四川大学计算机学院

研　究　与　开　发　实　践

PRTS 事相重建

PRTS Event Reconstruct

基于 C++11 和 Qt5 的塔防游戏开发 课程实践报告

学　　院：计算机学院
专　　业：人工智能
学生姓名：吉昱阳
学　　号：2019141460203
任课教师：张轶
课 序 号：06
电子邮箱：2553036255@qq.com

2021 年 12 月 25 日

一 项目概述

本项目名为《PRTS 事相重建》（PRTS Event Reconstruct），是参照游戏《明日方舟》（Arknights）进行设计和创作，所设计的关卡、角色、敌人、美术素材等均直接或简介源自于该游戏。本项目仅实现了欢迎界面、菜单界面、关卡界面这几个主界面。接下来，该部分会以“用户手册”的方式，对本游戏项目进行内容介绍、玩法介绍、关卡描述。

1.1 内容概述

该部分主要对游戏各页面进行介绍。

需要注意，本游戏锁定 1440x810 分辨率窗口化，且游戏元素定位多采用绝对大小和绝对定位，在不同分辨率的设备上表现可能不同。项目开发环境为 Windows10，尚未在其他操作系统环境下进行过运行测试。本游戏的不同界面均有不同的背景音乐（BGM）自动循环播放，且无相关点击或打击音效。

1.2 欢迎界面

此界面可视为标题界面或欢迎界面，显示了该项目的 LOGO 和标题。在考虑到**尚未加入或将来可能实现**的账号系统，此界面可扩展成为登录界面，即在此输入用户的账号和密码，登录成功后方可进入游戏。

此界面可交互内容不多，仅一个可单击的“开始唤醒”按钮，点击后便可进入菜单界面。

图 1: 欢迎界面

1.3 菜单界面

此界面为游戏的主菜单界面，展示了一张场景图，并可选择游戏难度、查看帮助。由于本游戏实现内容较少，所以菜单界面可操作的行为不多。若本游戏体量较大、界面较多，则可以通过该界面对其他界面进行跳转，并支持其他辅助功能。例如以下**未实现的功能**：进入账号管理界面，打开友方角色（干员）界面，打开基建界面等等。

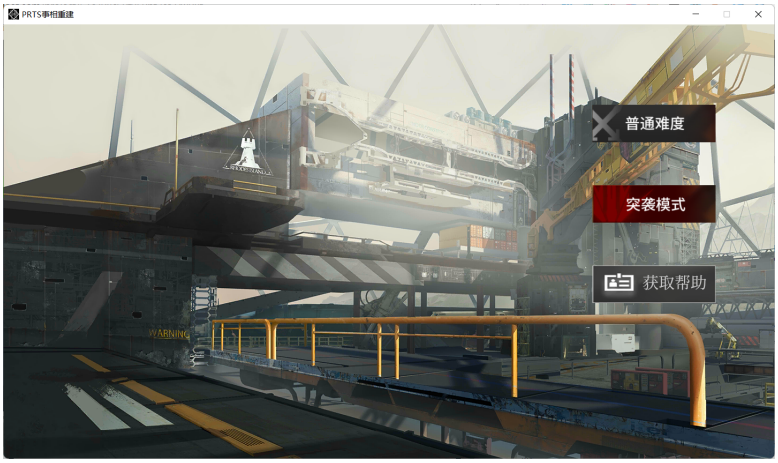

图 2: 菜单界面

此界面可交互内容仅三个按钮，从上到下依次为：

- 普通难度：点击进入普通难度的关卡界面。
- 突袭模式：点击进入突袭难度的关卡界面。
- 获取帮助：点击弹出帮助窗口，再次点击即可关闭。

弹出帮助窗口后的菜单界面如下图所示：

图 3: 帮助窗口

1.4 关卡界面

此界面为游戏的关卡战斗界面。受时间和技术力限制，本游戏仅实现了一个关卡地图（参照《明日方舟》主线关卡 1-7），敌人出现的时间、路线均固定，仅在关卡难度上有所调整。关卡界面如下图所示：

