**Java程序设计大作业实验报告**

**——Air Force飞机大战**

材五一 旷杰 2015011921

一 Air Force飞机大战简要介绍和操作说明

1.简要介绍

（1）游戏介绍：“Air Force”是一款双人联机的射击作战类游戏，玩家联机成功后即可操控战机协同作战。“Air Force”基于飞机大战设计，玩家共同摧毁敌机数达到20便可获得胜利，而当两架战机均被摧毁，游戏失败。

（2）角色介绍：

1. 玩家一(PLAYER1,P1)：蓝色战斗机；
2. 玩家二(PLAYER2,P2)：红色战斗机；
3. 敌机(ENERMY)：

* 绿色普通战斗机：血量1，奖励1，可发射子弹，移动随机；
* 绿色精英战斗机：血量2，奖励3，可发射子弹，移动随机；
* 自杀式战斗机：血量1，奖励1，不可发射子弹，会朝向任一玩家移动；

2.操作说明

玩家按住按键“↑，↓，←，→”进行移动，按住空格进行“SPACE”进行射击。

二 实现功能点

1. 联机和同步：玩家一和玩家二可以联机共同进行游戏，并能同步玩家双方的运动、命中判断、敌机运动、双方目前得分和共同摧毁敌机数等游戏内容；
2. 界面：实现图形用户界面且单机菜单栏相应按钮可控制游戏的开始、暂停、结束，玩家二负责开始游戏，玩家一和玩家二均可暂停和继续游戏；
3. 事件响应：玩家与界面的交互通过按键完成，如移动和开火；
4. 碰撞及出界检测：子弹和战机的碰撞、敌机和敌机的碰撞均可检测。敌机、友机、子弹出界均可检测；
5. 敌机自动控制：三种敌机的出发点随机，绿色敌机的移动随机且可自动按照固定时间发射子弹，自杀式飞机的目标随机。三种敌机的运动均自动进行；
6. 状态判断：可判断开始、暂停、继续、胜利、失败五种状态，并能实现状态的同步；

三 类和模块的设计

1.游戏相关类：

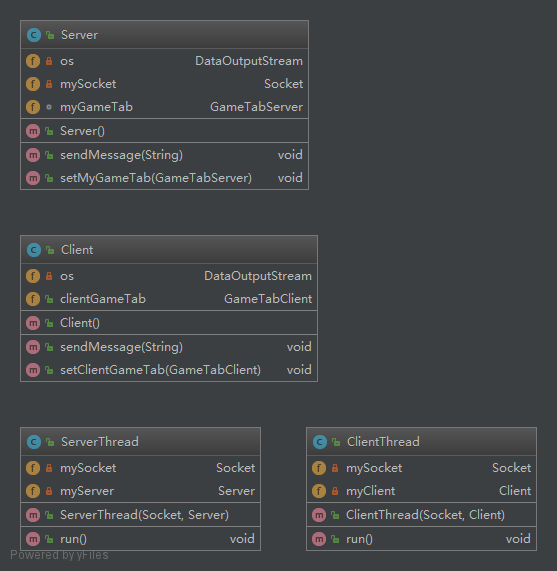
* GameTabClient：玩家一的界面、面板流程控制及显示类；
* GameTabServer：玩家二的界面、面板流程控制及显示类；
* FlyingThings：抽象类，定义飞行物的基本行为；
* MyPlane：继承FlyingThings，定义我方战机的行为；
* Enemy：抽象类，继承FlyingThings，定义敌军的基本行为；
* NormalEnemy：继承Enemy，定义敌军普通战机的行为；
* SuperEnemy：继承Enemy，定义敌军精英战机的行为；
* BoomEnemy：继承Enemy，定义敌军自杀战机的行为；
* Bullet：继承FlyingThings，定义我方战机子弹的行为
* EnemyBullet：继承FlyingThings，定义敌方战机子弹的行为；

