Qt大作业报告

组长：何子默

组员：李梓铭、徐奕辰（以上排名不分先后）

**1, 项目简介**

本项目基于王pro同学发明的桌面游戏, 使用面向对象的方法开发了Glory of pro游戏界面以及游戏内核程序, 使得程序能够在pc端正常运行, 并且带有精美的游戏画面. 游戏方式简便快捷、轻松益智、老少咸宜，是消磨时间、放松身心的不二选择。GOP玩法简单，进行一局游戏只需要几分钟，在当今快节奏、碎片化的时代中，该游戏有着广泛的市场和应用价值。为了更好的适应当今时代人们的游戏习惯, 我们增加了游戏的随机性, 设计了存档读档功能, 并且改动游戏规则使得2-4个人都可以愉快玩耍. 游戏规则介绍见附录1.

在游戏设计过程中, 我们先单纯用c++实现了游戏(2), 使之在终端可以运行. 然后根据c++游戏的输入输出逻辑, 把各个部分换成Qt中的输入输出, 从而完成了Qt版本(3). 此后, 根据Qt版本, 我们进行了一些测试和debug, 使得程序在面对各种情况时都能够正常反馈(4).最后, 我们将简要介绍我们的分工和总结感受

**2, 项目具体实现(基础c++部分)**

这一部分中, 我们将介绍如何使用c++搭建游戏运行的基本框架以及最终终端运行版本. 代码在gloryofpro\_shell文件夹中.

**2.1 程序基本架构**

程序的输入输出控制基本在main.cpp中. 在play函数中, 我们会进行读档/存档, 并根据存档情况新开始或者继续进行某次游戏. 在游戏没有结束时, play函数在每一轮游戏时会调用cin\_action函数.

Cin\_action函数中, 首先判定每个玩家是否有行动资格, 然后让每个有行动资格的玩家依次输入行动. 行动包含“move”(移动), “mine”(挖金币), “buy”(买东西), “attack”(攻击), “recover”(恢复), “gohome”(使用回城卷轴回家), “save”(存盘), “quit”(退出游戏). 根据输入不同, 玩家会进行不同的操作, 例如buy以后会要求玩家输入购买的物品和等级等, 此处不再赘述. 操作结束后, 终端会输出每个玩家的点数以及玩家操作的结果. 在每轮开始时, 终端还会更新玩家的状态, 包括血量, 武器, 金币等数据. 在每一轮中. 玩家都有存档和退出的机会. 同时, 我们在cin\_judge 中加入了一些判断条件, 以防恶意输入.

在一轮游戏结束后, judge\_end函数判断游戏是否结束. 如果游戏结束, 则会提示玩家是否继续进行游戏, 或者退出游戏. Cin\_judge和judge\_end函数都封装在board类中

**2.2 board类和player类**

为了实现游戏中的每种操作, 我们在big\_class.hpp文件中设计了board 类和player类. 包含函数如下:

* board:: board() 根据不同人数进行初始化的构造函数. Board类包含了player, bigshop, smallshop, home, mine等类的对象, 因此在board::board()函数中都进行初始化
* board:: print() 在终端输出地图
* board:: readboard() 读取存档, 利用了fstream类的输入流fin
* board:: saveboard() 保存存档, 利用了fstream类的输出流fout
* board:: cin\_judge() 前文已叙述
* board:: judge\_end() 前文已叙述
* player :: player() player类的构造函数, 包含了玩家各种信息. 在player类中,我们使用了STL中的vector动态存储player的weapon和magic, 使得我们可以随机访问某个玩家的所有武器和魔法, 并且可以很方便的添加/删除物品
* player :: print\_status() 输出玩家的位置, 血量, 金币, 武器, 魔法等信息
* player :: checkLevel() 给定装备的种类, 找这个装备的最大等级. 函数中利用了vector的随机访问迭代器, 能够很好对应武器种类和等级.
* player :: check\_tp\_num() 类似于checkLevel函数, 查找tp的数量
* player :: buy() 玩家购买物品后, 调整玩家装备以及玩家金币数量. 由于bigshop和smallshop物品价格不同, 还需要根据商店判断价格..
* player :: attackjudge() playerA攻击playerB时, 需要投两次点, 判断A能否成功攻击B.
* player :: attack() 攻击函数, 需要根据投点和武器类别造成血量变化. 我们通过weapon类的虚函数, 在attack函数中使用多态来减少代码量, 达到函数复用的目的. 在这个攻击函数中, 玩家需要制定武器类型. 在qt版本实现中, 我们加入了给定目标自动选择最优武器的程序, 减少操作复杂度.
* player :: dis() 计算两个player之间的距离.

