哈尔滨工业大学（深圳）

面向对象的软件构造实践
课程报告

学生班级：计算机类 4 班

学生学号：200110428

学生姓名：杨杰睿

评阅教师：

报告成绩：

实验与创新实践教育中心制

2022 年 4 月

1. 面向对象分析

1.1 需求模型

需求分析：

1. 选择单机游戏或者联机游戏；
2. 选择游戏难度；
3. 游戏中通过滑动屏幕移动英雄机对应的距离（无需在英雄机图标区域内触控）
4. 游戏中显示敌机血条、英雄机血条和英雄机道具有效蓝条
5. 游戏中英雄机可碰撞道具，然后道具生效
6. 游戏中可在护盾保护下撞击敌机，并扣除一定护盾值
7. 联机模式随机匹配已经加入到同一服务器的玩家，开始游戏后同步得分

用例图：

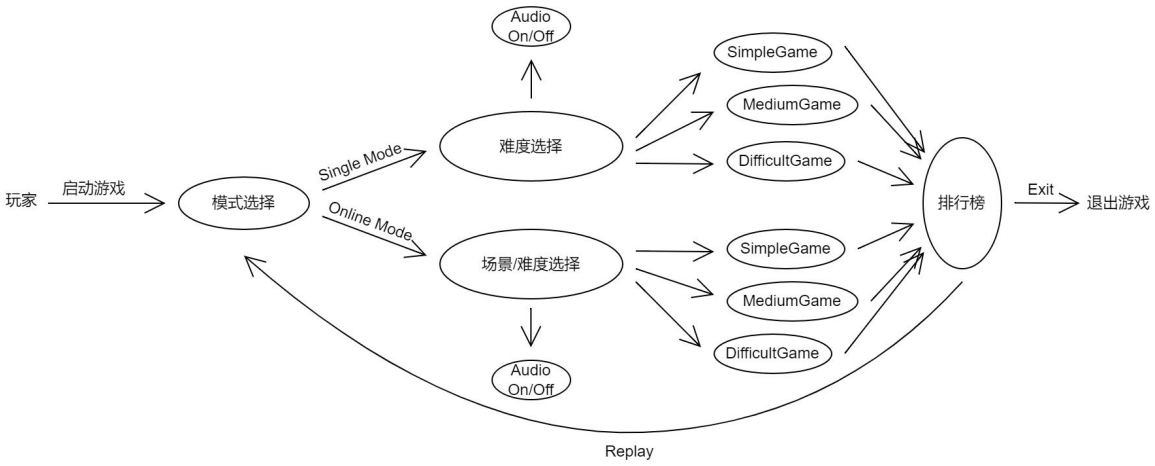


图 1：用例图，以玩家为参与者

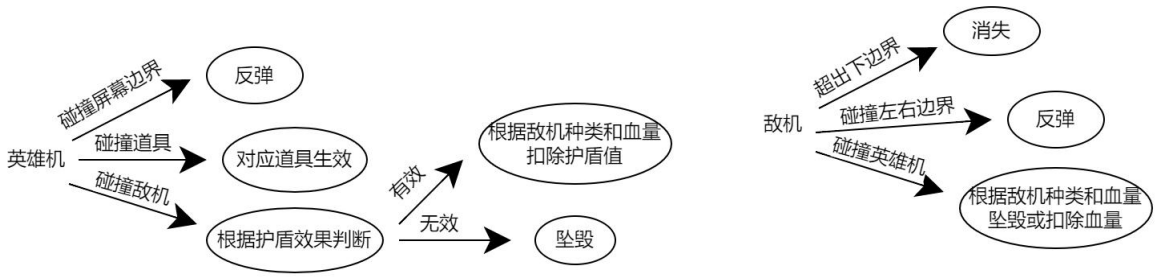


图 2：用例图，以英雄机、敌机为参与者

1.2 基本模型

与导论部分完全相同，此处仅绘制简要的继承关系图

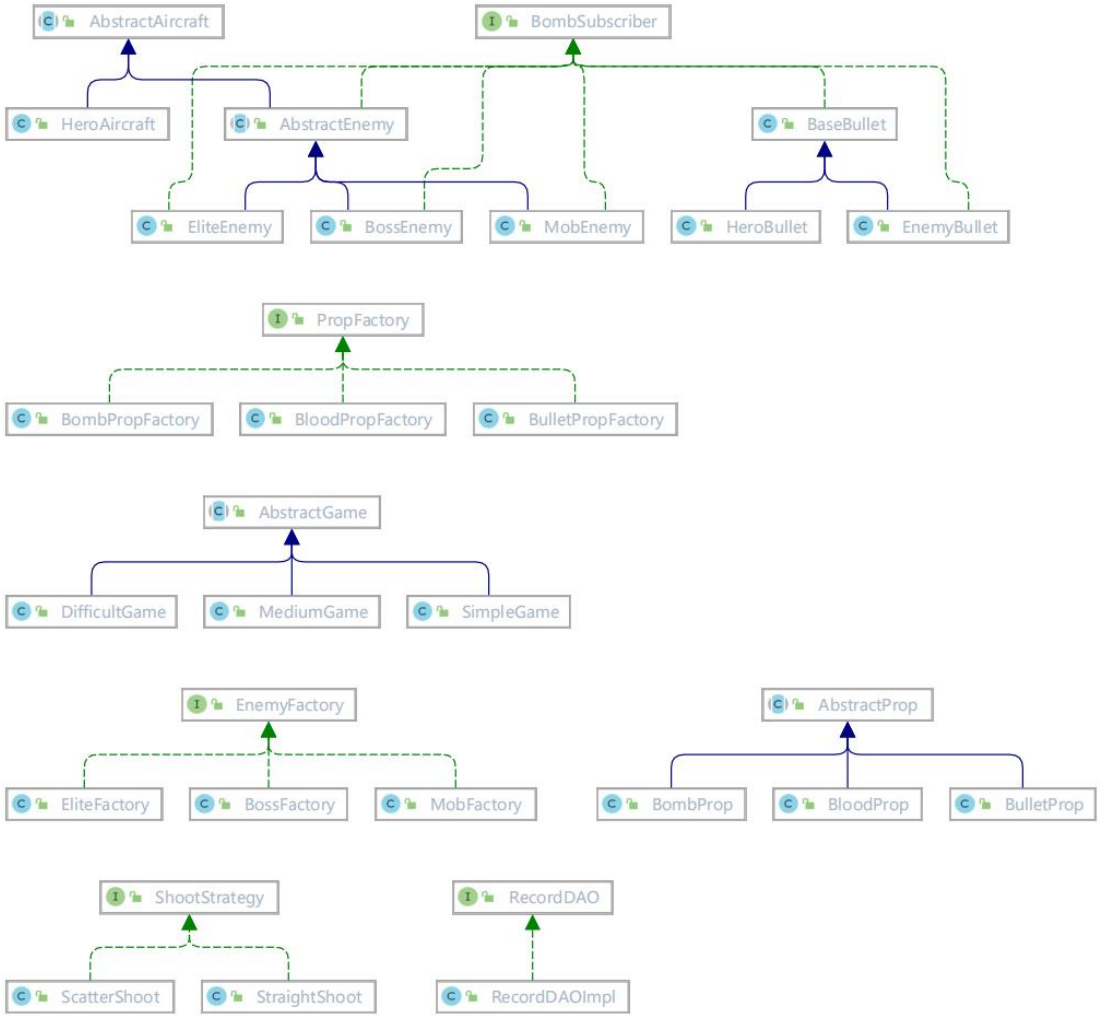


图 3：简要继承关系图

1.3 辅助模型

联机模式顺序图：

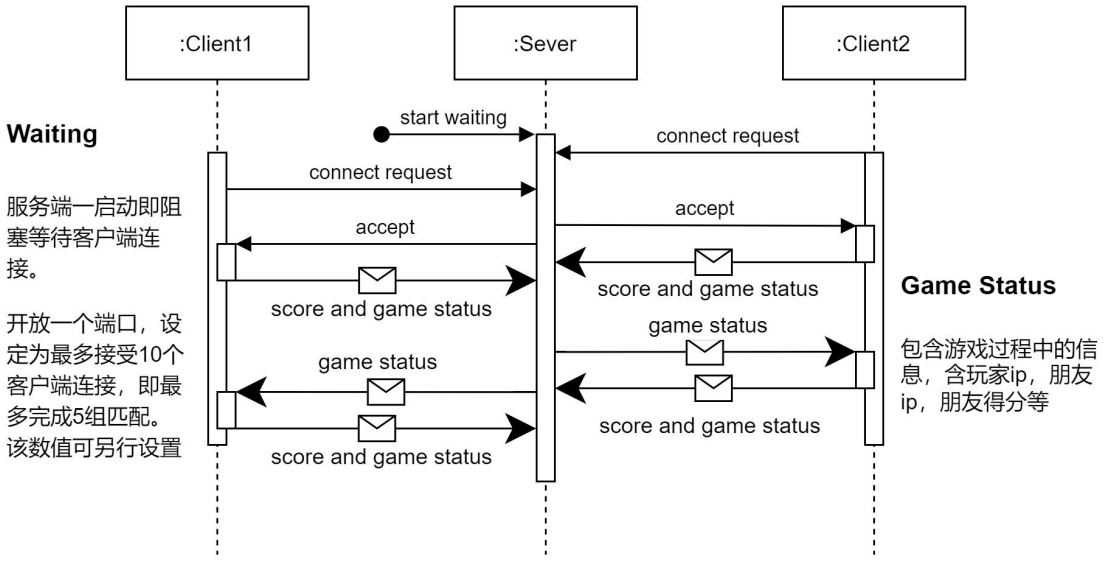


