

（深圳）

实验报告

开课学期： 2022春季

课程名称： 面向对象的软件构造导论

实验名称： 飞机大战游戏系统的设计与实现

实验性质： 设计型

实验学时： 16 地点： T2102

学生班级： 11

学生学号： 200111132

学生姓名： 吴桐

评阅教师：

报告成绩：

实验与创新实践教育中心制

2022年4月

# 实验环境

系统：Windows10

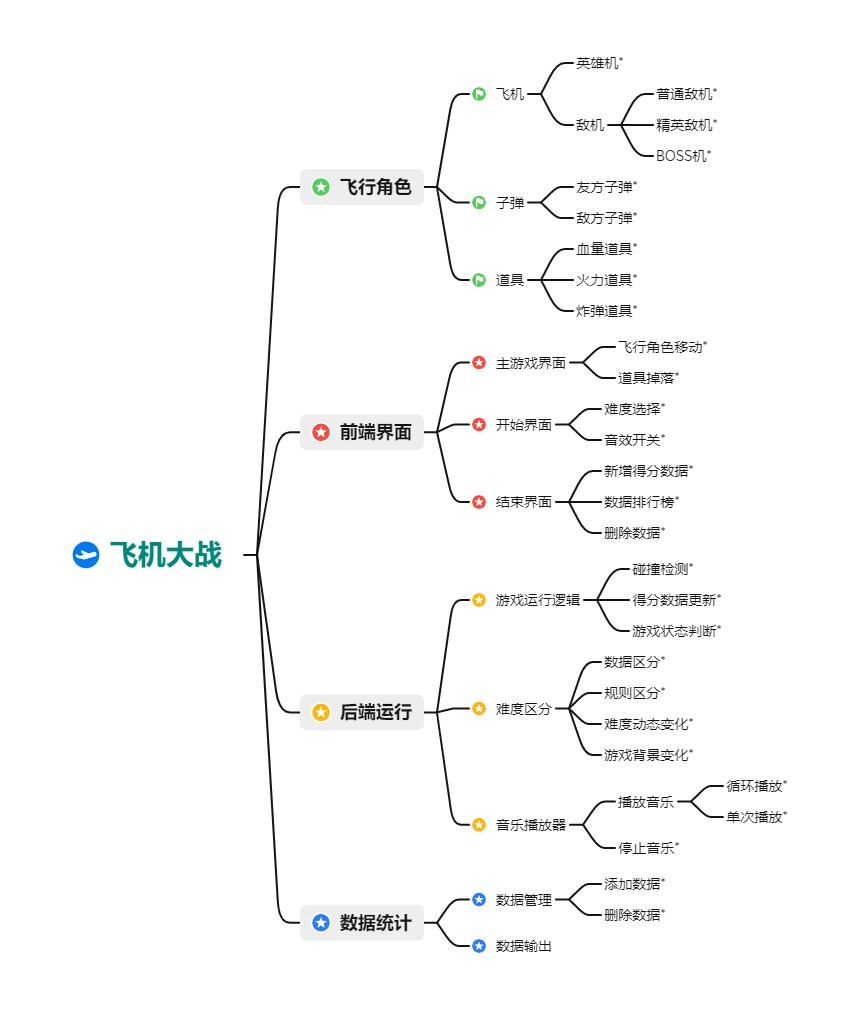
主要开发软件：idea2021->idea2022

# 实验过程

## 系统功能分析

*请结合文字、图表等方式（推荐使用功能层次图），清晰描述系统的功能。亮点功能请用\*标志。*

*(向下的层次图做出来字体太小了不方便看，所以做成了横向的)*

**

飞机大战是一款可爱又迷人的简单易操作的得分游戏

我将要实现的功能按照便于理解和实现分为四个部分：飞行角色，前端界面，后端运行，数据统计

飞行角色里实现了飞机和子弹等具体在游戏中承担实际演出对象的定义，属性，相关方法等。

前端界面实现了游戏运行中不断刷新的画面；游戏开始前选择难度，打开音乐的界面；以及游戏结束后，记录玩家得分的界面。

后端运行实现了游戏在运行过程中，程序在后台按照游戏规则和玩家操作的逻辑操作，其中除了游戏本身基础运行框架，还有根据上文难度选择的区分，背景音乐和音效。利用多线程的特点和优势得以实现。

