问题求解（二）项目-第三阶段报告

211240088 徐沐杰

补足第二阶段的完成情况

图形按钮

利用 PPT 艺术字完成了图形按钮，自建 GraphicButtons 类（继承于 QGraphicsObject）创建能借助鼠标事件发出被点击信号的图片按钮。

地图大小

借助第一阶段的 mapEditor 和第二阶段实现的地图转换器，成功将地图换版，并修改为 15*20 之大小。

计分板和帮助

计分板已用 QGraphicsProxyWidget 嵌入 QTextBrowser 实现，帮助则用了一个 QMessageBox 窗口简单实现。

存读档

本项目基于 JSON 的地图架构为实现存读档功能提供了便利，基于存读档功能，本项目实现了默认路径的快速存读档功能。

帧率显示

现在在状态栏可以显示帧率（fps）了。

游戏结束

只剩下一个或没有玩家存活时会出现游戏结束提示，显示获胜的玩家（或者无人获胜），并自动退回主菜单。

未完成的自建框架功能

这些功能的由来见第二阶段报告，但出于各种原因，这些功能没有完成，仅记录于此。

- 远程服务器
- 设置和高级选项（包括输入键值、偏好设置的修改等）

机器人的实现情况

- 机器人利用 `Ai` 类实现，当服务器发现一个机器人没有对应的 `Ai` 实例时，便会为其创建一个。
- `Ai` 是一个状态自动机，它被设置有如下状态 `MODE` 来让其在复杂的地图和地形中存活下来并试图击杀其它玩家：
    - 逃离 `FLEE` ，这个状态下 `Ai` 会往设定的目标逃离，并忽略其它一切可能的危险。

    - 攻击 ATK，这个状态下 Ai 会奔向一个选定的玩家，并试图在它身边放下炸弹。
    - 拾取 EAT，这个状态下 Ai 会奔向一个道具。
    - 开辟 OPEN，这个状态下 Ai 会奔向一个选定的软墙，并试图在它旁边放下炸弹。
    - 无 NONE，这个状态下 Ai 会试图进入其它状态。

  目标地点或目标玩家存储于 target 或 targetPlayerNum 中。

- 每一帧，服务器线程都会调用尚活着的机器人的 AI 实例，并将地图送入，要求其做出决策（Decision 函数），该函数首先对地图从自己开始进行广度优先搜索，算出地图某点是否可达以及通往其的路径的前一步，即 map 和 pre。

- 进入和退出各个状态的规则如下：
    - 无论处于什么状态（除了 FLEE），当 Ai 发现自己可能处于炸弹的威胁之中时，便会尝试进入逃离状态，它会优先选择于炸弹来向垂直的方向（就像躲避山洪），实在不行就选择反方向。
    - 在 NONE 状态下，Ai 会扫描地图，寻找一个可达的玩家进行攻击，进入 ATK 状态；如果没有可达的玩家，它就寻找可达的道具进入拾取状态；如果还没有，它就选择一个可达的软墙进入开辟状态。
    - 在拾取、开辟、攻击状态的目标达成（吃到道具或放下炸弹）或变得不可达（比如被其它玩家挡住）以后，Ai 会进入 NONE 状态，并立即（而不是等到下一帧）重新试图进入其它状态。

机器人的实现在 ai.h 与 ai.cpp 中。

心得体会

- Ai 类的抽象没有建好，导致出现了大量很难或根本无法修复的漏洞。
  所有状态转移全部被塞入 Desicion 函数，以至于状态的转移逻辑变得非常复杂，以及大量可以写成方法的地图判定逻辑也没有被封入函数，导致 Decision 变得非常臃肿，代码复用性低，以及容易出现漏洞。
  另外，在抽象层次上，Ai 类应当提供一个专门处理分析移动细节的中间层（主要是为了和自建框架的移动逻辑交互），而不是把这部分也揉到决策状态机中，令代码变得令人费解且无法实现预期功能，最后我甚至让机器人作弊修改地图（对自己运动状态的微调）才能实现部分功能（比如转弯）。
- 游戏核心逻辑和机器人的协同程度低，缺乏很多接口。
  这部分问题和上面的差不多，主要还是顶层设计缺乏一个机器人和核心逻辑用合乎游戏规则的规则交互的「中间层」（比如知道了 pos 和 posTo，负责操作如何移动到一个和这两个方格四联通相邻的方格），上层类应该只需要决定 movingTo，而不需要具体做出摁下哪个键的决定，否则将会使得逻辑很费解。