人工智能导论第一次大作业报告

目录

一、数学建模

1、基本模型

本次大作业我选择了第二题，兔子吃胡萝卜。

我们可以将兔子与胡萝卜所在的密闭空间建模为网格，每个网格上可以放置障碍物，也可以是空着的。此外，网格中有两个格子分别为兔子及胡萝卜，且有一定距离。这样，空间中的每个元素都有且只有一个坐标与之对应，通过这个一一映射便可以把原问题转化为数学上可解的问题。

进一步对元素的属性或行为进行建模，规定胡萝卜不可移动，且兔子可以获取其坐标。兔子的初始位置坐标亦是已知的，兔子每次只能移动一步，每一步限制在上下左右四个方向，且不可通过障碍物。

最终确定问题的目标：兔子通过移动到达胡萝卜的位置，且移动次数尽可能少，也就是把路径代价定义为移动步数。这依赖于搜索算法的实现。

至此，我们完成了对这个问题基本模型的建模。

2、模型拓展

在基本模型的基础上可以做以下拓展：

① 可以将胡萝卜的个数变为多个，此时需要用一个向量来描述它们的位置，问题的目标也变成了尽可能多找到胡萝卜（可能有胡萝卜被障碍物及地图边界环绕，无法到达），且移动次数尽可能少。当然，搜索算法也需要改进（见搜索算法设计部分）。

② 可以人为地设置兔子、胡萝卜以及障碍物的位置，也可以改变地图大小、初始随机生成的障碍物的密度，使得该模型可以解决更广泛的问题。

③ 可以采用多种搜索算法，进而可以比较它们的效率。

二、算法实现

1、搜索算法设计

1.1 基础算法——A*

本次大作业搜索的基础算法使用经典的 A*算法。

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

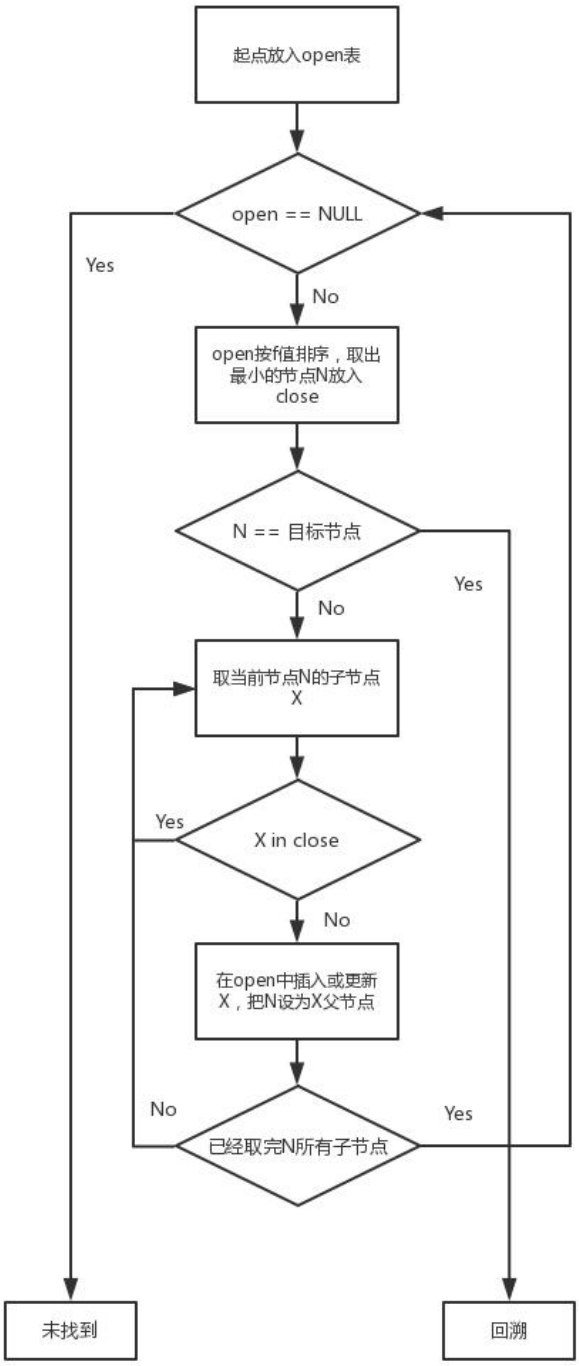
其中，f 为第 n 步时所需要的总代价，g 为实际经过的路径代价，启发函数 h 采用兔子和胡萝卜所在位置的曼哈顿距离，由于地图中有障碍物阻挡，且兔子不可以翻越障碍物、只能每次向上下左右移动一格，所以兔子到胡萝卜的真实代价一定大于等于曼哈顿距离，即