**2.3 其他类**

在这些函数之外, 我们将涉及到每个建筑和每个装备的函数进行了封装, 分别封装在building, magic, weapon三个基类中, 把随机过程放在了Dice类中.

**2.3.1 buiding类**

Building类为了使用方便, 仍然放在了big\_class.hpp中. buidling类派生出来了shop类, home类, mine类, 分别对应这不同的功能. 对于home类和mine类, 我们这里只把它们当成了封装数据的结构体, 并没有定义函数. 这样单纯的减少了调用数据的难度, 使得代码更加美观. 对于shop类, shop类又作为基类派生出了两个子类: bigshop类和smallshop类. 这是因为在bigshop中和smallshop中, 物品的价格不一样. 因此在这里又一次使用了多态, 通过商店的不同展示不同的商品价格, 减少代码量.

**2.3.2 magic类.**

Magic类派生出了wing, blood, tp类, 分别对应三种magic装备. 在这个程序中, 我们将magic类作为结构体储存数据, 没有定义函数. 用结构体储存的好处是便于添加新的magic

**2.3.3 weapon类**

Weapon类派生出了 fist, sword, gun类, 分别含有一个虚函数attack, 根据武器属性对被攻击者的血量造成影响. 用结构体储存的好处是便于添加新的weapon.

**2.3.4 Dice类**

这个类封装了一个随机掷骰子的函数, 可以选择一次投多少个骰子, 根据c++随机数返回骰子点数. 随机性通过c++中的随机数实现, 随机数种子在main.cpp中的main函数里面, 通过srand((int)time(0));保证种子的随机性

**2.4其他实现细节**

1,为了通过武器名称查找价格和范围, 我们使用了STL中的map容器, 以string类作为key, 实现了对于武器种类和范围的访问. map在board构造函数中进行初始化.

2, 为了输出商店价格, 我们使用了string类作为输出. String类的好处是重载了加法, 使得我们可以非常便捷的组织终端中的输出, 在最后只需要一次cout输出.换行操作只需要 +‘\n’即可. 这样减少了很多不必要的代码量, 详见big\_class.hpp中的showprice函数.

**3, Qt实现**

以下界面类均继承于QWidget类，用于可视化实现本项目中的各个界面。其中使用了UI设计界面对界面进行布局。

**3.1 mainWidget类**

mainWidget类代表游戏的开始界面，其中定义了4个QPushButton类对象，分别用于“开始新游戏”、“读取存档”、“打开规则说明”与“退出游戏”，以及1个QSpinBox类对象，用于选取玩家人数。

当主程序开始运行时，首先会定义出一个mainWidget类的对象，并调用该对象的show()函数，打开游戏开始界面。

1、“开始游戏”

打开GameWindow界面并根据QSpinBox的值初始化一个board对象。

2、“读取存档”

打开GameWindow界面并读取board.txt文件中储存的值更改board对象。

3、“规则说明”

打开一个规则说明文档。

4、“退出游戏”

退出程序。

**3.2 Gamewindow类**

Gamewindow类代表进行游戏的界面，当在开始界面中点击了“开始新游戏”或“读取存档”时，会打开GameWindow类的对象。本项目近一半的工作量都用于实现Gamewindow类各功能。

GameWindow类的界面布局主要由UI设计界面实现。我们在界面上方显示4名玩家的属性信息，在左下角450x450区域显示游戏地图，在右侧定义了一个QTextBrowser对象用于记录对局信息，在右下角的玩家操作台中定义了8个QPushButton对象用于实现“移动”、“攻击”、“挖矿”、“治疗”、“购物”、“回城”的游戏操作以及“存档”和“返回主界面”的功能.

