浙江大学
ZHEJIANG UNIVERSITY

C++ 项目管理及工程实践

最终总结

C++ Project Management and Engineering Practices

组长 ____ _

组员一 ___ ___

组员二 ___ ___

目 录

图目录

第 1 章 项目成果

1.1 概述

目前本项目完成了三轮的迭代

1. 第一轮首先完成了 Common 层的编写，随后并行开发 view 层和 ViewModel 层，最后在 app 层完成了完成了跑动与攻击动画的显示；
2. 第二轮迭代基于第一轮迭代完成了打击信号的绑定，实现了玩家和敌人的交互；
3. 第三轮迭代基于前两轮迭代，优化了背景以及击打效果血条，添加了以及完成了游戏胜利失败结算界面，并引入了重置按钮清空游戏状态重新开始，进一步完善了袁老师在的第二次验收时指出的框架在解耦合度上的不足。

1.2 成果展示

以下是项目游戏场景，目前完成了包括各种对象的动画实现，其中实现了玩家的方向键移动、小怪的随机生成及碰撞机制移动、Boss 的追踪玩家特性等动画逻辑。

最后一轮迭代，我们主要实现了如下功能：

1. 背景图片的全填充
2. 小怪的移动规则填充
3. 游戏胜利失败界面设计

1.2.1 主角的攻击动画

本小节展示玩家攻击动画截图，包括击打小怪和击打 Boss 的动画截图。

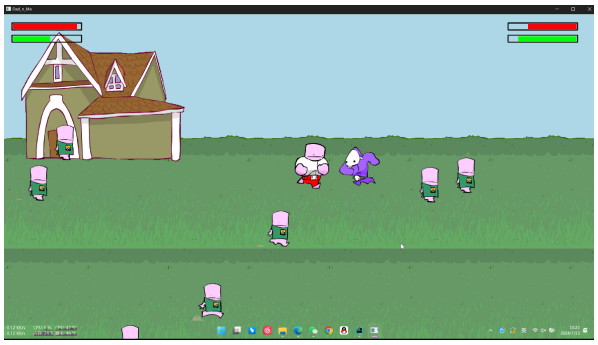

图 1-1 Game Scene

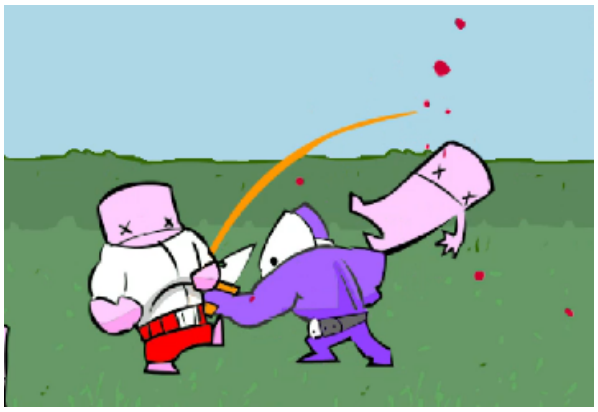

图 1-2 Player_Attack

1.2.2 人物血槽和精力条设计

本小节展示人物血槽和精力条展示，人物会被 Boss 攻击直至死亡，当人物死亡游戏即结束，反之，Boss 的生命清空及小怪被杀死时即游戏胜利

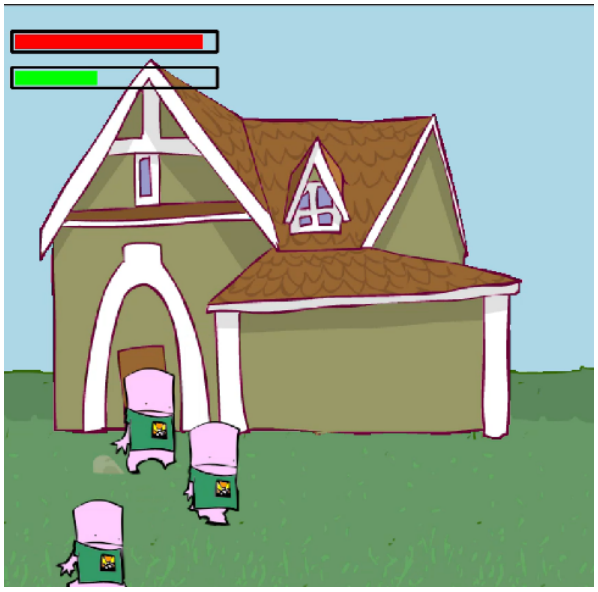

图 1-3 Blood&Power

1.2.3 胜利及失败界面

图 1-4 Win scene

图 1-5 Lose scene

1.2.4 背景显示与障碍物反弹机制

我们设计了一个房屋实例并完善了障碍物检测机制，使得人物在移动时无法穿墙以及小怪在碰到房屋时反方向反弹的机制，限于文档只能插入静态图片，我们不便于将小怪的反弹动画展现出来，只能在验收时完成。

