Predicting Ship Movement Using A* Algorithm in Anno 1800 Strategy

Muhammad Farid Adilazuarda/13518040
Informatics
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
faridadilazuarda29@gmail.com

***Abstract*—Anno 1800 is a city-building real-time strategy video game published by Ubisoft. Anno 1800 takes place in the 19th century at the dawn of Industrial Age. The main goal of this game is to conquer other player's territory by War or by Economical approach. As a part of Industrial Age-themed game, Anno 1800 came with ship expedition and ship trading for its main feature of the game. In the cause of it, predicting ship movement is a major part of winning this game. This paper will cover abut the concept of A* algorithm as well as its heuristic function and its implementation in predicting the Anno 1800's ship movement.**

***Keywords*—Anno 1800, A* algorithm, heuristic;**

I. Introduction

In this modern era, games are a form of play usually done for the purpose of enjoyment and relaxation. Apart from those intentions, people also play games to obtain the sense of achievement and challenge, or even reward, whether from playing alone, with friends, or against other people. The three major components that compose a game are rules, challenge, and interaction. Games provide either physical or mental stimulation, or both at the same time, which in turn helps in developing the players' practical skill and perform an educational, psychological, or simulational role.

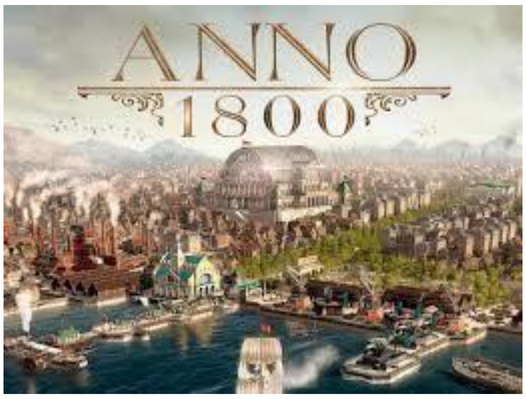


Image 1. Logo of Anno 1800 Game (source : https://gamebrott.com/meski-eksklusif-di-epic-store-jumlah-pemain-anno-1800-di-steam-ditaksir-mencapai-setengah-juta-lebih)

An example of an exciting yet challenging game is Anno 1800. Anno 1800 is a game where the players are supposed to beat each other by taking other's territory by firepower or by economical power in order to finish the game. Anno 1800 can be played offline with bots in order to challenge the player before joining the real world battle. The higher the difficulty, the smarter the AI of bots will be.[1]

The challenge of this game is the strategy if taking other player's territory/continent can't be done without ships. That is what Anno game is, the Anno's developer doesn't want the players only gather as much firepower as they can and then just blow each enemy's continent, it's more than that. Anno's developer wants the player make a very deep strategy that effect the flow of the game in the long period of the game. That is why ships is very important in this game.

The importance of ships in this game can be seen on every aspect of the game, for example if you want to take a territory by economical power, you have to improve your economical power by make a trade route, sell goods to other player, or even make a deals with pirates. In the other word if you want to take other player's territory by firepower, you have to build a very strong fleet in order to defeat enemy defenses.

With that said, there is a way to make the game won easier and make players could arrange their strategies more efficiently. In this paper, we will be providing one such method by predicting the ship's movement on the map by calculating the fastest path the ship will use to intercept the enemy's strategy. We will be implementing it using the concept of A* algorithm.

II. Theorical Framework

A. A* Algorithm

A* (pronounced "A-star") is a graph traversal and path search algorithm, which is often used in computer science due to its completeness, optimality, and optimal efficiency. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A* is still the best solution in many cases.

Informally speaking, A* Search algorithms, unlike other traversal techniques, it has "brains". What it means is that it is really a smart algorithm which separates it from the other conventional algorithms. This fact is cleared in detail in below sections. And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation).

