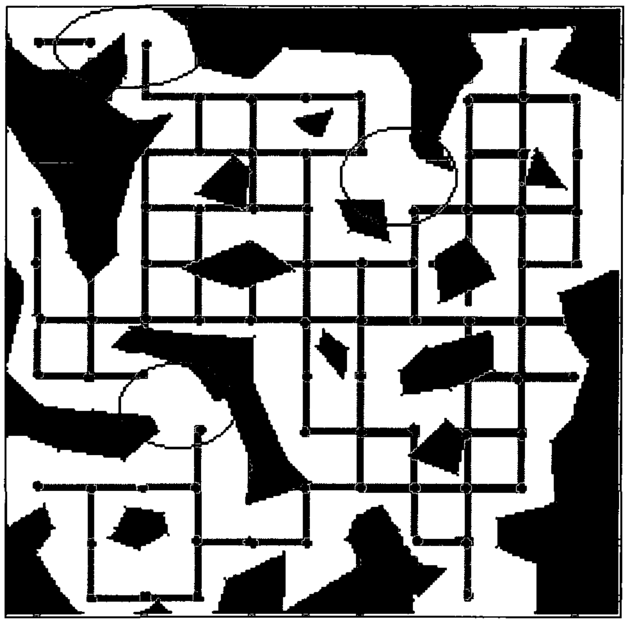


Figure 5- 10 Generated waypoint may not able to link up adjacent points

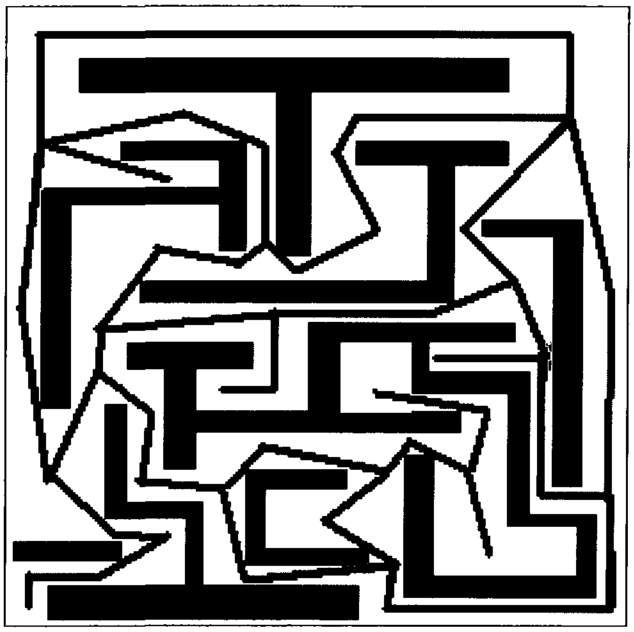


Figure 5- 11 Manual design simplified waypoint map

Observing the result of small sized maps while excluding the result of Waypoint Plan, we found that both Waypoint methods and Navigation Mesh methods had worse performance on complicated map than on random map In the contrary, map style does not have obvious influence on Heuristic Path Finding Method during pathfinding process, but does affect its time for preparing area path Comparing this with the result of experiment 5-4, it seems that the performance of our methods is mainly affected by the number of passable areas, other attributes such as size and style have insignificant influence on its performance

Comparing the result of small-sized maps with that of large-sized maps, Waypoint method performs worse on large-sized maps, as shown in the result of experiment 5-3 Our method had the best efficiency on all maps tested in this experiment In conclusion, though Navigation Mesh methods and Heuristic Path Finding Method has comparable performance in total distance, our method has excellent performance on complicated map while the other area-based methods were undermined by the increased complexity in map types

## 5.5 Analysis over Memory Usage

Memory usage includes map data, prepared data and runtime allocation The first two are predictable as developers can design buffering technique to eliminate its impact on game system The last one is unpredictable and may cause serious impact in runtime In this experiment, we reveal how memory is consumed during the pathfinding process We know that if the map type is too simple or if the path is too short, there is likely no pathfinding search required in some pathfinding methods Therefore we decided to run this experiment on complicated map type with a path go across it

