APPLICATION OF MAINTAINING THE SHORTEST PATH METHOD IN THE GAME MAP PATH-FINDING

Zheng-hong Hu,Jin Li
Computer Science Department
Taiyuan Normal University
Taiyuan, Shanxi 030012,China
E-mail:mary.hu.jin@163.com, syjinli2008@163.com

Abstractt— **Path-Finding algorithms mainly solve the problem of how to find a path from the starting point to the target point. If there is no connected path, the game will enter the deadlock state. This paper focuses on analyzing the rule of picture matching, giving the estimated formula and process of the A * algorithm in this game, and introducing a typical determining deadlock algorithm: the A* algorithm-based method to maintain the shortest path. Moreover, the paper pointed out the data structure of the game map and elaborated the algorithm thought of maintaining the shortest path .Then it described the specific steps of the algorithm.**

Keywords- the shortest path;path-finding; game map

I. THE QUESTION PROPOSED

In graph theory, path-finding is usually one of the hottest topics which people study. It focused on how to solve the problem of finding a road from the starting point (S) to the target point (T) in the graph. Picture matching game sometimes is at the deadlock state and the game cannot proceed. The so-called deadlock is that the two grids with the same pattern cannot connect by straight lines which turning number is smaller than 3, and means interface cards cannot be eliminated. Therefore, an algorithm, which determines whether the game is at the deadlock in order to rearrange cards, is important to the game. The deadlock is essentially a problem of judging the same cards can be eliminated or can't, and the most direct way is blind-type searching. For example, blind method of exhaustion, for cards which the game has not yet eliminated, calculated every two grids and determined whether they can be deleted，which almost travel the entire map. It was low efficiency, and CPU occupation run slowly for a long time, which proves that it is not a suitable path-finding method for online games. This paper points out to achieve deadlock judgment by maintaining the shortest path between two related grids, uses A* algorithm to calculate the shortest path between them, in every elimination of two cards, then updates data which need to maintain. In order to understand the algorithm to maintain the shortest path, a brief introduction of the A * algorithm to calculate the shortest path in the game map was discussed.

II. THINKING OF THE SHORTEST PATH ALGORITHM

More representative in the picture matching path-finding algorithm is breadth-first search algorithm. The search path needs to satisfy certain rule; therefore I briefly describe the game rule before introducing the path-finding algorithm.

A. The game rule

Picture matching game's rule is not actually complex. First, when the game starts, several different cards disperse stochastically on the map(and every card could appear occasionally several times, usually was 4 times). Second,the player only search two same cards(nodes) on the map and the two nodes may be connected by straight lines which not to surpass 3,then those two nodes may be eliminated. Last, if the player completes elimination of all cards then he win the game.

Analysis the rule of the game, the connection points in three ways: connect with a straight line, a break point of connection and two break points of connection [1].

B. Breadth-first search algorithm

Breadth-first search is a classical path-finding algorithm about picture matching.

First of all the graphics Start (x1, y1) is pressed into the queue. Then it extended the nodes which graphics Start (x1, y1) can reach by a straight line , with the set S0 = Find (x1, y1) explained that these nodes can be visited by a path which turning number is 0. If the node, Target (x2, y2), is included in the collection S0, by end of the search[2].

Otherwise, continue to expand the node which the blank block may arrive by a straight line in collection S0, with the set S1 = (Find (p) | p ∈ S0) said that S1 '= S1 - S0 (S1 contains S0), expressed that a link is exist through a path which turning number is 1 between the S1 ' nodes and that of Start (x1, y1) . If the node B (x2, y2) in the S1', then the node S and T may be connected by the path which turning number is 1, the end of search.

Above condition does not hold, continue to expand the node which S1' blank block can reach by a straight line, with set S2 = Find (Find (p) | p ∈ S1'), said that S2 '= S2 - S0-S1 = S2 - S0 - S1 '(S2 contains the S0 and S1). Collection of S2' is the grid that the node A (x1, y1) can reach through the path with 2 windings. If the node T (x2, y2) is in the set S2 ', then the node A, B may be connected by the path with 2

978-0-7695-4202-7/10 $26.00 © 2010 IEEE
DOI 10.1109/CASoN.2010.169

turns, otherwise the node A, B can not link through the path of less than 3 windings.