What A* Search Algorithm does is that at each step it picks the node according to a value-'**f**' which is a parameter equal to the sum of two other parameters – '**g**' and '**h**'. At each step it picks the node/cell having the lowest '**f**', and process that node/cell.[2]

We define '**g**' and '**h**' as simply as possible below **g** = the movement cost to move from the starting point to a given goal, following the path generated to get there. **h** = the estimated movement cost to move from that given goal to the final destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess. We really don't know the actual distance until we find the path, because all sorts of things can be in the way (walls, water, etc.). There can be many ways to calculate this 'h' which are discussed in the later sections.

A* algorithm is a combination of UCS algorithm and Greedy Best First Search algorithm. That is why the evaluation function used by A* algorithm to calculate the shortest path to goal are Makalah IF2211 Strategi Algoritma, Semester II Tahun 2019/2020 cost to reach current node and an estimated cost from current node to goal. The heuristic function h(n) is an estimate of the minimum cost from any vertex n to the goal. Dijkstra is a special case where h(n) is equal to zero.

We must decide the heuristic function carefully, for if the heuristic function h(n) always underestimates (smaller than) the true cheapest cost from vertex n to a goal (h*(n)), A* is guaranteed to find an optimal solution. So it is safe to conclude that, if the heuristic is accurate, then the algorithm will be efficient. If not, then the algorithm may be a worse way to find path than using Dijkstra's algorithm.[3]

There are generally three approximation that can be used to calculate heuristic functions. They are manhattan distance, diagonal distance, and euclidean distance.

A* algorithm is often used for the common pathfinding problem such as

1. **Video Game :** Search the shortest path for unit mobilizatoins.
2. **Route Planning :** Plan the shortest path to travel from a certain location to a destination.
3. **Natural Lannguage Processing** : Find the closest prediction to a possible solution

B. Heuristic Search

Heuristic search refers to a search strategy that attempts to optimize a problem by iteratively improving the solution based on a given heuristic function or a cost measure. A heuristic search method does not always guarantee to find an optimal or the best solution, but may instead find a good or acceptable solution within a reasonable amount of time and memory space. Several commonly used heuristic search methods include hill climbing methods, the best-first search, the A* algorithm, simulated-annealing, and genetic algorithms (Russell and Norvig 2003). A classic example of applying heuristic search is the traveling salesman problem (Russell and Norvig 2003).[3]

C. Manhattan Distance

The distance between two points measured along axes at right angles. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2).

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

D. Diagonal Distance

The diagonal of a square is a line drawn from one corner to the corner across and at the other side of the square. The length of the diagonal of any rectangle equals the square root of the sum of the squares of its length and width. A square is a rectangle with all sides of equal length, so the diagonal's length is the square root of twice the square of a side, which simplifies to the square root of two multiplied by the length of a side. We can compute the length of the diagonal merely by multiplying the length of a side by this constant.

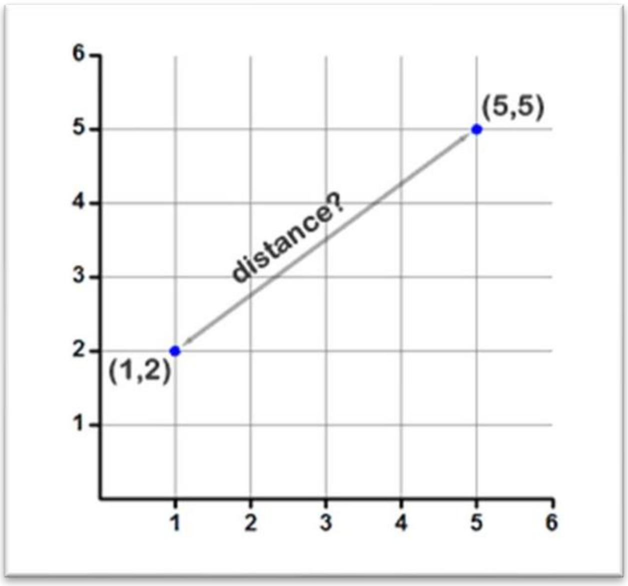
E. Euclidean Distance

The Euclidean distance between two points in either the plane or 3-dimensional space measures the length of a segment connecting the two points. It is the most obvious way of representing distance between two points.

