International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 5, May-2014, pp: (58-66), Impact Factor: 1.252, Available online at: www.erpublications.com

Artificial Intelligence Pathfinding Algorithm and Its Application Research in the Game

Li Xinyun[1], Naveer Kumar N[2]
[12]VIT University, Vellore, India

Abstract: Artificial intelligence is an important part of the game design process. Pathfinding is one of the most basic problems in artificial intelligence applied to the game. The A* Algorithm is the most widely used Pathfinding Algorithm of Artificial intelligence, and it is also one of most effective short-Pathfinding Algorithm. Actually, A* algorithm is a heuristic search algorithm, which is based on breadth-first search. The normal A* algorithm get a path according to Closed table. If there is a blind road, the node in which will be included in the Closed table, and result in crooked road phenomena of Pathfinding.

Keywords: ARTIFICAL INTELLIGENCE, PATHFINDING, A* ALGORITHM

Introduction

Modern computer games through integrated graphics, physics and artificial intelligence method to achieve realistic game. Decide the quality of the game is game sense of reality. And what is realistic of game? In general, realistic gaming experience refers to the immersive game and the player characters appearing in the game of intelligence. In order to enhance realism to the game now on the market most computer games are comprehensively used Manage technology, Graphics technology and artificial intelligence technology. In the past few years the game's graphics technology Physical properties and simulation technology have made great progress. For example the DirectX is made by Microsoft and OpenGL. But now, to the physical characteristics of the application of simulation technology and graphics technology is not enough to make a game has uniqueness, at the same time also can't satisfy the high requirements of the quality of game players. So the game developers need to from other ways to enhance the sense of real experience. The application of artificial intelligence technology in the game will gradually get attention. And thePathfindingof artificial intelligence is one of the most popular research content, so a large number of domestic and foreign scholars have in path searching technology research.

Literature review

In most parts of Asia, the game industry is an emerging industry. Most of the game industry is in growth period, and it wills rapidly maturing stage. A few years ago, when the world financial crisis, the world economy has taken a knock, but the game industry has not been the impact of financial crisis, it has become the leader of the entire economic developments; it obtained the swift and violent development. Every year the game industry's income is increasing rapidly. For example, in China, the game industry revenue is increased than 2006 game industry revenue by 67% in 2007. Game industry development trend: with the rapid development of the industry, Game players also higher to the requirement of quality of the Game, the Game to be competitive in the market, we have to increase the degree of the graphics fine! Entertaining and challenging, and improve the speed of the Game. In addition, more importantly, to enhance the real experience of Game players, only in this way can the Game player's recognized and resistance, otherwise, the Game will not be able to survive in the market, will be eliminated. In all types of games, especially in the RPG (Role Playing Game) role-playing games virtual character of the most basic tasks is required in the shortest possible time to find the most suitable path, namely must effectively find a path, and enable it to inside the Game in accordance with the established path. This is used in the field of artificial intelligence in the path to search for related knowledge, Game developers should be according to the specific needs of artificial intelligence in Pathfinding algorithm was improved and optimized, make artificial intelligence a bigger development space for Game development, make a virtual character in the Game map walk look more intelligent. Before the game lack of personalized, almost no difference between rival, even there's a difference, is purely statistical difference. Former rival is achieved by hard coding; starting point is the game balance and writes a matter of time. Game AI opponents, on the other hand, are realized on the basis of the learning system. Only basic knowledge about the game world, the opponent often responds according to their own environment, and they will always be in similar circumstances. Therefore, the final game characters with more personality, intelligence level of the system is also more satisfactory. One day, we can create a completely intelligence game system, the role of these games can accord different levels of human players show different style, creativity, personality and intelligent level. In all kinds of game, the player character is the basic task of must effectively find a path, game programmers should according to the development of the specific needs to improve

and optimize the Pathfinding algorithm of artificial intelligence, artificial intelligence for game development and brings innovation and larger development space.

A * Pathfinding Algorithm

In the Pathfinding algorithm there have some famous algorithm in it, in these review it is show A* algorithm, it is contain the A* algorithm introduction, function and diagram. In the last there have a program to show the A* algorithm. The A* Algorithm is the most widely used Pathfinding Algorithm of Artificial intelligence, and it is also one of most effective short-Pathfinding Algorithm.