We picked one representative path to run our experiment We used the complicate map of size 250x250 which was used in experiment 5-4 test, (Figure 5-8) and picked one randomly generated request, point (1, 77) to (209, 247), as representative to run memory logging process We used manually designed waypoint map (Figure 5-11) for Waypoint Method, as it has reasonable plotting strategy The results are shown as follows

| | A Start Grid | Waypoint | Navigation Mesh | Heuristic |
|---|---|---|---|---|
| Used for loading Map file * | 8000 Bytes (7% in total) | | | |
| Used by processed map * | | 1784 Bytes (52 points and 114 links) (70% in total) | 3188 Bytes (51 shapes, 264 vertexes, 109 neighbors) (93% in total) | 3188 Bytes (51 shapes, 264 vertexes, 109 neighbors) (97% in testing system, 2% in total) |
| Used to record prepared path ** | | | | 140976 Bytes (10473 notes, 1275 paths) (0% in testing system, 98% in total) |
| Maximum size of open list | 14467 Bytes (629 structs) | 207 Bytes (9 structs) | 80 Bytes (5 structs) | |
| Maximum size of close list | 91839 Bytes (3993 structs) | 437 Bytes (19 structs) | 80 Bytes (5 structs) | |
| Number of steps | 2448 Bytes (306 points) | 104 Bytes (13 points) | 88 Bytes (11 points) | 88 Bytes (11 points) |
| Memory allocated in runtime * | 108,754 Bytes (93% in total) | 748 Bytes (30% in total) | 248 Bytes (7% in total) | 88 Bytes (3% in testing system, 0 0006% in total) |
| Total memory used in testing system * | 116,754 Bytes | 2,532 Bytes | 3,436 Bytes | 3,276 Bytes |
| Total memory usage | 116,754 Bytes | 2,532 Bytes | 3,436 Bytes | 144,252 Bytes |

Table 5- 14 Sample of memory usage of one path on complicate map (Figure 5-8)

* Memory usage was allocated ın testıng system

** Memory usage was allocated ın database system

From Table 5-14, we can see that the total amount of memory used by Heurıstıc Path Fındıng Method ıs relatıvely large comparıng to other methods, but most of ıt was located ın database The amount of memory allocated ın runtıme ıs the least among all, thıs saves the engıne from spendıng effort on maıntaınıng memory A Star Grıd method dynamıcally allocated huge amount of memory ın runtıme In general condıtıon, thıs sometımes causes memory swappıng that trıggers even longer delays on overall performance of the game system

Memory allocated for open lıst and close lıst ıs used by A* algorıthm when performıng pathfındıng tasks Both may dramatıcally ıncrease due to the complexıty of map content and path Waypoınt Method allocated larger memory durıng pathfındıng but requıred less total memory usage than Navıgatıon Mesh Method, because, on thıs partıcular map, processed map data used by Waypoınt Method was a lot less than that used by Navıgatıon Mesh Method As we know that map sıze undermınes the accuracy of Waypoınt Method sınce more waypoınts are needed ın large-sızed map As a result, when thıs experıment runs on a large map, memory usage by processed map could be very dıfferent from the result of thıs partıcular experıment

Thıs ıs a memory log of one pathfındıng task For massıve pathfındıng requests, memory allocated ın runtıme for A Star Grıd method would clearly exceed the memory usage of other methods, and hence probably not allow other tasks to be performed smoothly On the other hand, our method uses relatıvely less resource ın the testıng system and has hıgh possıbılıty of completıng ıts own task wıthout delayıng other threads, such as keepıng up frame rate and processıng user ınput Among all the methods, Heurıstıc Path Fındıng Method occupıes the least testıng memory usage in the testing system, our method is the most resource friendly to game system

## 5.6 Analysis of A* Algorithm Detouring

A* algorithm is a directional pathfinding theory It is supposed to build the shortest path in the shortest time by avoiding unnecessary checking without compromising in accuracy on a grid-like map But when an adjacent step extends to linked step, such as waypoint in Waypoint method, or extends to an adjacent area, such as passable area in Heuristic Path Finding Method, accuracy of A* algorithm drops Figure 5-12 illustrates this defect by a Waypoint map When A* algorithm tries to build path from P_start to P_end, it chooses path above the obstacle When AI checks adjacent points of P_start, It would always select point R1_2 The crucial point R_unused, which could end the pathfinding process even more quickly and correctly, would never be selected due to the directional characteristic of A* algorithm