C. A algorithm

A * algorithm is a typical heuristic search algorithm. It is also an algorithm which the search state is least among heuristic search algorithms. The principle is to design a estimate function which may obtain the cost of every node[4]. Whenever it searches to carries on the appraisal one by one through the estimate function, for the node which next step possibly arrives. Then it takes out the minimum cost node to continue downward to search. Those which are not selected on the node can not be searched. Moreover it must have two tables which were Open table and Close table:

The Open table is composed of the node which has not been inspected and preserves the F value of the node opened.

Each time it takes the node with smallest F value from the Open table to expand the following node.

The Close table is composed of the node inspected. It joins the node that has been expanded (not including end point) [5].

In view of game's characteristic, it designs evaluation function firstly[3].

F (n) =G (n) +H (n) +C (n)

Description:

F (n): It is the minimum cost estimates in all paths which embarks from the initial node “Start” and restrained visits node n then reaches goal node “Target” .

G (n): It is the mobile-consuming which starts from the beginning node “Start” along the path created and moves to the designated node.

C (n):It is the number of windings which starts from the beginning Start to the specified node.

In picture matching game, the node can move in four directions. And the connection between two of the same pattern can not be more than two bends.

G value of settings: In regard of game’s request it is designed to move in four traverses. G value is infinity if the position node to move has the pattern, such node does not expand. G value is 10 if the position does not have pattern.

H value of settings: The sum of walking steps which from specified node to goal node in horizontal and vertical direction; For example T(Tx, Ty) is an end point, C(Cx, Cy) is the current spot. Then: h(n)=|Tx-Cx|+|Ty-Cy|.

C values of settings: The number of windings from beginning Start to specify node. C value is 0 if it has straight lines joining. Otherwise, the C value is 10 if it turns a bend. We could set C = 20 when it turns two bendings. Or its turns is 3, the C value is infinite.

A * algorithm specific implementation steps are as follows:

Suppose the starting grid for the Start and target cell for the Target. Current is each time the node which inspects, and emergent node Temp represents effective sub-node of Current four directions. The so-called effective sub-node means the block which does not connected with design the same pattern. When Current is equal to Target then the search is successful l [4].

(1) Start=Current. Calculate its G, H, and C to get its F value. Then puts them into the Open table, the redundant following circulate spatially until the Open table is empty.

(2)If Target is in the Open table, the search completes and the way is found. Then break circulation.

(3)Sort the Open table's element according to F value. The node Current, which with the smallest F value, is taken out from the Open table and put in the Close Table.

(4) Unwind Temp, the effective sub-node of Current in four directions.

(a) If F value of Temp is infinite or in the Close table, next effective sub-node is selected.

(b) If Temp is not in the Open table, we put it into the Open Table; Take Current as the father node of Temp; Record F, G, C and H value of Temp node.

(c) Otherwise, Temp is in the Open table and its H value compared to previous was smaller which means that has a more superior path.

(i) Set the "Temp" parent node for the current node "Current";

(ii) Recalculate G, H, C and F value of Temp for this computation value;

(iii) Rearrangement elements s of the Open table by F value sorted;

(5) Returns step (2).

If it didn’t found the target node and the Open table was empty which indicates the connection path does not exist between starting point and target point. Otherwise, the target node exists, then the node elected by a* algorithm in the way is a key point. That institute which constitutes the way is the most short-path [5].

III. MAINTAIN THE SHORTEST PATH ALGORITHM

When we determinate game's deadlock state, various factors must be considered, such as game’s data structure [2].

A. Data Structure

Through to Picture Matching game’s rule observation, we may regard the map as a square class dimension array, in which all cards (below said that “node”) may regard as a concrete element in the array. Then, in the game, the same cards may take as elements which have the different position with the equal value. As for the pattern state, we define class named square which owners attribute id, to correspond every position’s state. Like 1, means the pattern state and 0 means the no pattern state. We use Java program described the maps’ data structure:

```
public class Square {
public int id; // the identification number for every grid; the grid with the same pattern has the same id value
public Width = 40; public Height = 40;// grid width and height
public int x; // abscissa
public int y; // ordinate
```

int father=0; /express the father node and set its initially 0

byte way; // record node's direction

public Square Target; // the goal block

public Path=new array[]; } // record connection way }

int map[width+1][height+1]; // express map barrier data

int SquareID[m]; // record node subscript that has the same id value

Square PathTable[width*height] // define the shortest path table to maintain the shortest path-finding

B. *algorithm mentalities*

When it comes to determination of game's deadlock state, the algorithm usually checks cards in game and calculates whether a link exits between every two cards. If the link didn't find it explains the game was at the deadlock. When user eliminates a pair of cards each time, it is possibly has the new way. But we cannot determine which ways can be affected by those spatial checks, only calculate again. Obliviously, that method has more shortcomings, such as the big computation load, long consumption, as well as the massive CPU time.

