A*（Dijkstra with the heuristic）：在迪杰斯特拉的基础上，引入了启发式函数，根据累计的代价 g，和启发式函数 h（The estimated least cost from node n to goal state）之和，选择容器中代价最小的节点。

代码工作流程

·维护一个用来存储将要拓展的节点优先级队列（能根据优先级自动排列的容器）。

·首先进行初始化，清空容器，并且将起点放入容器之中，将起点的 g（Xs）设为 0，其他未拓展的节点的 g（n）设为 infinite（无穷大）。

·Loop（当维护的容器为空时，退出循环，返回 false 表示没找到终点）

•从容器中拿出优先级最高(**f（n）=g（n）+h（n）**最小的节点)的节点，将其标记为已经走过的节点，放入 CloseSet 中，以后无法再走。

•判断拿出的节点是否到达终点时，如果是，退出循环，返回 true 表示找到节点。

•根据拿出的节点拓展其未走过的周围邻居节点（拓展过和走过不是一个概念，走过表示该结点已经在路径上了，拓展表示的是可能会走的节点）

•对于每一个拓展的节点进行遍历

•如果邻居节点未拓展过 g（m）为无穷大

g（m）=g（n）+Cost（nm）

将其放入容器中，并将拿出的节点设为父节点。

•如果邻居节点拓展过 g（m）>g(n)+Cost(nm)

g（m）=g（n）+Cost（nm）

进行更新，并将拿出的节点设为父节点。

•结束遍历

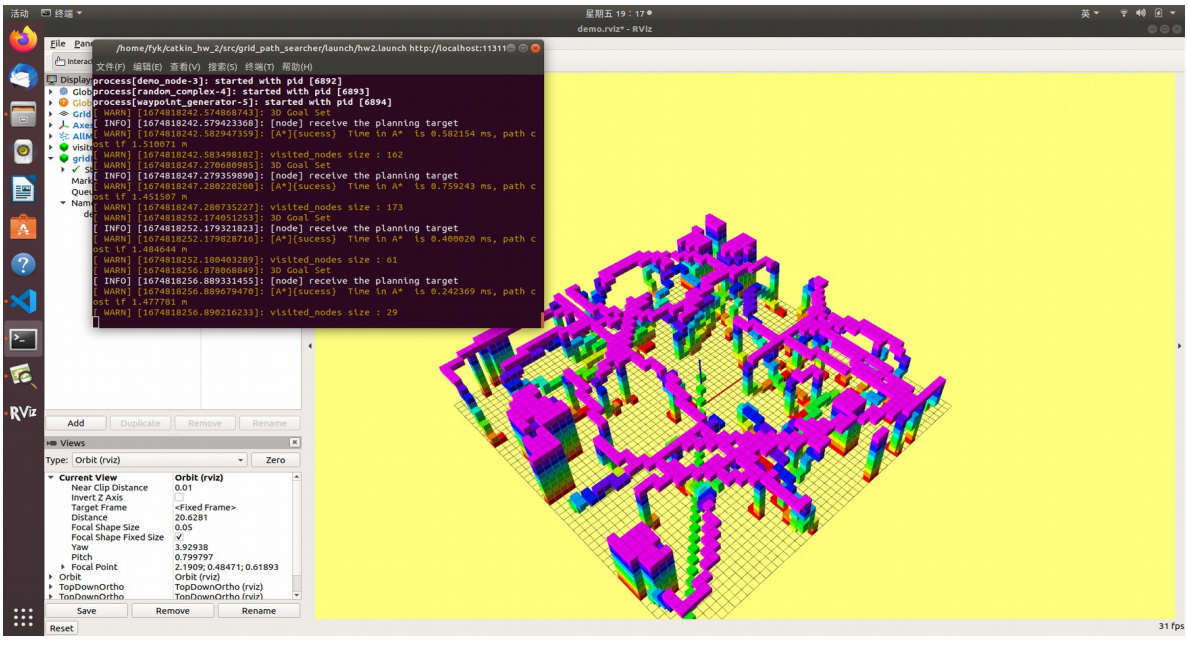
·End

最后进行节点回溯，找父节点，便可以找到路径

思考：为啥更新的不是 f（n）？

答：看课的时候突然冒出来这个问题，现在想来有点傻傻的，因为每个节点的 h（n）都是确定的，当更新 g（n）时，f（n）也就更新了。

运行结果：成功运行



实验条件：不同的启发式，相同的地图，选取了多个不同位置的点。

启发式函数	消耗时间（ms）	路径成本	访问节点个数
Diagonal Heuristic	0.4959	1.4810	106.25
Diagonal Heuristic（t）	0.3134	1.4751	58.25
Manhattan	0.1782	1.4864	27.25
Manhattan（t）	0.1890	1.5054	28.25
Euclidean	1.5109	1.4571	366.75
Euclidean（t）	1.1323	1.4571	290.50

