浙江大学

Generals 项目中期报告

授课教师：	袁昕
组长：	王造时
组员：	胡思远
	徐若禺
日期：	2024.7.11

1 项目介绍

1.1 项目说明

- 项目名称：将军棋
- 开发小组名称：Generals 组
- 开发小组成员：王造时、胡思远、徐若禺

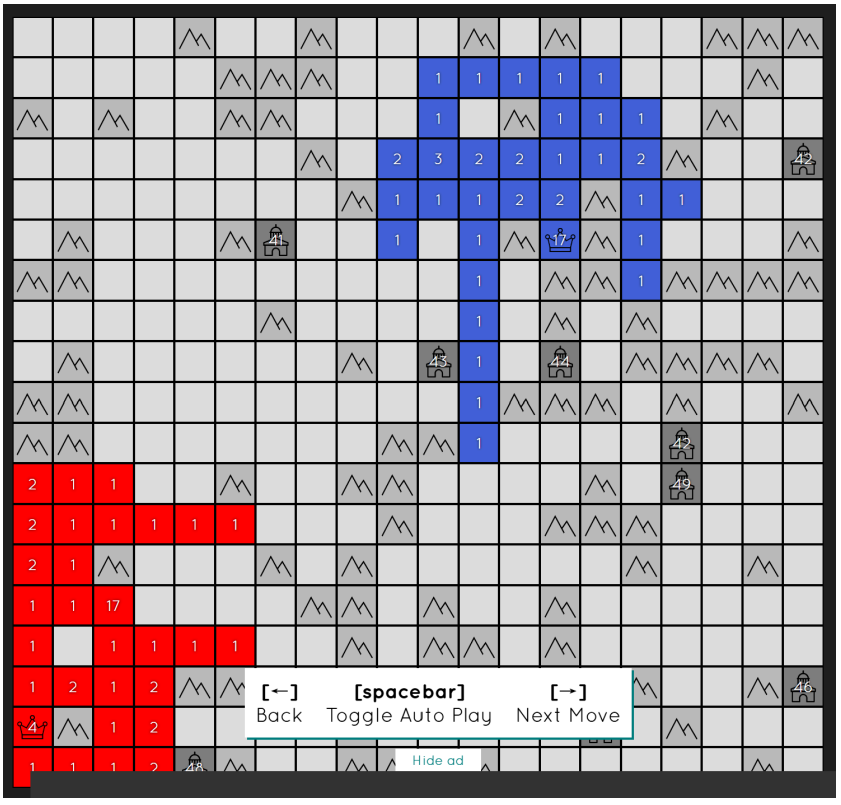
1.2 项目简介

本项目是对多人在线策略塔防游戏将军棋的复刻，玩家可以在本地进行人机游戏，也可通过局域网联机对战。

游戏在一块大小为 $n \times m$ 棋盘上进行，其中有四种格子：基地、障碍、要塞和空地。

- 基地：每名玩家有一座初始基地，一旦被其他玩家占领，该玩家出局
- 障碍：不可被经过，也不可被占有
- 要塞：初始时不被任何玩家占有且拥有若干兵力，可以被玩家占领，占领后开始产出兵力
- 空地：可以被玩家占领