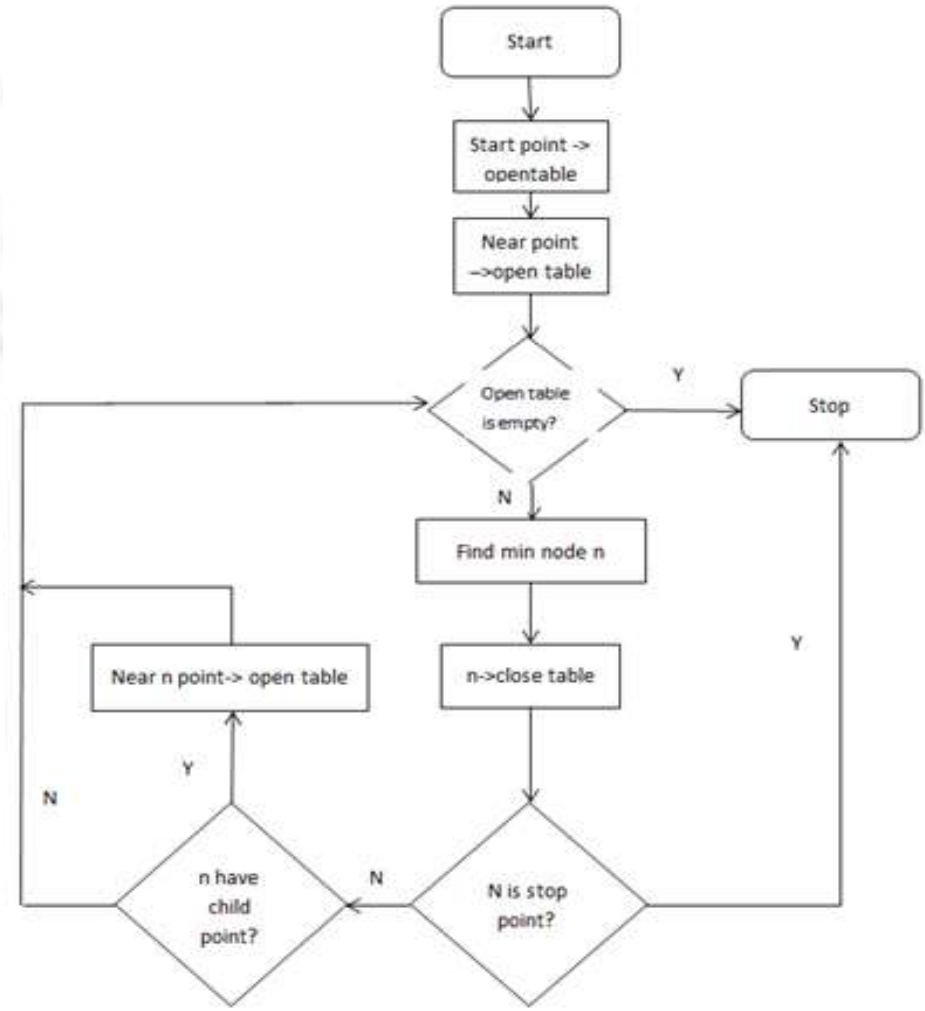
A* Algorithm is admissible and considers fewer nodes than any other admissible search algorithm with the same heuristic. This is because A* uses an "optimistic" estimate of the cost of a path through every node that it considers—optimistic in that the true cost of a path through that node to the goal will be at least as great as the estimate. But, critically, as far as A* "knows", that optimistic estimate might be achievable.