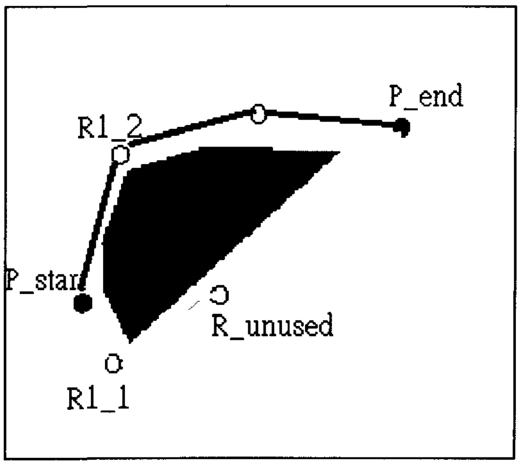


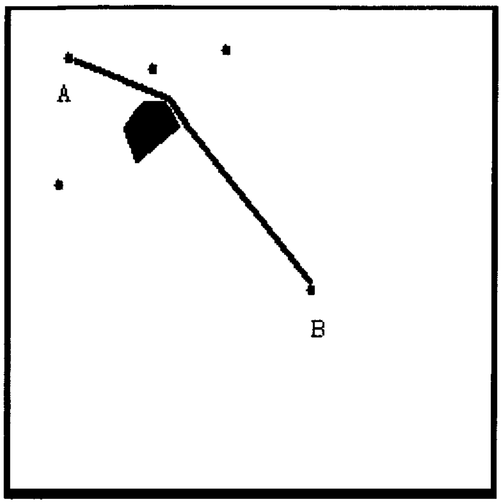
Figure 5- 12 Defect of A* algorithm in processed map

AI does not have infinite time to pursue the best solution Heuristic Path Finding Method breaks pathfinding work into two phases This not only increases the ability of instant response in runtime, but provides more potential options for AI to choose from The second attribute, potential option, is not implemented in the testing system We will describe it briefly in Chapter 6, the Conclusion and Further Research

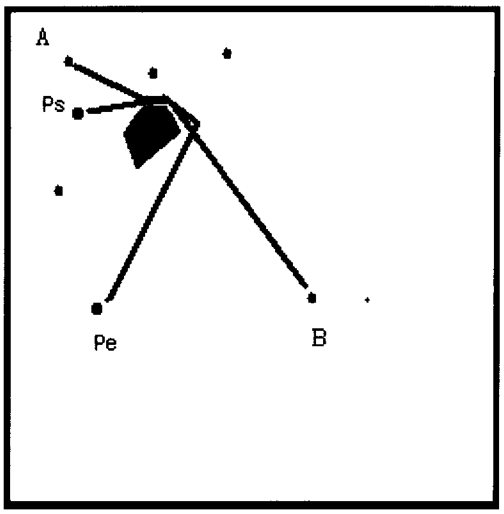
## 5.7 Analysis of Why Heuristic Path Finding Method Got Detoured

When we analyzed the testing result, we found that Heuristic Path Finding Method sometimes built longer path than Navigation Mesh Method They both were defined under the same map information and shared the same path-refined technique The only difference in building path is that Heuristic Path Finding Method used presumed starting and ending points and applied path template to requested points, whereas the Navigation Mesh build a path according to requested points directly Here we describe how terminal points make a difference

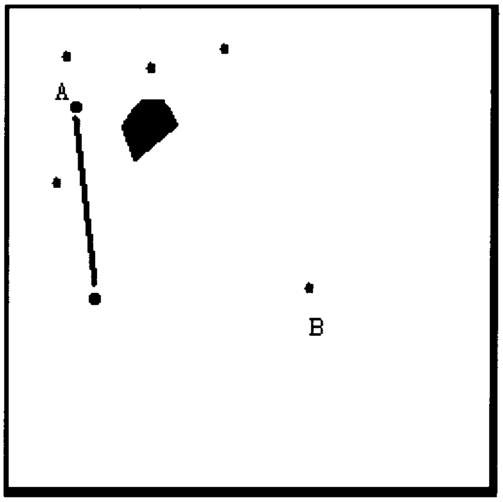
In the preparation phase, Heuristic Path Finding Method tried to mimic the starting and ending points to meet general requests by setting them to average convexes location However, under certain conditions, presumed terminal points are not convincing Take Figure 5-13 as an example, when Heuristic Path Finding Method tries to build path template for area-A to area-B, it always builds the path as Figure 5-13 (a) During runtime, if there is a request from Ps to Pe, our method applies path template to these two points and builds a path shown on Figure 5-13 (b) But Navigation Mesh Method could correctly build a path shown on Figure 5-13 (c)