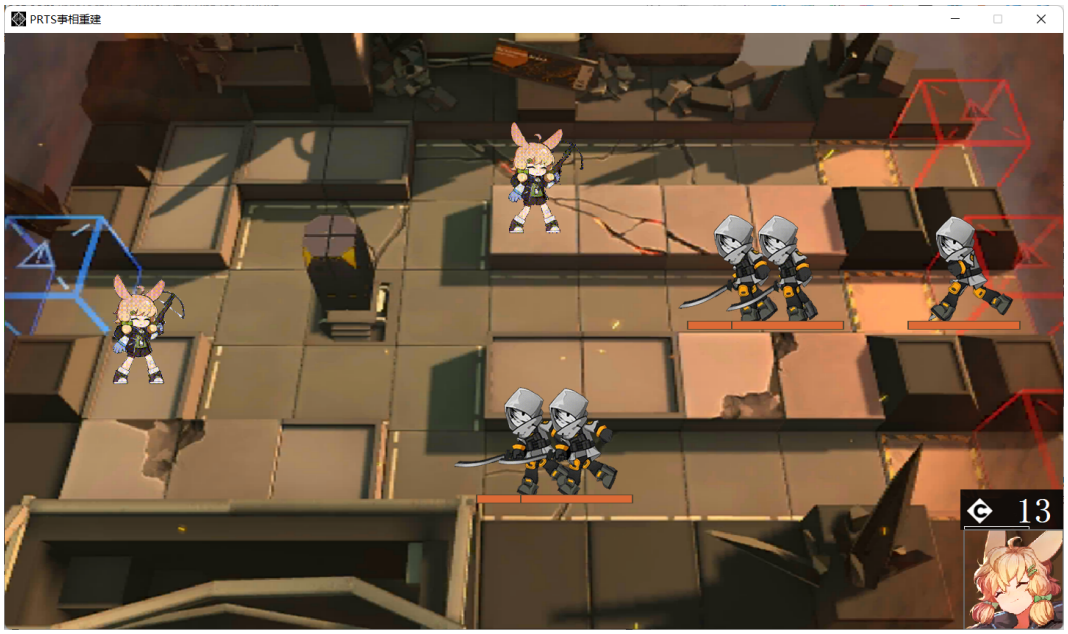

图 4: 关卡界面

该关卡界面有以下元素：

- 红色目标点为敌人出生点，蓝色目标点为我方保护点
- 右下角头像上方为部署费用，初始 10 点，每秒上升 1 点，上限为 99 点
- 右下角头像为我方干员，长按拖出人物模型并在高台放下即可部署，需要消耗 10 点部署费用；不足 10 点费用将无法拖出，不在高台松开鼠标则无法部署，鼠标经过可部署高台则人物模型会被吸附
- 部署后的我方干员以自身为基准，可对上中下 3 行、右侧 4 列，总计 3x4 的范围内敌人进行单体攻击，发射白色线状箭矢，飞至敌人即会造成伤害
- 部署后的我方干员优先攻击最先出现的敌人，无法撤退，不会被攻击
- 关卡出怪流程固定，敌人仅有生命值和行动路线的差异，突袭模式费用增加速度减半
- 敌人全部死亡视为任务成功；敌人进入蓝色目标点即视为任务失败
- 任务成功/失败后，在结算界面等待 3 秒即可回到主菜单界面

二　具体实现

2.1　技术与环境概述

- 操作系统：Windows11
- 语言标准：C++11，Qt5.9
- 集成环境 (IDE)：Qt Creator 4.3.0 (Community)
- 文本编辑：Visual Studio Code
- 编译环境：g++ 7.3.0
- 图片处理：Photoshop 2021

2.2　项目目录结构

本项目的所有源代码、资源文件、配置文件，均在文件夹 PrtsEventReconstruct 之中（下称根目录），依托 Qt Creator 进行文件和资源的管理，并用 Qt Creator 编写和运行项目，用 VS Code 编写 Json 配置文件。该项目的 *.cpp, *.h 等源文件和 *.pro, *.qrc 等项目配置文件，均在根目录下。关卡和敌人配置信息 *.json，音乐素材 *.mp3，图片素材 *.png, *.gif 等，均在根目录下对应的文件夹当中。文件夹结构如下图所示：

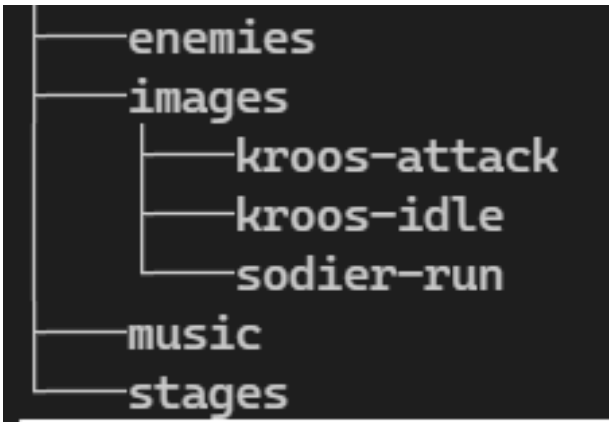

图 5: 文件夹结构图 (未展示文件)

- main.cpp：主程序，用于开启 Qt 应用，打开主窗口
- mainwindow.*：主窗口，主要进行页面间切换，以及切换时的资源管理
- page*：页面，管理一个界面的逻辑、资源，在主窗口中展示
  - pagewelcome.*：欢迎界面，用户打开游戏、进入游戏前的界面

  - pagemenu.*：菜单界面，用户进入游戏，进行选择的主要界面
  - pagestage.*：关卡界面，用户开始战斗，进行操作的关卡界面

- 其他：关卡中组件，由关卡页面集中管理，各自包含对应的数据和操作
  - grid.*：方格，为干员部署和敌人移动的单位
  - operator.*：干员，为关卡中我方可部署、可攻击的己方单位
  - enemy.*：敌人，为关卡中自动出现、可被攻击的地方单位
  - bullet.*：子弹，为干员发出、飞向敌人，并能在接触时造成伤害