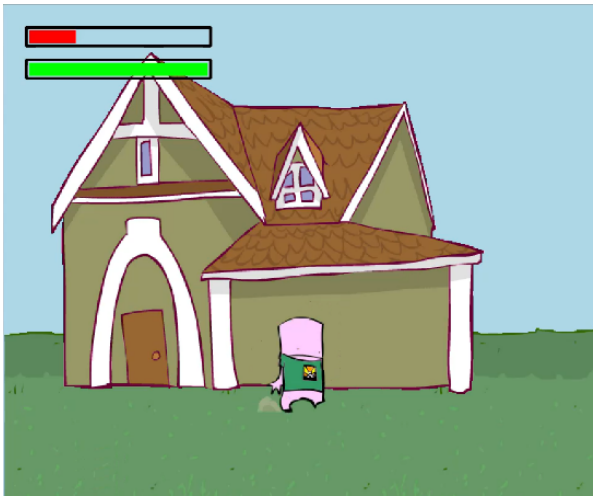

图 1-6 BackGround

第 2 章 开发过程

2.1 Git 协作情况

以下是主分支 Main 的推送情况，包括有 main 的自提交和 pr 的合并

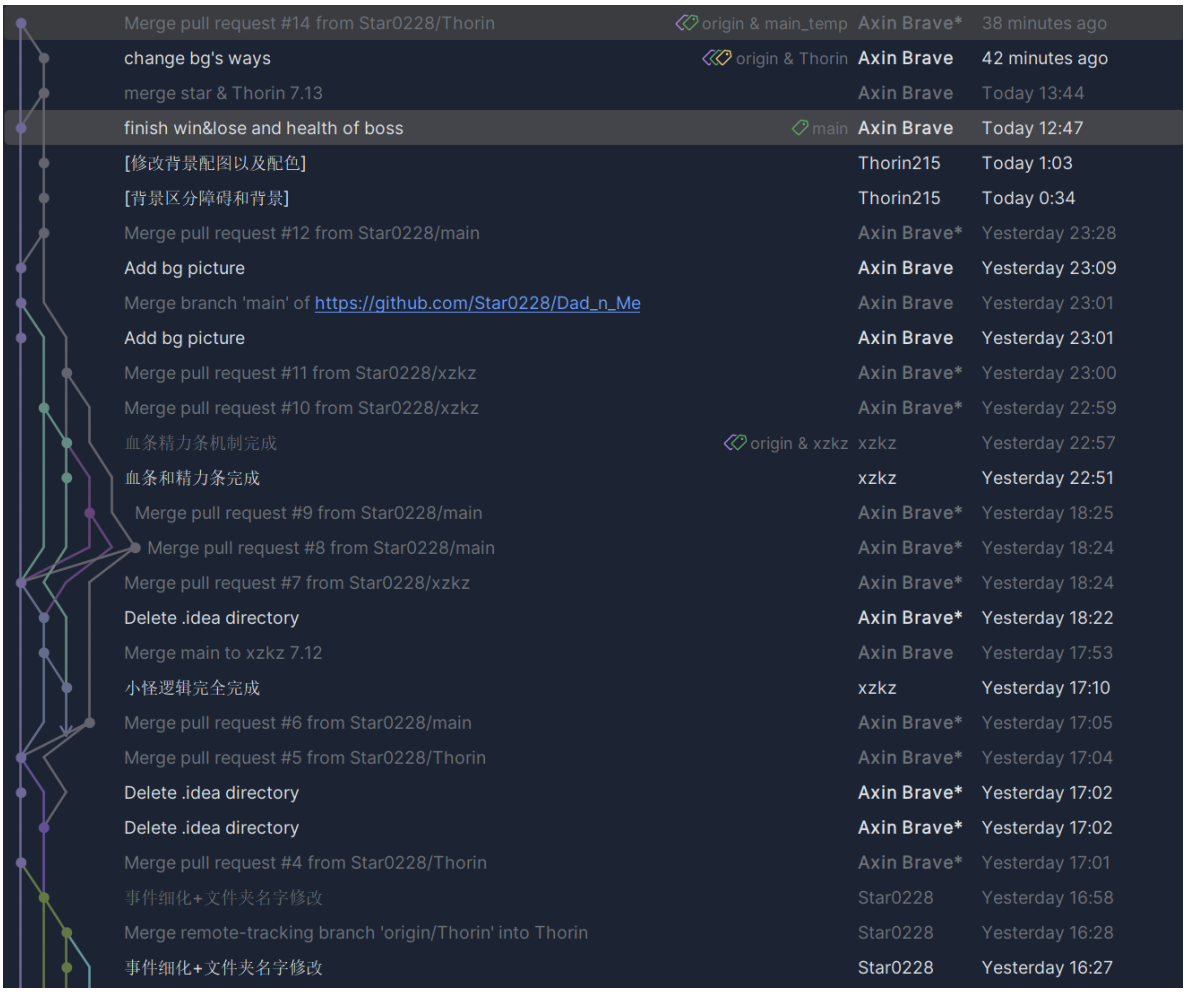

图 2-1 Git log 1/3

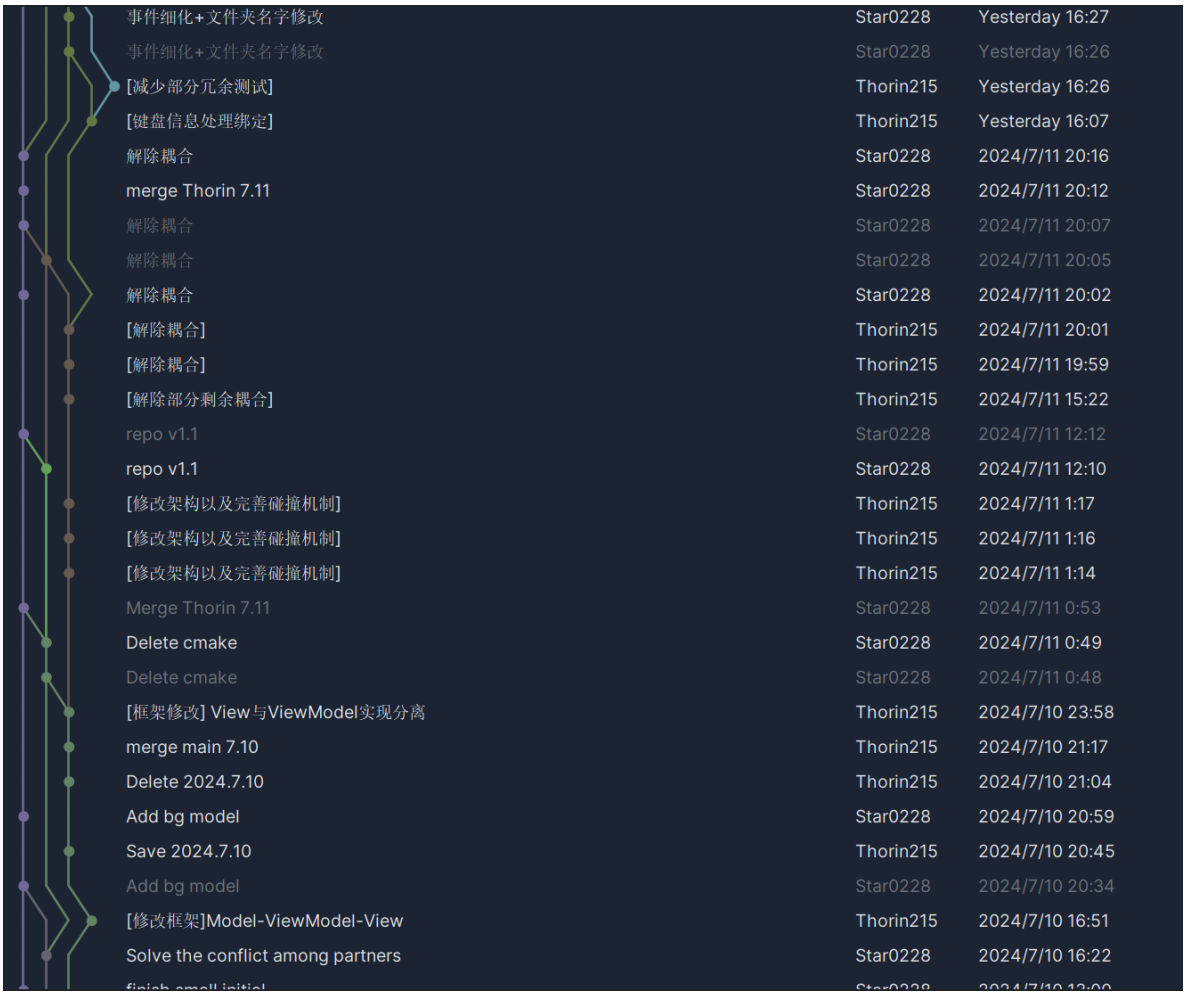

图 2-2 Git log 2/3

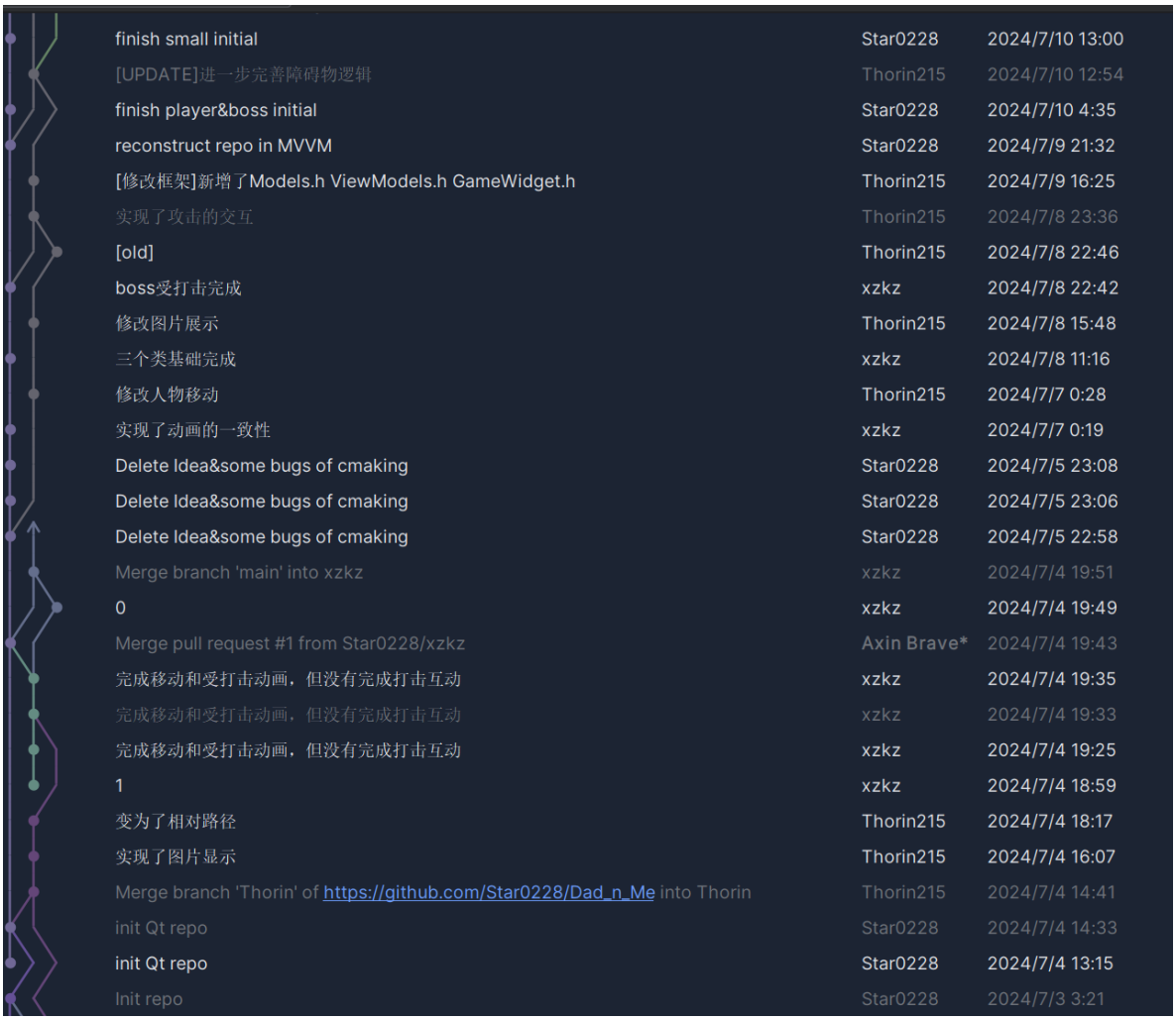

图 2-3 Git log 3/3

