3D A* Path Planning Algorithm Description

I. Program Design

(1) Introduction to the A* Algorithm

The A-Star algorithm is used to find the shortest path in a static road network. It starts from the starting position and evaluates the surrounding points for each point to find the best position, then searches from this position until it reaches the target position.

(2) Specific Process

1. According to the map data, mark the obstacle-free points into node.setWalkable(isWalkable);
2. Select the current point node from the known point set close based on the specified point range condition, traverse the points to be searched nodes[k][j][i], and select the point with the smallest evaluation function F as the next node;
3. When the path connects to the endpoint, stop searching and output the point set in reverse order to get the forward path. Otherwise, continue the search operation.

(3) Evaluation Function

$$F(n) = g(n) + h(n)$$

$g(n)$represents the movement cost from the starting point A to the specified point, $h(n)$ represents the estimated cost from the specified point to the endpoint B.

$$g(n) = \begin{cases} 10, & \text{orthogonal adjacent points} \\ 14, & \text{planar diagonal adjacent points} \\ 17, & \text{spatial diagonal adjacent points} \end{cases}$$

The heuristic function $h(n)$ is M times the Manhattan distance, where M is set to 10 in the program:

$$h(n) = M \times (|X_1 - X_2| + |Y_1 - Y_2| + |Z_1 - Z_2|)$$

The open set stores all generated but unvisited nodes, and the close set stores visited nodes. Specific operations are as follows:

1. Take a point from the open set and store it in the close set;
2. Exclude adjacent nodes from the close set, if the adjacent node of the current node is not in the open set, add it. Then set this node as the parent node of the newly added node in the open set;
3. If the adjacent node already exists in the open set, only check if the path through this node to the adjacent node has a smaller g value. If the g value is smaller, set the parent node of the adjacent node to this node, then recalculate the f and g values.

II. Program Instructions

Main Application Source File Star.cpp

(1) Reading the Map and Setting Related Data

Read the map information marked in the txt file into the 3D matrix array[i][j][k], where i, j, and k represent the x, y, and z coordinates of the current point, respectively $array[i][j][k] = \begin{cases} 0, & The\ point\ (i,j,k)\ is\ impassable \\ 1, & The\ point\ (i,j,k)\ is\ passable \end{cases}$

```
#define  xDepth  200
#define  yDepth  200
#define  zDepth  30

Node3D  nodes[200][200][30] ;
short   aray[200][200][30] = { 0 };

ifstream infile; //定义读取文件流
    infile.open("D:\inputt.txt", ios::in); //打开文件
    if (!infile.is_open())
    {
        cerr << "open error!" << endl;
        exit(1);
    }
    for (i = 0; i < 200; i++)//定义行循环
    {
        for (j = 0; j < 200; j++)//定义列循环
        {
            for (k = 0; k < 30; k++)
            {
                infile >> aray[i][j][k];
            }
        }
    }
    infile.close(); //关闭文件
   //起始点定义
    int startpointx;
    int startpointy;
    int startpointz;
    int endpointx;
    int endpointy;
    int endpointz;
    cout << "Please input the strat point:" << endl;
    cin >> startpointx >>startpointy >>startpointz;
    cout << "Please input the end point:" << endl;
    cin >> endpointx >> endpointy >> endpointz;

    for ( int i = 0 ; i < zDepth ; i++ )
    {
        for ( int j  = 0; j < yDepth ; j++ )
        {
            for ( int k = 0 ; k < xDepth ; k++ )
            {
                Point pi;
                pi.xPos = k ;
                pi.yPos = j ;
                pi.zPos = i ;
                bool isWalkable = aray[k][j][i] >= 1 ? true : false;//1为通，0为阻
                node.setWalkable( isWalkable );
                node.setPoint( pi );
                nodes[k][j][i] = node;
            }
        }
    }

    startp = nodes[startpointx][startpointy][startpointz];// 设置开始节点
    endp   = nodes[endpointx][endpointy][endpointz];//设置结束节点
```

(2) Path Finding Function Star::Find_path(Node3D * node)

Find the best path points according to the constraints, connect the path and output it.

```cpp
Find_path( &nodes[startpointx][startpointy][startpointz] );//执行函数

bool Star::Find_path(Node3D * node)
{
    if ( *node != endp  && endp.getWalkable() )
    {
        do
        {
            close[node] =  node->f;
            int sz = max(node->point.zPos -1 , 0 );
            int ez = min(node->point.zPos +1 ,zDepth-1);

            int sy = max(node->point.yPos -1 , 0 );
            int ey = min(node->point.yPos +1 ,yDepth-1);

            int sx = max(node->point.xPos -1 , 0 );
            int ex = min(node->point.xPos +1 ,xDepth-1);//点不能出界
            //确定周围点范围
            for ( int i = node->point.zPos-1; i <= ez ; i++ )
            {
                for ( int j  = node->point.yPos-1; j <= ey ; j++ )
                {
                    for ( int k = node->point.xPos; k <= ex ; k++ )
                    {
                        if ( !( i == node->point.zPos && j== node->point.yPos && k == node->point.xPos ) && !(j == node->point.yPos && k == node->point.xPos))//不能是点本身,不能垂直向上走
                        {
                            searchchNode( &nodes[k][j][i], node );//寻找最佳点
                        }
                    }
                }
            }

            Node3D * best = NULL;
            if ( open.size() >= 1 )//如果点集里面有点，则取出原来的第一个点，放入新的点
            {
                best = open.begin()->first;
                open.erase( open.begin() );//删除容器中position所指位置的元素。返回值是指向被删元素之后的那个元素(即下一个元素)的迭代器
            }
            node = best;
        } while ( node != NULL  && (*node != endp)) ;
    }

    if ( node != NULL  && endp.getWalkable() && (*node == endp))//路线连接到终点了
    {
        std::cout << " 找到路径 " << std::endl;
        Node3D *adjNode = node;

        path.clear();
        while ( *adjNode != startp )
        {
            if ( *adjNode != startp )
            {
                adjNode->setPath( true );
                path.push_back( adjNode->point );
            }
            if ( adjNode->parent == NULL )
            {
                break ;
            }
            adjNode = (Node3D*)(adjNode->parent);
        }

        path.push_back( adjNode->point );
        for ( int i = path.size()-1 ; i >=0  ; --i )
        {
            char str[1000];
            path[i].tostring(str);
            std::cout << str << std::endl;//输出路径
        }
    }
    else
    {
        std::cout << " 不能找到路径 "  << std::endl;
    }
    return true ;
}
```