A* Algorithm using the function:

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

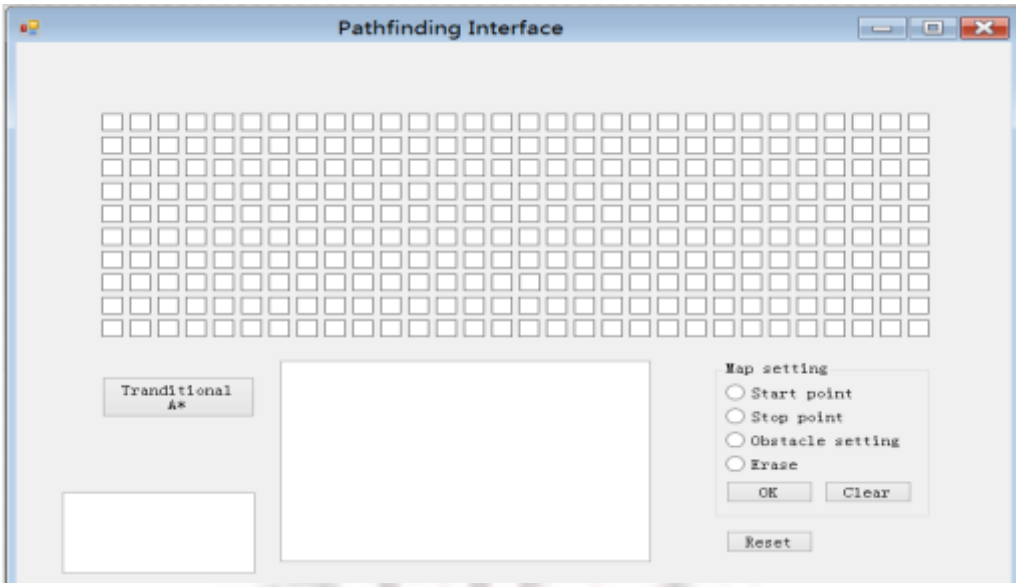
G (n) means the start node to the current node in the distance, H (n) said the current node to estimate the distance of the target node. Use this function to calculate the next step will be searched Dedicated to all the nodes of value, by comparing the option value of minimum node, as the node of the next to go.

Flowchart of A * Pathfinding Algorithm



Program of basic A* Pathfinding Algorithm

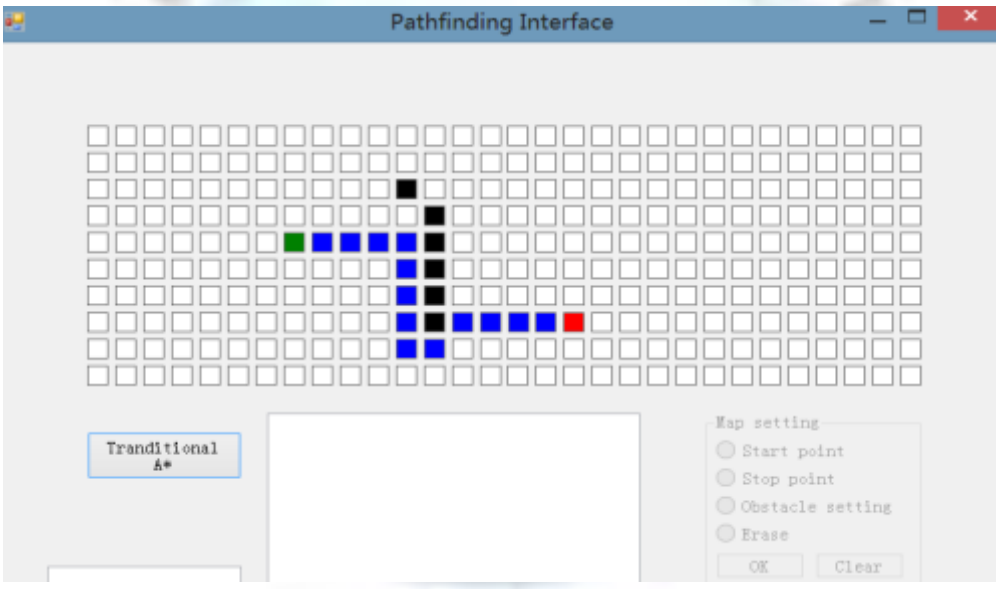
This is a program to show the traditional A* algorithm, it is contain the start point, stop point and obstacle.
This program can set the start point, stop point and obstacle, and then click the A* button and doing the A*



Pathfinding algorithm.