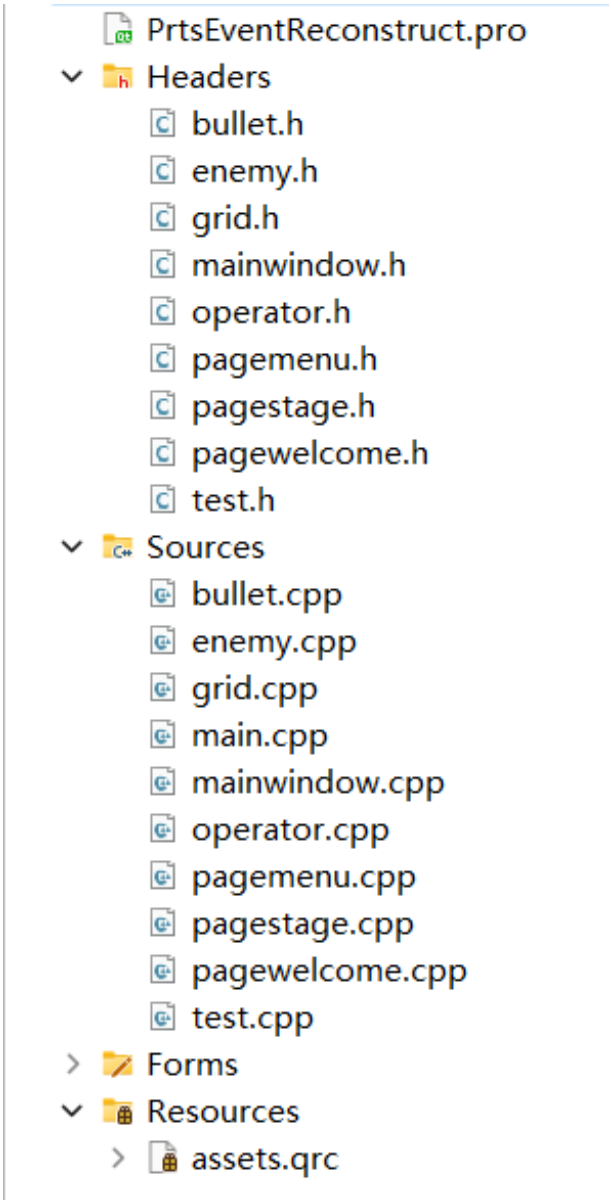

图 6: 源文件和项目文件逻辑结构

上图为 Qt Creator 浏览下，根目录文件的逻辑结构。每一对.h 头文件和.cpp 源文件，都分别代表一个类的声明和实现，具体代表内容在上面已经叙述。根目录下的.pro 和.pro.user 文件为 Qt 项目组织文件，可以无视；.qrc 文件为 Qt 资源管理文件，指明资源文件的列表和路径；*.ui 文件为相应的界面文件，可以用于可视化拖放部件，在本项目中未使用。

需要注意，如果使用 Qt Creator 的 Debug 或 Release 模式编译运行本项目，则会在项目根目录并列的文件夹出生成 build 文件夹，内含运行所需文件和可执行文件，稍作处理即可打包发行。

限于篇幅，下面各部分的实现仅对上述各类进行简要介绍，尽量不给出具体代码。

2.3 游戏页面实现

此部分仅介绍页面的布局逻辑、绘制逻辑、切换逻辑、交互逻辑。

2.3.1 页面切换

如2.2节所述，游戏中不同的界面均由不同的界面类实现，这些界面类均继承于 QWidget 类，亦即可进行 Qt 操作的图像组件。游戏的主窗口即 MainWindow 类，其继承于 QMainWindow，作为整个应用的主窗口，管理相应的页面对象和全局资源对象（如背景音乐）。主窗口在应用运行时便创建。

```
public slots:
    void showWelcome();
    void showMenu();
    void showStage1();
    void showStage2();
    void showStage(bool isHard);

private:
    Ui::MainWindow *ui;
    PageWelcome *pageWelcome = nullptr;
    PageMenu *pageMenu = nullptr;
    PageStage *pageStage = nullptr;
    QMediaPlayer *mplayer = nullptr;
    QMediaPlaylist *mlist = nullptr;
```

图 7: 主窗口类 MainWindow 所管理的对象和槽函数

此处主要用 Qt 的**信号与槽**机制实现页面切换的流程。当要引起切换的事件发生时（如点击按钮、关卡结束），对应发生事件的组件会发出信号 signal，然后由主窗口的槽函数 slot 接收，并调用槽函数进行处理。

而要实现页面的切换，仅需在槽函数中，并调用 setCentralWidget(page) 将上述管理的 Page 页面设置为窗口中间组件，并在此前后进行相关资源的控制（背景音乐切换、页面资源加载），即可完成页面的切换。