玩家可以操纵自己的兵力来占领空地和要塞，以及攻打其他玩家的领地。若玩家的基地被攻占，则该玩家被淘汰。最终存活的玩家游戏胜利。



2 项目目标

2.1 基本目标

- 实现游戏地图的随机生成
- 实现游戏的基本操作
- 实现图形界面的即时显示
- 实现离线人机对战（即实现游戏 bot）
- 实现自定义地图功能

2.2 进阶目标

- 实现对局回放功能
- 丰富游戏模式与地图元素
- 实现多人局域网联机对战

3 开发进度

第一轮迭代已完成

3.1 第一轮迭代

- [x] 地图随机生成
- [x] 游戏基本操作
- [x] 图形界面显示

3.2 第二轮迭代

- [x] 实时排行榜和公告栏的显示
- [ ] 支持用户操作快捷键
- [ ] 离线人机对战
- [ ] 自定义地图功能

3.3 第三轮迭代

- [ ] 对局回放功能
- [ ] 丰富游戏模式
- [ ] 多人局域网联机对战

3.4 说明

在实际开发中，我们发现“多人联机”这一功能实现难度较大。在经过充分考量以及与老师的沟通后，我们决定将这一功能作为本项目的进阶需求。相应地，我们对第二轮迭代的目标做了一些调整，使得项目开发的规划更加合理。

4 技术难点

4.1 随机地图生成算法

为了减少游戏的重复性，也就是尽可能地保证每一次游戏的地图都不相同，选择一个合适的随机地图生成算法是至关重要的。

同时，我们为了保证地图上的内容都是尽可能有效的（不存在无法到达的空地），我们制定了随机生成地图的基本要求：

- 将山区和要塞视为不可走的点时，生成的地图中的空地联通
- 地图包括至少 (16) 个空地

在实际算法中，我们先接收用户输入的地图尺寸，以及要塞和山区的个数。接着，我们随机选择不同的格子，将其类型设置为山区，剩下的格子设置为空地。每次新加入一个山区时，需要判断地图中空地的连通性，由此来判断此次加入的山区是否合法。这样就可以保证最终地图中空地是联通的。接着我们再从刚才生成的山区中随机选择若干个作为要塞即可。

4.2 首都分配机制

为了保证玩家的游戏体验，我们制定了以下要求：在最优的情况下，玩家可以在不经过其他玩家的领地的前提下到达所有的空地。换句话讲，也就是在不经过其他玩家的首都的前提下能够到达所有的空地。

如果将所有的空地都视为一个结点，并在相邻的空地之间连边（四连通）。则上述要求可以转化为：选择的所有首都都不能是这张图的割点。

寻找一张图的所有割点可以使用 *Tarjan* 算法。但为了简化算法的实现，我们在保证了上述要求的前提下，选择了一个更为简单的算法。

我们只需要找到那些一定不可能是割点的空地来作为首都的候选点即可。接着，我们再在这些候选点中随机挑选首都即可。需要注意的是，我们还需要保证首都之间不能太近，此处我们设置首都之间的曼哈顿距离不小于 5。

4.3 页面切换

游戏一共拥有四个界面，分别为主页面（MainWindow），地图创建页面（MapPage），游戏设置页面（SettingPage）及游戏页面（GamePage）。其中游戏设置页面暂时未开发，其余页面切换方法如下：在主页面中按下"Ready"按钮，切换至地图创建页面；地图创建页面中按下"Start Game"按钮，切换至游戏设置页面，或按下"Back"按钮返回至主页面。

对于已有的三个页面之间的关系，我们曾考虑过两种模式。

- 包含的模式，即主页面包含地图创建页面成员变量，地图创建页面包含了游戏页面成员变量。这种模式的好处是页面切换完全由View层内部决定，方便整体的测试；坏处是当外部要访问内层页面变量时比较麻烦。
- 独立的模式，即三个页面之间没有包含关系，独立存在。这种模式的好处是更加灵活，并且方便分别测试；坏处是页面之间信息的传递相比之下更加复杂。

由于能查阅到的资料较多地使用了第一种模式，并且在尝试第二种模式时产生了未能解决的错误，于是我们采用第一种模式。

对于页面切换的问题，我们的解决方案是使用Qt内置的信号与槽函数。Qt中的按钮类本身带有clicked()信号，当被点击时自动释放。当我们捕捉到这个信号时，我们隐藏当前页面，并显示目标页面。以主页面切换至地图创建页面的过程为例，相关代码如下：

```
connect(ui->Ready_Button, &QPushButton::clicked, this, [=]  {
    this->hide();
    mappage->show();
});
```

