A star 算法

1. 广度优先 (Breadth list)

   逐点打散

2. Dijkstra 算法

   Node 移动代价的优先队列

3. Best First

   Node 到终点的距离做为优先级 (贪婪. Path 不是最优)

4. A*

   Node 先搜索 优先级

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

F = G + H 每个 Node 都包括

H ↓ heuristic distance[current node][end node]

G ↓ distance[current node][start node]

F ↓ the total cost

待遍历 open_set 已 close_set

Step1: 初始化 open_set、close_set　　起点添加到列表

Step2: Openlist 找最小的 lowest F. 优先级最高. node_n

2-1　node_n 终点

从 node_n 回溯 parent Node → Path.

2-2　node_n 非终点

2-2-1　node_n 从 Open_set → Close_set

2-2-2　node_n 相邻 adjacent Nodes

A. node_adjacent 在 close_set 中:

Skip, 选取下一个 node_adjacent.

B. 不在 Open Set 中:

node_adjacent. { parent_node　node_n

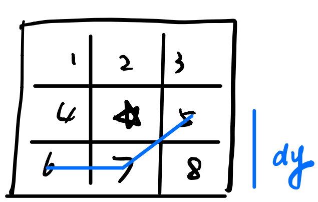
$F = G + H$

放入 Open_Set

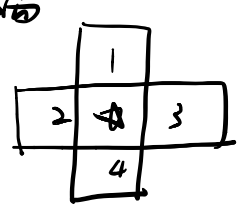
H　要比实际距离小, 能寻找到最短路.

理想的是相等

对角距离



曼哈顿距离



dx

dx = abs(Node.x - goal.x)
dy = abs (Node.y - goal.y)

dx = abs(Node.x - goal.x)
dy = abs (Node.y - goal.y)
Cost = α * (dx + dy)

$dx + dy + (\sqrt{2} - 2)\min(dx, dy)$

self. cost 的几个数.

A* map row, col 看成 Node 坐标

map function 判断是否 obstacles

open_list
close_list
Node 又 Node 邻接顶点

① parent_Node 从 open set → close_set

② adjacent_nodes

a、是否有效. b、是否 close_set

c、如果在 Open_set 中

adjacent_node . parent

p. cost : $f(n) = g(n) + h(n)$

放入 Open_set 中