There have 10*30 matrix in here is instead of the Map, and each rectangle is instead of the pixel of Map.

Running is:



Green point is start point, Red point is stop point, black is obstacle. Blue is path road.

According to the program, we can clear to know the effect of the A* algorithm, and know how to finding the road using A* algorithm.This program is only show the basic of Pathfinding, this path is maybe not the best path, but it is can finding the path. If we must find the best path in it we must let the point find the path by itself, it is means let it intelligence, and how to do it? The answer is in next.

Advance of A* Pathfinding Algorithm

There have some element in the A* Algorithm. In the last chapter we know the A* Algorithm function F=G+H. Actually there have two more important lists in it, is called:

Open list: The one of list that to write down all the squares that is being considered to find the shortest path.
Close list: The one of list that to write down the square that does not have to consider it again.

The man (the point of Pathfinding) begins by adding his current position (we'll call this starting position point "A") to the closed list. Then, he adds all workable tiles adjacent to his current position to the open list.
After this the man need to find the shortest path. And how to find the shortest path? The answer is using the function F=G+H.
If we simplifying the search area, like the program we could divide the search area into pixels, but that's a granularity which is too high (and unnecessary) for our a tile-based game like this.

So instead, we will use the tile (a square shape) as unit for the Pathfinding algorithm. Variants with other type of shapes are possible (such as triangles or hexagons), but the square is the best fit for our needs and is also the simplest.

We'll give each square a score G + H where:

•G is the movement cost from the start point A to the current square. So for a square adjacent to the start point A, this would be 1, but this will increase as we get farther away from the start point.
•H is the estimated movement cost from the current square to the stop point This is often called the heuristic because we do not really know the cost yet – it's just an estimate.

Now we can using A* to finding the path: (In the last Chapter we know what is G and H)
Allow me to write the A* algorithm step again:
The man will find the shortest path by repeating the following steps:

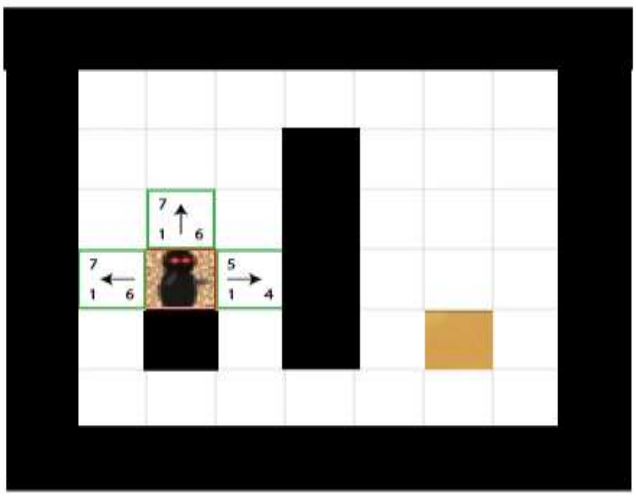
1. Get the square on the open list which has the lowest score. Let's call this square S.
2. Remove S from the open list and add S to the closed list.
3. For each square T in S's walkable adjacent tiles:

A. If T is in the closed list: Ignore it.
B. If T is not in the open list: Add it and compute its score.
C. If T is already in the open list: Check if the F score is lower when we use the current generated path to get there. If it is, update its score and update its parent as well.

Step by step using A* Algorithm

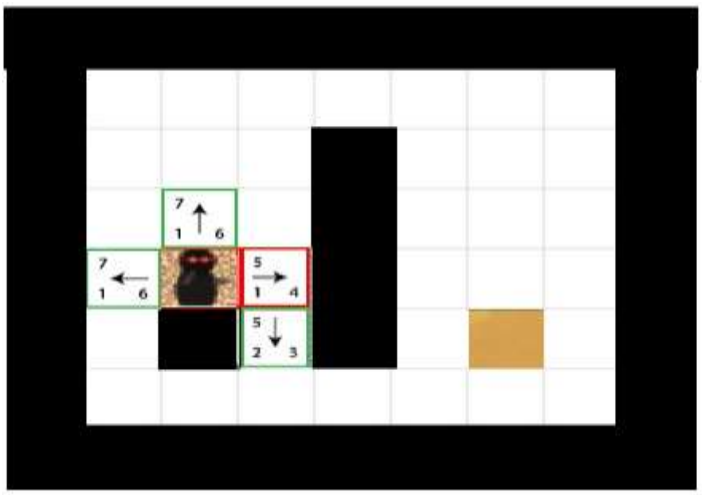
Step 1:

As A first step, the man will determine the relative to the starting position (point A) of the adjacent squares, computes their F scores, and then add them to the open list:



You can see that the H value is listed for each square (two have 6 and one has 4). I recommend counting out the squares according to the "city block distance" to make sure you understand how that part works.
Also note that the F value (in the upper right) is just the sum of G+H (lower left and lower right).

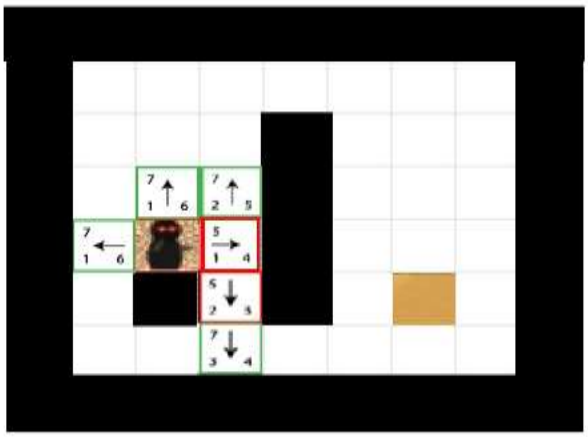
Step 2:
In the second step, the man chose the minimum value F square, add it to the closed list, and then the numerical values of the adjacent squares to retrieve it.

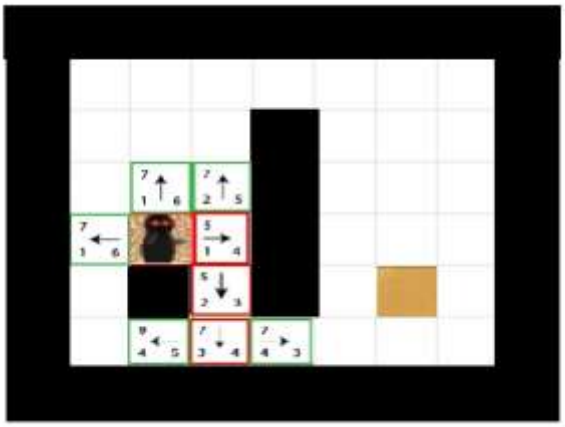


So you'll see here that the square with the lowest cost was the one that had F value as 4. It tried to add any tile adjacent to this to open list (and calculate their score), except notice that it couldn't add the cat tile (because it was already on the closed list) or the wall tile (because it wasn't walkable). Note is added to the open list two new square, their G value add 1, because they are far from the starting point has two squares.

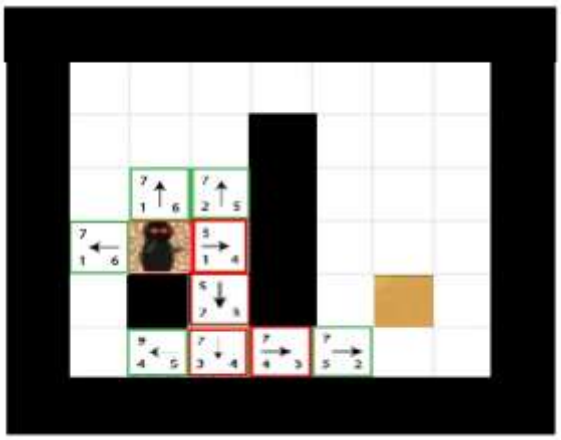
Step3:

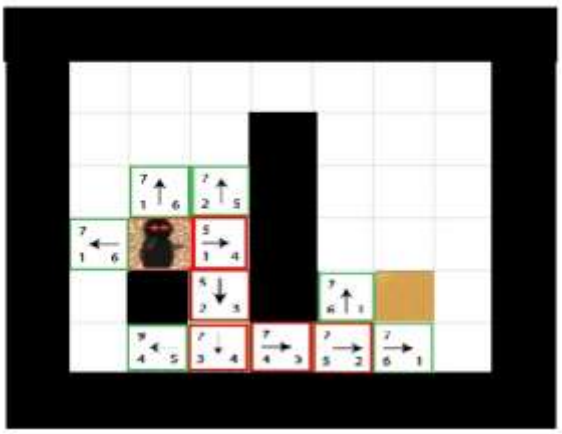
Again, we choose the tile with the lowest F score (F=5) and continue to iterate:
Now we see there have to squares G and F is same, and how we choose the next squares? I think the easiest is choosing the square that nearest addition to the open list so next is:





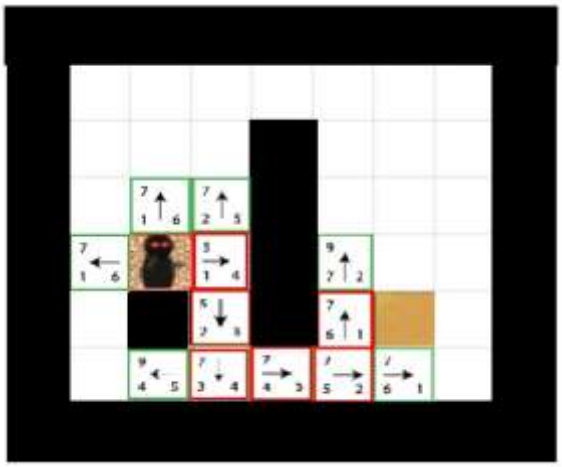
Step 4:
Again we choose the tile with the lowest score and in case of a tie choose the most recent:





and then:

Now there also have two squares, and which square is best? The answer is near start point square, so:



Then see the red squares, all there consist the best road form start point to stop point.

The shortest path is construct by starting by the final destination and go backward from parent to parent.
And we can see the pseudo code of A* Algorithm (Object-C):

```
[openListadd:originalSquare];
do {
	currentSquare = [openListsquareWithLowestFScore]; // Get the square to lowest F score

	[closedListadd:currentSquare]; // add the current square to the closed list
	[openListremove:currentSquare]; // remove it from the open list

	if ([closedListcontains:destinationSquare]) { // if we added the destination to the closed list, we've found a path
		// PATH FOUND
		break; // break the loop
	}

	adjacentSquares = [currentSquarewalkableAdjacentSquares]; // Retrieve all its walkable adjacent squares

	foreach (aSquare in adjacentSquares) {

		if ([closedListcontains:aSquare]) { // if this adjacent square is already in the closed list ignore it
			continue; // Go to the next adjacent square
		}

		if (![openListcontains:aSquare]) { // if not in the open list

			// compute its score, set the parent
			[openListadd:aSquare]; // and add to the open list

		}
else { // if its already in the open list

			// test if using the current G score make the aSquare F score lower, if yes update the parent because it means its a better path

		}
}
 } while(![openListisEmpty]); // Continue until there is no more available square in the open list (which means there is no path)
```

Now we easy to know What is A* Algorithm and the step of it.

International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 5, May-2014, pp: (58-66), Impact Factor: 1.252, Available online at: www.erpublications.com

Application of A* Pathfinding Algorithm

It can create a simper game to show the A* Pathfinding. Most of games like WOW, LOL, DOTA need the Pathfinding Algorithm, it is also basic of them.