$$h(n)\le h^*(n)$$

所以该启发函数是可采纳的，故树搜索 A*算法是最优的。

该算法需要 open、close 两个表来存储节点，而且要对每个节点的代价进行排序，在程序中我使用了 QVector 实现，并根据节点的数据结构重写了 qSort 的比较算法。又由于 QVector 中的 contains 方法不可用于自定义的数据结构，我使用了两个 QVector<int>来存储 open 与 close 中节点的序号，从而通过查询这两个表就实现了 contains 方法。

算法实现的程序框图如下：



1.2 一致代价搜索与贪婪搜索

在

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

中，令 g 为 0，即得贪婪搜索；令 h 等于 0，即得一致代价搜索。

在 1.1 的基础上，在搜索函数中加入一个选择参数 type，来决定 f 的计算方法，这样，便把之前的一种算法拓展为三种算法，从而可以进一步比较三种算法的优劣。

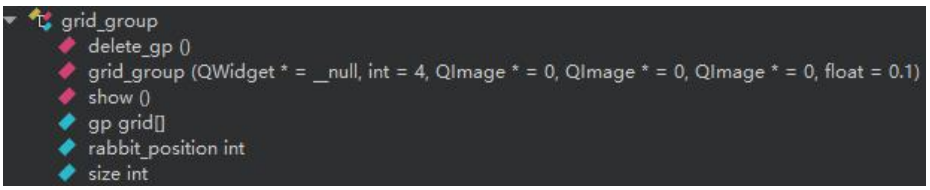
1.3 多个胡萝卜的搜索

与 Tower of Hanoi 问题类似，通过问题归约，将这个问题转化为本原问题，即兔子用最少的路径代价先找到其中一个胡萝卜。在找到一个胡萝卜后，问题与原问题相同，进行类似的问题分解，继续使用一样的方法直到找完最后一个。最后将这些子问题的结果进行合并得到原问题的解。

对本原问题的求解：在原算法的基础上，在计算启发函数时，将当前节点到所有胡萝卜的 h 都进行计算，并取最小值来计算 f，这样，启发函数是可采纳的，而且兔子每一步都走向当前距离它最近的胡萝卜。

