问题求解（二）项目第三阶段

付博
211098200

2022 年 7 月 3 日

1 游戏实现

1.1 具体实现

在第二阶段已经完成图形化界面游戏的基础上，对游戏内容与设定进行了进一步优化。

其中最主要的工作在于对机器人算法的设计，使得机器人最终拥有一定的智能操作，其余附带性质的工作还有诸如对炸弹爆炸光波动态效果的设计、当前人物状态显示的窗口设计以及游戏背景音乐的添加。

以下是对机器人策略算法详情的描述：

首先从人的角度思考取胜策略，分析决定当前移动方向的依据无外乎地图当前状况以及移动后所期望达到的效果；类似地分析炸弹策略的依据也就无外乎放下炸弹后所能达到的效果。为了能在连续移动中达到离散考虑的最佳化采取了间隔设计，即让机器人只有正好在格点时才分析下一步策略，这依靠将格点时的移动方向 *Move_dir* 设置为方向枚举类中特殊的 *center*[1] 来实现，也因此可以确保我们每次更改策略时，都是正好处于格点的位置。

在保证了离散化判断后，具体分策略而言：首先对于机器人的移动，我们分析机器人移动所考虑的“需求”并将其总结为四种，分别为去到安全区域以摆脱危险/靠近其他玩家以放炸弹攻击/获取道具/靠近软墙以炸破软墙获得分数，接下来便用到个人认为很新颖也很巧妙的方法，即根据作判断时地图状态分权重赋优先级值，具体做法为利用最简单的二进制表示，将需求按照紧迫度排序为安全[2]/获取道具/靠近其他玩家/靠近软墙并对符合要求的位置分别加上权重 $2^3 = 8$，$2^2 = 4$，$2^1 = 2$，$2^0 = 1$，之后便可直接根据总权值大小来判断此位置的优劣，省去了布尔逻辑的复杂判断，这一点是我认为设计优秀的一点。之后便是老生常谈的设计出 *DFS* 算法，并采用边搜索边记忆到达当前位置所用最小步数花费，然后用一个引用变量在函数内部记录当前的最大权值、相应最小花费以及对应的起始运动方向，只有在遇到权值大或者权值相等但是花费步数小的情况才更新以上三个值，最后搜索完所有可行步段后结果就是最优情况；其次对于放炸弹便是最简单的布尔逻辑判断，机器人的最大需求是安全，所以我们考虑如果放下炸弹不能再找到可行安全区域，就不放炸弹，在这里由于以上搜索的可复用性，直接假设已经放下炸弹，然后搜索一遍最优路径，如果搜到的最大权值小于安全情况下的权值[3]，那么就不放炸弹，之后便

[1]这个方向代表目前为停滞“待机”状态

[2]这里具体的实现时采取了危险区域（即炸弹的爆炸区域）减 8 的做法，效果一样但是时间复杂度更加优秀。

[3]在这里由于采取了减 8 的策略，直接与 0 相比即可

对能否炸到软墙/能否炸到人[4]进行判断，如果可以就放下炸弹，其余情况均不放。

在以上算法的指导下，实现过后还是得到了很不错的效果[5]。

1.2 实现过程

首先完成了前置的离散化需求工作，然后在此基础上，着重分析了算法核心，并逐渐成型，最后实现起来还是比较轻松的。

对旧代码的改进方面，我认为这个确实是需要时间积累来进行的，有一些功能类似的代码段最好复用，会很大程度地精简原代码，这一阶段已经改了我认为最丑的道具状态显示部分，其他的慢慢来了。

但是仍然遇到了一些困难包括但不限于 *Qt* 扩展库的使用细则[6]，具体体现在背景音乐的添加等需要很繁琐的查询；内存管理异常困难，游戏会陷入一些离谱的 *crash* 中，具体原因不明。目前打算以后还会继续磨练精简这个项目。

2 收获

2.1 知识

2.1.1 代码精简训练

2.1.2 特定寻路算法设计实践

2.1.3 *Qt* 扩展库的安装使用方法

2.2 心得-总结

在第二阶段的较好基础上，这一阶段的实现还是比较容易的，只要实现一个较好的寻路算法以及相应的炸弹判断即可，附加的还有对原代码的改进升级。在算法方面我的体会是前期的设计与预想很重要，这会很大程度上减少一改再改的繁琐，还有尽可能思考复用问题；代码改进方面体会是数组的方便快捷性还有枚举类的广泛应用带来的好处，提前设计好一个所需的常数（宏定义）文件真的会带来极大的方便，无论是在需要更改异或添加时。

3 鸣谢

排名不分前后：

感谢助教们的悉心解答

感谢Internet里的解答分享

感谢我自己的坚持

[4]在这里为了能尽量炸到人，将判断为能炸到人的区域放大了一些

[5]人类玩家玩不过它们 ()

[6]*Search* 后发现可用资料太少了，甚至用的 *Qt* 新版都没有*.pro* 文件