Principle and Realization of A* Algorithm

Lehui Huang[1,a], Xinxin Wang[2, b]

[1]Education School, Jiangxi Science&Technology Normal University, Nanchang, China

[2]Education School, Jiangxi Science&Technology Normal University, Nanchang, China

[a]HLH8899@163.com , [b]1270731275@QQ.com

Keywords: A* algorithm, Principle, Realization

Abstract. In 3D games, pathfinding function is very common. The design can not only let the game player need not control the main body role through the keyboard and mouse blindly and arrive at the designated location more quickly to complete the task, but also can active AI roles and make the games fun to make the game players experience the better game fun. This paper introduces a popular pathfinding algorithm named A* and its function realization.

Introduction

Algorithm refers to the accurate and complete description of problem solving plan. It is a series of clear instructions to solve the problem, and the strategy mechanism representing using system method to describe the problem solving. That is to say, it allows a certain standard input, and gets the output in a limited period of time. If an algorithm has defects, or is not suitable for a particular problem, executing this algorithm will not solve the problem. Different algorithms may accomplish the same tasks with different times, space and efficiency. Space complexity and time complexity is used to measure the pros and cons of an algorithm. An algorithm includes two aspects: on one hand, arithmetic operation of data objects includes arithmetic, the logic operation, relational operation, and data transmission. On the other hand, the control structure of the algorithm. Meanwhile, an algorithm should have the following five important characteristics: finiteness, definiteness, input, output and effectiveness. To solve the same problem, different algorithms can be used, and the quality of an algorithm will affect the efficiency of algorithm and program directly. Analysis of the algorithm aims to select the appropriate algorithm and improve the algorithm. Evaluating an algorithm mainly considers the time complexity and the space complexity, which means the computing workload in executing the algorithm and memory consumption of the algorithm.

A* algorithm, as one pathfinding, refers to the most effective method in direct searching the shortest path from one place to another in the static network. Many kinds of preprocessing algorithm, like ALT, CH, HL and so on, are not include in it. This algorithm is classified as direct searching algorithm, heuristic algorithm and static graph of searching algorithm. Actually, it is proved that it can also be applied to the dynamic graph searching later. It can directly search in the real terrain map without any pretreatment, and guide the direction of the algorithm through the heuristic function.

Principle of A* algorithm

Pathfinding is very important in game designing, and how to find the shortest path in lots of methods needs to design a corresponding algorithm. At present, the most popular routing algorithm is A* algorithm. When the evaluation value and the actual value get closer, valuation function is better to acquire [1]. The A* algorithm is closely combined with the state space searching.

The state space searching, is looking for a path from the initial state to the goal state in the process of solving problem. Popularly, it means that finding an optimal solving process of solving the problem from the beginning question to the end answer [2]. But in the solving process, there may be uncertain or incomplete conditions, causing many branching process and producing many corresponding plurality of solution paths. The paths is connected to form the process state space. At

this point, the performance of problem solving is to find a path from the beginning to the end in this graph, and this search process is called the state space searching.

Common state space searching includes breadth first and depth first. Breadth first means to search from the initial state, layer upon layer down, until you find the target; depth first refers to finish searching a branch from beginning to the end in order, if not find it, then next branch until you find the target. These two kinds of methods are enumeration in the established state space and exist very big limitation, only applicable to the small state space. When the space is very large with unpredictability, it is difficult to use these two methods to complete the search. But another algorithm, heuristic search, can effectively avoid the low efficiency of exhaustion even the situation is unallowable, resulting in the development of popular algorithm. Specifically, it refers to the evaluation of each searching node existing in the state space, until finding the best, then searching from this node until finding the target position. Evaluation of the searching node is extremely important in heuristic searching, and using different valuations may produce different results. The evaluation function is: F (n) =g (n) +h (n)[3], namely the valuation of the searching node is equal to the actual cost from the initial searching point to the current point with the valuation price form this point to the target point. Because the former is known, while the latter is far greater than the former, the former can be neglected in the actual operation, so as to improve the efficiency of evaluation.

There are also a variety of heuristic searching algorithm, local first searching and best first searching both belong to this class, and the A* algorithm is derived from this. These algorithms all utilize heuristic function, but the searching strategies are different in the specific selection of the best node. Local fist algorithms abandon other brother nodes after selecting the best node in the search process, and will search downwards along the selected node. Because it takes a shekel much when selecting the best searching node, causing the best node may be rejected, so it has some defects. Best first algorithm is different. It will only discard the dead nodes before no further searching point evaluated. It will compare the value of the node to be selected with the previous node value, and then select the better, so to ensure the best node not to be lost.