其中Ready_Button是一个按钮类，但其被按下时，会释放clicked信号。当主页面（this）接收到这一信号时，其会完成槽函数的内容。此处我们的槽函数使用lambda表达式书写，两行分别表示关闭当前页面和显示地图创建页面的动作。

其余几处页面切换类似。

4.4 地图绘制

地图是一个w×h的方阵（由若干个正方形排列而成）。由于点击地图的某个单元格会产生特殊效果，于是我们选择使用按钮类来完成地图的绘制。假设我们已经接收到了地图的信息（例如长宽、单元格内容等），并约定在下文中单元格与对应坐标的按钮含义一致。

我们使用一个二维按钮指针数组来存储按钮信息。为了防止多次创建指针导致不必要的空间消耗，我们直接在页面的构造函数中创建指针。当得知具体的长宽后，我们显示其中的一部分按钮、隐藏另一部分按钮，以实现地图单元格数量的正确显示。接下来我们根据窗口大小、长宽等信息来计算单元格的边长，以便适应不同屏幕的大小和地图的大小。

每个单元格的类型有四种可能：空格（Blank），城市（City），城堡（Capital）和山地（Mountain），以及其是否被选中（Focus）。对于每一种不同的单元格及选中情况，我们都有对应的图片和QSS代码来展示内容，可以使用switch-case块方便地实现。

4.5 服务器联机

在项目规划阶段，我们组原本计划在第二轮迭代中使用 QWebSocket 类实现多人联机对战。在对 MVVM 框架理解不够深入的前期阶段，我们本以为服务器和客户端可以放进同一个程序框架中，但在自己摸索的过程中发现这一目标遇到不小的阻碍。事实上，受游戏特性所限，若想实现多主机在线对战，就必须对服务器和客户端进行分离，这意味着项目在第一轮迭代后需要进行较大的结构性变动，且缺乏计算机网络知识也会为代码编写过程带来极大的挑战。

在询问老师的建议之后，我们组对这一功能进行重新评估，考虑到项目完成的时间并不充裕，最终决定将多人联机对战作为第三轮迭代的进阶目标。同时，在第二轮迭代中我们将重点实现“人机对战”这一功能，尽量完成一个强度较高、策略智能的自动对战机器人。重新评估项目可行性、及时调整目标计划的过程虽然为项目开发带来一定的波折，但也是实际开发场景中常见的问题。相信有此次经历后，我们在今后项目开发中遇到类似问题时能够更加从容地应对。

4.6 MVVM 框架搭建

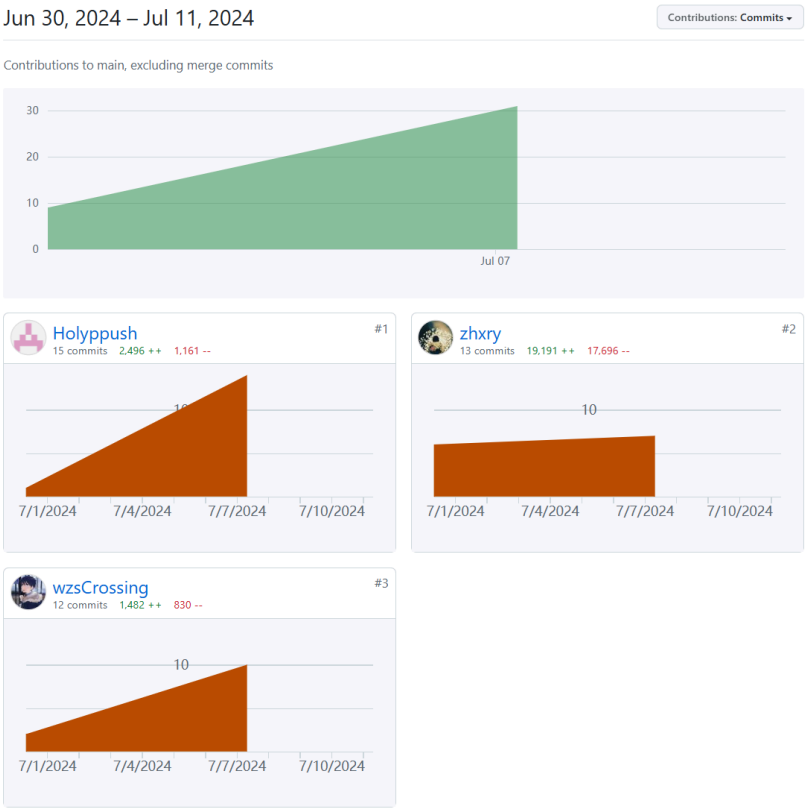
我们在搭建 MVVM 框架时遇到了不少问题。在项目开发的前期，我们对框架中各层之间关系的理解存在一些问题，导致不同层级之间存在耦合，开发的效率也比较低下。比如，我们没有正确地认识到数据绑定的作用，而是在 View 层中直接调用了 ViewModel 层来获取游戏信息。但实际上，View 层与 ViewModel 层应该是感受不到彼此的存在的，这样才能保证并行开发。而数据绑定使得 View 和 ViewModel 之间的通信更加简洁和高效。例如，玩家的资源、单位数量和地图状态都可以通过数据绑定实时更新UI。