图 4：顺序图，联机模式

2 系统设计方案

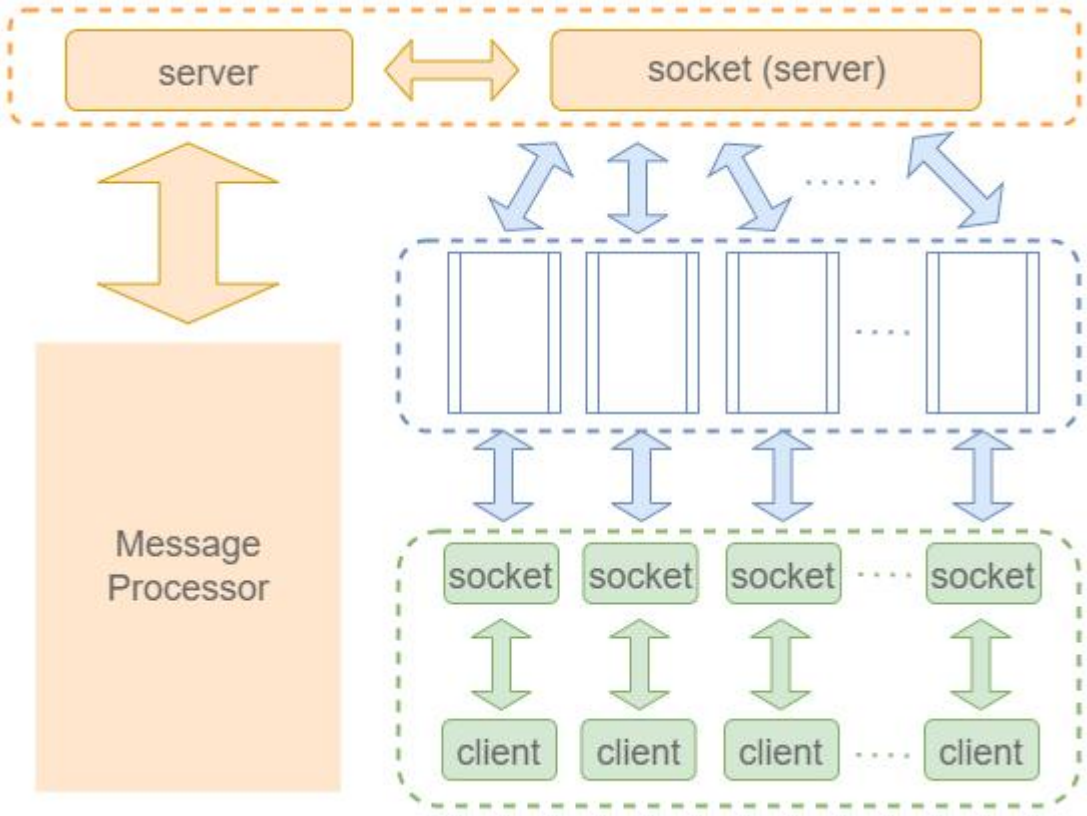


图 5：服务端、客户端架构

本系统和导论的差异主要体现在网络通信部分，网络通信使用 socket 进行。服务器为 centos9，服务端可执行文件环境为 Python 3.9.10，无预装第三方库。服务端阻塞等待客户端主动连接，在建立连接后新建线程处理客户端传输的信息，在内部处理玩家与游戏的交互逻辑，当前仅需处理将传输的字节流解码为字符串，并且解析 JSON 为字典，用于更新玩家列表中对应的玩家信息。