最后还有玩家游戏结束后的得分记录的增删查改，利用java排序,shuju1jiexi1等api通过java文件输入输出流，文件读写的方式简易完成。

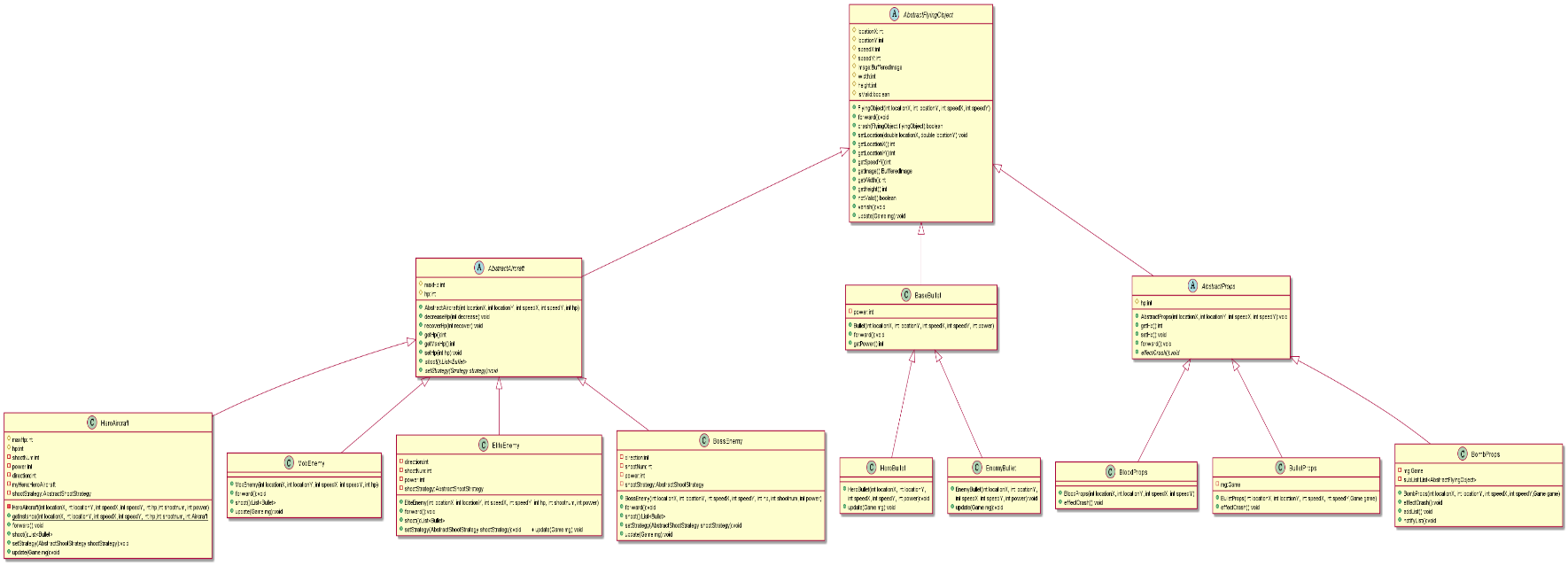
以上功能均已实现，并且可以方便后续有更加interesting的想法的时候更改一些更有意思的东西进去