（图1 游戏相关类关系、具体参数及方法图）

2.联机相关类：

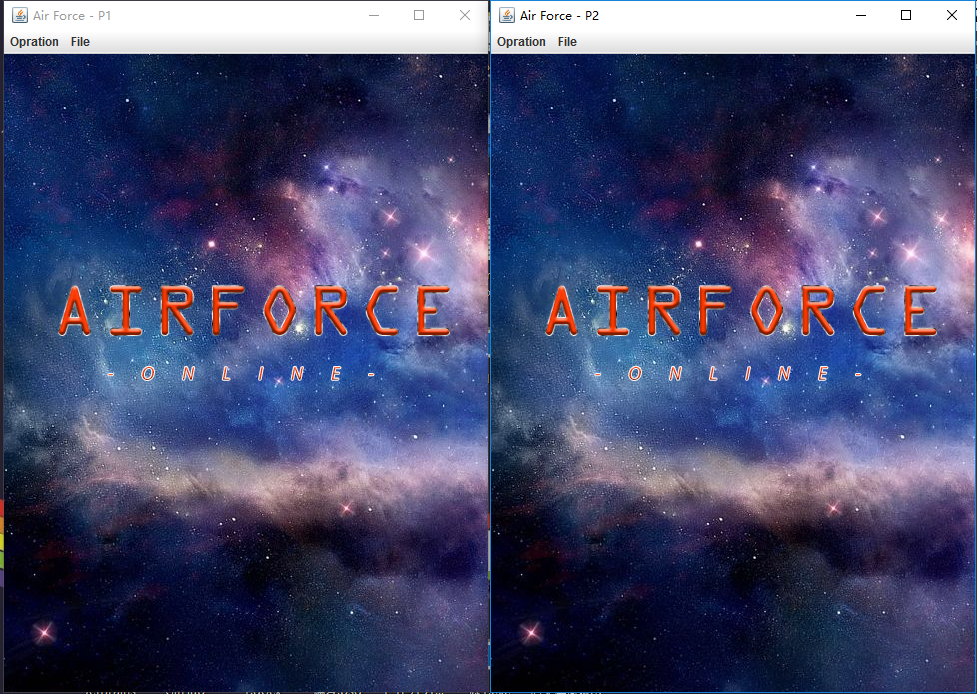
* Server:玩家二的联机类；
* Client:玩家一的联机类；
* ServerThread:玩家二的接收信息同步类；
* ClientThread:玩家一的接收信息同步类；

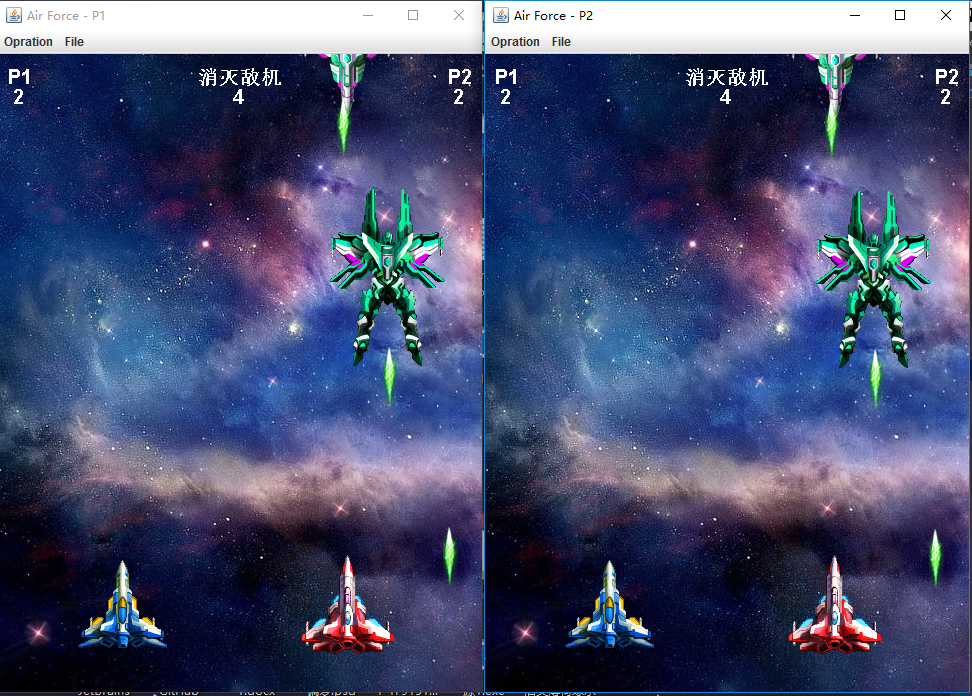
Server类的构造函数中定义一个ServerThread对象用于即使监听信息并同步信息，Client类也如此，具体构造见下图。