玩家抽象为 Player 对象，提取的字典内容通过标签添加到对应的属性中，当前因未实现登录注册功能，暂以玩家 IP 为唯一标识。

```
class Player:

    def __init__(self, ip) -> None:
        self.ip = ip
        self.fip = ""
        self.score = 0
        self.fscore = 0
        self.matched = False

    def __str__(self) -> str:
        return str(self.__dict__).replace('\'', "\"").lower()

    def __repr__(self) -> str:
        return str(self)
```

图 6：Player 类的定义，当前暂以 ip 作为唯一标识

3 调试分析

在移植绘制图像功能的过程中，遇到了一些问题，即在绘制图像时游戏直接卡住，而只保留英雄机的绘制则可正常进行。经过测试，我们发现问题在于英雄机的绘制与其他飞行物不同，直接调用了 drawBitmap 函数传入英雄机的图像，而其他飞行物如敌机，子弹，道具等则是通过调用 paintImagePositionRevised 函数，传入飞行物的列表，通过飞行物的 getImage 函数返回对应的图像再传入 drawBitmap 函数进行绘制。而要实现这个功能，需要在 ImageManager 函数中建立类与图像的映射，并存在一个 get 方法返回对应图像。在遇到该问题时我们并没有一一建立映射，所以 get 方法访问映射 CLASSNAME_IMAGE_MAP 访问了 null，导致了程序异常停止。由于只能在 AppCompatActivity 类的实例中加载资源，因此将加载资源和添加映射的代码添加在 GameActivity 中，问题得到了解决。在此前也遇到过类似访问 null 的问题，如导论内容中 bossBgm 停止时如果 bossBgm 并未创建，则对象变量指向 null，此时如果调用停止 bgm 的函数游戏则会异常终止。因此在不确定对象变量指向是否为 null 时都应在操作前加上判断是否指向 null，如果发生了这种情况就要做出相应处理。

4　系统核心功能运行结果与分析

软件主要包括页面切换，游戏逻辑，画面音效，联机对战四个功能。

页面切换部分首先进入游戏首页，选择联机模式或单机模式，此后再选择不同的难度。选择完模式和难度后就会切换游戏页面，游戏进行中绘制图像，播放音乐，游戏结束后切换至排行榜页面，选择结束游戏或重新开始。而由于跳转至排行榜时会导致游戏异常终止，因此切换排行榜功能暂未实现。

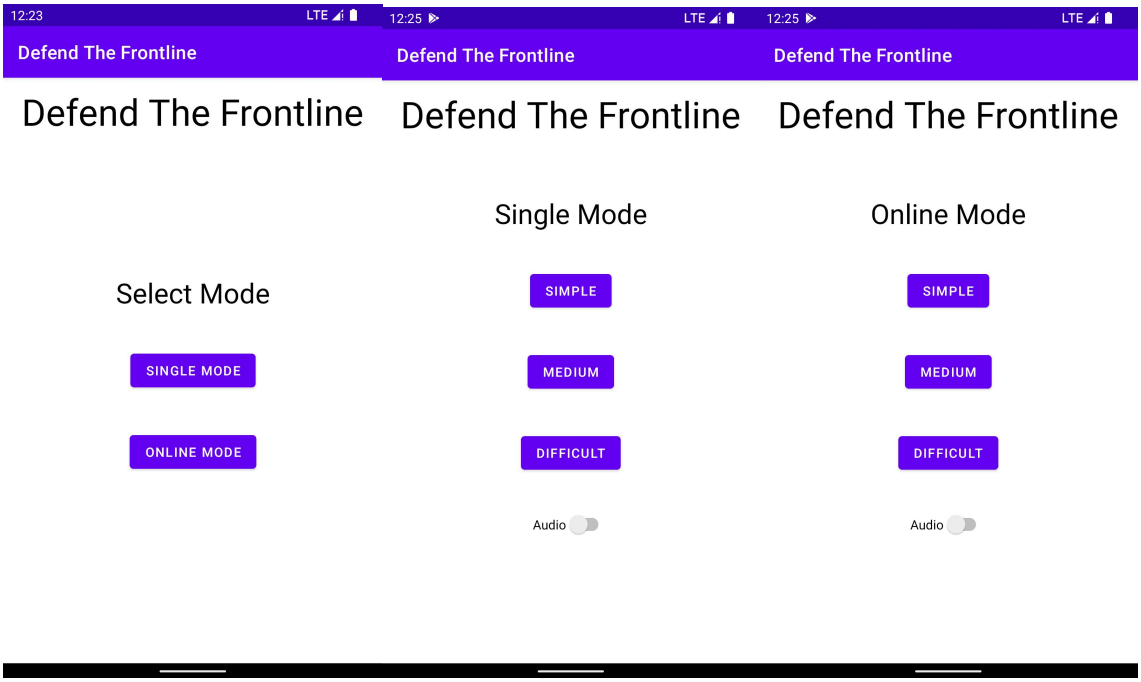
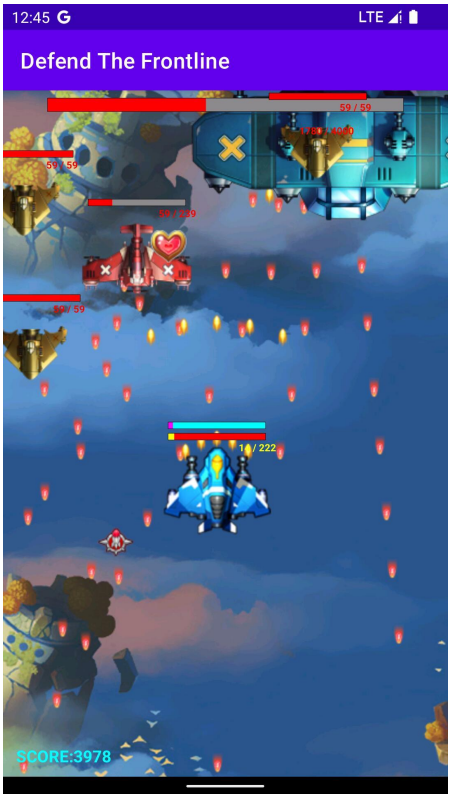