后来，在与老师多次的交流沟通之后，我们才正确地认识到了 View 层和 ViewModel 层的关系，并逐步地解决了代码耦合的问题。在后续的开发中，我们也发现在正确实现这一框架后，游戏操作的添加也会变得更加便捷，也深刻地体会到了框架式开发的重要意义。

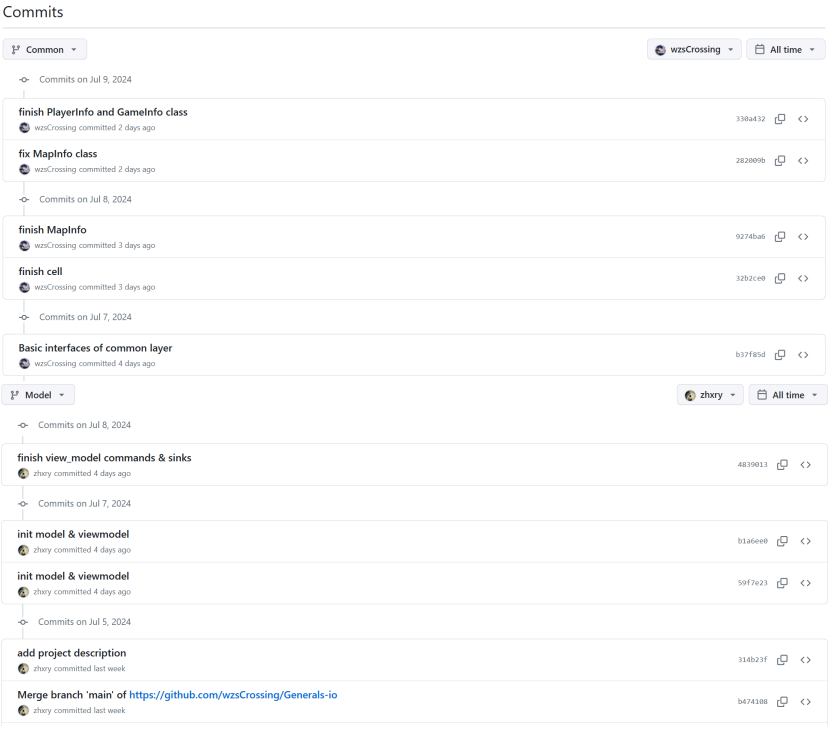
5 协作情况

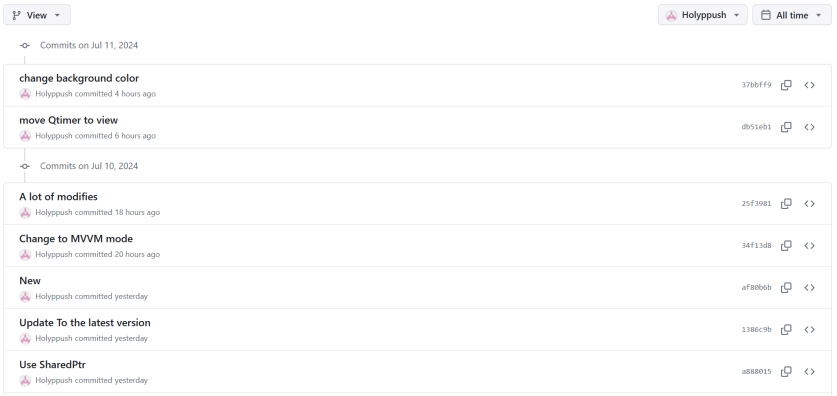
成员之间通过 Github 进行协作开发。根据事先安排好的分工，我们为每个成员新建分支来完成相应的工作，大家都在自己对应的分支进行代码的提交。由于 MVVM 框架的特性，各个成员之间的工作并不需要相互等待，也不会产生冲突，这极大地提高了协作开发的效率。