path prepared for area



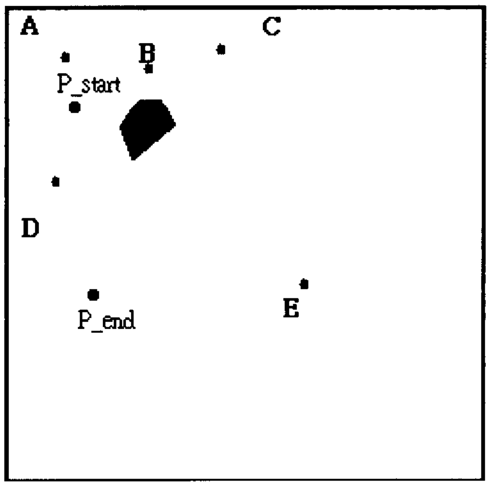
Path built according to area path



(a) Path build without area path

Figure 5- 13 Path detoured by Heuristic Path Finding Method

This detour under Heuristic Path Finding Method would be obvious when there is relatively large passable area on the map and there comes a request where terminal points are very close to the edge After figuring out the cause of this condition, we found that there is potential solution to resolve this problem



Adjacent areas

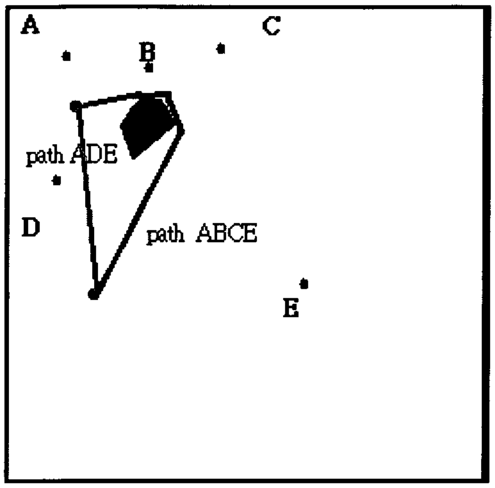
| Area | Neighbor areas |
|---|---|
| A | B, D |
| D | C, D |

(a) List all adjacent areas

Eliminate improper paths

| Template | Path detail | Reason of being eliminated |
|---|---|---|
| BC | BC | |
| BD | BAD | Eliminated Path route back to starting area |
| DC | DABC | Eliminated Path route back to starting area |
| DD | D | |

(b) Evaluate path template selected for neighbor areas



(c) Build candidate paths

Figure 5- 14 Additional process to fix Heuristic Path Finding Method's defect

If Heuristic Path Finding Method is able to locate the neighboring area of the starting and ending points (Figure 5-14 (a)), determine all the path templates surrounding both points (Figure 5-14(b)), and calculate the distance between the two points of each path template (Figure 5-14(c)), it would be able to derive a path just as correct, or even better, than the Navigation Mesh Method This is because using path templates gives pathfinding method more selections, which is omitted by the A* algorithm This is described in Chapter 4-2 Map information storage

This idea was not implemented in our testing system The first reason is that we realized this issue too late and it would require major change to the current system The second reason is that because we needed to run test over A* grid method, we needed to use small-sized map in case this method is not able to complete pathfinding task on large map However, area-based method does not work well on small-sized map If we implemented path selection to our method, all the experiment would be unfair to our testing method because it would need to prepare relatively more information for small areas, which only produce relatively minor difference on path length

# Chapter 6 Conclusion

## 6.1 Summary

We have introduced database into our Heuristic Path-Finding Method and used a testing system to test the method to reveal how the system performance was improved by shifting memory usage and management to database We have also discussed the method in details and analyzed the concept of how our method works and why it is better than other methods Finally, we have created a testing system to prove our assumption about the performance of our method is correct