2.2 工具使用

2.2.1 开发环境

开发环境使用了 QT 6.5.3，以提供良好的图形用户界面支持。

2.2.2 编程语言

项目的主要编程语言为 C++ 17，兼具性能和现代编程特性。

2.2.3 框架结构

采用了 MVVM 架构（模型-视图-视图模型），实现了前后端分离，高复用性和低耦合性。

2.2.4 文档撰写

文档撰写在 overleaf 在线平台协作进行，使用了 Markdown 和 LaTeX，确保文档的清晰性和可维护性。

2.2.5 交流讨论

团队内部的交流和讨论通过钉钉视频会议和线下进行，以保证沟通的效率和质量。

2.2.6 开发工具 CLion

智能代码编辑器

Clion 智能代码编辑器提供了自动补全、语法高亮、错误提示和代码重构等功能。Clion 还可以自动检测代码风格细节，提供 clang-tidy 建议。

智能代码导航

本工程使用了 Clion 强大的代码导航功能，它支持快速跳转到函数、变量或类的定义，以及查找所有引用和使用情况。同时，Clion 还提供了的搜索和替换功能，以帮助开发者在整个项目中查找和修改代码。

调试工具

本工程使用了 Clion 提供的断点、单步执行、变量监视和堆栈跟踪等功能。

版本控制集成

本工程使用了 Clion 提供的 Git 版本控制系统的功能与 GitHub 和其他代码托管平台的集成，提高了小组协作的效率。

2.2.7 构建工具 CMake

CMake 是一个项目构建工具，并且是跨平台的。关于项目构建我们所熟知的还有 Makefile（通过 make 命令进行项目的构建），大多是 IDE 软件都集成了 make，比如：VS 的 nmake、linux 下的 GNU make、**Qt 的 qmake** 等，如果自己动手写 makefile，会发现，makefile 通常依赖于当前的编译平台，而且编写 makefile 的工作量比较大，解决依赖关系时也容易出错。而 CMake 恰好能解决上述问题，其允许开发者指定整个工程的编译流程，在根据编译平台，自动生成本地化的 Makefile 和工程文件，最后用户只需 make 编译即可.

