Global Path Search based on A* Algorithm

Xiaoyan Guo[1, a], Xun Luo[2, b]

[1]School of Changsha, Hunan Normal University, Hunan 410000, China;

[2] School of Changsha, Hunan Normal University, Hunan 410000, China.

[a]1426645300@qq.com, [b]30053441@qq.com

Abstract. In the field of games and GIS development, the shortest path search is often involved, and the A* algorithm with heuristic functions is one of the more commonly used search algorithms. By studying the path search algorithm, on the basis of the simulation of the theory, combined with different heuristic algorithms to achieve the path planning strategy. Taking the terrain environment designed by self-design as the experimental platform, the path search algorithm is studied and the four measurement methods are compared. The final experimental results show that the path finding with the diagonal distance as the heuristic function is better reduced. The scope and scale of the algorithm search effectively improve the operational efficiency of the A* algorithm.

Keywords: A* algorithm; heuristic; shortest path search; terrain environment; diagonal distance metric.

1. Introduction

With the rapid development of technology, online games have become an important part of people's lives. Pathfinding is a very important element in game development. How to find the shortest path efficiently is one of the foundations of game in AI design [1]. The A* algorithm, as one of the heuristic search algorithms, is a path with multiple nodes on the graphics plane to find the lowest cost algorithm[2]. The A* algorithm is often used in mobile computing for NPC in games, or for mobile computing of online games. The algorithm can find a shortest path just like the Dijkstra algorithm[3]; it also performs a heuristic search like BFS.

At present, many scholars have conducted in-depth research on the A* algorithm and have achieved many results. In [4], the Euclidean distance is used as the heuristic value to calculate the optimal path of the grid topography. The literature [5] uses the parallel search method to optimize the ordinary A* algorithm, which shortens the consumption of time to a certain extent. This process uses Manhattan distance as the heuristic value. The literature [6] uses the diagonal distance as a heuristic value to solve the trap problem. In [7],Chebyshev distance, Manhattan distance and Euclidean distance were used as heuristic functions, and pathfinding experiments were carried out. In order to make the path-finding efficiency more efficient, a three-dimensional terrain environment is established in this paper, and the heuristic functions of different metrics are compared and analyzed.

2. A* Algorithm

The evaluation function of the traditional BFS algorithm only considers the distance between the starting point and the end point. The strategy is to select the point closest to the end point to search[8]. The Dijkstra algorithm only focuses on the distance between the current point and the starting point, and selects the point closest to the starting point to start the search. The A* algorithm combines the two for analysis and calculation.

The heuristic search of A* algorithm formula is shown in (1):

$$\mathrm{F}(n) = \mathrm{G}(n) + \mathrm{H}(\mathrm{n}) \tag{1}$$

In equation (1), it is assumed that the starting node is S, the target node is E, and the current node is n. G(n) represents the cost of reaching the current node in the process of searching from the S node to the E node, and H(n) represents the estimated value of the optimal path from the current node to the target node E. That is, the estimated cost to be paid from the current node to the target node D, and F(n) represents the valuation function of the current node, which is the sum of the estimated values of G(n) and H(n). Under the condition that the shortest path exists, the efficiency of the A* algorithm lies in the selection of the heuristic function H(n). Assuming that D(n) represents the actual value of the distance from n to the target node, then the selection of H(n) is three cases: if H(n)>D(n), the fewer the number of nodes to search, the smaller the search range, the higher the efficiency of the search, but the optimal path cannot be guaranteed. If H(n) < D(n), the more nodes that will be accessed, the wider the search range and the lower the efficiency, but the optimal path will certainly be obtained. If H(n) = D(n), the distance estimate H(n) is equal to the shortest distance, the search will be performed strictly along the shortest path, and the search efficiency is the highest at this time. The pathfinding process of the A* algorithm requires the use of an openlist and a closelist . The openlist is used to store the extended nodes, and the closelist is used to store the appropriate target nodes obtained after the calculation. Pathfinding steps of the A* algorithm:

1. Add the starting point to the openlist;

2. Traverse the openlist, find the node with the smallest F value, use it as the currently processed node, and add the node to the closelist[9];

3. Determine the 8 adjacent grids of the node. If the grid is unreachable or in the closelist, ignore it. Otherwise, do the following:

a. If the adjacent grid is not in the openlist, add it and calculate the F, G, and H values.

b. If the adjacent grid is already in the openlist and the new G value is smaller than the previous G value, then the parent of the adjacent grid is set to the node and the f value is recalculated.