A* algorithm is one of the best first algorithm, which adds some specific constraint conditions. The aim of the A* algorithm is to quickly find the shortest path from the initial point to the target point in the state space, to solve the problem quickly [4]. The formula is: F'(n) =g' (n) +h'(n), namely the performance of the node evaluation is the sum of the shortest path value between the starting point and the target point with the shortest path heuristic value between this node and the target node. Usually, the actual cost from the initial point to the selected best point is greater than the shortest path value from the beginning node to the end node. When the shortest distance heuristic value between the starting point and the end point is greater than or equal to the cost valuation of the best path form the best node to the target node, using the evaluation function can find the shortest path. When evaluating a node, if the constraint conditions are more, the more other nodes can be excluded, so as to find the optimal node, to achieve the optimal algorithm, as the A* algorithm does, so this pathfinding function has been widely used in the game.

Realization of A* algorithm

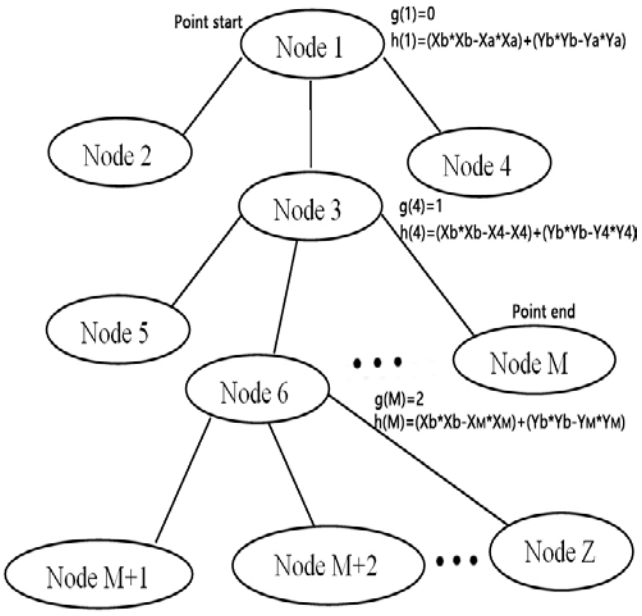
Then how to use A* algorithm? Here is an example:

First, create two tables. One of them named OPEN, to save all the nodes generated without investigation, the other named CLOSE, to record the nodes have been accessed. Then count the evaluation value of the starting point, and save the point in the OPEN table. While the OPEN table is not equal to null, take the minimum assessment value code as father node from the OPEN table, if it is equal to the target node, then break. Iterate each child node of the current node, count assessment value of the child node. If the evaluation value of child node is less than measured value in OPEN, set the father node to child node's father and update the estimated values in the OPEN table. If the child code is in CLOSE, then continue. While the child node is not in both tables, set the father node to child node's father, count the valuation of child code and insert the child code into the OPEN table. Then insert the father node into table CLOSED, order the nodes according to the assessed value. Save the path, starting at the end, each node moves along the parent node until the

beginning point, this is the path which has been built.

The scripts is as follows:

```
List<Point> CloseList;
List<Point> OpenList;
public class Point{
    public Point ParentPoint {get; set; }
    public int F {get; set; }
    public int G {get; set; }
    public int H {get; set; }
    public int X {get; set; }
    public int Y {get; set; }
    public Point(int x,int y){
        this.X = x;
        this.Y = y;    }
    public void CalcF(){
        this.F =this.G +this.H;    }    }
public Point FindPath(Point start, Point end,bool IsIgnoreCorner){
    OpenList.Add(start);
    while (OpenList.Count != 0){
        var tempStart = OpenList.MinPoint();
        OpenList.RemoveAt(0);
        CloseList.Add(tempStart);
        var surroundPoints = SurrroundPoints(tempStart, IsIgnoreCorner);
        foreach (Point pointin surroundPoints){
            if (OpenList.Exists(point))
                FoundPoint(tempStart, point);
            else
                NotFoundPoint(tempStart, end, point);    }
        if (OpenList.Get(end) !=null)
            return OpenList.Get(end);    }
    return OpenList.Get(end);    }
```



Pic.1

Summary

In the procedure design, algorithm is the core. Game AI and player role designing are both related to the algorithm. Selecting an optimal algorithm in all kinds of algorithm in the current designing of the game, can not only reduce the work of the programmer, but also can increase the fun of the game, enhance the gaming experience, increase the number of registered users and expand the influence of the game. Automatic pathfinding algorithm has always been the discussion focus of game developers. Although the A* algorithm is the most popular algorithm, it still has a lot to be improved. But I believe that a new better algorithm will emerge with the continuous efforts of programmers.

References

[1] Rui Kuai, Jinmin Hong. Application and optimized designing of automatic routing A* algorithm[J]. Journal of Shanghai Institute of Technology (NATURAL SCIENCE EDITION).2014,14 (2): 159～162.

[2] Xiaojing Zhou. Research on routing in game map based on the improved A* algorithm[D]. Southwestern University in ChongQing, 2011.

[3] Yun Meng, Bangquan Liu. Research on Automatic Routing Algorithm of role in 3D scene[J]. Journal of Wuhan University of Technology.2011,33 (12): 125-130.

[4] Jiangbo Yuan, Panliang Mu, Yue Shi, Hui Li. Efficient Algorithm of Automatically Routing of the Virtual Vehicle in Big Scene [J]. Computer Engineering and Designing.2008,29 (10): 2622-2625.