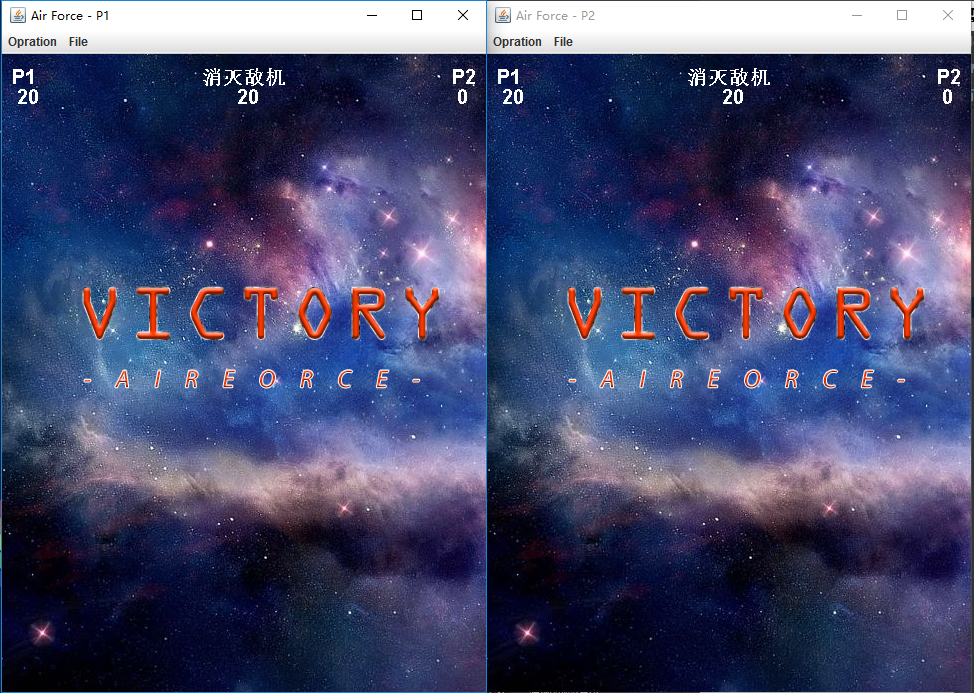
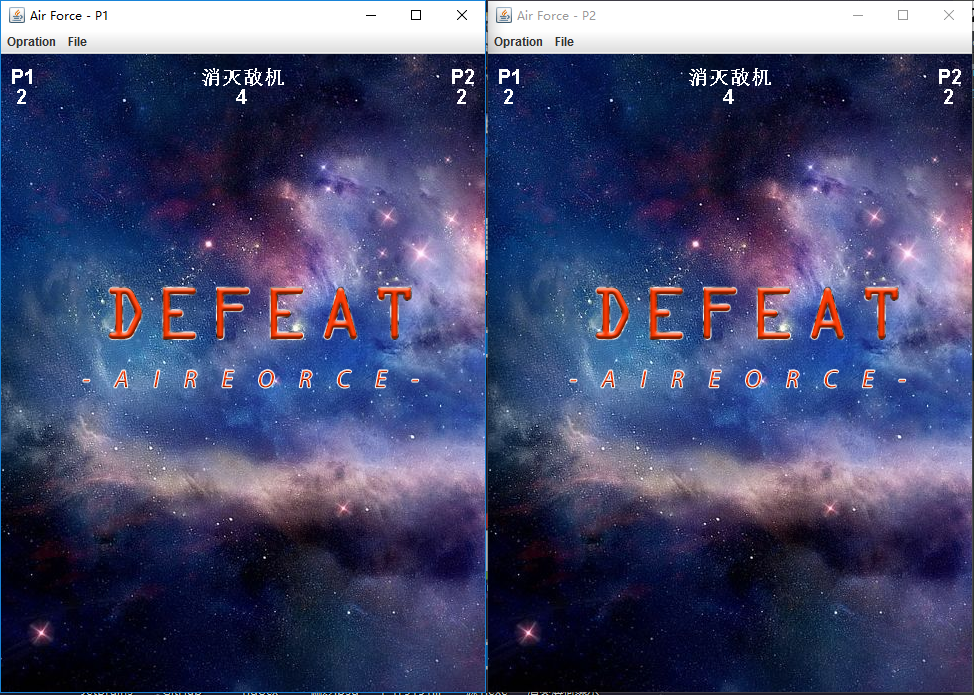