4. Repeat steps 2 and 3 until the end point is added to the openlist to indicate the path; if openlist is empty, no path is reachable.

The heuristic estimation function H(n) in the A* algorithm is the distance moved from the current position to the final point. There are usually four ways to calculate distances:

(1) Diagonal distance

The diagonal distance combines the Pythagorean theorem and the Manhattan algorithm. The principle is to go diagonally first, and then go straight to the horizontal or vertical parallel with the end point, and then go straight[10]. The expression in two-dimensional space is shown in (2):

$$\begin{gathered} \mathrm{d_x} = |\mathrm{x_n} - \mathrm{x_{final}}| \\ \mathrm{d_y} = |\mathrm{y_n} - \mathrm{y_{final}}| \\ \mathrm{m} = \min(\mathrm{d_x}, \mathrm{d_y}) \\ \mathrm{h(n)} = \mathrm{d1} * \mathrm{m} + (\mathrm{d_x} + \mathrm{d_y} - \mathrm{d2} * \mathrm{m}) \end{gathered} \quad (2)$$

In equation (2), d1 is the cost of moving on a straight line, and d2 is the cost of diagonal movement.

(2) Manhattan Distance

In the two-dimensional coordinates, the Manhattan distance is the sum of the absolute wheelbases of the two points in the coordinate system[11]. The expression is as shown in (3):

$$\mathrm{h(n)} = |x_n - x_{final}| + |y_n - y_{final}| \quad (3)$$

(3) Euclidean distance

Euclidean distance is a commonly used distance formula that refers to the true distance between two points in m-dimensional space, or the natural length of the vector (the distance from the point to the origin)[12]. The expression in the two-dimensional space is shown in (4):

$$\mathrm{h(n)} = \sqrt{(x_n - x_{final})^2 + (y_n - y_{final})^2} \quad (4)$$

(4) Chebyshev distance

Chebyshev distance is a measure in vector space. The distance between two points is defined as the maximum value of the absolute value of each coordinate value difference[13]. The expression is as shown in (5):

$$\mathrm{h(n)} = \max(|x_n - x_{final}|, |y_n - y_{final}|) \tag{5}$$

3. Experiment and Analysis

3.1 Experimental Platform

Unity3D is a multi-platform integrated game development tool developed by Unity Technologies that allows players to easily create interactive content such as 3D video games, architectural visualization[14], real-time 3D animation, etc. It is a fully integrated professional game engine[15]. In the system of program generation road, in order to achieve better visualization effect, a three-dimensional terrain is created by using Unity3d, and then the terrain information of each coordinate point is obtained by ray casting, and the information in the three-dimensional scene is mapped to two-dimensional by transformation. In the coordinates, it is convenient for the user to operate. The specific process is shown in Fig.1:

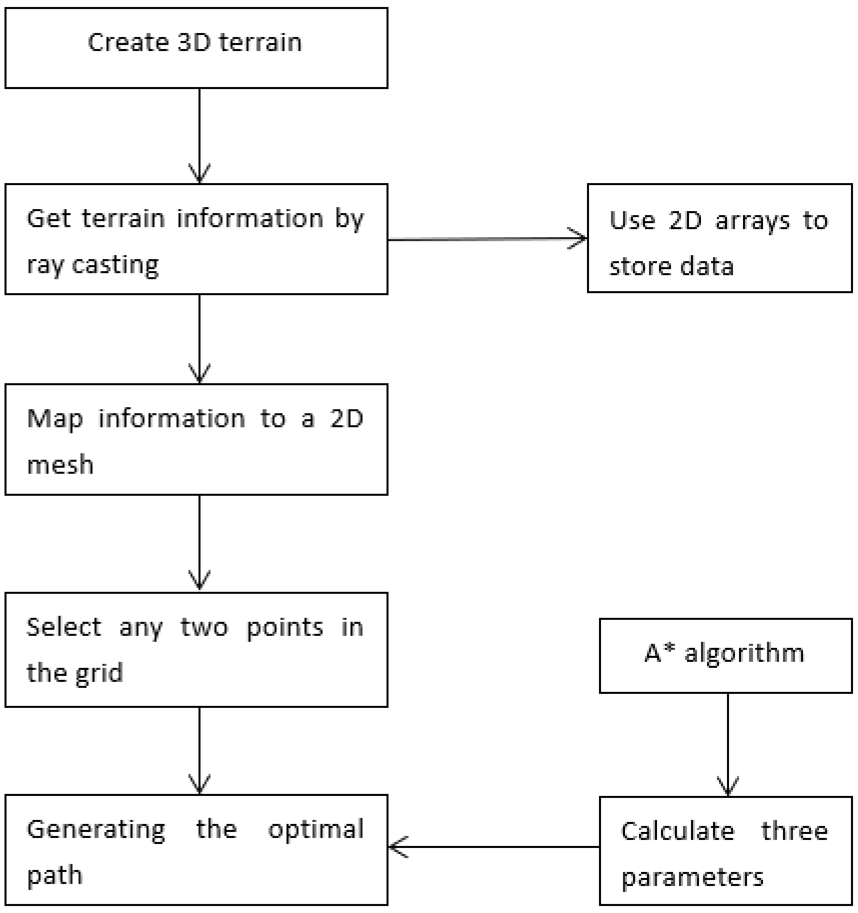
Create 3D terrain
Get terrain information by ray casting
Use 2D arrays to store data
Map information to a 2D mesh
Select any two points in the grid
A* algorithm
Generating the optimal path
Calculate three parameters