Actually, the prerequisite of erasing two cards is that they can be matched with the same pattern, so maintaining the shortest path does not need to find a link for every two cards. We only calculate the shortest path between the two same pattern blocks and join them to the shortest path table in preserves.

In game process, when user selects the starting grid and target grid then the program find them from records of the shortest path table. If they present, the program deletes them from the table and calculate the new shortest path which possibly produces.

If two same pattern cards successfully eliminate, the new connection, which takes the two grids' four direction sub-nodes as the beginning point (as shown in Figure 1,Figure2),will produce. Thus, the computation on connecting way is running among the grids which match to those sub-nodes with the same ID. If the connection exists, join them to the shortest path table. When the shortest path table is empty, it shows that the deadlock state emerges.

C. *algorithm descriptions*

According to that algorithm mentality, we design a specific algorithm to maintain the shortest path and use an array of object types saved it.

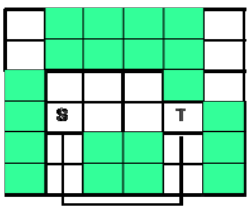

Figure 1. S to T connect condition

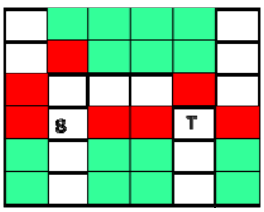

Figure 2. After S and T liminated, the node to expand

Algorithm steps are as follows:

(1)Initialize the shortest path. Cycle through the grid that game has not yet eliminated, then give the implementation of steps (a) - (e):

(a) Take out Start node.

(b) Search the grid with the same Start ID as Target in turn. If Start and Target are not in the shortest path, we execute (c), otherwise carry out (e).

(c) Using A* algorithm to calculate the connection path between Start and Target.

(d) If there is the most short-path from Start to Target, put them into the shortest path table.

(e)Or not, select the next cell as the Start Repeat steps (b).

(2) Determine the shortest path table is empty or not. The empty means the game is at the deadlock and need to rearrange the blocks.

(3) When user selects two blocks S, T then program runs to find whether they are in shortest path. Don't do anything if they aren't. Otherwise, delete them and erase their shortest path from the shortest path table.

(4) Calculate the shortest path to create possibly. First, expand S' four-direction nodes, then perform steps (a) - (e) if the node has not been investigated and is not empty.

(5) If the node is at connected condition, we can expand non-spatial node which its straight line arrives and execute steps (a) - (e).

(6)Expand T node and execute steps (4) and (5).

IV. SUMMARY

Suppose the game map has n rows and m columns, so that let N = m * n. If the same design's redundant number is 4 and s is the total number of different patterns, we could set N = 4 * s. When it is initialized, the node with the same ID enters the shortest list altogether 6 times, and every node, which join the shortest path as the expand node, to enter the Open table is no more than 3 times . Therefore, Open table handles nodes only 3 * N. The most time-consuming of algorithm is O (3 * N) because expanding every node needs O (3) of the time. To construct the corresponding shortest path requires O (L) of the time, and L is the length of the shortest path. Thus, we may draw a conclusion that the algorithm's complexity is O (N / 4 * 6 (3 * N + L)).

Based on A * algorithm, this paper judged game's deadlock state by maintaining the shortest path table, and gave the corresponding data structure as well as algorithm steps. That can be applied to areas of game map find-pathing. As for matters such as rearranging the game map and calculating the number of turns(whenever it enters deadlock state) , the paper did not discuss because of space limitations. In the following article we will continue to research. As a kind of shortest path algorithm , A * algorithm combines heuristic and formal methods. For the map of fixed obstacles, it also has some practical value.

REFERENCE

[1] Hu Zhenghong, “Application of a Path-Finding Method in Game Development”, Shanxi Electronic Technology,vol.147,Dec.2009,pp.53-54.

[2] Beauty of Programming [M] Beijing: Electronic Industry Press, 2009.

[3] Hu Zhenghong, “Application and implementation of A* algorithm in Picture Matching path-finding”, unpubished.

[4] Chen Hepin and Zhang Qianshao, “Application and Implementation of A* Algorithms in the Game Map Path-Finding”, Computer Applications and Softwre,vol .22,Dec.2005,pp.118-120.

[5] Cai Zixin and Xu Guangyou, Artificial Intelligence and Its Applications (Third Edition)[M] Beijing: Tsinghua University Press, 2004.