Application and Implementation of A* Algorithm in Picture Matching 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

***Abstract*—In game design, Artificial intelligence is an important and complex module, and path-finding algorithms are one of the most fundamental problems applied to video games. According to the feature of path –finding in Picture Matching game, this paper analyzed a* algorithm applied to the game in detail, and pointed out the composition of estimate function. Combined with practical application, the paper given its realization.**

Keywords-path-finding; Breadth-first search; A algorithm; Estimate function

I. QUESTION DESCRIPTION AND ANALYSIS

Picture Matching game, also named LianLiankan, is composed of many small rectangular blocks with a pattern. It can test player's eyesight, and its rule is as follows: in the limited time, two blocks with the same pattern can be eliminated if they connect together, only completing elimination of all patterns can the player win. The so-called connection refers to either horizontally or vertically, the connection between two of the same pattern can not be more than two turns, and it can not pass through the block with a pattern. [1]

The analysis of the game's rule showed three ways of the connection: straight line connected a break point of connection, or two break points of connection.[2]

It can be seen that the game's design lies in: How to design data structures; How to judge whether two blocks can be deleted; How to find the shortest path between the same pattern blocks; How to determine the deadlock state and relieve it?

II. BREADTH-FIRST SEARCH ALGORITHM

From the above analysis, the essence of the deadlock is calculate every two blocks ,which exist to the game, can be eliminated or not. It converts to the path search. Thus, well-designed search algorithms are the focus of game design.

Breadth-first search is a classical path search algorithm about Picture Matching.

It will modify the objective function to turning frequency which from one point to another point. First, the node Start (x1, y1) is pressed into the queue. Then, it extends the node which Start (x1, y1) can arrive at straightly. Setting S0 = Find (x1, y1) explained that these nodes can be visited by the path which turning number is 0. It may end the search if the node Target (x2, y2) is included in the collection S0.

On the contrary, it continues to expand the node which space blank can reach by a straight line in the collection S0. It sets S1 = (Find (p) | p ∈ S0) and S1 '= S1 - S0 (S1 contains S0) .That expressed the S1' node and the node Start (x1, y1) can be linked by the path which turning number is 1. If the node B (x2, y2) is in the S1 ' then the node S and T may be connected by the path which the curve number is 1.It is end of search.

Above condition does not hold, it goes on expanding the node which space blank can arrive at by a straight line in the collection S1. Setting S2 = Find (Find (p) | p ∈ S1') explained S2 '= S2 - S0-S1 = S2 - S0 - S1 '(S2 contains the S0 and S1). And collection of S2 ' is the block which A (x1, y1) can walk to through the path of 2 turns. If the node T (x2, y2) is in the set S2 ', then a connection exists between the node A and B which the windings is 2. Otherwise, the node A and B can not connect by the path which turning number is less than 3[3].

III. A* ALGORITHM APPLIED TO PICTURE MATCHING GAME

A. Design of Estimate function

A* algorithm has many merits. Firstly, if there is an effective path between the start and the end, A* algorithm certainly find the path. Secondly, so long as H (n) is may accepted, it must find the shortest path. Lastly, it is an algorithm which the search state is least among heuristic search algorithms, and enabled the heuristic function to obtain the most effective application.

The heuristic search's core is the estimate function -F (n). In Picture Matching game, the connection cannot pass through the block with a pattern; the expand node only move in four directions; the link is more than two bends between the two same pattern cards.

Thus, for the estimate function, it is necessary to consider the mobile cost between the node current and Target (the goal node), as well as the number of turns between them. Hereby, we modify the estimate function formula applied to Picture Matching game:

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

Description:

F (n): It is the minimum cost estimates in all paths. That sets out the initial node "Start" and arrives at the goal node "Target" with restrained visiting node n.

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 leaves the beginning Start to the specified node. The four directions refer to the grid S can move along four directions such as the upward, the downward, the left, and the right in Picture Matching game. And every grid is at a pattern state or no pattern state.

So we design the two-dimension array map[Rows][Cols] to express the corresponding grid is at the pattern state or not. If the array element value takes 1 then the corresponding position is at the pattern, or takes 0 express the position is at no pattern.