Fig.1 algorithm flow chart

3.2 Creating Terrain

On the unity3d platform, create a 3D terrain using internal tools. In order to make the 3D terrain authentic, you need to add textures, texture drawing, bump operations and collision detection on this terrain. The topographic map drawn is shown in Fig.2:

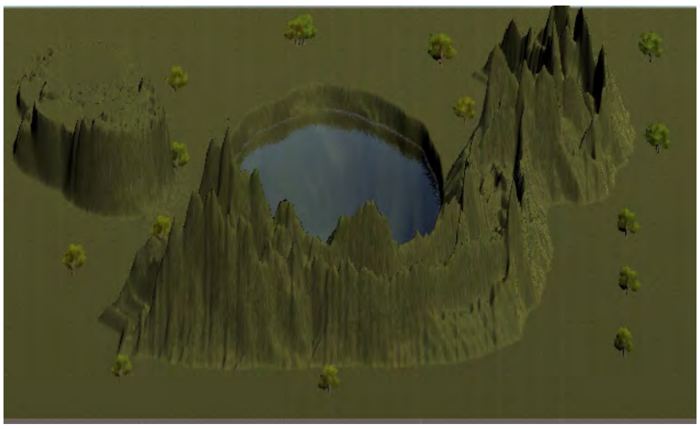
Fig.2 3D topographic map

3.3 Generation Grid Model

In order to more easily and intuitively see the real situation of the generated road, we use ray casting to obtain information about each coordinate in the terrain. Then create a 2D mesh in the 3D terrain where you need to convert the 3D coordinates into 2D coordinates. In three-dimensional coordinates, (x, y, z) represent the length, height, and width of the terrain. In the conversion process, you need to replace (x, z) with (x, y). The height of the three-dimensional coordinates can be stored using an array. The Grid model generated by transcoding is shown in Fig.3:

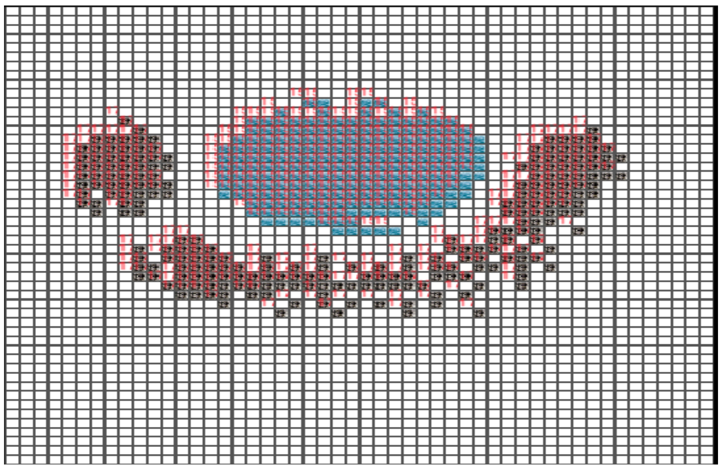
Fig.3 Grid model diagram

The Grid model diagram in Figure 3 is converted from the 3D topographic map in Figure 2. In the image above, the white grid represents flat terrain, the blue grid represents water, and the black grid represents mountains.

3.4 Experimental Results

In this paper, four different metrics are used for the heuristic function: diagonal, Huffman, Euclidean, Chebyshev. The two-dimensional grid map is used to simulate the number of grids traversed by the four metrics in the presence and absence of obstacles.

Experiment 1: Select a two-dimensional grid map without obstacles for comparison. Some experimental results are shown in Table 1.