在项目开发过程中，组长负责协调组员之间的协作，负责各个分支的合并工作，并在发生合并冲突时及时解决。



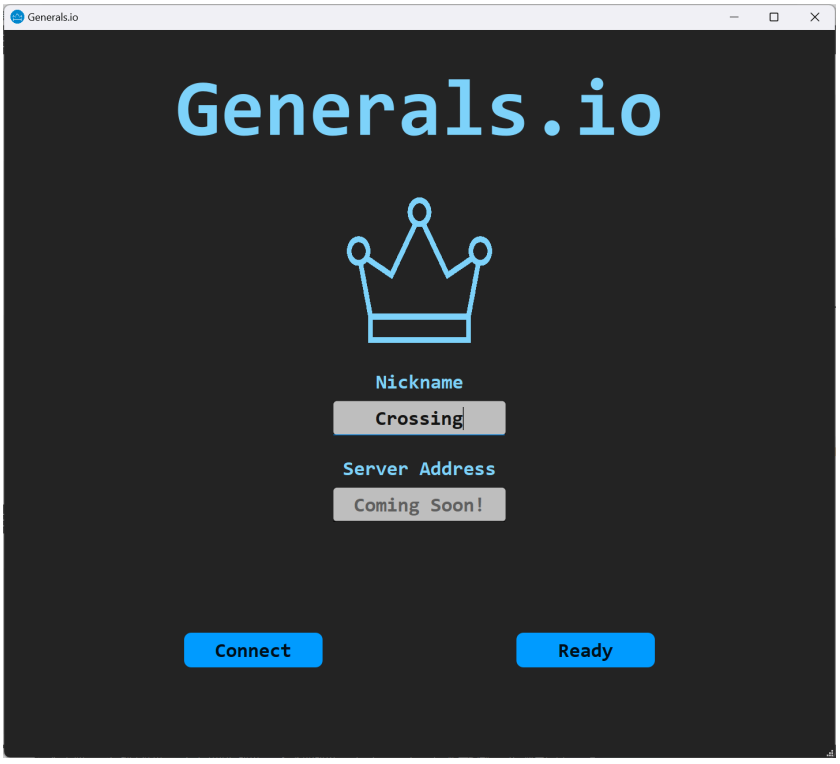
5.1 部分代码提交记录





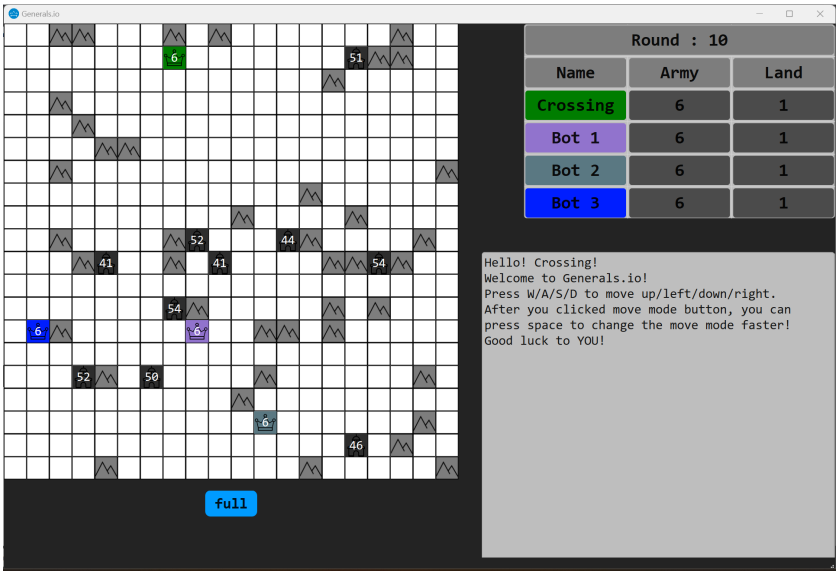
6 部分效果图

6.1 开始界面



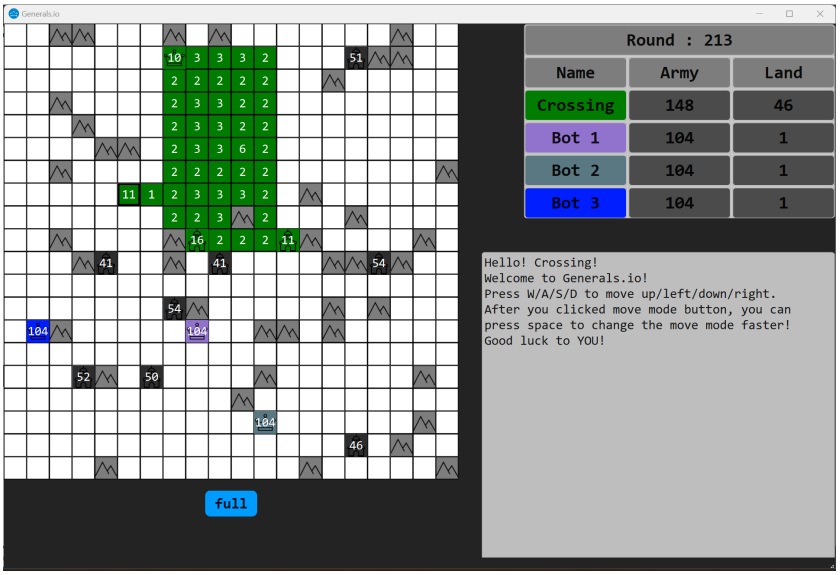
6.2 游戏界面

随机地图生成、排行榜和公告栏显示



6.3 游戏进程

- 玩家操作：鼠标单击选择当前单元格，WASD操纵兵力的转移，并可以选择转移全部兵力或一半兵力（之后还会完善撤销、清除等玩家操作）
- 地图更新：（1）根据玩家的移动操作更新地图的显示（2）每固定时间间隔增加单元格上的兵力
- 排行榜更新：获取地图信息，动态更新玩家的排名并显示（以兵力为第一关键字，领地数量为第二关键字）



7 个人心得

王造时：

本项目源于我们在高中时接触到的一款网络联机小游戏 Generals-io，当时在机房局域网联机这一游戏的时光给我带来了很多快乐。借本课程的契机，同时也考虑到该游戏的界面和规则设计并不是很复杂，我们便选择这一游戏作为开发项目。稍有遗憾的是，我们暂时没有能力实现网络联机的功能，所以我们选择开发一个人机作战的版本。