These are values setting of estimate function- F (n):

G value of settings: According to the game's request, the expand node moves along four directions. G value is infinity if the position node to move is at the pattern state, which does not consider to expand . Or G value is 1 if the position does not have a pattern. As shown in Figure 1.

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

C value of settings: It presents the number of windings from beginning Start to specify node. The C value is 0 if they join with a straight line, and the value is 1 if the turning number is 1. Or the C value is 2 if the connected path turns two bendings. Otherwise, C value is infinite[4].

For example, the beginning point, S, and the end point, T are in a horizontal line. The grids' state around them is shown in Figure 1. When we calculate of the nodes' F (n) in four directions of S, we can see the upward and left nodes' cost function is infinity. Thus, we may set the right node of f(n) = 1 +3 +0, and the downward node of f (n) = 1 +5 +0; the calculation shown in Figure 2 - Figure 4.

New node is expanded from the node which F value is 4. We can calculate its right node's evaluation function: f = 4 + (1 +2 +0) = 7.The calculation shown in Figure 5.

Continue to expand the node which F value is 7 and calculate its right node's evaluation function: f = 7 + (1 +1 +0) = 9. The process is shown in Figure 6.

Lastly expand the node which F value is 9. It can be seen that the end node is in the expanding collection, then completed the shortest path- finding.

B. Design mentality

A* algorithm is a typical heuristic searching algorithm. Except for using that inspiration function to appraise every node in the search process, moreover, it must have two tables such as Open table and Close table.

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

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

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

Incorporating the game's requirement, we can use pseudo code describe the implementation of A * algorithm applied to Picture Matching.

We assume that Start refers to the start node and Target stands for the goal node. Current means the node inspected each time, and Temp represents effective sub-nodes in Current's four directions. The so-called effective sub-node means the space blank which is at no pattern state. When Current is equal to Target then the search is successfully complete.

A* algorithm, the path-finding method in Picture Matching, expresses as follows with the Pseudo code:

First, it calculates G, H and C value about Start to conclude its F value, and puts Start in the Open table;

```
finish=false;      // it expressed the search condition about path-finding. It said the path has not found.
while (Open table is not empty)
if (Target is in the open table)
finish=true;                    // it completed path-finding successfully.
break;          // breaks up circuit and executes next step
End if
sorting Open table element according to F value ;
// quickly find the node with the minimum of F.
Take the node Current from Open table which with the smallest value of F , and put it in Close table;
for(evry Current's valid child node in Temp)
calculate G, H, C and F value of Temp;
if (Temp is infinity of F or in Close table)
// explained that Temp node does not conform to the game rule or it has been investigated.
 conutie;       // stop this cycle and select the next node
End if
if(Temp is not in Open table)
// explained that Temp is not in Open table and Close table which has not been visited.
{puts Temp in Open table;
takes Current as Temp father node;
records the value about F, G, C and H of Temp;
}
e1se                // Temp is in Open table
if(C value about Temp calculated in this circuit is smaller than that of the last)// it implies a better path
set Temp's parent node as the current node- Current;
recalculated the G, H, C and F value of Temp for this computation value;
sorted Open table elements again on F value;
        End if

   End if
End for
Loop

If not finish       // it did not find the target node. Open Table is empty and the connected path does not exist
```

IV. REALIZATION OF A * ALGORITHM APPLIED TO PICTURE MATCHING

A* algorithm's data structure setting and programming have their certain pattern, so we give implementation codes of A* algorithm applied to Picture Matching.

A. Digital direction of movement

The node can move in four directions in the game, thereby, we set an array with constant element which stores every direction's migration. We define the class method and transfer parameters: coordinates of starting point and target point . It as shown in Figure 6.

```
int Move[4][2]={{-1,0},{0,1},{1,0},{0,-1}}
```

Before expanding the node, some data need to initialize:

```
Int OpenListCount=1;  //defined variable to record the number of expanded nodes in OpenList
Int NewOpenNodeID=1;  // defined variable to record the node's ID . its initial value is 1
If OpenStadus=1;  //if variable is 1,it means the node is at the open condition
```