1.玩家信息区域

每个玩家都有8个QLabel对象用于显示其当前各属性，我们定义了成员函数updatePlayerInfo()用于在玩家每一次操作后对所有玩家的信息进行实时更新。玩家信息均储存在board类中定义的player类对象中。

2.游戏地图

我们设计了游戏的背景地图，并为每位玩家定义了Qlabel对象，将其设为人物图片。当玩家进行移动操作时，通过调用move()函数，其人物图片会在地图上移到相应位置。我们用QStackedWidget在游戏地图区域设置了人物图片层、移动按钮层和商店层。

3.对局信息

在QtextBrowser对象中输出玩家已经进行的操作描述。

4.操作台

i. “移动”

打开QStackedWidget的移动按钮层，当前玩家可以点击相应按钮进行移动，同时更改player的pos。

ii. “攻击”

打开攻击界面AttackWidget，其中包含4个按钮用于选择攻击目标。点击其中按钮后，会调用当前player的attack()函数进行攻击。

iii. “挖矿”

先判断玩家是否位于金矿上，若是，当前player的money+=1。

iv. “治疗”

先判断玩家是否位于自己家以及玩家是否未满血，若皆是，当前player的hp+=1。

v. “购物”

打开QStackedWidget的商店层，其中包含购买sword, gun, wing, tp和hp的按钮以及离开按钮。点击购买时会调用player类的buy()函数。

vi. “回城”

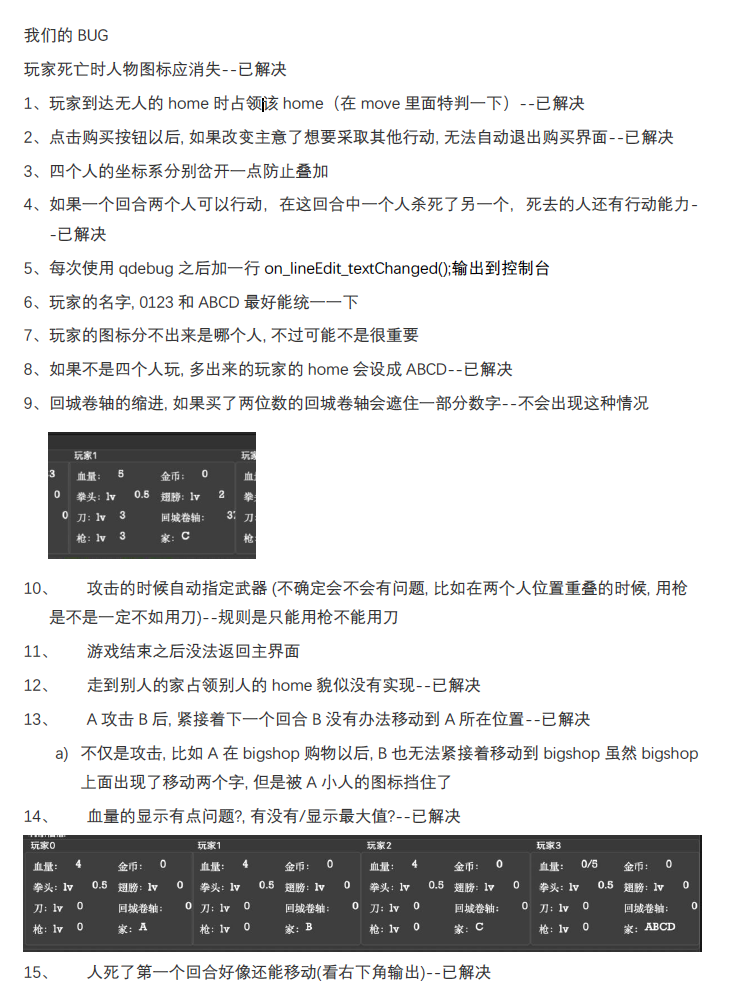
先调用player类的check\_tp\_num()判断当前玩家是否有回城卷轴，若是，打开回城界面TPWidget，其中包含4个按钮用于选择回城地点。点击其中按钮后，会消耗回城卷轴并更改玩家位置。

vii. “存档”

调用bd类的saveboard()函数，将当前所有玩家信息写入board.txt文件中。

**4, 测试及debug**

在代码从命令行版本迁移至Qt时, 需要改动很多输入输出结果, 加上我们对于Qt的不熟悉, 容易出现bug.为了减少玩家在玩耍时的不愉快体验, 我们专门进行了极端情况模拟来发现Qt版本的问题. 例如, 玩家购买了很多回城卷轴, 玩家来回反复移动等等.经过这些模拟, 我们发现了操作界面, 游戏图标, 等共计15个问题.



