A*Start

```
namespace NewAStart
{
    public enum EAStartNodeType
    {
        walk, stop
    }
    public class AStartNode
    {
        //坐标点
        public int x, y;
        //格子类型
        public EAStartNodeType type;

        //寻路消耗
        public  float f, g, h;

        //父节点
        public AStartNode father;

        public AStartNode(int x, int y, EAStartNodeType type)
        {
            this.x = x; this.y = y; this.type = type;
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using UnityEngine;

namespace NewAStart
{
    public class AStartManager
    {
        private static AStartManager instance;
        public static AStartManager GetInstance()
        {
```

```
12          if (instance == null)
13              instance = new AStartManager();
14          return instance;
15      }
16      //地图的宽高
17      public int mapW, mapH;
18
19      //格子对象
20      public AStartNode[,] mapNodes;
21      //开启和关闭列表
22      public List<AStartNode> openList = new List<AStartNode>();
23      public List<AStartNode> closeLists = new List<AStartNode>();
24
25      //初始化地图
26      public void InitMap(int w, int h,int stopNum)
27      {
28          //初始化地图
29          mapNodes = new AStartNode[w, h];
30          mapW = w;
31          mapH = h;
32
33          //实例化地图
34          for (int i = 0; i < w; i++)
35          {
36              for (int j = 0; j < h; j++)
37              {
38                  mapNodes[i, j] = new AStartNode(i, j, EAStartNodeType.walk);
39              }
40          }
41          for (int i = 0; i < stopNum; i++)
42          {
43              int x = UnityEngine.Random.Range(0, w);
44              int y = UnityEngine.Random.Range(0, h);
45              mapNodes[x, y].type = EAStartNodeType.stop;
46          }
47
48      }
49
50      //寻找路径
51      public List<AStartNode> FindPath(Vector2 startPos, Vector2 endPos)
52      {
53          //清除列表
54          openList.Clear();
55          closeLists.Clear();
56
57          int spX = Mathf.FloorToInt(startPos.x);
58          int spY = Mathf.FloorToInt(startPos.y);
59          int epX = Mathf.FloorToInt(endPos.x);
```

```
        int epY = Mathf.FloorToInt(endPos.y);

        var startNode = mapNodes[spX, spY];
        var endNode = mapNodes[epX, epY];

        //判断起点或者终点是否在地图外
        if (InMapExternal(spX, spY) || InMapExternal(epX, epY))
        {
            Debug.Log("起点或者终点再地图外");
            return null;
        }
        //判断起点或者终点是否是障碍
        if (startNode.type == EAStartNodeType.stop || endNode.type == EAStartNodeType.stop)
        {
            Debug.Log("起点或者终点是障碍");
            return null;
        }
        closeLists.Add(startNode);//把起点加入关闭列表中
        bool success = FindEndPoint(spX, spY, epX, epY);
        //是否找到终点
        if (success)
        {
            //从终点开始往前，一直找初始点
            var pathEndNode = closeLists[closeLists.Count - 1];
            var pathList = new List<AStartNode>();
            var curNode = pathEndNode;
            //路径绘制
            while (true)
            {
                pathList.Add(curNode);
                if (curNode.father == null)
                {
                    break;
                }

                    curNode = curNode.father;

            }
            //因为是从终点开始往起点找的，画路径点的时候要从起点开始，反转一下列表
            pathList.Reverse();
            return pathList;
        }
        return null;

    }
    bool FindEndPoint(int sx, int sy, int ex, int ey)
    {
        var startNode = mapNodes[sx, sy];
```

```
108         for (int i = -1; i < 2; i++)
109         {
110             for (int j = -1; j < 2; j++)
111             {
112                 //中心点跳出
113                 if (i == 0 & j == 0)
114                 {
115                     continue;
116                 }
117                 int cx = sx + i;
118                 int cy = sy + j;
119 
120                 //如果当前点等于终点，结束
121                 if (cx== ex && cy == ey)
122                 {
123                     return true;
124                 }
125                 //判断点是否在地图内 是否是障碍，是否再列表中
126                 if (InMapExternal(cx, cy) )
127                 {
128                     continue;
129                 }
130                 var curNode = mapNodes[cx, cy];
131                 if ( curNode.type == EAStartNodeType.stop )
132                 {
133                     continue;
134                 }
135                 if (openList.Contains(curNode) || closeLists.Contains(curNode))
136                 {
137                     continue;
138                 }
139 
140                 //设置父节点
141                 curNode.father = startNode;
142                 //f=g+h;  g=父节点的g+当前节点到父节点的距离
143                 float d = 1;
144                 if (i != 0 || j != 0)
145                 {
146                     d = 1.4f;
147                 }
148                 float g = startNode.g + d;
149 
150                 float h1 = Mathf.Abs(cx - sx);
151                 float h2 = Mathf.Abs(cy - sy);
152                 float h = h1 + h2;
153 
154                 float f = g + h;
155                 curNode.g = g;
```

```
156                 curNode.h = h;
157                 curNode.f = f;
158 
159                 openList.Add(curNode);
160             }
161         }
162         //判断是否为死路
163         if (openList.Count == 0)
164         {
165             Debug.LogError("这是一条死路");
166             return false;
167         }
168         //对openList进行排序 从如果返回一，这说明node1大于node2 ，交换位置node2在前面
169         openList.Sort((node1, node2) => { return node1.f >= node2.f ? 1 : -1; });
170         //把最小的从开放列表移出，放入关闭列表
171         var minNode = openList[0];
172         openList.RemoveAt(0);
173         closeLists.Add(minNode);
174 
175 
176         //找到了目标点
177         if (minNode.x == ex & minNode.y == ey)
178         {
179             return true;
180         }
181 
182         //递归，寻找下一个点
183         return FindEndPoint(minNode.x,minNode.y,ex,ey);
184     }
185     //判断坐标点是否在地图外
186     bool InMapExternal(int x, int y)
187     {
188         if (x < 0 || x >= mapW || y < 0 || y >= mapH) return true;
189         return false;
190     }
191   }
192 
193 }
194 
```