B. Construction of data

The A* algorithm's request for data structure is a Open Table, a Close table, and lists of F, G, H and E value, so we set some variables to improve the data of the game map:

```
public class Square {
public int id; // it is the identification number for every card. some cards with the same pattern which id is the same
public Width = 40; public Height = 40;// card's width and height
public int x; // the node's x-coordinate
public int y; // the node's y-coordinate
int father=0;  // explained the node's parent node, and its initial value is 0
byte way;  // it records the node's movement direction. The value is 1 express X direction and it is -1express Y direction
public Square Target; //target node
public Path=new array[]; } // record connection way }
int map[width+1][height+1];
```

// defined a two-dimensional array to express obstacles of the map. its value is 1 indicated that the corresponding position of the map is at a pattern state, and its value is 0 said that the position is at no pattern state. When initialized data , we can set the elements outside blocks are 0 and other elements are 1.

```
Square Node[width*height]  // defined a Square type one-dimensional array which expressed every node about Picture Matching game:
Int OpenList[width*height]  // defined a Open Table to save subscript ID numbers of the node, which is an ordered heap
Int whichNode[width][height];  // it said that the node is at open state or close state . 0 means the close state and 1 means the open state.
Int F[width*height];// the F value of every node
Int G[width][height]  // the G value of every node. if it is at a pattern state ,set the G value 9999.
Int H[width*height]  // the H value of every node
Int C[width*height]  //the C value of every node
```

C. Calculated turning number

It is essential to calculate turning number about the node in Picture Matching game. Thus, we definite the class method, GetCrossing, which introduced the node's ID , to calculate the turning number about the node and return it to the program.

```
Class GetCrossing{
Int Crossing public GetCrossing(int index,int Crossing)
{Squre n;  byte way,crossing;
n=Node[index];way=n.way;// taken the node and its direction corresponding index
crossing=0;
do
{ if (n.father<>0) and (way<>n.way)
  {way=n.way;
   Crossing=crossing+1;}
 n=Node[n.father];
}
while n.father<>0
if crossing<=2 return crossing
else return 9999  // returns value of infinity if the number of windings is bigger than 2
}
}
```

D. Path-finding design

When the node is expanded, we could definite a class method, FindPath, which passed parameters, to calculate the node's coordinates and record its ID. Then , add the node to OpenList and calculate its G, H, E and F value. Lastly, the expanded node joins to whichNode .

```
Public class FindPath{
// defined path-finding method
Public FindPath (int startX,int startY,int targetX,int targetY);
{……Variable definition slightly
G[startX][startY]=0;
OPenList[1]=1;// corresponding the first node
Node[1].x=startX; Node[1].y=startY;// set the initial value to the first node
Int NewOpenNodeID=1;  // defined a variable to record the node's ID and set its initial value is 1
parentXval=Node[1].x;parentYval=Node[1].y;
ifOpenStadus=1;  // expressed the node is at the open state
  for (int i=1;i<=4;i++)  // the node moves in four directions
 { MoveX=parentXval+Move[i][1];
MoveY=parentYval+Move[i][2];
NewOpenNodeID = NewOpenNodeID + 1;
// the new addition of a node
  Node[NewOpenNodeID].x= MoveX;
// preserve the expanded node
  Node[NewOpenNodeID].y= MoveY;
  Node[NewOpenNodeID].father= openList[1];
If (Move[i][1]==0) Node[NewOpenNodeID].way=-1;
```

```
If (Move[i][2]==0) Node[NewOpenNodeID].way=1;
//if a node moved in X traverse then way value is 1, otherwise, the way value is - 1
m = OpenListCount +1;
// record the number of the open node after expanding node
OpenList[m]= NewOpenNodeID; // Places open list the new node(ID No.) to be final
   If map[MoveX][MoveY]=1
AddG=9999;     // if the node is at the pattern state, G value is infinity
    Else AdddG=1;          //if the node is at no pattern state ,G value is 1
G[MoveX][MoveY]=G[ParentXval][ParentYval]+ AddG; // calculated G value
H[OpenList[m]] = abs(MoveX - targetX) + abs(MoveY – targetY);// calculated H value
GetCrossing Crossing=new GetCrossing();          // Instantiated object
C[OpenList[m]]=Crossing.GetCrossing(OpenList[m]);
// calculated C value
F[OpenList[m]]= G[MoveX][MoveY]+H[OpenList[m]]+ C[OpenList[m]];
// calculated F value
If F[OpenList[m]]<9999     // If F value <9999 then the node can be expanded,
 {OpenListCount = OpenListCount +1;// add a node to OpenList
  whichNode[MoveX,MoveY]= ifOpenStadus;  }     // Expand the node
    }
  }
 }
```