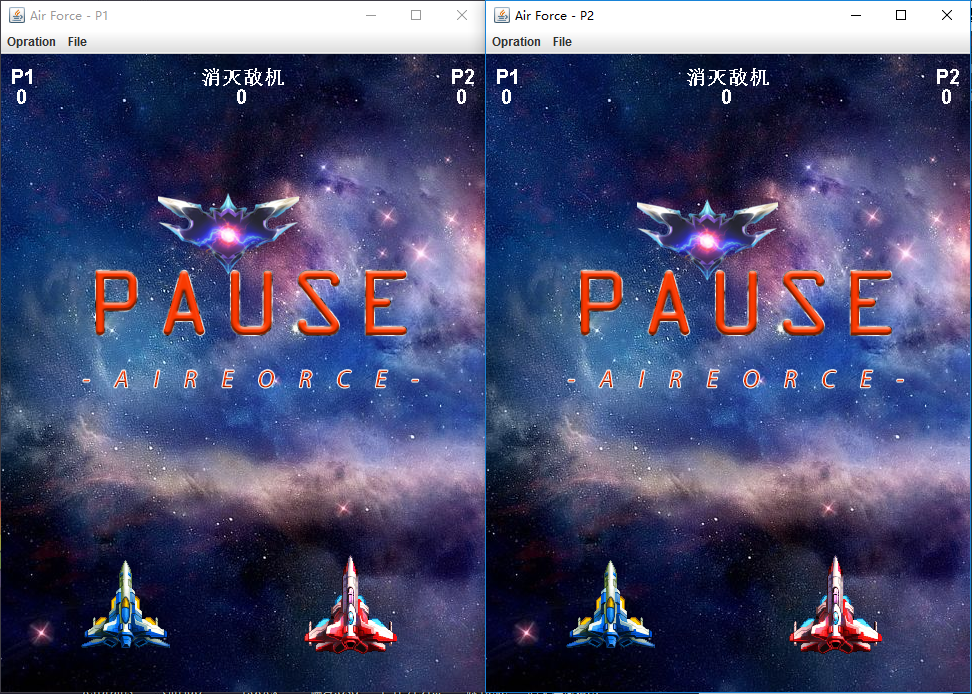
（图2 联机相关类具体参数及方法图）

四 界面设计

1.开始界面图:

2.游戏界面图:

3.游戏胜利/失败界面图:

4.游戏暂停界面图：

5.游戏模型图：

|  |  |  |  |
| --- | --- | --- | --- |
| 玩家一战机 |  | 玩家一战机（左倾） |  |
| 玩家一战机（右倾） |  | 玩家二战机 |  |
| 玩家二战机（左倾） |  | 玩家二战机（右倾） |  |
| 敌方普通战机 |  | 敌方精英战机 |  |
| 敌方自杀式战机 |  | 玩家一子弹 |  |
| 玩家二子弹 |  | 敌军子弹 |  |

五 网络通信和状态同步策略

由于本次游戏仅支持双人联机作战，因此考虑采用玩家一作为Client、玩家二作为Server进行联机，使用的是java.net内带的Socket类和ServerSocket类。

对于信息的接收，Server和Client类的构造函数中分别创建并运行有负责即时接收数据和同步界面的ServerThread和ClientThread类的对象，它们分别是两个代码无限循环的线程；对于信息的发送，Server和Client类只负责提供方法sendMessage(String message)和DataOutputStream，具体信息内容的发送与玩家的交互和状态的判断有关。

ServerThread类和ClientThread类可访问相应GameTab的方法，也即可以直接调用游戏界面的方法。

以下是信息的收发和状态的同步过程举例——以玩家一（Client）按下方向键“↓”为例：

1.玩家一和玩家二建立连接，游戏视窗建立，ClientThread和ServerThread线程建立并循环监听对应端的信息，玩家二开始游戏；

2.玩家一按下“↓”键，通过调用Client的sendMessage()方法将信息(“1down”+“\n”)发送至ServerThread类的对象serverThread；

3.对象serverThread接收到来自玩家一的信息并读取一行得到“1down”，将其与各个条件对比，在符合条件的情况下便对玩家二的GameTab中的plane1调用plane1.goDown()方法，玩家二面板中的plane1便向下运动，实现状态同步；

游戏中也有许多状态同步请求来自于流程循环中的条件判断，比如子弹命中、敌机坠毁、游戏暂停等等，这些状态同步与前文的举例类似，都是通过：条件判断——发送信息——比对信息——调整界面的流程来具体实现。

六 补充信息

1.由于项目检测是在本地进行，于是没有为Client和Server指定的或可自选的IP和端口，Client连接的IP是本地的8888端口，Server监听的是8888端口；

2.考虑到连贯性，游戏的开始只能由作为Server的玩家二执行，玩家一可执行的操作有暂停和继续；

3.仅有Server端会自动生成敌军战机，Client端仅同步Server端的生成信息，绘制敌机并实现敌机自动控制；

4.由于切图的问题，在命中和碰撞判定上引入了一些具体数值来使其更准确；

5.相应源代码和jar文件位于code文件夹内，代码位于src\airplane内,jar文件位于AirForce\_jar文件夹内。

七 个人感想

经过一学期Java语言的学习和学期末大作业的完成，我觉得收获很大，主要体现在对面向对象思想的体会和项目功能分离两个方面。以下结合大作业稍作叙述。

面向对象的思想贯穿于整个大作业。在大作业的实现过程中，许多模型具有相同的行为和描述方法，通过建立抽象类和继承一步一步自上而下、自抽象至具体来描述各个模型。这样的方法不仅准确而且简洁。通过子类对父类方法的重写，能够准确地描述子类对象。

项目功能分离是实现项目的高效手段。通过对界面的不断刷新，实现各个图形的运动，而每次运动都调用了经过流程循环、条件判断、修改参数等逻辑层步骤控制并修改的参数。这样一来，视图操作仅仅影响视图，逻辑层仅仅负责对各个对象参数的判断和修改，对代码的维护和编写都有很大的帮助，提高了项目的效率。

虽然没有完整地完成整个大作业，但还是觉得很有成就感。在此，感谢老师和助教一个学期以来的的辛勤付出和耐心辅导！