浙江大学
ZHEJIANG UNIVERSITY

C++ 项目管理及工程实践

中期总结

C++ Project Management and Engineering Practices

组长 ____ _

组员一 ___ ___

组员二 ___ ___

目 录

图目录

第 1 章 迭代成果

1.1 概述

目前本项目完成了两轮的迭代，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；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。

1.2 成果展示

以下是项目初步的游戏场景，目前完成了包括玩家、boss、小怪的图片导入 (于项目./resource 目录下) 和动画实现，其中实现了玩家的方向键移动、小怪的随机生成及单方向移动、Boss 的追踪玩家特性等动画逻辑

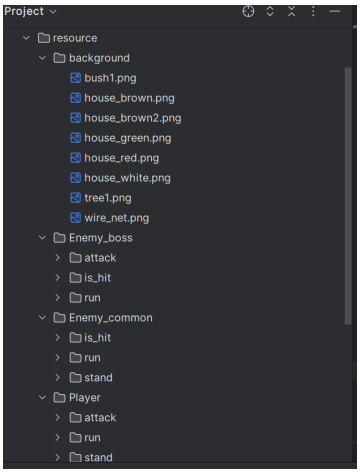


图 1-1 Project picture resource

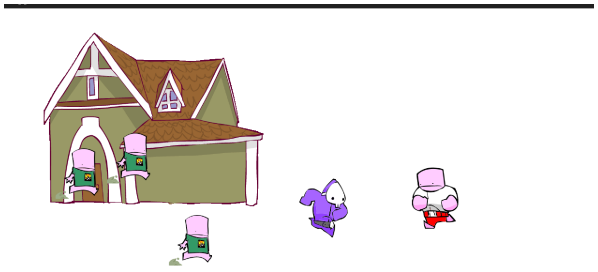


图 1-2 Game Scene

我们将在本轮迭代以及框架的基本建设完成后继续进行背景优化和细节，具体来说我们将会实现

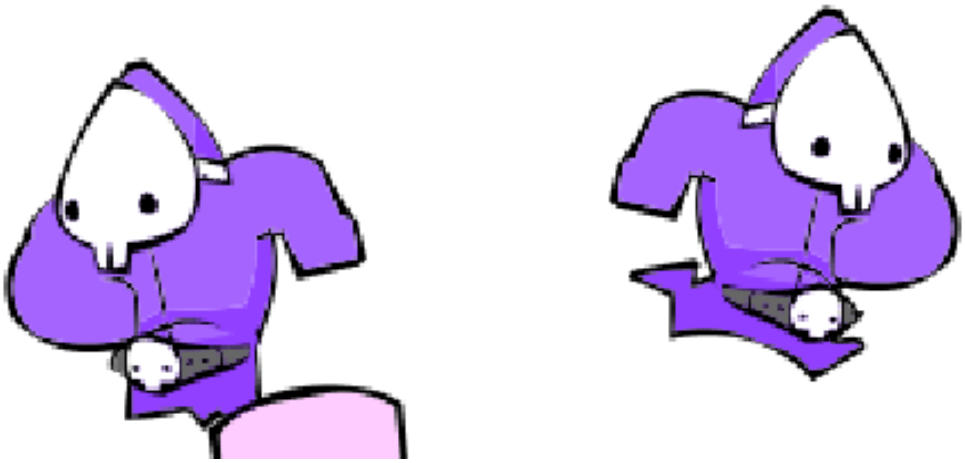
1. 背景图片的全填充

2. 小怪的移动规则填充

3. 游戏胜利失败界面设计

1.2.1 主角的移动与攻击动画

本小节展示玩家移动 & 攻击动画截图



(a) Player moves left (b) Player moves right

图 1-3 移动动画

图 1-4 Player attacks image
攻击动画

1.2.2 小怪的移动与死亡动画

本小节展示小怪移动 & 被攻击攻击动画截图

(a) Simple monster moves (b) Simple monster is hit

图 1-5 小怪动画

1.2.3 Boss 的索敌机制与攻击动画

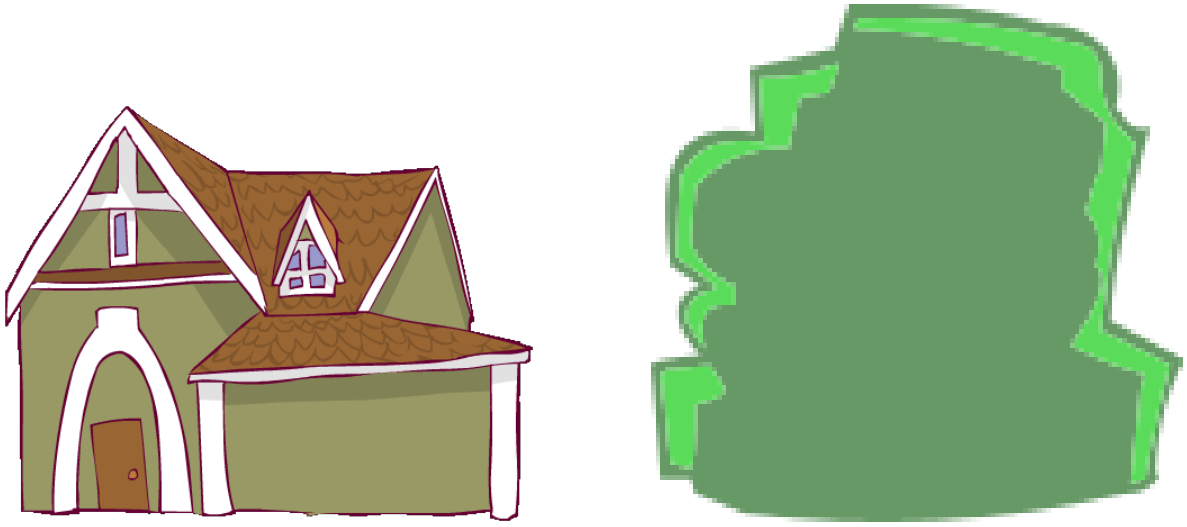
本小节展示 Boss 索敌移动机制、攻击动画截图

通过计算与玩家之间的 x、y 方向上的距离以及归一化两物体的欧式距离来调整 Boss 的移动路径，从而实现 Boss 的自动移动机制。

(a) Boss run (b) Boss attack

图 1-6 Boss 动画

1.2.4 背景显示与障碍物逻辑



(a) Background1 (b) Background2

图 1-7 背景显示图片

图 1-8 Obstacles: 人物无法穿过障碍物，实现了背景的实物化

第 2 章 协作情况

2.1 分工情况

- 开发 App 层，组装其他两层的代码
- 参与度占比 1/3

- 开发 View 层和 Window 层
- 参与度占比 1/3

- 开发 ViewModel 和 Model 层
- 参与度占比 1/3

2.2 协作过程

- 本次协作通过 Git 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。