Prepare:

This game I using the Xcode software create a cocos2d program, so it is need a map, and some sound file. So Download the Tile software and editing the map:

See the map and it is contain elements:

Land: 

Tree: 

Water: 

Desert: 

Gravel: 

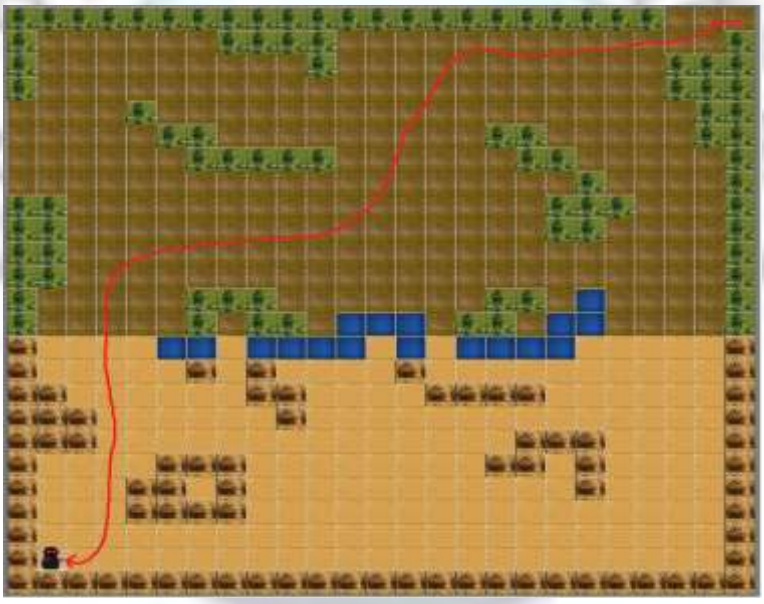
In here the land and desert is walkable. Tree water and gravel arebarrier. The protagonist is go on food in this map:

Protagonist: 

Using the mouse click any point in the map, if the protagonist can arrive, then the protagonist can go there, else the protagonist can stand immobility. This program is the best to show The A* pathfinding Algorithm application in the game. Most of RGB like game can using this program function to control the man move.
OK, let running the game:

First the man is stand in the top right corner of map:

Then we using the mouse click the finish point: (here setting in the bottom left)
Next you can see the man move to the finish point:



This moving is include the pseudo code in page 5, if you want create like this game, you must be study how to do the cocos2dx game building. Study well and do it is easy

Supplement

If around the road have not any barrier the man is may be to do diagonal movement, because it is decrement the road number and timesaver, it is also let the man move not mechanization. But if the map is like this:

The man is not move between two of land squares.
See a example:

According to the Pythagorean Theorem, $C^2 = A^2 + B^2$, so:

$C = \sqrt{(A^2 + B^2)}$
with A = B = 1 (The movement cost to move from a square to another = G cost)
$C = \sqrt{(2)}$
$C \approx 1.41$

Conclusion

In this paper, we can know the importance of Pathfinding Algorithm in the Game, It is not only the basic point of the game, it is also a Artificial Intelligence application. This paper is important research the A* Pathfinding Algorithm, but it is only one of Pathfinding function in the Pathfinding Algorithm. The essence of A* algorithm is the "best and precedence" depth of the search.

The core of implementation, is the two tables, and consists of the parent node of a tree structure:

Two tables is the OPEN list and the CLOSE list, including the OPEN list has not passed, can be used as the nodes of the next games. The CLOSE list contains those to walk through is already in use, and it is made of the best ways to achieve the set of nodes. Every decision, from the OPEN list inside take out one of the best point, generally is the smallest price points, as the next attempt. Due to node in the OPENlist, the record of the parent node, is the best way (can use recursive method), as a result, the point being tried, can naturally, also should be put into the CLOSE list. The process over and over again, until you reach the goal, or no option. So this is A* flow. And in the page2 also have a flow chart of this Algorithm.

References

[1]. Hierarchical path planning for multi-size agents in heterogeneous environments Harabor D. and Botea, A.IEEE Symposium on Computational Intelligence & Games, 2008 J. Clerk Maxwell, “A Treatise on Electricity and Magnetism”, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[2]. Alexander, Thor,. “GoCap: Game Observation Capture”, AI Game Programming Wisdom, Charles River Media, 2002.
[3]. Ray Wenderlich,.”Introduction to A* Pathfinding”,blog, on September 29, 2011.
[4]. Donald Hearn .ComPuter Graphie(Third Edition),USA : Addsion Wisley, 2005.
[5]. Zhang qianshao. “Map based on A * algorithm for the diameter of the research” (master's degree thesis). Wuhan university of scienceand technology, 2005.