2.2.8 Git 版本控制

- 本次协作的前期通过 Git 分支管理完成版本控制。在工程仓库中创建三个分支，对应每位组员的工程任务。在组员的任务完成后将代码推送到自己的分支上，随后由 app 层的开发者控制工程的架构合并到 main 分支上。

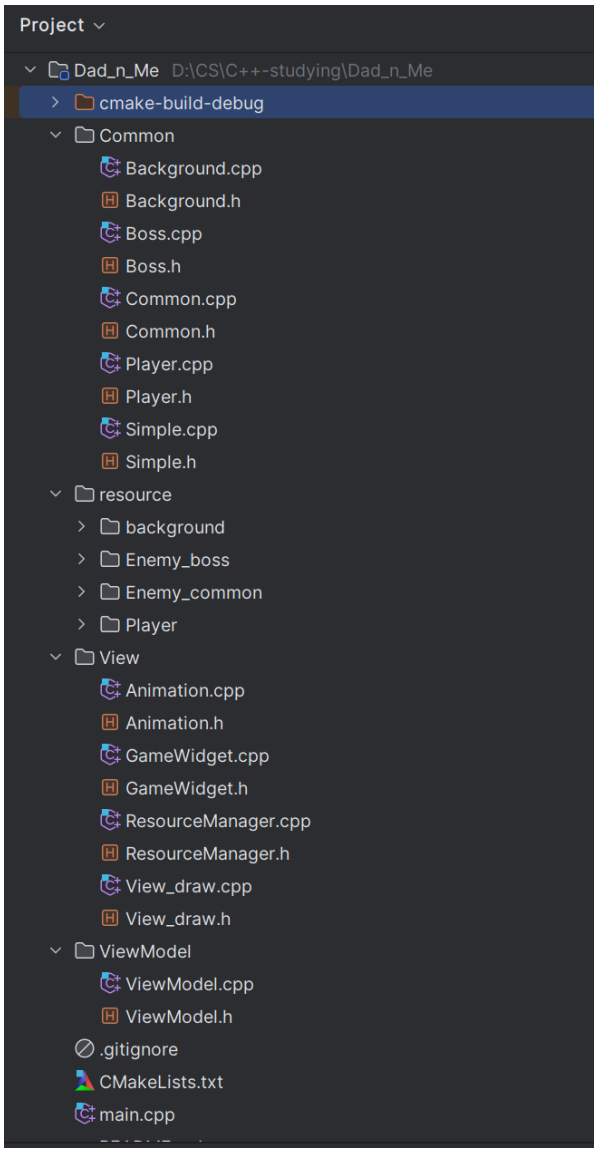

图 2-4 Index

- 本次协作的后期通过 Git PullRequest 完成版本控制。小组成员分别 fork 原仓库，在自己的本地维护远程分支和本地修改分支，当产生最新修改后可以添加 Pull Request 的标题和描述，解释所做的更改，并提交请求到原仓库，由小组成员之一进行仓库的维护，一旦 Pull Request 被批准，它可以被合并到原始仓库中。

第 3 章 总体心得

本次课程可以分为理论学习阶段和实践阶段。在课程的初始阶段我们跟随袁老师系统地学习了 C++ 开发的规范、工具、经验以及架构等。随后我们在 MVVM 的架构下从零开始展开游戏项目的设计。在整个开发流程的每一轮迭代中，袁老师都为我们提出了宝贵的修改建议，并帮助我们理清了框架设计思路，纠正了对于各层关系的错误认知，我们最终实现了低耦合、代码可重用、使用现代化开发工具并经历多轮迭代的游戏项目。

1. **MVVM 架构的深入理解：**MVVM（Model-View-ViewModel）架构模式为大型项目开发提供了一种清晰、高效的设计思路。它将数据模型、用户界面和业务逻辑分离，使得各个部分可以独立开发和维护。