- 首先完成 common 层数据结构的编写，随后分别开始基于 common 中各个类的 view 层和 viewmodel 层的隔离并行编写，最后在 app 层中实现各层对象的绑定。

第 3 章 难点攻克

3.1 框架难点

Common 层

存放数据结构。这里在 common 层中放置玩家、小怪、boss 以及背景地图四个类的声明以及定义。在 common 层完成编写后 view 层和 viewmodel 层的编写将根据 common 层中定义的数据结构分别独立进行。

View 层

view 层中隔离地实现了底层的图片管理（Animation 类、ResourceManager 类）与动画播放（draw 方法），同时也实现了键盘事件的响应。

ViewModel 层

加载和改变需要呈现的各种数据，提供 common 对象集合，提供可用来显示的数据类，提供了一系列 get,set 函数暴露可用来绑定的数据属性。

App 层

创建 common 层、view 层、ViewModel 层的对象，并实现它们之间的相互绑定。

3.2 信号槽机制难点

signal-slot 信号槽机制是 Qt 的核心机制之一。通过继承 QObject 类并定义 signals 信号函数和 slots 槽函数可以实现不同对象之间的通信。本工程中为了实现键盘信号从 view 层向 ViewModel 层的传递使用了 connect 方法将 view 层产生的键盘事件信号连接到 ViewModel 层中根据信号产生行为的 slot 中，这样可以保证 view 层开发完全独立于 ViewModel 层并且确保了状态的同步性。

3.3 动画帧控制难点

动画的播放本质上需要实现图片替换逻辑。这里使用 timer 变量来标识是否达到了替换图片的时间间隔，即控制了同一张图片将会持续显示多帧。在 view 层中，对于 common 层的不同的数据结构进行了图片的分类管理与绘制方法重写。view 层可以非耦合地控制动画的播放。

第 4 章 总体心得

在经过与袁昕老师的深入探讨和项目实践的前两轮迭代后，以下是我们总结的几点心得：

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 5 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。我也将在后续的迭代中不断完善我负责的部分，提供给更高效的 common 层以及 ViewModel 层.

在前两轮迭代的过程中，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。通过和袁老师的交流，我对于 MVVM 架构有了更深刻的理解。在实际的大型项目开发的过程中往往需要降低耦合度以避免各个模块开发的滞留等待情况，在写好 common 层之后就可以实现 view 层和 viewmodel 层的并行开发，整个工程中的主要部分相对独立，实现了低耦合和可重用性。

通过对项目的编写，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！