图 7：以上三图分别为初始界面、单机选择界面和联机选择界面

游戏逻辑部分直接从导论移植，包括移动，射击，碰撞等飞行物的行为，并且通过绘制画面和播放音效的功能体现。右图为单机模式下的游戏界面。玩家通过滑动屏幕控制飞机移动距离，无需直接触控飞机本体，这样避免了手指的视觉遮挡，更便于用户的使用。该功能核心位于游戏的 AbstractGame 类中，重写了 SurfaceView 的 onTouchEvent 方法。该算法参考自唐楷杰同学。



联机对战功能实现了单个服务端与多个客户端（指定连接上限为 10）通过 socket 进行通信。在建立连接之后，服务端获取客户端传输的信息，信息以 JSON 格式编码为字节流进行传递，包含用户得分、匹配状态、朋友得分等信息，因在服务端以字典方式存放，并抽象为 Player 对象，可方便的增加所需的信息种类，为之后添加传输的信息种类预留了空间。

图 8：右图为单机模式游戏画面

服务器的程序运行时，通过 IP 随机匹配建立了连接的玩家，同步匹配玩家的得分，因未实现注册功能，暂定通过 IP 唯一标识用户，此后会更改为注册方式实现对用户的唯一标识，避免同一设备多个用户的识别问题。