通过解决这些bug, 我们的Qt程序页更加成熟稳定, 能够给玩家带来更好的使用体验.

**5, 人员分工**

何子默:负责实现Qt版本中回城, 治疗, 挖矿, 攻击，更新玩家信息，存档读档. 普通c++版游戏中weapon, magic, dice类的初步代码, 以及buy(购物)和attack(攻击)的初步代码. 最终报告的Qt部分以及游戏简介.

徐奕辰: 负责在Qt版本中实现移动（包括人物图标），商店界面，输出对局信息，地图，界面美化. 最终报告的Qt部分.

李梓铭: 负责普通c++版游戏的输入输出逻辑,代码整体框架,以及各个操作. big\_class中board类和player类设计, 优化weapon类和player类中的attack函数, buy函数. 代码环境测试(普通版和Qt版), 撰写最终报告 .

**6, 总结和感受**

在学习了程序设计实习第一部分的内容(类和对象)后, 我们了解了一种新的编程方式: 面向对象. 面向对象的设计方式让代码的复用性强, 在维护和编写上都比面向过程的编程更有优势. 因此, 这个大作业《glory\_of\_pro》为我们提供了一个实战的机会, 让我们切身感受到了面向对象编程的好处和意义. 在这个项目的代码中, 我们几乎用到了所有类与对象讲述的知识. 包括类的成员函数和构造函数, 封闭类, 派生类和基类, 虚函数和多态, string类, 重载, fstream文件流输入输出, STL的容器和容器适配器, 迭代器等. 我们真正做到了活学活用, 把课堂的知识转化为实际的应用.

在项目开始时,我们首先研究了这个游戏的游戏规则. 处于趣味性的考虑, 我们增加了游戏的随机性. 通过不断的试玩这个游戏, 我们感受到了修改游戏的乐趣, 以及思考如何用代码实现游戏功能的快乐. 在写代码的过程中, 当我们使用了非常简洁的方式完成代码, 也让我们感到心旷神怡, 宠辱偕忘. 完成了c++版代码是我们这个项目的一个里程碑, 标志着我们真的用计算机模拟出来了一个游戏, 大家都非常激动. 但在迁移到Qt上的时候, 我们还是遇到了很多困难. Qt的很多自定义需要我们从头学起. 我们一点点照着网上的视频学习, 经过无数个不眠之夜, 自己动手完成了Qt所有类的编写和实现.

然而, 疫情的出现让我们的协作变得非常困难. 有时不能很好的沟通, 也造成了一些合作的麻烦. 为此, 我们使用了github组织代码, 让小组合作变得更加简单, 也解决了代码修改不同步的问题. 在期末考试后, 我们的成绩很不理想,因此我们更加努力完成这个Qt大作业, 希望弥补期末考试的遗憾. 最终我们实现了原定计划的所有功能. 在此过程中我们的代码除了输出对局以外, 都是一行一行手敲, 没有在别处复制粘贴, 实现了代码的基本完全自主.

通过这次大作业, 我们磨练心智, 锻炼编程能力, 更好的吸收了程序设计实习这门课教给我们的知识. 我们在未来的学习和工作中, 也一定会用到这门课中学习的类和对象以及算法设计的知识, 让郭老师, 刘老师和张老师教给我们的知识发光发热. 我们祝愿《程序设计实习》这门课越办越好!