2.3.2 欢迎页面

如1.2节所述，欢迎界面较为简单，可交互部分仅有一个按钮，其他部分均为静态的标签和贴图，因此也是采用了 Layout 布局来保证一定的可扩展性。即：页面整体应用垂直布局分为上中下三部分，上下部分用纯色标签挑中。中间部分又应用垂直布局，分别加入图标、标题的图片和按钮对象。

创建的按钮需要设置常态图片和按下图片，然后在 MainWindow 中使用 connect 函数，绑定该按钮的按下信号和窗口的页面切换槽函数。其中，按钮的按下信号为 Qt 的 QPushButton 类自带，窗口的切换槽函数为自己实现，已在2.3.1节中叙述过。

2.3.3 菜单页面

如1.3节所述，菜单界面也较简单，主要为一张背景图、三个按钮、一张帮助图片。由于 Layout 布局一定程度上的不可控性，此处及之后均采用绝对位置布局，即调用 move 函数指定部件的横纵坐标值。

两个关卡难度的按钮同样由 MainWindow 进行 connect，均间接调用关卡页面的 loadStage 函数，通过参数区分普通模式和突袭模式。帮助按钮则绑定该页面自身的槽函数，控制帮助图片的显示和隐藏。

2.3.4 关卡页面

受项目结构的影响，关卡战斗的逻辑和关卡界面的布局均放在该关卡页面中。为了保证逻辑清晰，**此处仅介绍关卡页面的布局和交互相关**，关卡战斗逻辑请参见2.5节。

布局绘制逻辑 该页面的所有部件均采用绝对布局。每次循环均会重新绘制画面：

- 背景图片：每次循环最先绘制以放在底层，铺满整个页面
- 费用变化：先将图片绘制到右下角，然后再绘制相应的费用数字；费用进度条根据当前计数数值与最大数值计算得到（如每 60 帧增加 1 点费用，当前为第 20 帧，则费用进度条仅绘制 1/3 的长度）（该费用进度条规则也适用于血条）
- 角色动作：每数帧为动画计数器加 1，同时绘制对应的图片即可实现动画（干员和敌人均适用）；干员攻击/待机状态切换需将动画计数器置 0，并切换为对应状态的图片；干员攻击状态下，检测扣下扳机那一帧并发射子弹
- 子弹移动：在子弹对象的起点和终点间绘制直线，子弹位置由逻辑部分维护

拖放交互逻辑 该页面的交互部分主要是拖放干员，整个拖放过程由以下 Qt 自带事件维护：

- 鼠标按下事件（mousePressEvent）：鼠标按下时触发。检测按下的位置是否为右下角干员头像：如是，则将图片和位置信息传递给 QMimeData 用于拖动事件的数据传递，并产生一个拖动事件
- 拖动移动事件（dragMoveEvent）：拖动事件产生后，长按拖动时触发。检测鼠标当前的位置是否在可部署高台上：如否，则在鼠标位置绘制干员模型；如是，则不在鼠标处绘制，而在对应高台上绘制，代表干员被吸附
- 拖放放下事件（dropEvent）：拖动事件产生后，长按松开时触发。检测鼠标当前的位置是否在可部署高台上：如是，则在对应高台上生成干员对象

2.4 游戏组件实现

此部分简要介绍关卡战斗逻辑的类组件，与布局和绘制关系不大。

2.4.1 方格

在本塔防游戏当中，地图以方格作为干员部署和敌人移动的基本单位。由于本项目无法做到 3D 立体、透视、不规则边缘等，因此只能手动标注地图中的方格地块，且这样标注的方格不是无缝的，也无法保证完全对齐，但能通过测量像素基本保证位置正确。

在方格 Grid 类中，保存了该方格在地图中的行列信息、中心位置的横纵像素坐标、是否为高台或地面、是否可部署等信息。这些数据成员方便了关卡中的逻辑控制，关卡页面中方格的数组 (Vector) 就是对地图的抽象。

在方格 Grid 类中，也提供了一些公用的函数接口，主要是改变可部署状态的 deploy 和 retreat 函数；同时也提供了从 Json 对象中读取并设置自身信息的 loadJson 函数，用于在地图初始化时，读取 Json 并对方格信息进行设置。

2.4.2 干员

干员为可部署的我方单位，在本项目中仅实现了一个远程单体攻击单位，且将数值和攻击逻辑等写死在了代码中。

在干员 Operator 类中，保存了图片渲染的坐标、中心判定坐标、当前状态、攻击目标、帧数计数器、图片组等。坐标、图片等是控制干员的动作和图像显示，由关卡页面进行绘制；状态、目标等是控制干员的攻击和待机逻辑，用于完成相应指令。

在干员 Operator 类中，提供了更新图片、帧数计数器和状态的 updateSelf 函数，用于逻辑控制和图像控制；也提供了 doAttack 信号，在进行攻击时发出该信号，通知关卡界