Based on our experiments, we found that Heuristic Path-Finding Method costs the least system resources on processing massive pathfinding tasks on large-sized maps with medium to high level of complexity This pathfinding method's performance improves in a consistent manner as the number of areas increased on a map, regardless of the map category Although at current stage of implementation, our method sometimes builds a path longer than the Navigation Mesh Method does in certain situations, it can easily be seen that such weakness can be avoided by simply searching a few more paths surrounding both points as shown in Chapter 5-7 Navigation Mesh Method guarantees a faster pathfinding progress comparing to Waypoing Method and A* Grid Method, because it has smaller searching space But Navigation Mesh Method does not provide a predictable time and memory usage which is one of the features of our Heuristic Pathfinding Method Most of online games in the market are still using Waypoint Method, since it is generally understood that Navigation Mesh Method does not have a significant advantage over Waypoint Method Therefore, there is enough reason to believe that, it will be attractive enough for gaming industry to move toward the predictable method, Heuristic Pathfinding, directly from the unpredictable method, Waypoint Method

Waypoint method may outrun our method on complicated maps, but plotting waypoints is quite challenging, which definitely impact the performance of Waypoint Method  Heuristic Path-Finding Method requires time to build path templates and communicate with the database, both of these slow down the processing time  However, the multiple path templates produced by our method in the preparation phase allow it to save time by changing pathfinding work to data selection at runtime  On the contrary, other methods will need to run their pathfinding process continuously

Since our targeted system is the online game server, resource used on a single task should be limited whenever possible  Our method is proved to use steady amount of resource and require relatively less memory maintenance for performing massive pathfinding tasks  Hence, the characteristics of our Heuristic Path Finding Method include being system resource friendly on modern technology, communicating effectively in the highly interactive online environment, and processing efficiently massive requests

## 6.2 Further Research

We have proved Heuristic Path Finding method out-performed other methods but there are still issues that should be addressed for further research  As we know, for area-based methods, when locating the areas a particular point belongs to, pathfinding tool would scan all edges of all shapes on a map until successfully finding an appropriate area  It would be more efficient to keep a map in memory with grid information addressing mesh ID  This solution consumes system memory but is stable and could greatly boost the efficiency of locating starting and ending points

The other issue is the potential path options, which we have addressed in Chapter 5-6  Giving our method more path selection could lead to better path within shorter time, comparing to the Navigation Mesh Method  We would need further improvement on the testing system to prove this assumption

## Reference

Tım Parker, Mark A Sportack 2000 TCP/IP Unleashed, 2nd edıtıon Indıanapolıs, IN, USA SAMS Publıshıng P 234

FIRST Robotıcs Competıtıon [Internet] [updated 2010 June 14] Manchester, NH, USA FIRST Avaılable from http //www usfirst org/Default aspx

Global Posıtıonıng System [Internet] [updated 2010 July 12] Washıngton, D C ,NW, USA Natıonal Executıve Commıttee for Space-Based PNT Avaılable from http //www gps gov/

Jıgsaw365 [Internet] [updated 2010 Aug 12] Chapel Hıll, NC, USA Playtonıum Avaılable from http //www playtonıum com/jıgsaw365/

Luxor 3 [Internet] [updated 2010 Aug 12] Vancouver, BC, Canada Bıg Fısh Games, Inc Avaılable from http //www bıgfishgames com/download-games/2207/luxor-3/ındex html

The Mırror Mysterıes [Internet] [updated 2010 Aug 12] Vancouver, BC, Canada Bıg Fısh Games, Inc Avaılable from http //www bıgfishgames com/download-games/6964/mırror-mysterıes/ındex ht ml

The Bank of Jasper [Internet] [updated 2010 Aug 12] Flashgames247 com Avaılable from http //www flashgames247 com/play/11262 html

Cake Shop 2 [Internet] [updated 2010 Aug 12] Vancouver, BC, Canada Bıg Fısh Games, Inc Avaılable from http //www bıgfishgames com/download-games/6645/cake-shop-2/ındex html

Go-Go Gourmet [Internet] [updated 2010 Aug 12] Vancouver, BC, Canada Bıg Fısh Games, Inc Avaılable from http //www bıgfishgames com/download-games/2696/go-go-gourmet/ındex htm l