*puml图除了在out里面，我也放在外面了，如果报告图片太小不方便放大可以到外层文件夹看*

## 类的继承关系分析

*请根据面向对象设计原则，分析和设计游戏中的所有飞机类、道具类和子*

*弹类，并使用 PlantUML 插件绘制相应的 UML 类图及继承关系，类图中需包括英*

*雄机、所有敌机、道具、子弹及它们所继承的父类。*

## 设计模式应用

### 2.3.1单例模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

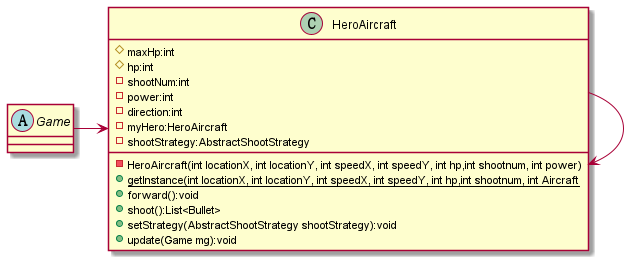
*应用场景：在游戏开始的时候创建自己的英雄机实例*

*存在问题：目前使用的是对象自身的构建方法，不能够保证实例只被创建一次，不够安全*

*所以我选择使用单例模式来保证只创建一个英雄敌机的实例*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。*



*2.描述*

*老师设置的一些飞机属性，移动射击方法除外*

*核心修改：*

*将HeroAircraft的构造方法改为private，不允许外部调用*

*加入静态的 getInstance 方法，在类内部使用构造方法，返回一个HeroAircraft 的对象*

*使用懒汉式的手段创建，当第一次被需要使用的时候才创建实例*

*考虑到后续加入多线程，以及性能的充足条件下，我们采用了线程加锁的方式来保证线程安全*

*这里HeroAircraft 构造方法也进行了重写（虽然最后要改用策略者模式），但是这里可以在创建的时候修改射击子弹次数以及攻击力。*

*(其实敌机也做了类似的修改，后续的工厂模式这里就不再介绍了)*

*后续为了和策略模式配合，加入了属性Strategy 和 setStrategy方法，更新射击策略以及shoot在策略不为空的时候调用策略的射击方法*

update方法是为了配合观察者模式重写的虚拟飞行物品函数

### 2.3.2工厂模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

*当游戏进行的时候我们需要不断地产生敌机，以及精英敌机坠毁后产生的道具*

*如果每次都是用构造方法的话并没有很好的利用创造者模式的思想*

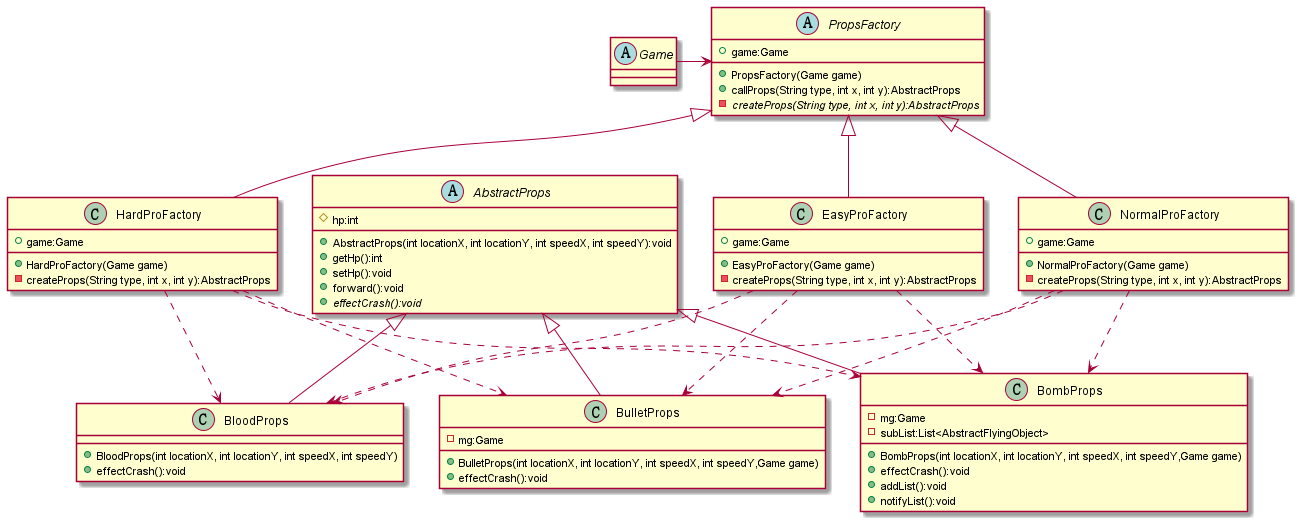
*并且如果我们需要修改难度，则对应创建敌机和物品的属性等等，需要修改主程序过多，耦合性过强*

