A-STAR SHORTEST PATH ALGORITHM (C++ RECIPE) BY FB36

ACTIVESTATE CODE (HTTP://CODE.ACTIVESTATE.COM/RECIPES/577457/)

A-star (A*) is a shortest path algorithm widely used for RTS games, GPS navigation etc.

C++, 292 lines

```cpp
// Astar.cpp
// http://en.wikipedia.org/wiki/A*
// Compiler: Dev-C++ 4.9.9.2
// FB - 201012256
#include <iostream>
#include <iomanip>
#include <queue>
#include <string>
#include <math.h>
#include <ctime>
using namespace std;

const int n=60; // horizontal size of the map
const int m=60; // vertical size size of the map
static int map[n][m];
static int closed_nodes_map[n][m]; // map of closed (tried-out) nodes
static int open_nodes_map[n][m]; // map of open (not-yet-tried) nodes
static int dir_map[n][m]; // map of directions
const int dir=8; // number of possible directions to go at any position
// if dir==4
//static int dx[dir]={1, 0, -1, 0};
//static int dy[dir]={0, 1, 0, -1};
// if dir==8
static int dx[dir]={1, 1, 0, -1, -1, -1, 0, 1};
static int dy[dir]={0, 1, 1, 1, 0, -1, -1, -1};

class node
{
    // current position
    int xPos;
    int yPos;
    // total distance already travelled to reach the node
```

```
33      int level;
34      // priority=level+remaining distance estimate
35      int priority;  // smaller: higher priority
36
37      public:
38          node(int xp, int yp, int d, int p)
39              {xPos=xp; yPos=yp; level=d; priority=p;}
40
41          int getxPos() const {return xPos;}
42          int getyPos() const {return yPos;}
43          int getLevel() const {return level;}
44          int getPriority() const {return priority;}
45
46          void updatePriority(const int & xDest, const int & yDest)
47          {
48               priority=level+estimate(xDest, yDest)*10; //A*
49          }
50
51          // give better priority to going strait instead of diagonally
52          void nextLevel(const int & i) // i: direction
53          {
54               level+=(dir==8?(i%2==0?10:14):10);
55          }
56
57          // Estimation function for the remaining distance to the goal.
58          const int & estimate(const int & xDest, const int & yDest) const
59          {
60              static int xd, yd, d;
61              xd=xDest-xPos;
62              yd=yDest-yPos;
63
64              // Euclidian Distance
65              d=static_cast<int>(sqrt(xd*xd+yd*yd));
66
67              // Manhattan distance
68              //d=abs(xd)+abs(yd);
69
70              // Chebyshev distance
71              //d=max(abs(xd), abs(yd));
72
73              return(d);
74          }
75  };
76
77  // Determine priority (in the priority queue)
78  bool operator<(const node & a, const node & b)
79  {
80    return a.getPriority() > b.getPriority();
81  }
82
83  // A-star algorithm.
```

```
 84 // The route returned is a string of direction digits.
 85 string pathFind( const int & xStart, const int & yStart,
 86                  const int & xFinish, const int & yFinish )
 87 {
 88     static priority_queue<node> pq[2]; // list of open (not-yet-tried) nodes
 89     static int pqi; // pq index
 90     static node* n0;
 91     static node* m0;
 92     static int i, j, x, y, xdx, ydy;
 93     static char c;
 94     pqi=0;
 95 
 96     // reset the node maps
 97     for(y=0;y<m;y++)
 98     {
 99         for(x=0;x<n;x++)
100         {
101             closed_nodes_map[x][y]=0;
102             open_nodes_map[x][y]=0;
103         }
104     }
105 
106     // create the start node and push into list of open nodes
107     n0=new node(xStart, yStart, 0, 0);
108     n0->updatePriority(xFinish, yFinish);
109     pq[pqi].push(*n0);
110     open_nodes_map[x][y]=n0->getPriority(); // mark it on the open nodes map
111 
112     // A* search
113     while(!pq[pqi].empty())
114     {
115         // get the current node w/ the highest priority
116         // from the list of open nodes
117         n0=new node( pq[pqi].top().getxPos(), pq[pqi].top().getyPos(),
118                      pq[pqi].top().getLevel(), pq[pqi].top().getPriority());
119 
120         x=n0->getxPos(); y=n0->getyPos();
121 
122         pq[pqi].pop(); // remove the node from the open list
123         open_nodes_map[x][y]=0;
124         // mark it on the closed nodes map
125         closed_nodes_map[x][y]=1;
126 
127         // quit searching when the goal state is reached
128         //if((*n0).estimate(xFinish, yFinish) == 0)
129         if(x==xFinish && y==yFinish)
130         {
131             // generate the path from finish to start
132             // by following the directions
133             string path="";
134             while(!(x==xStart && y==yStart))
```

```
135             {
136                 j=dir_map[x][y];
137                 c='0'+(j+dir/2)%dir;
138                 path=c+path;
139                 x+=dx[j];
140                 y+=dy[j];
141             }
142 
143             // garbage collection
144             delete n0;
145             // empty the leftover nodes