客户端测试

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.IO;
5 using UnityEngine;
6 
```

```
public class AStartTest : MonoBehaviour
{
    public int mapW = 10;
    public int mapH = 10;

    private bool firstPoint = true;
    private Vector2 clickPos;

    private Dictionary<string, GameObject> goDic;
    // Start is called before the first frame update
    void Start()
    {

        NewAStart.AStartManager.GetInstance().InitMap(mapW, mapH,15);
        StartCoroutine(CreatCube());
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            RaycastHit hit;
            var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            if (Physics.Raycast(ray, out hit))
            {
                var go = hit.collider.gameObject;
                var names = go.name.Split('_');
                var x = int.Parse(names[0]);
                var y = int.Parse(names[1]);
                var pos = new Vector2(x, y);
                print(pos);
                //如果是第一次点击，保存起来，第二次点击就是终点，进行寻路
                if (firstPoint)
                {
                    clickPos = pos;
                    firstPoint = false;
                    var goName = x + "_" + y;
                    var goPath = goDic[goName];
                    goPath.GetComponent<MeshRenderer>().material.SetColor("_Color", Color.blue);//设置物体的颜色
                }
                else
                {
                    Debug.Log(".....");
                    var pathlist = NewAStart.AStartManager.GetInstance().FindPath(clickPos, pos);
                    if (pathlist != null)
                    {
```

```
                for (int i = 0; i < pathlist.Count; i++)
                {
                    //通过名字来查找物体
                    var goName = pathlist[i].x + "_" + pathlist[i].y;
                    var goPath = goDic[goName];
                    Debug.Log(pathlist[i]);
                    Debug.Log(goPath.transform.position);
                    goPath.GetComponent<MeshRenderer>().material.SetColor("_Color", Color.green);//设置物体的颜色

                }

            }
            var goName2 = x + "_" + y;
            var goPath2 = goDic[goName2];
            goPath2.GetComponent<MeshRenderer>().material.SetColor("_Color", Color.blue);//设置物体的颜色
        }

    }
  }
  IEnumerator CreatCube()
  {
   var nodes=  NewAStart.AStartManager.GetInstance().mapNodes;
    goDic = new Dictionary<string, GameObject>(mapH * mapW);
    for (int i = 0; i < mapW; i++)
    {
      for (int j = 0; j < mapH; j++)
      {
        //创建立方体
        var go = GameObject.CreatePrimitive(PrimitiveType.Cube);
        go.transform.position = new Vector3(i + 0.1f * i, 0, j + 0.1f * j);
        go.name = i + "_" + j;
        goDic.Add(go.name, go);
        var node = nodes[i, j];
        if (node.type == NewAStart.EAStartNodeType.stop)
        {
          go.GetComponent<MeshRenderer>().material.SetColor("_Color", Color.red);
        }

        yield return null;
      }
    }
  }
}

```