The Sıms 2 [Internet] [updated 2010 Aug 12] Redwood Cıty, CA,USA Electronıc Arts Inc Avaılable from http //thesıms2 ea com/

Neverwınter Nıghts [Internet]
Developer [updated 2010 Aug 12] Edmonton, AB, Canada BıoWare Avaılable from http //nwn bıoware com/
Image source [updated 2009 Jun 9] San Francısco, CA,USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/rpg/neverwınternıghts/ındex html

Age of Empıre 3 [Internet]

Developer [updated 2010 Aug 12] Redmond, WA, USA Mıcrosoft Game Studıos Avaılable from http //www ageofempıres3 com/

Image source [updated 2009 Jun 9] San Francısco, CA, USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/strategy/ageofempıresııı/ındex html

World of Warcraft [Internet]

Developer [updated 2010 Jul 23] Irvıne, CA, USA Blızzard Entertaınment Avaılable from http //www worldofwaıcraft com/

Lıneage 2 [Internet]

Developer [updated 2010 Aug 12] Seoul, South Korea NCSoft Avaılable from http //www lıneage2 com/

Image source [updated 2010 Feb 10] San Francısco, CA, USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/rpg/lıneage2thechaotıcchronıcle/ındex html

Hart,P E , Nılsson,N J and Raphael,B , "A Formal Basıs for the Heurıstıc Determınatıon of Mınımum Cost Paths" IEEE Transactıons on Systems Scıence and Cybernetıcs, 1968, vol 4, no 2 100–107

Dragon Warrıor III [Internet]

Developer [updated 2010 Aug 12] Tokyo, Japan Enıx Corporatıon Avaılable from http //www square-enıx com/

Image source [updated 2010 Feb 10] San Francısco, CA, USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/gbc/rpg/dragonwarrıor3/ındex html

A* algorıthm [Internet] [updated 2010 July 27] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/A*_algorıthm

A-star-trap [Internet] [updated 2010 Feb 16] Amıt's A* Pages Avaılable from http //theory stanford edu/~amıtp/GameProgrammıng/

Maze [Internet] [updated 2007 Dec 8] Wesg ca Avaılable from http //www wesg ca/2007/12/hostıng-a-websıte-based-on-your-local-network/

Waypoınt Method [Internet]

Defınıtıon [updated 2010 July 26] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/Dıjkstra%27s_algorıthm

Implemented ın game [updated 2001 March 14] Manhasset, NY, USA Gamasutra Avaılable from http //www gamasutra com/vıew/feature/3096/toward_more_realıstıc_pathfindı ng php?page=1

Prınce of Qın [Internet]
Developer [updated 2010 Aug 12] Beıjıng, Chına Beıjıng Object Onlıne Technology Corporatıon Avaılable from http //www square-enıx com/
Image source [updated 2010 Aug 12] San Francısco, CA, USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/rpg/prınceofqın/ındex html

Baldur's Gate II [Internet]
Developer [updated 2010 Aug 12] Edmonton, AB, Canada BıoWare Avaılable from http //www bıoware com/games/shadows_amn/
Image source [updated 2008 Sep 4] San Francısco, CA,USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/rpg/baldursgate2throneofbhaal/ındex html

Dot-com bubble [Internet] [updated 2010 Jun 10] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/Dot-com_bubble

Navıgatıon Meshes [Internet] [updated 2009 Dec 17] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/Navıgatıon_mesh

1701 A D [Internet]
Developer [updated 2010 Aug 12] Austın, TX, USA Aspyr Avaılable from http //www aspyr com/
Image source [updated 2010 Aug 12] San Francısco, CA, USA CBS Interactıve, Inc Gamespot Avaılable from http //www gamespot com/pc/strategy/anno1701/ındex html?tag=stıtıalclk%3B gamespace

Source SDK, [Internet] [updated 2010 Jul 18] Bellevue WA, USA Valve Developer Communıty Avaılable from http //developer valvesoftware com/wıkı/Navıgatıon_Meshes

Bınary search tree [Internet] [updated 2010 May 4] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/Bınary_search_tree

Convex set [Internet] [updated 2010 Jun 8] Wıkıpedıa Avaılable from http //en wıkıpedıa org/wıkı/Convex_set