面进行攻击，并产生相应的子弹对象。由于仅实现了一种干员，因此数值直接写死，没有 loadJson 等读取并设置配置的函数。

2.4.3 敌人

敌人为可移动、可被攻击的敌方单位。因本项目没有制作敌人远程攻击单位，也没有近战友方单位，因此敌人没有攻击相关参数和行为。此外，由于每关敌人数量和流程固定、总数量较少，因此为了便利一次性创建了全部敌人，通过设置敌人的状态控制其是否可被选中攻击、移动等。

在敌人 Enemy 类中。保存了图片坐标、中心判定坐标、帧计数器、图片组、血条对象等用于绘制和图像控制的信息，也保存了出场时间、移动速度、当前和最大生命值、当前状态等信息，用于游戏中的逻辑控制。

在敌人 Enemy 类中，同样提供了 updateSelf 进行图像和逻辑控制，提供了 loadJson 进行 Json 对象的数据读取与设置，其逻辑和行为和其他组件类相似，不再赘述。

2.4.4 子弹

子弹为友方干员射出、飞向敌人的中间单位，有飞行过程、有绘制逻辑，也有碰撞时造成伤害的逻辑，因此单独独立出一个类来控制。同样由于本项目仅设计了一个友方干员，也仅有一种子弹，所以没有设置参数和加载图片。

在子弹 Bullet 类中，保存了子弹绘制起点坐标、子弹绘制终点坐标对绘制进行控制，也保存了攻击目标，用于实时获取目标的位置，进行飞行方向的控制。类中同样也保存了伤害值、飞行速度等逻辑参数，此处是写死在代码中。

在子弹 Bullet 类中，仅提供了 updateSelf 作为更新接口。在该函数中，通过攻击目标 target 获取目标坐标，然后让子弹的起点和终点均向目标坐标移动，并在碰撞时设置子弹状态。实际的绘制操作、状态对应的伤害造成，交给关卡界面进行处理。

2.5 游戏逻辑实现

此部分简要介绍关卡界面中游戏的运行逻辑。

2.5.1 关卡初始化

关卡初始化大致可分为三部分：图片等资源的加载、配置文件的读取、关卡循环的开启。

在关卡界面创建时（即构造函数中），便进行图片等资源的加载，以及计时器的创建。加载的图片资源主要是干员、敌人移动和攻击动画的各帧图片，从资源文件中读取并保存

到 QPixmap 中，之后再将这些保存的图片传递给对应干员或敌人对象（图片对象的拷贝不耗时，图片文件的加载才耗时）。计时器的创建主要工作位：设置计时器触发时间间隔，此处锁定 1 秒 60 帧；然后绑定计时器触发事件，此处是2.5.2节中将介绍的 updateStage 函数。

在关卡界面创建后，还需要调用 loadStage 函数读取指定关卡的配置文件。读取配置文件分为两部分：读取地图信息和敌人信息。此处使用的配置文件均为 Json 文件格式，使用 Qt 的 QJsonDocument 类族的方式读取，不做过多叙述。读取配置文件的过程中，分别会调用方格对象 Grid 和敌人 Enemy 对象的 loadJson 函数，在解析出 Json 对象后分别交由各个对象自己进行参数和配置的设置。

和构造函数中初始化不同的是，构造函数中初始化的是普遍的资源，如各个关卡中都需使用角色图片资源、关卡计时器等。读取配置时读取的是特定关卡的配置文件，取决于调用时传递的参数，通过参数实现可复用机制。

2.5.2 关卡主循环

在资源加载完毕、参数设定完成之后，游戏的进程就可以开始了。关卡通过 QTimer 锁定 60 帧，在每一帧分别进行逻辑上的更新和画面上的更新。

逻辑更新操作，主要依靠计时器触发函数 updateStage。关卡界面维护了干员数组、敌人数组、子弹数组，在 updateStage 中，关卡将会遍历这些数组的每个元素，并依次调用它们的 updateSelf 函数以更新它们的状态。

干员对敌人的攻击交互放在了干员的 updateSelf 函数中，干员的攻击范围由其中心坐标直接计算得到。在更新时，干员会遍历敌人的数组，并找到首个在其攻击范围内的敌人，并且进入攻击状态，不断更新并发射子弹。直到攻击范围内没有敌人，则进入待机状态。

敌人移动与路径点选取放在了敌人的 updateSelf 函数中，每个敌人内存放了路径点数组，一直向着下一个路径点移动。如果和下一个路径点发生了碰撞，则更新下个路径点，并朝新路径点的方向前进。

子弹和敌人的碰撞判定放在了子弹的 updateSelf 函数中，如果子弹和敌人碰撞则将状态 isDone 置为 true。交互效果就在 updateStage 函数中，如果检测到子弹状态 isDone 为 true，则对子弹的目标敌人造成伤害，并销毁子弹。

部署费用的更新同样放在 updateStage 中，每一次调用就会使费用计数器加 1，如果计数器的值到达最大值（普通模式为 60，突袭模式为 120，即每 1 点费用的回复时间分别为 1 秒和 2 秒），则会将计数器归零，并增加 1 点部署费用。