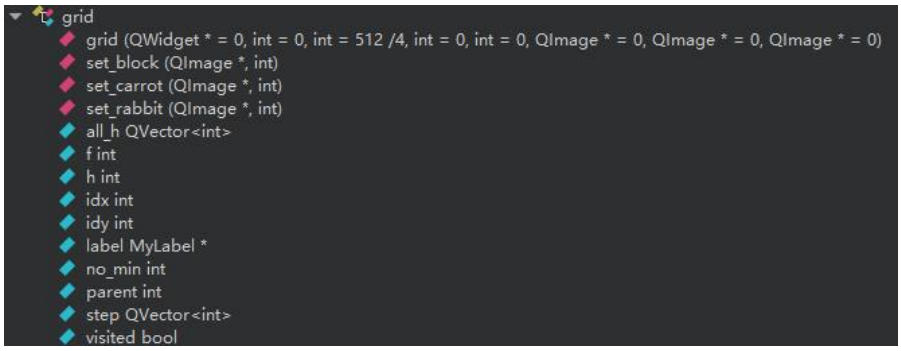
2、类设计

2.1 grid_group 类



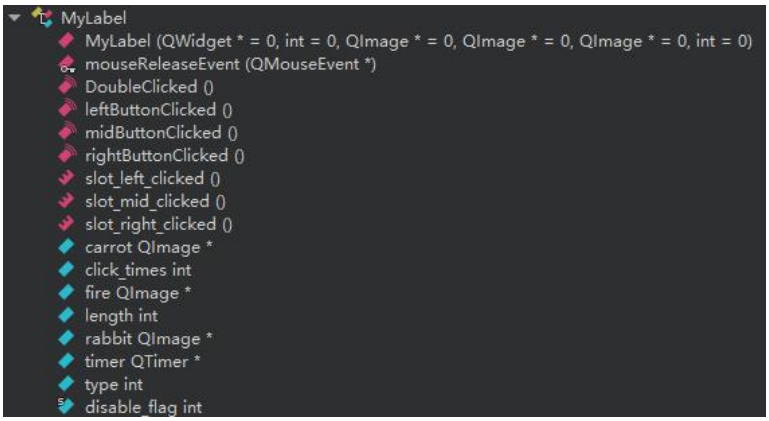
这个类对应着整个地图，用于动态地根据用户输入的地图大小及障碍物密度生成所需要的网格以及构造其中每一格相应的数据结构 grid，同时还具有所有格子的 delete 与 show 的方法，便于从宏观上对整个地图进行管理并获取相应信息。

2.2 grid 类



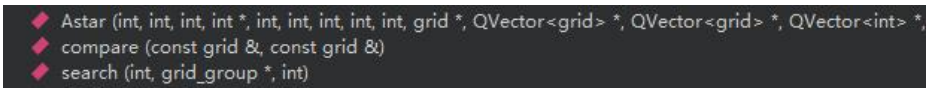
这个类对应地图中的某一格，存储着这一格在 ui 中的相关信息以及搜索算法中所需要的 f、h、parent 等节点信息，集成了格子层面所需要的所有方法。

2.3 mylabel 类



这个类继承自 QLabel 类，其对象的指针存于每个 grid 中，用于对来自 ui 的各种操作进行响应。此外，借助该类中的 type 值对兔子、胡萝卜、障碍物、可通过区域进行区分。

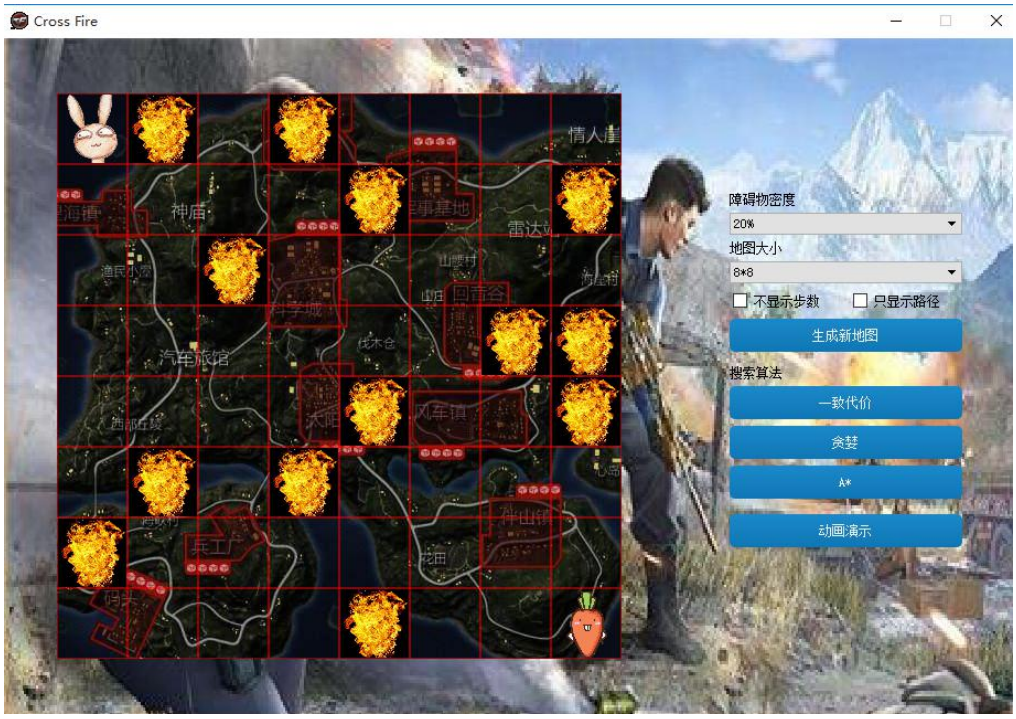
2.4 algorithm.h



这个头文件中封装了三个函数，从上到下依次为 A*单步搜索函数（可以拓展到其他算法）、重写的代价比较函数、全局搜索函数。通过封装可以更好地调用这些函数，也提高了代码可拓展性和可读性。