146             while(!pq[pqi].empty()) pq[pqi].pop();
147             return path;
148         }
149 
150         // generate moves (child nodes) in all possible directions
151         for(i=0;i<dir;i++)
152         {
153             xdx=x+dx[i]; ydy=y+dy[i];
154 
155             if(!(xdx<0 || xdx>n-1 || ydy<0 || ydy>m-1 || map[xdx][ydy]==1
156                 || closed_nodes_map[xdx][ydy]==1))
157             {
158                 // generate a child node
159                 m0=new node( xdx, ydy, n0->getLevel(),
160                              n0->getPriority());
161                 m0->nextLevel(i);
162                 m0->updatePriority(xFinish, yFinish);
163 
164                 // if it is not in the open list then add into that
165                 if(open_nodes_map[xdx][ydy]==0)
166                 {
167                     open_nodes_map[xdx][ydy]=m0->getPriority();
168                     pq[pqi].push(*m0);
169                     // mark its parent node direction
170                     dir_map[xdx][ydy]=(i+dir/2)%dir;
171                 }
172                 else if(open_nodes_map[xdx][ydy]>m0->getPriority())
173                 {
174                     // update the priority info
175                     open_nodes_map[xdx][ydy]=m0->getPriority();
176                     // update the parent direction info
177                     dir_map[xdx][ydy]=(i+dir/2)%dir;
178 
179                     // replace the node
180                     // by emptying one pq to the other one
181                     // except the node to be replaced will be ignored
182                     // and the new node will be pushed in instead
183                     while(!(pq[pqi].top().getxPos()==xdx &&
184                            pq[pqi].top().getyPos()==ydy))
185                     {
```

```
                        pq[1-pqi].push(pq[pqi].top());
                        pq[pqi].pop();
                    }
                    pq[pqi].pop(); // remove the wanted node

                    // empty the larger size pq to the smaller one
                    if(pq[pqi].size()>pq[1-pqi].size()) pqi=1-pqi;
                    while(!pq[pqi].empty())
                    {
                        pq[1-pqi].push(pq[pqi].top());
                        pq[pqi].pop();
                    }
                    pqi=1-pqi;
                    pq[pqi].push(*m0); // add the better node instead
                }
                else delete m0; // garbage collection
            }
        }
        delete n0; // garbage collection
    }
    return ""; // no route found
}

int main()
{
    srand(time(NULL));

    // create empty map
    for(int y=0;y<m;y++)
    {
        for(int x=0;x<n;x++) map[x][y]=0;
    }

    // fillout the map matrix with a '+' pattern
    for(int x=n/8;x<n*7/8;x++)
    {
        map[x][m/2]=1;
    }
    for(int y=m/8;y<m*7/8;y++)
    {
        map[n/2][y]=1;
    }

    // randomly select start and finish locations
    int xA, yA, xB, yB;
    switch(rand()%8)
    {
        case 0: xA=0;yA=0;xB=n-1;yB=m-1; break;
        case 1: xA=0;yA=m-1;xB=n-1;yB=0; break;
        case 2: xA=n/2-1;yA=m/2-1;xB=n/2+1;yB=m/2+1; break;
        case 3: xA=n/2-1;yA=m/2+1;xB=n/2+1;yB=m/2-1; break;
```

```
        case 4: xA=n/2-1;yA=0;xB=n/2+1;yB=m-1; break;
        case 5: xA=n/2+1;yA=m-1;xB=n/2-1;yB=0; break;
        case 6: xA=0;yA=m/2-1;xB=n-1;yB=m/2+1; break;
        case 7: xA=n-1;yA=m/2+1;xB=0;yB=m/2-1; break;
    }

    cout<<"Map Size (X,Y): "<<n<<","<<m<<endl;
    cout<<"Start: "<<xA<<","<<yA<<endl;
    cout<<"Finish: "<<xB<<","<<yB<<endl;
    // get the route
    clock_t start = clock();
    string route=pathFind(xA, yA, xB, yB);
    if(route=="") cout<<"An empty route generated!"<<endl;
    clock_t end = clock();
    double time_elapsed = double(end - start);
    cout<<"Time to calculate the route (ms): "<<time_elapsed<<endl;
    cout<<"Route:"<<endl;
    cout<<route<<endl<<endl;

    // follow the route on the map and display it
    if(route.length()>0)
    {
        int j; char c;
        int x=xA;
        int y=yA;
        map[x][y]=2;
        for(int i=0;i<route.length();i++)
        {
            c =route.at(i);
            j=atoi(&c);
            x=x+dx[j];
            y=y+dy[j];
            map[x][y]=3;
        }
        map[x][y]=4;

        // display the map with the route
        for(int y=0;y<m;y++)
        {
            for(int x=0;x<n;x++)
                if(map[x][y]==0)
                    cout<<".";
                else if(map[x][y]==1)
                    cout<<"O"; //obstacle
                else if(map[x][y]==2)
                    cout<<"S"; //start
                else if(map[x][y]==3)
                    cout<<"R"; //route
                else if(map[x][y]==4)
                    cout<<"F"; //finish
            cout<<endl;
```

```
288         }
289     }
290     getchar(); // wait for a (Enter) keypress
291     return(0);
292 }
```