V. SUMMARY

Suppose the game map is n rows and m columns, we can let N = m * n. Every node as the expanded node enters Open list is no mare than 1 time, thus, the number of nodes Open table processed is no more than N. Expanding every node needs O (1) the time, therefore the algorithm possess t time-consuming no less than O(N). Constructing the corresponding shortest path requires O (L) the time, and L is the length of the shortest path. When the number of nodes is N , the time complexity of breadth-first search algorithm is O (N * N) ,and that of A * algorithm is O (N + L). That computation does not increase rapidly with the increase node expanding. If only an effective path exits between the beginning and the end node, A * algorithm could find it quickly.

This paper analyzed rules about Picture Matching and the breadth first searching algorithm. Based on those, it gave the application of A* algorithm applied the game, and offered the corresponding data structure and the program improvement. It could use O(N+L) time to complete the shortest path-finding and adapted the game's request well. A* algorithm is a kind of algorithm which calculate the shortest path-finding and it unified the heuristic and the formalized method, used for many video games. We may modify the direction variable applied eight traverse path-finding. And it has a certain application to the game.

REFERENCES

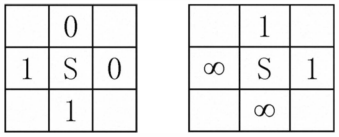
[1] Ma Chuanxiang and Li Gangqiang, “An Improved Algorithm Based on Maze in ‘Lianliankan’ game,”Computer and Digital Engineering,vol.35,Oct.2007,pp.18-20.

[2] Hu Zhenghong, “Application of a path-finding method in game development,”Shanxi Electronic Technology,vol.147,Dec.2009,pp.53-54.

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

[4] Hu Zhenghong, “Application of maintaining the shortest path method in the game map path-finding”, unpubished.

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



four directions of S G correspond S

Figure 1. Around the four-node status and value

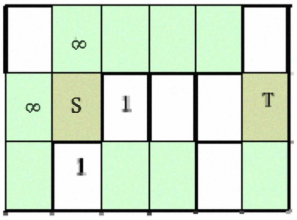


Figure 2. G calculation

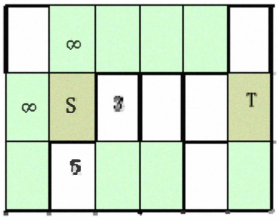


Figure 3. H calculation

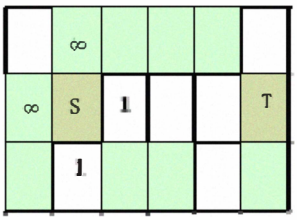


Figure 4. G calculation

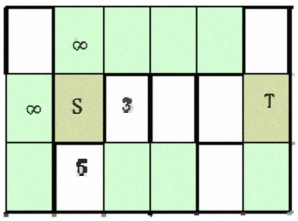


Figure 5. H calculation

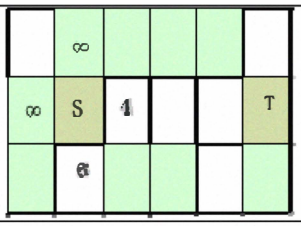


Figure 6. The estimate expressed

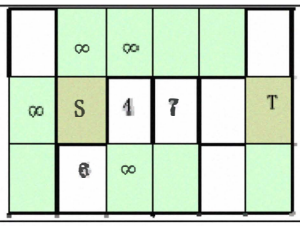


Figure 7. Expand the node f = 4

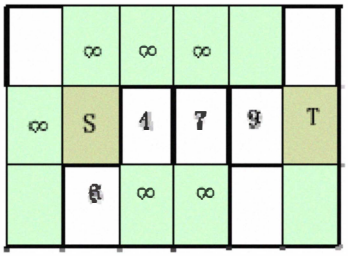


Figure 8. Expand the node f = 7

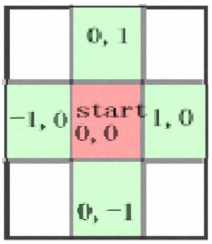


Figure 9. Digitized direction