在项目开发中，我主要负责 Common 层和 App 层的开发工作。MVVM 不仅帮助我们实现了代码的高可维护性和可测试性，还使得游戏的开发过程更加高效和结构化。尽管在实现过程中遇到了一些挑战，但通过团队的努力和对架构的深入理解，我们成功地解决了这些问题。这个项目不仅让我在技术上有了很大的提升，也让我对软件架构设计有了更深刻的认识。

胡思远：

由于我对游戏规则的理解并不是最深入的，并且对图形界面颇有兴趣，因此在项目开发中我主要负责 View 层的工作。在使用 Qt 的过程中，我第一次发觉图形界面的设计是如此方便和神奇（在此之前，我仅在 C 大课程中接触过图形界面，但是使用课程提供的库函数非常麻烦别扭），并且觉得将整个外观掌控在自己手里的感觉非常令人兴奋。虽然设计的过程查阅了不少资料，产生了不少问题，但是整体来看还是非常愉悦的。

但是由于对 MVVM 理解的不到位，我经历了不少次对代码的大规模修改。不过在这个过程当中，我也逐渐一步步理解了这个模式，从对它的质疑变成了接受与认可。相信在这几天的不断试错之后，我们的工作能进展地更加顺利。

徐若禺：

在第一轮迭代中，我负责 Model 和 ViewModel 层的编写。我需要根据 Common 层中的基础类在 Model 层中实现游戏的基本逻辑，并在 ViewModel 层中将 Model 层的内容进行整合，为 View 层提供简洁的接口。项目开始的前几天，我在配置带有 Qt 库的 VSCode 环境上耗费了较多的时间，好在最终搭建成功，让后续的编写体验大大提升。

虽然上学期有 OOP 的先修基础，但由于缺乏大型项目的协作编写经验、对 MVVM 框架理解不够深入，我在 Model、ViewModel 两个类的框架搭建过程中迟迟无法推进。不过在多次与同伴讨论、向老师请教之后，我逐渐理解了 MVVM 框架的设计思想，也明白了各个层级之间的关系，最终成功地完成了 Model 和 ViewModel 层的编写。在第一轮迭代对框架进行规范化后，之后进行功能扩展时便无需对现有框架进行较大调整，组员之间的协作效率也一定会提高。

8 总体心得

通过上学期数据库系统课程的 MiniSQL 大作业，我们对 C++ 的项目管理已经有了一定的了解和掌握，但对于软件开发的架构模式并没有太多的认识。以前完成的项目往往都是模块化的设计，实现的也都是偏后端的内容（类似于 MVVM 中的 Model 层）。

而经过本课程的学习，我们了解到了不同的软件架构模式，并尝试将 MVVM 这一框架应用到实际的项目开发中。MVVM 架构通过将数据逻辑、业务逻辑与用户界面分离，项目的代码结构更加清晰，维护和扩展变得更加容易。Model 负责处理游戏数据和规则，View 负责显示游戏界面，ViewModel 则作为桥梁协调两者。这种清晰的职责分工使得我们能够更好地管理和扩展代码。

但在实际的项目开发中，我们搭建这一框架的过程并不顺利。在项目开发的前期，我们对框架中各层之间关系的理解存在一些问题，导致不同层级之间存在耦合，开发的效率也比较低下，同时在添加操作时也发现比较麻烦。后来，在与老师多次的交流沟通之后，我们才正确地认识到了 View 层和 ViewModel 层的关系，并逐步地解决了代码耦合的问题。在后续的开发中，我们也发现在正确实现这一框架后，游戏操作的添加也会变得更加便捷，也深刻地体会到了框架式开发的重要意义。

总的来说，此次课程学习和项目开发的经历让我们受益匪浅，不仅了解到了软件开发的过程，同时也锻炼了我们团队协作的能力。唯一有些遗憾的是，“服务器联机”的功能由于技术水平有限而无法实现。希望在下学期学习过网络部分的知识后，我们能够补充完善这一功能。