The Pythagorean Theorem can be used to calculate the distance between two points, as shown in the figure below. If the points (x1,y1)(x1,y1) and (x2,y2)(x2,y2) are in 2-dimensional space, then the Euclidean distance between them is

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

X1 = horizontal position of the startingcoordinate
Y1 = vertical position of the starting coordinate
X2 = horizontal position of the destined coordinate
Y2 = vertical position of the destined coordinate



F. The Anno 1800 Game

Anno 1800 is a *real-time based* strategy game, developed by Blue Byte and published by Ubisoft, and launched on April 16, 2019 in North America. It is the seventh game in the *Anno* series, and returns to the use of a historical setting following the last two titles, *Anno 2070* and *Anno 2205*, taking place during the Industrial Revolution in the 19th century. Following the previous installment, the game returns to the series' traditional city-building and ocean combat mechanics.

Below is the image from the scene of the game trailer that was published in the Ubisoft's YouTube channel.

Image 3. Captured scene of Anno 1800 Game Trailer (source : https://youtu.be/UowsqoV0egc)

This game take place in 19th century in the dawn of the Industrial Age. *Anno 1800* takes place in the 19th century at the dawn of the Industrial Age.[2] Like other Anno series games, *Anno 1800* is a city-building and strategy game. While it is set in the context of colonial trade, the featured architecture is Victorian Era and the economic engine is factory labor. The *Anno 1800* is set in the Old World, where the needs of the citizens, workers and artisans are central to the management of production and supply chains. A parallel New World city exists, which produces products that laborers in the Old World want to purchase, thus trade routes need to be established. Unlike its colonial 18th century predecessor *Anno 1701*, the game has a blueprint feature[3] that helps the player to plan out the city.[5]

The game also features a story campaign, a sandbox mode, and multiplayer mode. Like *Anno 2205*, the game features multisession gameplay, though unlike its predecessor, combat and city-building sessions are not separated. *Anno 1800* integrates into a classic city-building game randomly generated maps and trade routes, artificial intelligence (AI) opponents that build on the same map as the player, and a military.

Some important object of the game that is frequently used in this game :

1. Map

   Map contains the area of the game that consist of islands, legends, player positions, and object/entity position

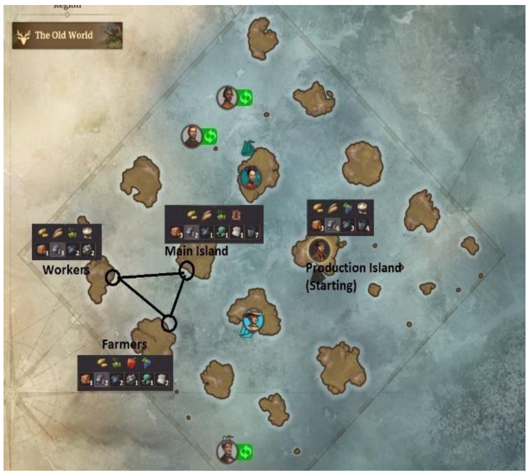


Image 3. Anno 1800 game map

2. Ships

   Ships are an essential part of Anno 1800 and the backbone of both the trading and warfare systems in the game.

Image 4. Captured scene of Anno 1800 Game Trailer (source : https://www.reddit.com/r/anno/comments/bmon72/i_present_to_you_the_great_eastern/)

3. Island

   Every game starts in an island, which should be a place of residence for majority of the population.

Image 5. Captured scene of Anno 1800 Game Trailer (source : https://www.reddit.com/r/anno/comments/bmon72/i_present_to_you_the_great_eastern/)

III. IMPLEMENTATION

The main idea of the strategy that we're using is to intercept the enemy's vessel by predicting its movement by using A* algorithm and heuristic distance of that vessel. This strategy is a very effective strategy against enemy that depend on economical approach to win the game because their main power is their trade routes, so if we intercept the route where the enemy's vessel is passing by, they will not grow on their economy and it will weaken them the entire game.