备注：（t）表示加入 tie breaker 的情况
加入 tie breaker 后效率有所提升，但是曼哈顿作为启发式函数效果为什么会更好一些?

补充：JPS 算法与 A*算法的比较

算法	启发式函数	消耗时间（ms）	路径成本	访问节点个数
A*	Diagonal Heuristic（t）	1.1210	1.0682	1083.5
JPS	Diagonal Heuristic（t）	42.4197	5.6341	4425.5

Diagonal Heuristic 推导过程：

每一步的距离在三维的情况下只会有三种情况，根号三，根号二，一，当距离为根号三时，会往 xyz 三个方向分别移动一格，如果距离为根号二时，会往两个方向移动一格，如果距离为一，则会向一个方向移动一格。
我们假设有 dz 个根号三的距离，那么 x，y，z 方向上分别移动一格，z 方向上已经移动到了终点，而 x 方向上还可以移动 dx-dz 个格，y 方向上还可以移动 dy-dz 个格。此时可以发现 dz 必须是最小的。假设有 dx-dz 个根号二的距离，此时 xz 方向上都已经移动到了终点，只能向 y 方向上移动（dy-dz）-（dx-dz)=dy-dx 个格。此时可以发现 dy 必须大于 dx。
因此我们可以得到以下的距离公式
Diagonal=√3*dz+√2*（dx-dz）+（dy-dx）
因此我们可以得出规律以下规律，max 为 dx，dy，dz 中的最大值，mid 为 dx，dy，dz 中的中间值，min 为 dx，dy，dz 中的最小值。
Diagonal=√3*min+√2*(mid-min)+(max-mid)

学习过程中遇到的问题：

问题一：超出边界的问题?
在进行节点拓展时，在访问节点之前一定要进行是否越界判断 isFree()，否则会出现访问出界现象，否则在终点设置较远时，程序会因为访问越界崩溃。
问题二：何时进行设置 cameform?
最初**只是**在 AstarGetSucc（）函数中设置的，但是经过实验可以发现，这样做路径会出现问题，
路径中会出现很多没必要的节点。在 AstarGetSucc（）函数中设置 camefrom，周围的节点可能为拓展过的，也可能是为拓展过的，如果为拓展过的，没有进行 cost 比较便 camefrom，会出现问题。
我们需要在遍历周围节点时进行 camefrom 的设置与更新。

・对于每一个拓展的节点进行遍历

•如果邻居节点未拓展过 g（m）为无穷大

g（m）=g（n）+Cost（nm）

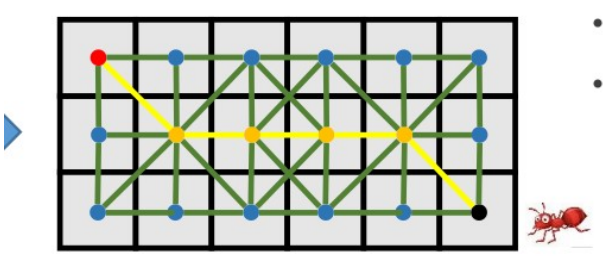
将其放入容器中，并将拿出的节点设为父节点。

•如果邻居节点拓展过 g（m）>g(n)+Cost(nm)

g（m）=g（n）+Cost（nm）

进行更新，并将拿出的节点设为父节点。

•结束遍历



- You don’ t need to search the path.
- It has the closed-form solution!

dx=abs(node.x −goal.x)
dy=abs(node.y −goal.y)
h=(dx+dy)+(√2−2)∗min(dx,dy)

当 dx 最小时

h=√2*dx+dy-dx

当 dy 最小时

h=√2*dy+dx-dy