(3) Point finding function: Star::searchchNode(Node3D * adjNode , Node3D * current)

Determine the position of the next node by evaluating the relative size of the total cost function adjNode->f of the current node Node3D * current and the candidate node Node3D * adjNode, and select the best point.

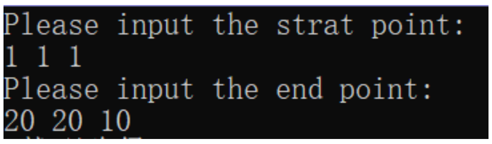
```cpp
void Star::searchchNode(Node3D * adjNode  , Node3D * current)
{

    bool xZDiagonal = ( adjNode->point.xPos != current->point.xPos && adjNode->point.zPos != current->point.zPos );
    bool xYDiagonal = ( adjNode->point.xPos != current->point.xPos && adjNode->point.yPos != current->point.yPos );
    bool yZDiagonal = ( adjNode->point.yPos != current->point.yPos && adjNode->point.zPos != current->point.zPos );
    bool corner = false;
    Node3D * tmp = current;
    //相邻节点不在open中
    if(  adjNode->getWalkable() &&  !findItem(open,adjNode) && !findItem(close,adjNode) && !corner)//corner为true
    {

        adjNode->parent = tmp;
        adjNode->g = current->g + (xZDiagonal || xYDiagonal || yZDiagonal) ? 14 : 10;

        int difX = endp.point.xPos - adjNode->point.xPos;
        int difY = endp.point.yPos - adjNode->point.yPos;
        int difZ = endp.point.zPos - adjNode->point.zPos;

        if( difX < 0 )   difX = difX * -1 ;
        if( difY < 0 )   difY = difY * -1 ;
        if( difZ < 0 )   difZ = difZ * -1 ;

        adjNode->h = (difX + difY + difZ ) * 10;
        adjNode->f = adjNode->g + adjNode->h ;

        open.insert(  pair<Node3D*,int>( adjNode , adjNode->f ) );
    }
    //相邻节点在open中
    else if  ( adjNode->getWalkable() && !findItem(close,adjNode) && !corner ) //
    {
        int g = current->g  + ( xZDiagonal || xYDiagonal || yZDiagonal ) ? 14 : 10;
        if ( g < adjNode->g )
        {
            adjNode->g = g;
            adjNode->parent = current;
        }
    }
}
```

III. Usage Instructions

Provide a txt file with map data, data storage format as follows:

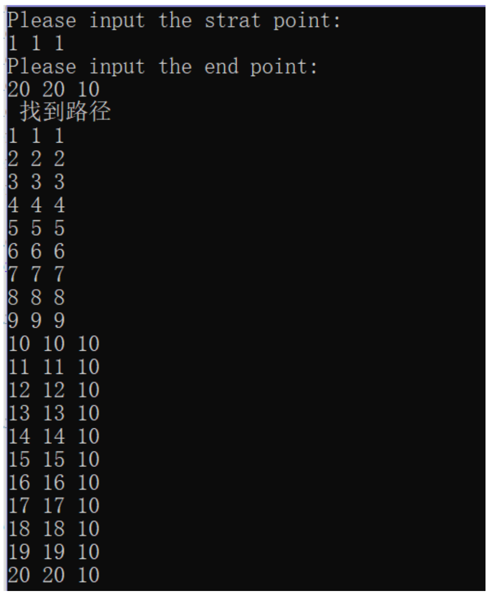
（x,y）increase direction↓	Height z increase direction→
（0，0）	0/1 (1 indicates no obstacle, 0 indicates an obstacle)
（0，1）	
（0，2）	
……	
（1，0）	
（1，1）	
（1，2）	
……	

Input the three-dimensional coordinates of the starting and ending points:



```
Please input the strat point:
1 1 1
Please input the end point:
20 20 10
```

The program's output results are as follows:



```
Please input the strat point:
1 1 1
Please input the end point:
20 20 10
 找到路径
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
6 6 6
7 7 7
8 8 8
9 9 9
10 10 10
11 11 10
12 12 10
13 13 10
14 14 10
15 15 10
16 16 10
17 17 10
18 18 10
19 19 10
20 20 10
```