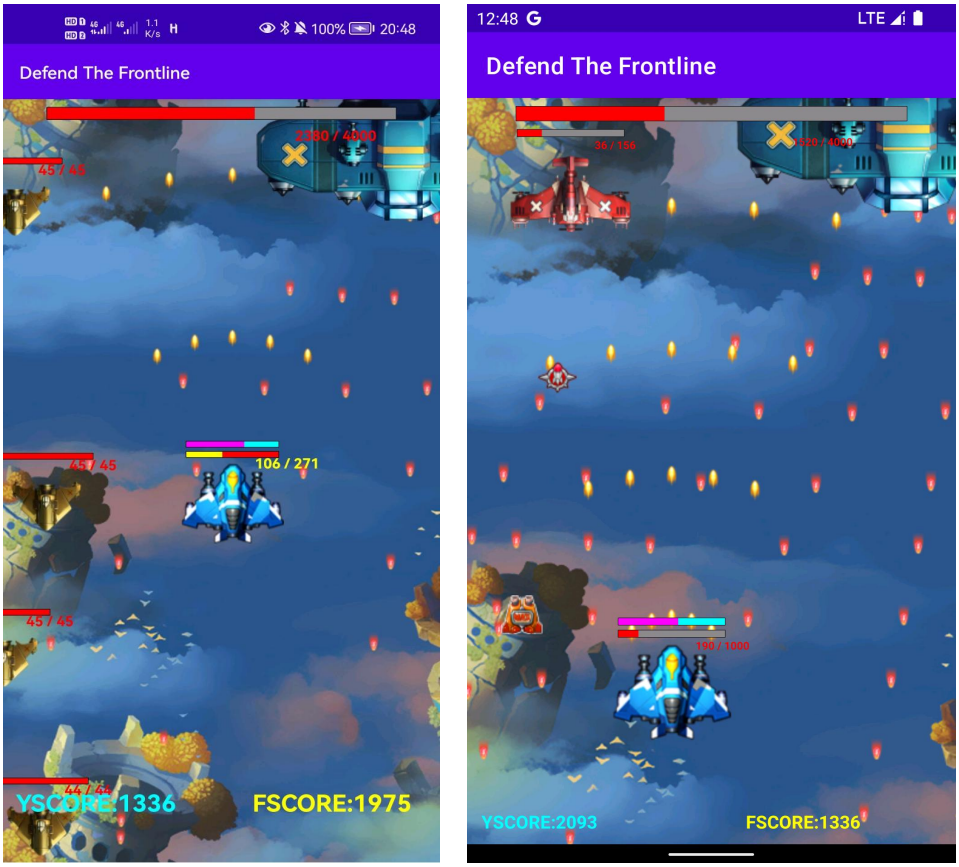


图 9：左图为实体机游戏界面，右图为模拟器运行的游戏界面，可见分数完成了同步

在截图时因手动的时延不能精准的同时获取两屏幕的界面，可能导致些许的分数差异，我们可从服务器的控制台输出查看具体的信息传输内容，如下图所示。

```
Raw data from client:
        @{"score": 2323}
now_player:
        {"ip": "58.60.1.[illegible]", "fip": "14.26.78.1[illegible]", "score": 2323, "fscore": 1381, "matched": true}
Message to client:
        @{"ip": "58.60.1.[illegible]", "fip": "14.26.78.1[illegible]", "score": 2323, "fscore": 1381, "matched": true}
Raw data from client:
        @{"score": 1381}@{"score": 1381}
now_player:
        {"ip": "14.26.78.1[illegible]", "fip": "58.60.1.[illegible]", "score": 1381, "fscore": 2323, "matched": true}
Message to client:
        @{"ip": "14.26.78.1[illegible]", "fip": "58.60.1.[illegible]", "score": 1381, "fscore": 2323, "matched": true}
```

图 10：图为服务器控制台输出

服务器在和各个客户端建立连接之后，持续的接收来自客户端的字节流，由于字节流的发送和接受没有起止分割的标记，会导致多条 JSON 信息粘合，在调试程序时解码后的字符串出现了 JSON 解析异常，本人使用无意义的字符“@”作为各条信息的分隔符，以此便于服务端切割单条信息段。

5 总结

本次安卓开发实践经历了 PC 至安卓平台的移植工作、安卓平台的游戏逻辑更新、以及最后的联机对战功能添加。总代码行数为 3234，Java 代码 2510 行，XML724 行，工作量各参半：游戏主体移植由本人负责，移植期间功能添加和逻辑修改由队友孙朔阳负责；联机对战由本人负责服务端和客户端相关代码，游戏运行逻辑重写由队友孙朔阳负责。

本次实验过程中较为疏漏的地方是对 PC 端代码未能充分利用，因服务端本应该独立运行游戏逻辑来实现同步，而非双方的安卓端各自运行逻辑，此处是因期末时间有限，为服务端配置方便，采用未安装任何第三方包的原生 Python3.9 环境，而非重新配置 Java 环境，而在服务端未使用 Java 的情况下，再重写游戏逻辑将是较为复杂的工作，故仍将游戏逻辑放在安卓端进行。

6 致谢

首先感谢俞文博同学在本人代码移植期间的倾心指导，他清晰明了的指导使得本人能快速的将导论代码的游戏主体框架移植到安卓平台。其次感谢队友在代码移植后对游戏逻辑的修改以及与我共同 debug 的过程，让安卓端游戏代码能够较完整的呈现 PC 端原游戏逻辑。与此同时，感谢老师在 qq 群提供的 demo，虽然因时间原因仅对移植部分代码进行了阅读和使用，对于后续发布的参考代码未能充分利用颇感遗憾。