Tags: algorithm, algorithms, game, graph, graphs, routes

9 COMMENTS

Léo LETARO 6 years, 3 months ago

I updated your code, to remove all memory leaks.

Need a `delete n0' before L113 Need a`delete m0' after L168 (now 169) Need a `delete m0' after L199 (now 201)

(line numbers are always *your* line numbers)

I also needed some more includes (<cstdlib> <cstdio>)

Jirka 5 years, 10 months ago

I think the major memory leak is caused by line 213, delete text "else delete m0" and add "delete m0" after line 214. So every new m0 will be deleted.

Mark Anthony 3 years, 8 months ago

Wow! This is very great! Many many thanks for this :D

Uhm, question here...

What does this suppose to do:

```
open_nodes_map[x][y]=n0->getPriority(); // mark it on the open nodes map
```

At line **116** ?

I think *x == n && y == m*? And shouldn't be the *x, y* were *xStart, yStart* ?

I have integrated your code in my project but most of my **ROWS** and **COLS** are dynamic and I got a random error somehow in MSVC saying *debug assertion failed* ?

Paul Hutchinson 3 years, 5 months ago

You are trashing memory (stack memory) on line 110:

```
open_nodes_map[x][y]=n0->getPriority(); // mark it on the open nodes map
```

y=m, x=n which are out of bounds. This should be:

```
open_nodes_map[xStart][yStart]=n0->getPriority(); // mark it on the open nodes map
```

Kristian Kristola 2 years, 7 months ago

There's also a bug on line 266. j=atoi(&c); is wrong because there must be a null after c in order for this to work. It should be: j=c-'0';

AJW 1 year, 9 months ago

Hey, I incorporated some of these helpful comments from other commenter's posts, this compiles on Dev C++ and now runs without causing abnormal termination on Windows 10

https://gist.github.com/soundmasteraj/8233fdd6fb939c126843e76beb986801.js

Thanks everybody! Hope this is helpful in some way. :)

https://code.activestate.com/recipes/users/4192555/ https://code.activestate.com/recipes/users/4180055/

shubham jaiswal 1 year, 2 months ago

include these two library then work correctly.

INCLUDE <STDIO.H>

INCLUDE <STDLIB.H>

yosua andreas 7 months, 4 weeks ago

why switch(rand()%8) in L231 cannot be used as a fuction ?

MTR 7 months, 2 weeks ago

Thanks for your great efforts in this code :). But, I had an issue of run time error in this code, and after modifying Line (265) to be: j= (int)c - int(48); it worked well ,hope this help somebody :).