画面更新操作，依靠 updateStage 函数调用 update 函数，进而发生 paintEvent 绘图事件进行画面的重新绘制。干员、敌人、血条的贴图由各对象自主完成，子弹由于需要进行绘画所以在此处完成。整个绘图事件只需要绘制：背景图片、角色头像、费用及进度条、

子弹。绘制主要是进行贴图或者直线绘制，不再赘述。

2.5.3 关卡结束判定

关卡有两种结束方式：成功或失败。暂停并退出关卡或者关闭游戏不在讨论范围内。

每次循环都会判定已死亡的敌人人数，如果敌人已全部死亡，则成功并调用 endStage (true)。如果有敌人进入蓝色目标点，亦即没有下一个目标点，则失败并调用 endStage(false)。在函数 endStage 中，会根据参数展示胜利或失败的图片，并在 3 秒后进行界面切换。

至此，游戏的页面框架、逻辑链条已经完整构建完毕。

2.6 项目代码统计

项目完成后，使用 VSCode 的 VSCode Counter 插件对代码量进行了统计，统计结果如下图所示。

language	files	code	comment	blank	total	备注
JSON	4	1,656	0	5	1,661	地图与敌人配置文件
C++	9	765	173	110	1,048	类的实现
CUDA C++	8	313	13	79	405	类的声明
Markdown	1	10	0	1	11	部分说明文档
Python	1	6	14	4	24	素材文件批量重命名

图 8: 行数统计

可以看到，Json 配置文件的行数占了大头，但重复性内容居多。代码工作量主要集中于 C++ 头文件和源文件中，总计有 1500 行以上。当然，借助于 Qt 的绘图机制、计时器、事件处理、信号与槽等特性，一定程度上减小了代码的工作量。

2.7 游戏素材获取

本项目的原素材主要来自于网站 PRTS、明日方舟游戏解包、网络资源下载、自制等。获取素材后，往往需要经过较大程度上的更改和修饰操作，主要通过 Photoshop 完成；部分素材转化工作使用在线网站 Convertio。相应的链接参见4.1节。下面以友方干员的攻击动作为例，展示部分素材处理流程。

第一步 在 PRTS（明日方舟非官方 WIKI 网站）上下载指定角色、指定模型、指定动作的动画，格式为导出的.webm 格式。这一步导出的动画背景不透明、背景有杂色、动画不完整、格式不好用，需要进一步的处理。

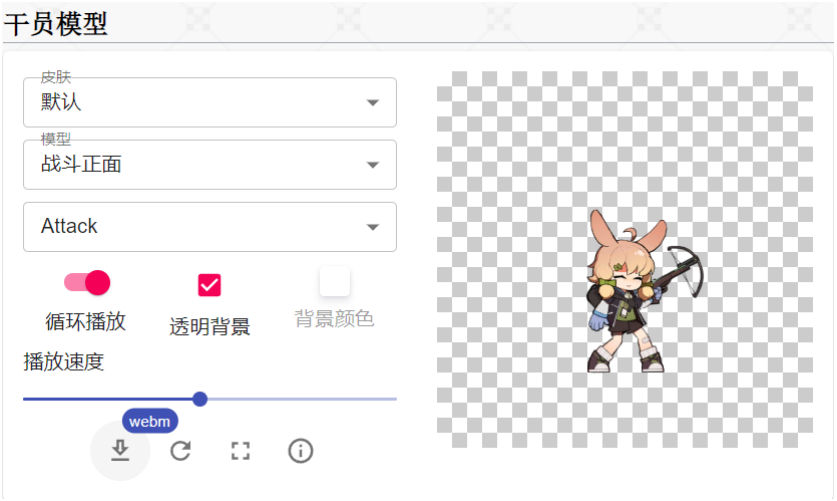

图 9: 模型下载

第二步 在 Convertio（在线格式转换网站）上，将获取的.webm 格式文件转化为.gif 文件，下载并保存。此处使用该网站的免费额度进行转换，暂不清楚何种规模的文件会导致付费。

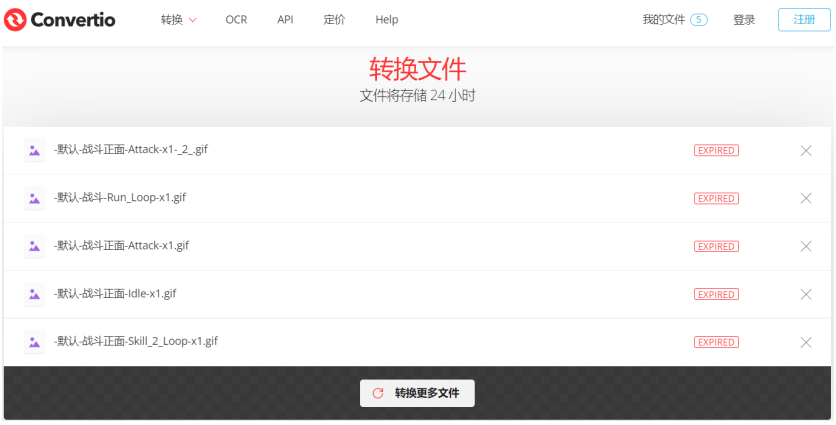

图 10: 文件转换

第三步 在 Photoshop 打开上述转换后的 gif 文件，将图层导出为 png 文件。随后编写 Python 脚本，对导出的 png 文件进行批量重命名，并保证文件的顺序不变。