**附录1: 游戏规则介绍**

1. **游戏地图（概念图）**

（1）家(Home)

（2）金矿(Mine)

（3）大商店(Big Shop)

（4）小商店(Small Shop)

（5）道路(Road)

1. **初始设定**

游戏开始时，每名玩家（2-4名）血量(hp)为4，血量上限为5，金币数(cash)为0，行动力(act)为1，初始武器为“拳头”，按照顺序依次选择家 A、B、C、D之一作为其出生地点（不能选择已经被选择过的家），自动占领出生点。完成初始设定后，进入行动轮。

1. **行动轮(move round)**

在每个行动轮开始时，每位玩家投掷一枚色子，点数最大的玩家（可以为多名）获得行动资格，行动力提升至当前上限。

获得行动资格的玩家按照出生地顺序依次选择进行以下行动中的一项：

1. 移动(walk)最多为行动力的距离（玩家仅能停留在特殊建筑所在位置）；
2. 进行玩家所在位置的特殊建筑的可选操作（例如，玩家处在大商店的位置，则可在大商店中购买(buy)武器）；
3. 攻击(attack)其它玩家，进入战斗轮。

进行任一行动后，玩家失去行动资格，当所有玩家均失去行动资格后，进入下一个行动轮。

1. **特殊建筑(buildings)**

地图中的每个位置都有特殊建筑，分别为家(Home)、金矿(Mine)、大商店(Big Shop)、小商店(Small Shop)，它们有着不同的功能。

1. 家(Home)：当玩家到达一个未被占领或占领者不在内部的Home时，可选择立即占领该Home；玩家在一个已被自己占领的Home中，若拥有行动资格，可选择回复1点血量;
2. 金矿(Mine)：玩家位于金矿且拥有行动资格时，可以选择获得1金币。
3. 商店(Shop)：玩家位于大（小）商店且拥有行动资格时，可以选择购买装备。小商店的所有装备均比大商店中的同种装备贵1金币。
4. **移动**
   1. 玩家初始移动距离最长为一个单位, 也就是地图上一段道路的长度.
   2. 在获得了传送之翼后, 玩家可以不按照道路飞行, 而是根据传送之翼行动范围的大小, 使用欧氏距离计算目标位置与当前位置的距离. 如果小于传送之翼的范围, 则可以移动, 反之不可以.
   3. 玩家可以移动到当前已有玩家的位置, 即每个位置玩家数量无上限.
5. **战斗轮**

玩家进行攻击时，首先进行攻击判定。攻击玩家与被攻击玩家各投掷一枚色子，若攻击玩家点数不小于攻击玩家用于攻击的装备等级的负一次方乘以被攻击玩家点数，则攻击判定为成功，否则为失败。即 时成功.

攻击判定成功时，依据攻击玩家所使用装备对被攻击玩家造成相应伤害；攻击判定失败时，视为被攻击玩家成功防御或闪避了此次攻击，不受到伤害。

玩家使用近战武器进行攻击时，需与被攻击玩家距离为0；使用远程武器进行攻击时，需与被攻击玩家距离不大于其远程武器射程。在游戏中, 程序会根据玩家位置自动选择最优攻击武器, 保证攻击效果最大化.

1. **商店清单**

* 拳头(fist)（非卖品）0.5级 100% 1伤害
* 驱魔大刀(knife)（近战）1级 1金币 100% 1伤害

2级 3金币 75% 2伤害 25% 1伤害

3级 5金币 75% 3伤害 25% 1伤害

* 烈火之枪(gun)（远程）1级 2金币 50% 1伤害 50% 0伤害

2级 4金币 50% 2伤害 25% 1伤害 25% 0伤害

3级 6金币 50% 3伤害 25% 2伤害 25% 0伤害

枪类武器射程等同于其等级，只能攻击水平或垂直方向目标

* 传送之翼(wings)（飞行道具）1级 3金币 增加0.5行动力上限

2级 5金币 增加1行动力上限

* 生命药水(blood) 1金币，购买后玩家立刻回复1点血量, 已到达血量上限则无变化.
* 回城卷轴(tp) 1金币 玩家拥有行动资格时，可选择使用回城卷轴，移动到任一自己占领的家，该行为视为一次行动

1. **玩家死亡**

当玩家血量降低至0（含）以下时，该玩家死亡。当一名玩家的攻击造成一名其它玩家死亡时，该玩家血量上限+1，血量+1。当只有一名玩家未死亡时，游戏结束，该玩家胜利。

**附录2: 代码github开源地址(包含Qt版本, c++版本, 游戏介绍等文件)**

<https://github.com/ZimingLi1204/glory_of_pro>