*不能很好地满足开闭和单一原则*

*所以我们就利用工厂模式来创建敌机以及道具*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。*

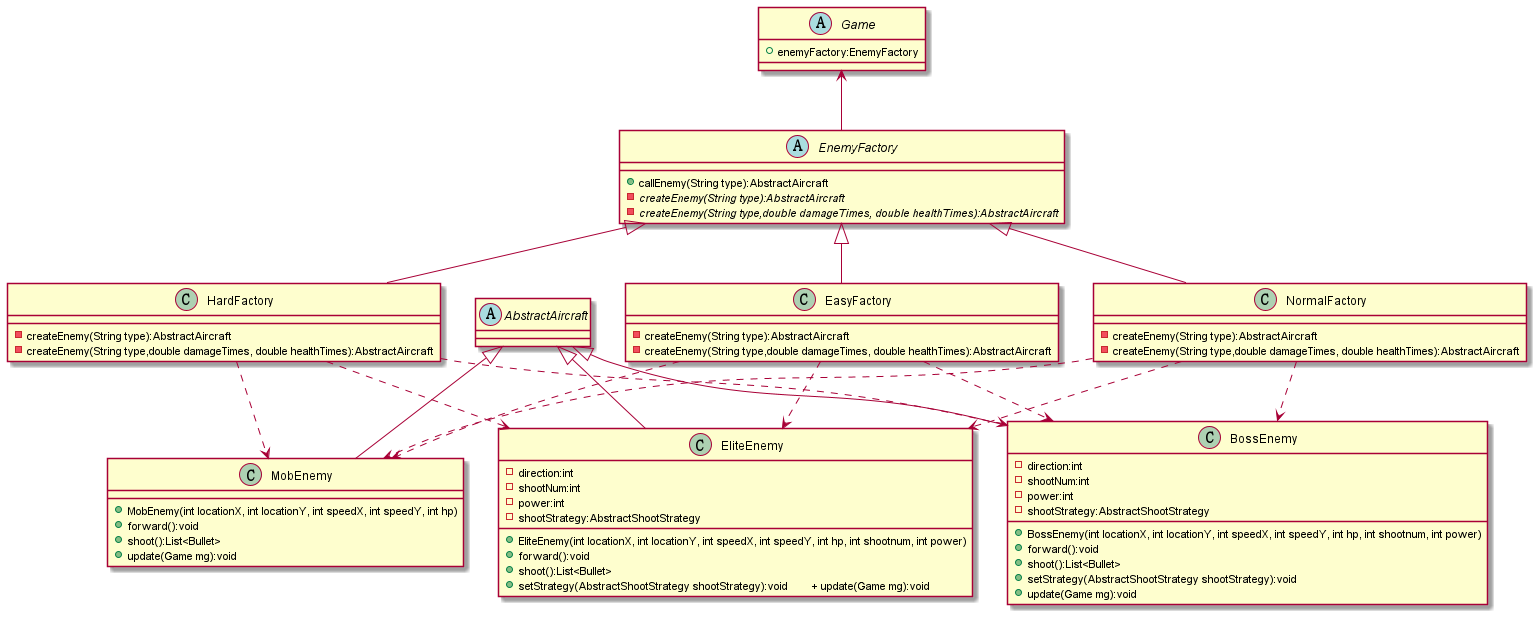
*EnemyFactory是一个抽象类，抽象的是所有敌机工厂的一个父类并且被主要游戏逻辑类Game所关联，用于当前游戏难度下的一个产生敌机工厂的一个属性*