三、UI 设计

1、界面组成



界面左侧为地图，右侧为参数及算法选择区，默认初始化为大小 4*4，障碍物密度 10%的地图。

在选择好地图大小以及障碍物密度参数后，点击“生成新地图”按钮可以生成新地图，默认兔子位置在左上角，胡萝卜在右下角且只有一个。在地图的格子上，单击左键放置兔子，单击右键放置胡萝卜，单击滚轮放置障碍物，双击左键变为空地。

调整好地图后，点击某一种算法按钮，进行搜索。搜索完成后弹出窗口，显示最短路径步数、累计拓展节点数、找到胡萝卜个数。如果有部分胡萝卜被围在障碍物中，也会弹出相应提示，并显示搜索完所有节点最多可以找到的胡萝卜个数。据此可以对比不同算法的搜索效率。

同时，地图中用蓝色方块及数字表示出最短路径，用白色方块表示尚且在 open 表中的节点，用绿色方块表示在 close 表中的节点。如果遇到同一个方格被多次途径的情况，格子上的数字会依次显示相应的步数，用空格隔开。如果想只显示路径或者不想显示步数，可以勾选相应选项实现。

在生成最短路径后，可以点击动画演示按钮，动态展示兔子找到所有萝卜的过程。

详细的软件建议测试流程及注意事项见 readme 文件。

此外，为了 UI 的美观性，在界面中加载了 CSS 文件，对按钮、复选框等元素进行了样式设置，整体呈蓝色调，与窗口背景图相适配；对界面中的图片素材以及方块涂色时做了半透明的处理，界面更加美观。

2、交互设计

为了提升程序的交互性，在 mylabel 类中重写了各种鼠标响应，特别是其中的左键单击响应与双击响应的区分：接收到鼠标单击事件后定时器 QTimer 开始倒计时，待超时后检测点击次数，实现了对二者的区分。有了各种鼠标响应，用户可以随心所欲地设计想要的地图以及兔子和胡萝卜的位置。

动画演示功能也使用户可以更加清晰地看到兔子沿最短路径行进的情况，直观生动，用户体验有很大的提高。

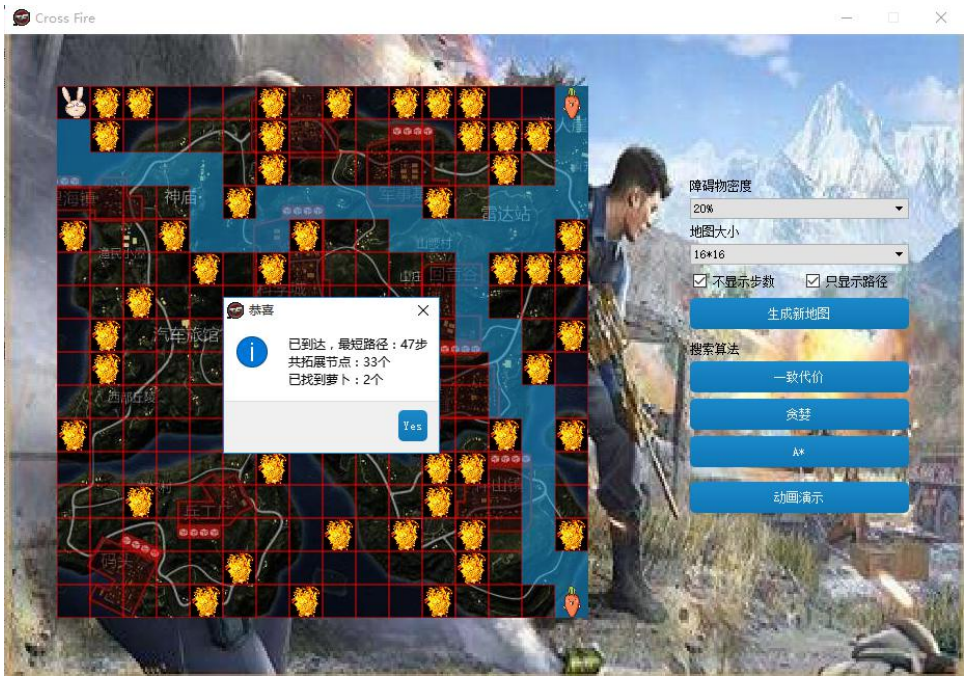
另外，对于用户的一些误操作也会进行相应提醒，如：放置多个兔子、删除了兔子或者所有胡萝卜、没有搜索就进行动画演示等，增强了程序的鲁棒性，也给出了正确操作的提示，交互性良好。