图 11: 导出图层

第四步 在 Photoshop 对导出的 png 文件进行处理，使用批处理对各图片进行裁剪，然后将背景替换为透明背景。由于背景噪点的存在，此步较为繁杂且无法用批处理。

图 12: 图批处理

第五步 根据动画中缺失的部分，使用前面的帧进行补帧。例如，下载的动画仅有放下弓弩的动作，则选取该动作的几帧倒放，作为抬起弓弩的动作，从而完成完整的动画。

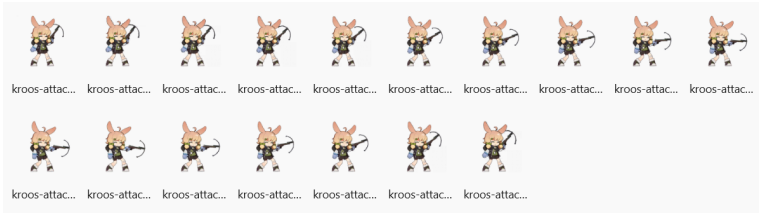

图 13: 攻击图片

三　开发感悟

本次开发过程较为漫长，涉及知识较多，整体项目完成度较高，也算是小小地完成了目标，也因此感悟颇多。下面将从开发中的困难、开发时的收获、开发后的目标这几个方面展开叙述，并在最后抒发一些感想。

3.1　昨日之失

总体来说，开发过程中的困难是多方面的。从开始前素材的处理，到整体类关系的架构；从 Qt 整体框架的学习和使用，到各部分功能的具体实现。

项目准备阶段的困难不占少数。项目大致方向的确定很快，但具体项目的实现功能和完成度却难以抉择。而当这些都考虑完毕后，素材收集和处理又成了一大难题。正如2.7节所说，哪怕是单一一个动画的处理流程已经足够复杂，更不需要说整个项目完整的素材处理了。流程摸索过程中解决方法的尝试，也耗费了我大量时间。素材来源的获取难以抉择，素材的处理工作同样绞尽脑汁。最终在精致度和工作量之间做了妥协，也还算是基本完成目标。Json 配置文件的编写则是大量重复工作居多，不断复制粘贴并修改那几个常量值。

项目开荒阶段同样困难重重。初始目标定得很高的我，放弃了使用课上所用的 EasyX 图形库，而是选用了看似更高级的 Qt。其间数周的资料查询、学习过程暂且不谈，缺乏实践经验的开发过程也足够令人折磨。例如，看似简单的页面切换功能，不同于只进行图片绘制的 EasyX 只需要状态机控制绘图即可完成，面向对象的 Qt 需要审视各对象之间的所属和呈现关系。又如，开始我想借助 Qt 的图形视图框架（QGraphics*）简化碰撞等逻辑判定，却直接在布局上无从下手，最后被迫将项目全部迁移到朴素的 Qt 图形框架，然后使用绝对位置绘图定位。但即便如此，绝对位置的测量也很费时间，只能用截图功能测量标记像素点的位置。原作明日方舟是 Unity 开发，可以利用建模 + 透视完成地图，物理引擎实现功能；然而这个游戏还只能地图贴图 + 手测方格，利用像素级操作完成功能，令人感叹。

项目扩展阶段相对平和。项目的主体代码框架确定后，各功能的实现至少没有出现完全无从下手的困难，基本上通过查阅文档、翻阅书籍、求助论坛便可解决。例如，鼠标拖放功能虽然棘手，开发和测试时间也较长，但跟着书上的指示也不算太困难，有了困难面向百度也算是基本解决。但略显遗憾的是，有一个问题至今没有得到解决。当拖放干员时，如果鼠标低速或不移动，则游戏主循环会被阻塞，运行速度及其缓慢，1 秒仅有 2 到 3 帧。这个问题的出现几乎莫名其妙，也是多方求助无果，只能当做游戏的特性“子弹时间”了。

3.2 今日之拾

历经一番独自开发的洗礼（折磨），不能说毫无收获，只能说各方面均有提升。至少看完的书、敲的 1500 行源代码、能运行的游戏，都证明着这不是无用功。如果说开发就像海边的沙滩，那么 ddl 则是呼啸的风暴和海浪，而我只能匍匐着小心前进。不过风平浪静之后沙滩上的贝壳，还是值得拾捡起来，收藏品味一番的。

素材处理和资料收集上，也算是小小地提升了我的一点能力。以后查资料、看书籍，会略微得心应手吧。gif 和 png 的处理，Photoshop 的一部分使用，说不定以后可以用来整个好活，修修图改改画什么的，至少不算是 0 基础了吧。

代码能力上，也还算是有所提升。面向对象的 Qt 让我更加熟悉了指针的使用和类的使用，也让我在图形界面的处理上有一点领会。不过虽然用 Qt 摆脱了部分造轮子的命运，但在别人限定的框架下（指 Qt）行事也比较难受。Qt 难能可贵地让我习惯了翻官方文档，一是 Qt Creator 自带离线文档且翻阅方便，二是 Qt 东西实在太多，网上找着又乱又杂还麻烦。虽然说 Qt 现在的讨论度和应用面不是很多，但好歹也是熟悉了 C++ 和图形框架吧，说不定后面看 DirectX 和 Unity 能触类旁通一点呢?