*callEnemy为一个public方法，用于输入所需要敌机type类型后，返回相应的飞行飞机实例，并且会输出创造了什么实例*

*createEnemy 为一个私有的真正调用构造函数创建实例的方法，但是需要在实际类中被重写实现*

*EasyFactory等为实际继承的子类，重写了createEnemy的方法，主要是为了针对玩家后续选择了不同难度，在返回相应普通，精英，BOSS类敌机的时候，相对应实例具有不同的参数属性（血量，攻击力，子弹数目，飞行时间等）*

*并且在后续还可以通过修改不同难度的工厂，精英敌机的多样性也可以得到保障，且不影响到Game实际游戏的调用*

*Game只需要通过参数敌机类型即可创造敌机*

*道具的工厂同理于敌机工厂*

*PropsFactory 为虚拟工厂父类*

*callProps为一个public方法，用于输入所需要敌机type类型后，返回相应的飞行飞机实例，并且会输出创造了什么实例*

*createProps 为一个私有的真正调用构造函数创建实例的方法，但是需要在实际类中被重写实现*

*同样的我是为了区分不同难度所需要产生的道具具有不同的属性而有不同难度的工厂*

*在实际的工厂里我为不同的道具预设不同的属性*

*道具继承抽象飞机道具类，成品版本拥有新构造属性参数Game game*

*以及碰撞后的活动effectCrash（），分别用于调用后续多线程火力道具，以及观察者模式炸弹爆炸*

### 2.3.3策略模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

*现在飞机的射击子弹的方式仍然是和飞机对象绑定的，让飞机（无论是自己还是敌机）在中途改变设计方式较为困难*

*策略模式将责任与算法分开，射击是统一的动作，但是我们需要不同的算法返回不同飞行方式的子弹*

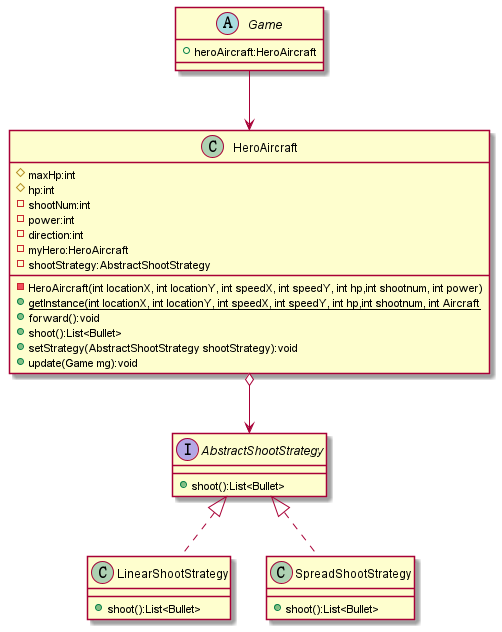
*所以我们利用策略模式来设计射击子弹，给飞机添加一个策略的接口，在创造实例，或者是游戏过程中接受道具改变方式，只需要更换飞机的策略实例即可。*

*射击策略主要利用飞机自身自带的属性，传入参数，策略的射击函数方法返回子弹序列。*

*这里我设计了直线射击和散射两种最基础的（后期再慢慢加入奇奇怪怪的射击方法）*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。*