3、运行演示

如图，可以完成对多个胡萝卜的搜索，显示最短路径以及拓展的节点。

当有胡萝卜搜索不到时（地图右下角），会有相应提示，同时会给出可以搜索到的胡萝卜的个数以及对应的最短路径。

改变地图大小和障碍物密度，勾选搜索显示设置选项，生成新地图并分别使用三种搜索算法得到的结果如下：



A*搜索

贪婪搜索

一致代价搜索

经对比可以发现，有信息搜索比无信息搜索的效率有了大大的提高，而在有信息搜索中，A*的效果还可以比贪婪算法还可以更好一些，但是也会付出拓展更多节点的代价。因而虽然这三种算法的时间效率和空间效率都是指数量级 $O(b^m)$，但实际实现效果的差异还是很大。由于时间有限没有完成不同启发函数间的比较，但可以预见，对于有信息搜索，启发函数的好坏对算法的效率也会有很大的影响。

四、实验总结

1、问题与解决

除了以上内容中提及的问题及解决方法外，其他罗列如下：

① 在程序开发初期，搜索的过程直接在按钮点击的响应事件中实现，而且每次开拓子节点时上下左右四个节点的代码几乎完全一致，大量近似代码的重复使得 MainWindow.cpp 文件一度达到了上千行。在我把搜索函数抽象为上下左右四个节点都可调用甚至不同的搜索算法都可以调用的通用函数后，写入专门的 algorithm 头文件中，大大精简了代码行数，MainWindow.cpp 缩短到 100 行左右，且增加了代码的可读性和可拓展性。这也提醒我以后在做类似开发时，提前就把多次调用的函数单独写出来，并且设计好通用的接口以便调用，这样，也有助于程序的调试。

② 由于 QVector 中的 contains 方法不支持自定义的数据结构，我曾经尝试使用继承 QVector 重新定义一个 MyVector 并重写 contains 方法，但发现根本问题在于 QVector 的拷贝构造函数是私有的，禁止从外部调用。虽然很理解这是为了数据结构的安全性，但还是不得已采用了不那么优雅的方法来解决，多开新的空间存储 open 表和 close 表中的序号，从而可以直接使用 contains 方法进行查询。当然我也相信应该会有更好的解决方法，之后也会继续思考这个问题。

2、心得体会

通过亲手编程实现搜索算法无疑是对课上所学的理论知识最好的回顾，用自己实现的模型对不同算法的效果进行对比，直观且生动，很有成就感。

同时，这也是一次锻炼界面开发的好机会，在不断的测试、修改的过程中，我也一直在思考如何提高交互性，让用户体验良好，相信这对于我以后类似项目的开发也会大有裨益。

当然，限于时间，没能把自己在开发初期的所有设想都实现出来，例如：采用并比较不同的启发函数、设置不同类型的障碍或者奖励，可以进行积分并升级兔子（如可以斜着走）、设置“虫洞”或“迷雾”，引入随机性、采用新的迷宫生成算法而不是随机放置障碍物从而确保连通性，等等。这也是这次大作业最大的遗憾了。

五、参考文献

[1] 维基百科 A*算法 https://en.wikipedia.org/wiki/A*_search_algorithm

[2] Qt 基础教程之 Qt 学习之路
https://blog.csdn.net/mars_xiaolei/article/details/79424315