2. **耦合度的降低：**在大型项目中，模块间的高耦合度往往导致开发效率低下。通过 MVVM 架构，我们学会了如何降低耦合度，实现模块间的松散耦合，从而避免了模块开发中的等待和依赖问题。

3. **并行开发的实现：**在编写好 common 层之后，view 层和 viewmodel 层可以并行开发，这大大提升了开发效率。并行开发不仅加快了项目进度，还允许团队成员专注于自己的专业领域。

4. **模块化与独立性：**架构的模块化特性可以使得项目可以被分解为独立的部分，每个部分都可以独立开发和测试。这种独立性不仅提高了开发流程的灵活性，也增强了项目的可维护性。

5. **代码的可重用性：**设计可重用的代码模块，减少了重复劳动，提高了开发效率，并且在未来的项目中可以快速复用已有的代码。

6. **持续迭代的重要性：**软件开发是一个持续迭代的过程，每次迭代都是对现有架构和代码的优化，是提升产品质量的关键。

第 4 章 个人心得

在前两轮迭代的过程中以及和袁老师交流的过程中，我对于 MVVM 架构有了更深刻的理解。我也对 Qt 中的 Qwidget 以及 QObject 的类的运用有了更多的理解，使用 connect 在 app 层面进行连接，对于企业级别的并行以及团队合作有了更深刻的认识。也要感谢袁老师不厌其烦的解答我们关于接解除耦合的疑问以及宝贵的指导。我也在后续的迭代中进一步感受到了 MVVM 框架相比较于传统的 MVC 的优势就是其模块具有更好的独立性，更适用于这种规模项目的并行运作。

本次工程是我的第二个协作项目。与我的上一个协作项目 MiniSQL 不同，本项目需要在 MVVM 架构下进行框架的设计与分工。我们需要考虑以低耦合的方式进行并行开发，更加贴近企业大型项目的开发思路。在袁老师的亲自指导下我们对于绑定信号、整体构建等技术知识有了更加深刻的理解，收货满满。在每一轮迭代的过程中，我们的框架在袁老师的指导下不断完善。我们提出的所有问题袁老师都一一进行了耐心的解答，帮助我们消除了一些对框架的错误认识乃至与 MVC 框架的混淆。相信我们能带着本次课程的收货与经验行稳致远。

在本次工程中我担任的角色之一是框架设计。通过与袁老师的交流和项目实际编写时的不断碰壁与试错，我对 MVVM 架构和 Qt 图形化构建能力有了更加深入的认识。MVVM 不仅提升了代码的组织性，还增强了开发过程中的灵活性和可维护性。在大型项目开发中，通过合理设计 common 层、view 层和 viewmodel 层，有效避免了模块间的依赖和等待，提升了开发效率；MVVM 架构的模块化特性能够将项目分解为独立的部分，每个部分都可以独立开发和测试，大大提高了开发流程的灵活性和项目的可维护性，同时我学会了如何设计可重用的代码模块，减少了重复劳动，提升了开发效率。在此再次感谢袁昕老师对项目细致无私的指导与帮助！

第 5 章 致谢与建议

本次实验袁老师对我们进行了细致入微的指导，我们对于框架的错误理解都被袁老师进行了一一纠正，而具体到本次实验的一些实现细节袁老师也给出了具体的建议。我们认为本次课程的理论与实践结合十分紧密，在修改框架时我们反过头来再次研读 ppt 又有了新的理解与收获。

关于验收方面 我们也在实践过程中慢慢理解了袁老师细致检查框架的良苦用心，一个低耦合度的框架给我们的合作带来了事半功倍的效果。而且也在验收的过程中加深了对于面向对象程序设计的理解。

理论课方面 在理论课上，我们对于企业中的工程开发有了深刻的认知，也对并行实践有了更深刻的认知。

对于本节课的建议 袁老师提供了很多有趣的例子，希望在理论课的前期可以有更多的例子。