The detailed explanation of the implementation is as follows:

A. A* Algorithm implementation for predicting the route

The implementation of the A* algorithm for route planning using python is as follows.

```
def search(grid, init, goal, cost, heuristic):
    closed = [[0 for col in range(len(grid[0]))] for ro
    w in range(len(grid))]
    expand = [[-1 for col in range(len(grid[0]))] for r
    ow in range(len(grid))]
    plan = [[' ' for col in range(len(grid[0]))] for r
    ow in range(len(grid))]
    for r in range(len(grid)):
        for c in range(len(grid[0])):
            closed[r][c] = grid[r][c]
    reached = False
    fail = False
    path = None
    selected_list = []
    plan_path = []
    f_value = 0 + heuristic[init[0]][init[1]]
    e = 0
open_list = [[f_value] + [0] + init]
   for child in children:
      if len([visited_child for visited_child in
      visited_list if visited_child == child]) > 0:
       continue
       child.h = (((child.position[0] -
       end_node.position[0]) ** 2) +
       ((child.position[1]-end_node.position[1])**2))
       child.f = child.g + child.h
       if len([i for i in yet_to_visit_list if child
       == i and child.g > i.g]) > 0:
           continue
```

B. Heuristic function

The implementation of heuristic function to find heuristic distance on the map is as follows

```
def heuristic(node):
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D*(dx+dy) + (D2-2*D) * min(dx,dy)
```

Map that we were used contains 0 and 1 number means 0 is the path the ship can pass by, and 1 means the obstacle/island the ship can't pass by.The map we used for collecting the heuristic distances is as follows

```
grid = [[0, 1, 0, 1, 1, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 1, 0, 0, 1, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 1, 0]]
```

The result contains the heuristic distance from every point on the map.The heuristic distances as result os above function is as follows

```
heuristic = [[9, 8, 7, 6, 5, 4],
             [8, 7, 6, 5, 4, 3],
             [7, 6, 5, 4, 3, 2],
             [6, 5, 4, 3, 2, 1],
             [5, 4, 3, 2, 1, 0]]
```

IV. EXPERIMENT

We tested the algorithm just on the command line for the sake of simplicity of this test, but it's quite illustrate how the algorithm will work on the real game.

A. How the algorithm work

Here is the steps on how the simulation works for the game :

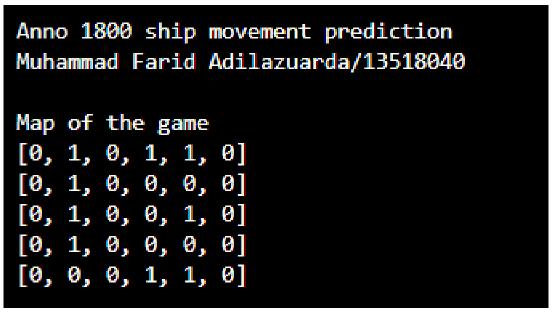
1. The initialization of the map of the game, described by grid list.
2. The initialization of where is the ship and its destination is located.
3. If the grid is null or not declared, the program will be stopped.
4. If the grid has been declared. The program will execute the heuristic function on every grid on the map, relative from the grid to the destination.
5. Using the heuristic distance that has been calculated, The ship will now move to the destined goal by each grid that doesn't contain any obstacle. The rule of the ship's movement is.
   - If Δx is positive ([0,1]), the ship position will move 1 grid to the East direction by Δx grid.
   - If Δx is negative ([0,-1]), the ship position will move 1 grid to the West direction by Δx grid.
   - If Δy is positive ([1, 0]), the ship position will move 1 grid to the South direction by Δy grid.
   - If Δy is positive ([-1,0]), the ship position will move 1 grid to the North direction by Δy grid.
6. For every move made, the state will be stored in a list variable called path_list.
7. The algorithm will find the closest distance between the ship and the destination, if its not the closest distance, then the state will be deleted and killed from the expanded list.
8. Back to step 3.

9. If the goal node isn't reachabke, the program will show a fail message.
10. If the program is finished, it will show the closest path using the('<','>','^','v').