Table 1. Experimental results of some data on obstacles

	diagonal	Manhattan	Euclid	Chebyshev
20*20	83	97	83	83
30*30	133	160	133	133
40*40	183	216	183	183
50*50	233	277	233	233

Table 1 is the number of grids traversed by these four metrics in different sized grids without obstacles. From these data we can see that when the heuristic function is diagonal, Euclidean and Chebyshev, the number of grids traversed without obstacles is the same, while the number of grids traversed by Manhattan is more than the first three. This is because in a two-dimensional grid, Manhattan is only allowed to move in four directions, while the diagonal distance allows movement in eight directions, and Euclidean and Chebyshev allow movement in any direction.

Experiment 2: Select the map grid with obstacles for comparison. Some experimental results are shown in Table 2:

Table 2. Experimental results of obstacle data

	diagonal	Manhattan	Euclid	Chebyshev
20*20	112	154	124	112
30*30	187	253	218	201
40*40	313	415	360	331
50*50	464	627	560	518

Table 2 shows the number of grids that these four metrics need to traverse in different sizes of grids based on the same density of obstacles. From these data, we can see that the number of grids traversed by Manhattan is the most, followed by Euclidean, then Chebyshev. On the whole, if the heuristic function uses a diagonal distance, the number of grids that need to be traversed is the least.

4. Summary

This paper creates a simple terrain on Unity3d, arbitrarily selects two points, calculates the distance between two points by four different metrics in A* algorithm, and compares and analyzes them. The experimental results show that the diagonal distance is a heuristic method that is more efficient in generating a road connecting the initial point and the final point. Not only reduces the size of the algorithm search, but also reduces the number of accesses to the grid.

References

[1]. LIU Na, WANG Yufang. Application of A* Pathfinding Algorithm in Games[J]. Digital Technology and Application.Vol.06 (2012), p. 135.

[2]. YUE Shuang: Research and application of multi-constraint dynamic path planning [D] (Master, University of Electronic Science and Technology of China, China, 2012). p.01.

[3]. LI Shuangliang: Research on the shortest path selection algorithm of GIS navigation system[D] (Master, Changchun University of Science and Technology, China, 2012). p.01.

[4]. SHI Junwei, BAO Shitai, FENG Wei. Analysis of Mountain Optimal Path Based on A* Algorithm[J]. Modern Computer.Vol.14 (2013), p. 9–11, 15.

[5]. HUANG Hai-wei, DENG Kai-fa. Application of improved A* algorithm in game map path-finding development[J]. Information Technology. Vol.04 (2015), p. 188-191.

[6]. ZHANG Haitao, CHENG Yinhang. Path Finding Using A* Algorithm[J]. Microcomputer Information. Vol.17 (2007), p. 238-239+308.

[7]. QIU Lei. Implementation and performance comparison of game maps based on A* algorithm[J]. Journal of Shaanxi University of Science and Technology (Natural Science Edition). Vol.06 (2011) No. 29, p. 89-93.

[8]. LIU Yunxiang, DU Jie, ZHANG Qing. Performance comparison between A* algorithm and Dijkstra algorithm based on path optimization[J]. Modern electronic technology. Vol.13 (2017) No. 40, p. 181-183+186.

[9]. LI Zhen: Research on path planning algorithm of pedestrian guidance system in traffic hub [D] (Master, University of Electronic Science and Technology of China, China, 2016). p.01.

[10]. CHEN Junmei: Research on Intelligent Production Logistics System Model and Its Key Issues [D] (Master, Southwest University of Science and Technology, China, 2016). p.01.

[11]. DONG Shuzhao: Research on content-based 3D medical image retrieval technology [D] (Master, Harbin Institute of Technology, China, 2013). p.01.

[12]. CHEN Hongguang, LI Feng. Implementation of a content-based image retrieval system [J]. Fujian Computer. Vol.07 (2017) No. 33, p. 37-39.

[13]. ZHANG Zhendong: Research on dynamic navigation of indoor service robot using laser scanner[D] (Master, Harbin Institute of Technology, China, 2014). p.01.

[14]. LIN Shenhua, FAN Zhishang, JIANG Jianbing, ZHU Yachao. Design and Implementation of Unity3D Game Based on Android Platform[J]. Enterprise Technology and Development.Vol.10 (2013), p. 40-42.

[15]. LIANG Cheng. Research and Development of Client in the Framework of the Basic Game Based on Unity3D[J]. Modern Computer (Special Edition). Vol. 25 (2015), p. 76-80.