开发流程上，可以说是稍微体验了一番完整流程吧。项目立项、资源准备、结构搭建、功能开发、文档编写、应用发行、运营维护，多少算是都涉及一点。只是没有那么多项目经理让改代码，也没有甲方改需求，也不用和别的程序员搞协作，总体体验上还算好。文档编写有点费劲，不过写到这里应该也快写完了。（看目录就知道确实快写完了）

3.3 明日之始

我不想说“重铸未来，方舟起航”之类的应援词，太中二了。虽然这章的标题也好不到哪去。总之还是展望一下项目的拓展方向。

与其说是游戏的功能拓展，不如说是高开低走最后没能实现的功能部分。想添加近战干员，可以部署在地面方格（当前只有能部署在高台的远程干员）。随之而来的，就是敌人和近战干员的阻挡机制、敌人的攻击机制、我方干员的生命值、不同干员的攻击范围和攻击方式差异，等等。想添加技能机制，技能充能完毕后可以释放并产生特殊效果。随之而来的，还有技力值、技能状态、手动释放与自动释放、自动回复与攻击回复、游戏内技能使用界面，等等。仅仅是添加一小点的新功能，就引入了这么多未实现的功能，当初没能实现的原因很明确了。（写的不错，下次不要再写了）

引入了敌人攻击机制和技能机制，就可以制作 Boss 关卡了，还可以加入更多的友方角色，激动人心。但想到要重新抠素材、量像素、写配置文件，还是算了吧。复杂的功能何不交给更复杂的框架呢? 至于暂停功能、二倍速、中途退出，以及一些配套 UI 的添加，只是单纯没时间写了。（摸了）

3.4 未竟之事

想说的有很多，但最后还是删了。只是一点吐槽和碎碎念。（怎么感觉这一章都是博客般的吐槽）

中国游戏行业环境并不算好，至少说不适合真正的理想主义者、真正热爱中国游戏事业的人。被现实溺毙的理想主义者毛星云便可以说明很多事情。或许大环境我们个人无法改变，或许改变的未来遥不可期，但我也真诚期待有那么一天。

我们都还年轻，该做的梦及时去做，该追寻的梦想用力去追寻。

我有一个梦想，将来的某一天，大家都能玩到拥有自己本土文化的优质游戏。

我有一个梦想，有一天，西游记能出 ACT，让老外去体会中国文化西游记中“斗战胜佛”的打击快感，那一定比西方的动作巅峰之作《战神》、《鬼泣》更加深邃。

我有一个梦想，有一天，上海滩能出沙盒游戏，而不是玩《GTA》感受美国梦，亦或是玩着《热血无赖》体验国外公司强行塞给我们的“中国文化”。

我有一个梦想，有一天，不少 3A 大作不需要汉化，因为是我们自己的游戏，配音是中国的，文化也是中国的。

我有一个梦想，将来的某一天，国产游戏能像中国的其他产业一样，以一个领跑者的姿态，面对全世界，面对全宇宙，器宇轩昂，扬眉吐气。

这会是由我们一起去完成的梦想。

等着我们的好消息！

图 14: 毛星云书

四 相关索引

4.1 参考资料与相关链接

[1] 参考游戏：明日方舟官网链接

[2] 素材获取：同人 Wiki 网站 PRTS 链接

[3] 格式转换：在线格式转换 Convertio 链接

[4] Qt 下载：Qt 官网 5.9.0 版本下载链接

[5] Qt 学习：《Qt Creator 快速入门》第三版

[6] 代码仓库：本项目 Github 仓库链接

4.2 运行方法

4.2.1 源代码编译与运行方法

1. 安装 Qt 配套环境，参见4.1节
2. 从 Github 拷贝仓库，参见4.1节
3. 使用 Qt Creator 打开文件夹，打开项目 PrtsEventReconstruct.pro
4. 在 Qt Creator 左下角选取 Debug 或 Release 模式编译运行
5. (该 Qt 源码应当在各系统下均能编译运行)

4.2.2 发行版应用运行方法

1. 进入 Github 仓库，下载右侧 Release 的压缩包（非源码包）
2. 在任意地方解压源码包，进入文件夹
3. 双击运行 PrtsEventReconstruct.exe 即可
4. (该发行版仅在 Windows 下可使用)