B. The Testing

The result of the testing is as follows,

This is the initial state of the map, the ship is located on (0,0) and the destination is located on (5,5). Just a friendly reminder, the 0 grid is where the ship can pass by, and the 1 grid is where the obstacle is located.

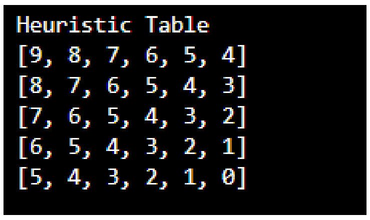


```
Anno 1800 ship movement prediction
Muhammad Farid Adilazuarda/13518040

Map of the game
[0, 1, 0, 1, 1, 0]
[0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 0]
```

*Image 6. **The initial state of the ship and the map (source : writer)***

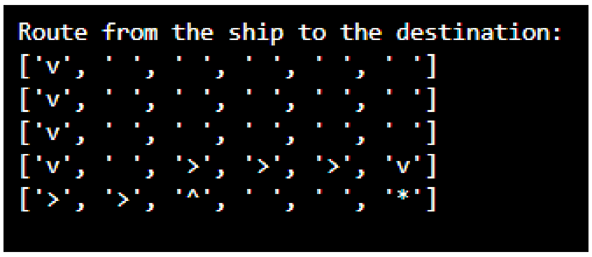
From the image 6 we can get some information of where is the location of the ship, the obstacles, the destination, and the size of the map. Theoretically, the A* algorithm will show the shortest path between the (0,0) to the (5,5) using the heuristic table follows.



```
Heuristic Table
[9, 8, 7, 6, 5, 4]
[8, 7, 6, 5, 4, 3]
[7, 6, 5, 4, 3, 2]
[6, 5, 4, 3, 2, 1]
[5, 4, 3, 2, 1, 0]
```

*Image 6. **The heuristic distance map (sorce:writer)***

And hereby is the ship movement on the map as the result of the A* algorithm.

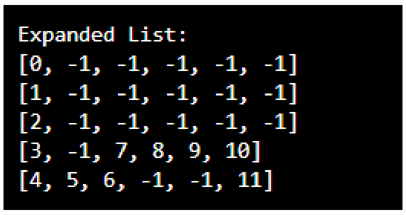


```
Route from the ship to the destination:
['v', ' ', ' ', ' ', ' ', ' ']
['v', ' ', ' ', ' ', ' ', ' ']
['v', ' ', ' ', ' ', ' ', ' ']
['v', ' ', '>', '>', '>', 'v']
['>', '>', '^', ' ', ' ', '*']
```

*Image 7. **The path predicted by A* algorithm (sorce:writer)***

From the image 7 above, we can see that the program search for the shortest path to the destination. For this, we can look in the grid (4,3), in this grid the program choose (4,4) instead of other grid (3,3). That is because from the heuristic table shown above, the heuristic distance of point (4,4) is smaller than the heuristic distance on point (3,3), that is 3 compared to 5.

Then from point (4,4) the prediction goes on until it reaches the destination point. The program made 12 steps from the starting point to the destination point.



```
Expanded List:
[0, -1, -1, -1, -1, -1]
[1, -1, -1, -1, -1, -1]
[2, -1, -1, -1, -1, -1]
[3, -1, 7, 8, 9, 10]
[4, 5, 6, -1, -1, 11]
```

*Image 8. **The Expanded list (sorce:writer***

Above is the Expanded List of the program. The positive values of the grid means that that node is alive in the decision tree, but the negative value means that the node is dead in the decision tree.

C. The comparison between the simuation and the game

This simulation is similar with how the ship in the game moves. In the game, ships move from the starting place to the destined place by moving accross the map and avoiding obstacles. Below shown how the ship moves in the game.

Compared to the game, the simulation won't add other outer factors such as windspeed or bad weather, but for the average cases the clear weather and the good windspeed is more common.

Back to the strategy topic, this simulation will give us a very good advantage because by predicting the ships movements, we can intercept our enemy's trade route or important ships that carries valuable items. But this strategy can only be used in campaign game mode, because this usage of algorithm will be considered cheating in online mode agains other people.