*b.详细*

*这里英雄机添加了策略接口的属性，还有设置策略的方法。*

*射击这里自己重构代码的时候为了避免策略为空，但是保留了默认射击的策略，如果策略不为空，则调用策略的射击函数*

*直射和原来差距不大，散射的射击函数主要依靠根据子弹射出位置线性改变子弹x方向上的速度不一样，来造成散射的效果*

### 2.3.4数据访问对象模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

*后续我们需要为每个玩游戏的人保存他的游玩数据，并且我们需要打印结果，游戏需要排行榜！*

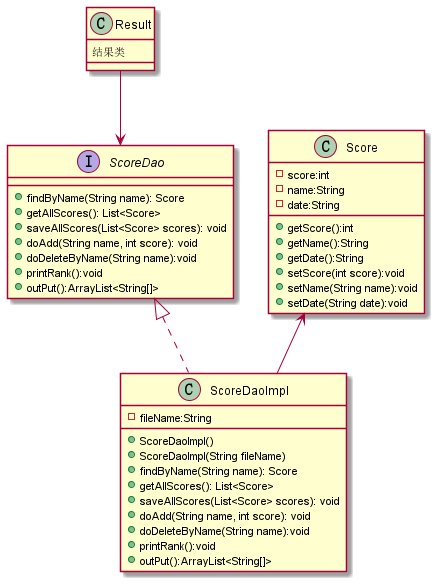
*但是目前游戏并没有加入保存游戏数据的能力，所以我们利用数据访问对象模式*

*有一个类是我们需要存储的数据（一切事物对象化），另一个接口是我们需要操作数据的方法。*

*操作数据实际方法的实现可以有很多种，通过具体的类来规定合适的算法*

*目前没有规划说用哪种数据库驱动，那就直接写入txt文件吧*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。。*

*Score 类*

*属性分别为得分，玩家名字，日期*

*方法为标准的构造方法*

*ScoreDao接口*

*写入文件主要是通过写入对象的方法实现（学习一下序列化）*

*findByName 通过名字查询返回当前排行榜的得分对象*

*getAllScores 返回一个列表，元素为所有当前排行榜的对象*

*saveAllScores 传入一个列表，将列表中的元素全部写入文件*

*doAdd 传入名字和得分参数，插入排行榜记录文件*

*doDeleteByName 传入名字参数，删除该名字下的记录*

*printRank 按名次打印当前排行榜中所有得分记录*

*outPut方法为配合GUI排行榜输出成绩的核心接口，输出list<String>用于为表格的模型更新数据*

### 2.3.5观察者模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

*飞机大战中，我们拥有一个炸弹道具，即能够消灭所有的敌机子弹和敌机（除boss机）*

*此时我们想到了使用学过的观察者模式*

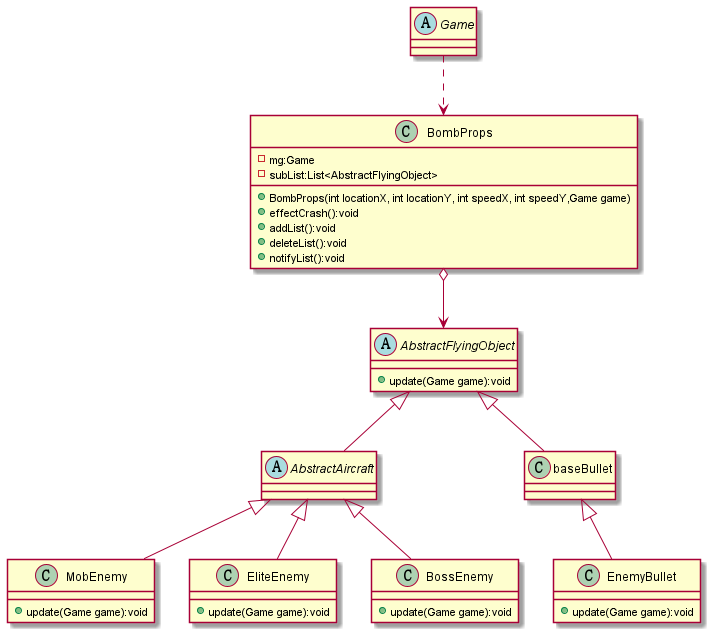
*将炸弹作为被观察者对象，将抽象飞行道具类作为观察者*

*添加一个方法update用于统一接收到观察对象的变化*

*重写炸弹代码，添加list保存观察者对象，以及触发方法*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。*