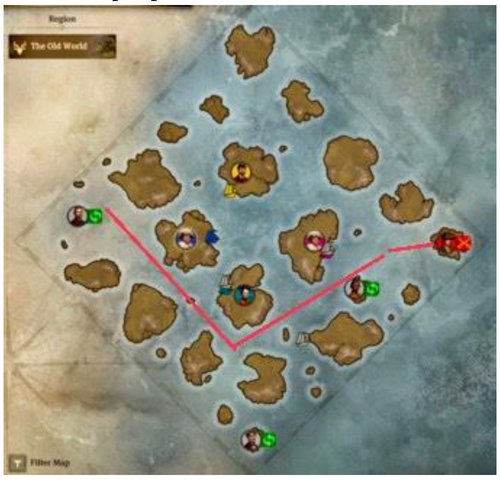


Image 9. The ship movement line (sorce:writer)

IV. ANALYSIS

From the section IV, we have shown that our prediction and strategy can work well in certaiin circumstances. But there are several weaknesses from our prgram as follows.

1. The program that we developed does not consider the outer factor of the game such as windspeed, bad weather, pirate ships, or etc.
2. To modify or change our desired location and starting point, we have to change it in the code by ourselves.
3. The program won't work against real human that has a very detailed micro game interaction. That such player will be very aware not to take their ships to the predictable locations
4. This program will be ineffective for such a big map, because the map/the grid table would be so big it won't fit the command prompt.

V. IMPROVEMENT IDEAS

To improve this chatbot we have several ideas for future improvements as follows.

1. Make the algorithm adaptable for the bigger map by connectng it directly to the game. By making the algorithm adaptable, it will make the game easier for the player to predict the enemy's ship.
2. Make the algorithm conaider the other factors that occurs in the game.
3. Make the program practical and can detect where is the ship in the game and its desired destination.

VI. CONCLUSION

In Anno 1800, the game time would take very long for common people that not a gamer. The AI created by Blue Byte is so strong that the bots keep developing such a powerful economy and vast technologies by using its trade routes To make the game easier for common people to enjoy the game, we create a program that predicts the enemy's movement on the map. From our testng, our program can work well on a certain circumstances.

s

VII. VIDEO LINK AT YOUTUBE

https://youtu.be/R4iw22ixDMI

VIII. ACKNOWLEDGMENT

I would like to thank all the lecturers of the knowledge that is shared regarding algorithm strategies, especially Mrs. Masayu Leylia Khodra as my IF2211 algorithm strategies class lecturer. This subject given me the chance to explore more regarding several topics in the subject and its application as well as given me the chance to practice my English writing skill. I would also like to thank my friends and families who support me in the process of learning and also making this paper.

IX. REFERENCES

[1] Problem Solving and Search A Star Best FS dan UCS (2018),https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf, accessed on 1 May 2020

[2] Ubisoft, "Anno 1800 overview and gameplay", accessed from https://www.ubisoft.com/en-us/game/anno-1800 on 1 May 2020.

[3] Brown University, "A* Algorithm", accessed from http://cs.brown.edu/people/pfelzens/papers/astar.pdf on 1 May 2020

[4] Makeabilitylab Washington University, "Route Predictions",https://makeabilitylab.cs.washington.edu/media/publications/Route_Prediction_from_Trip_Observations_p1hrOb7.pdf accessed on 1 May 2020.

[5] Krause, Eugene F. (1987). *Taxicab Geometry*. Dover. ISBN 978-0-486-25202- accessed on 1 May 2020

[6] Anno 1800 fandom, "The game rules and Entities", https://anno1800.fandom.com/wiki/Anno_1800_Wiki accessed on 1 May 2020.

X. STATEMENT

I hereby declare that the paper I wrote is my own writing, not an adaptation, or a translation of someone else's paper, and not plagiarism.

Blitar, 2 May 2020

[illegible]

Muhammad Farid Adilazuarda
13518040