subList中保存了所有观察者（非boss敌机以及敌机子弹）

addList 方法用于更新list列表里面的观察者

deleteList 方法用于删除列表里保存的观察者（仅仅删除观察列表，但是对象本身并没有丢失）

notifyList 方法用于调动list所有观察者的update方法

effectCrash 方法为爆炸主方法，在game主线程中被触发后调用

update 方法为每个观察者对象独有的相应炸弹爆炸的方法

主要为加分以及将自己摧毁（敌机子弹和非BOSS敌机）

### 2.3.6模板模式

1. 应用场景分析

*描述飞机大战游戏中哪个应用场景需要用到此模式，设计中遇到的实际问题，使用该模式解决此问题的优势。*

*游戏中我们有三个难度选项，分别为简单，普通和困难*

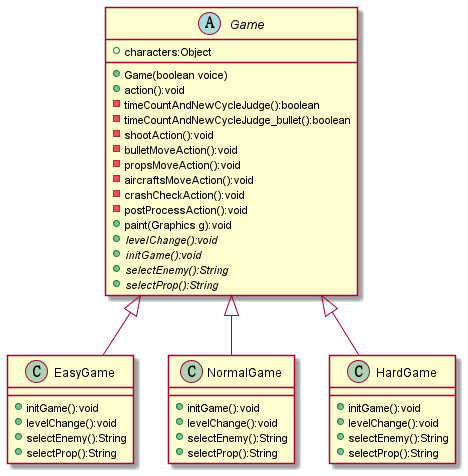
*三者的主游戏逻辑相似，但是在数值或者部分方法上存在差异，所以这里我们选用模板模式*

*将老师一开始给我们的game类抽象化，作为一个抽象父类，然后里面设置有关难度变化的方法作为抽象方法*

*实际实现每个难度的时候继承game类，并分别重写自己难度的相关方法。*

1. 设计模式结构图

*结合飞机大战实例，绘制该场景下具体的解决方案（UML类图）。描述你设计的UML类图结构中每个角色的作用，并指出它的关键属性和方法。*

*四个重写方法为*

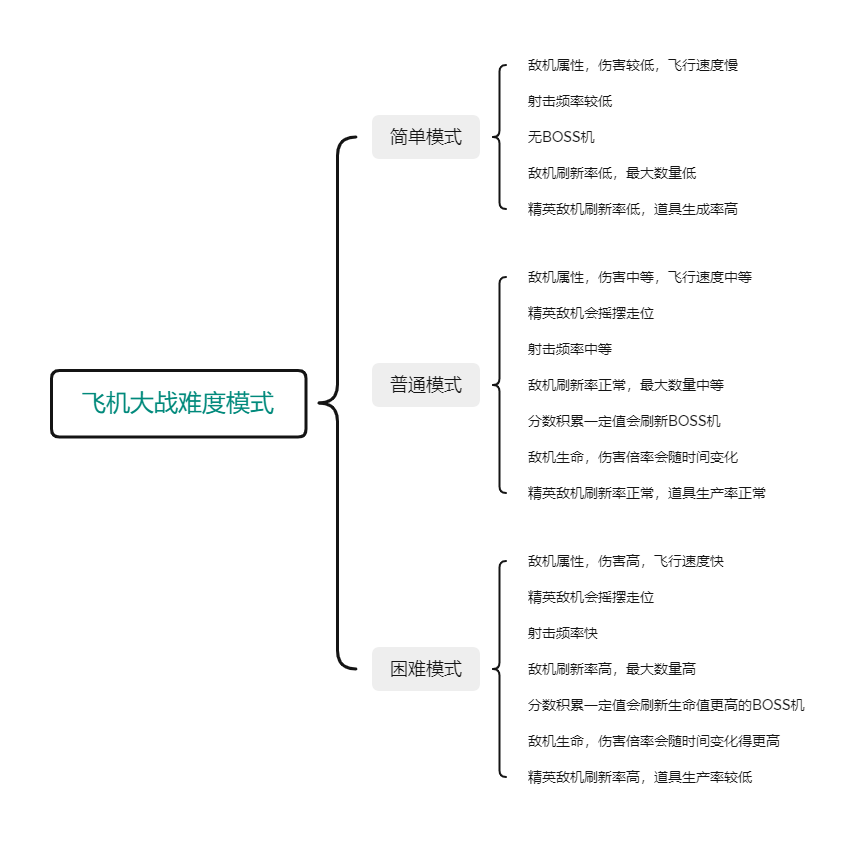
*initGame 初始化游戏数据，根据游戏难度可以在该函数中更新属性*

*levelChange 根据游戏进行，对难度进行动态变化*

*selectEnemy 生成敌机种类概率的重写*

*selectProp 生成道具种类概率的重写*

*做了一个图来反应不同游戏难度的区别*



# 收获和反思

*请填写本次实验的收获，记录实验过程中出现的值得反思的问题及你的思考。*

*欢迎为本课程实验提出宝贵意见！*

老师，收获很多，但是我想先说说对于这个课的建议！！！希望下一届能上更好的课，但是也都是我个人的声音，可能没有老师考虑的周全，如果有什么误解希望老师不要在意。

1.实验之间的配合我认为不是很好，为了完成前面的任务，虽然有版本控制，但是还是写了很多后期下一个实验需要重构的代码，我认为这在一个实验里加大了很多的工作量。

虽然每个阶段有每个阶段的意义所在。

建议可以提前公布整个实验更加详细的一个流程布局，我做实验一的时候，也可以去思考如何方便我完成后续实验的一个重构，减小一定工作量。

更加有利于我们把握一整个项目开发流程的完整性，而不要为了每次都是只学习一个部分，最后增加了一些不必要的时间开销，打消同学们为了做好这一个项目的积极性。

2.希望实验提交之后能有一个老师带着验收的过程，我每一次实验做完，虽然我自己没有验证出bug，或者我认为我自己达标了，但是最终实验前我认为都是一个对某一个部分的专项学习，在项目进展和对这个专项的突出上，我感觉很难把控，最好能老师带着验收，保证我上交版本的作业得分的保障。

3.希望可以和实践课老师沟通一下，当我听到实践课仍然是做飞机大战的时候，其实还是有一点失望，一是又要做一个飞机大战，二是感觉实践课的自由度的缺失。我很希望的是老师给一个方向，或者是给几个选题让我们挑选。实践课课程设置为的是考查课，还是应该积极鼓励我们做一点感兴趣的东西为主。

4.我个人认为自己的基础还是可以独利胜任这个实验的，也有过一些相关的经验，然后对一些更厉害的大佬来说，那就更厉害了。但是在我身边也有相当的人，做着就很痛苦，经常会来问我，有的也不大好意思来找老师。

虽然这个的确是需要大家一个学习的过程，但是我们这个课程可以设计的有一个分享的过程或者说大家一起上课的，可以有一个更加轻松的氛围。可以让更加厉害的来分享，或者是有问题的大家愿意群里面一起来解决。

就是我个人这门课上完，我希望这门课不仅让大家初步熟悉，更多的是让大家不要丢失去开发，完成一个东西的积极性，还有培养一个修改错误的耐心。但是我在做到后面，都感觉到大家的积极性都在丧失。我想不出来什么好的确切的措施，但是希望可以通过给老师的反馈，能帮到老师如何去完善做好这一门课。

收获也是很大的，虽然我只上了老师一节线下课，但是老师们都很热心帮同学们解决问题，我问了一些关于如何评分的问题老师也都细心解答。

然后开发过程中也对正课上的很多知识点进行了复现和熟悉

可惜的是网络，泛型，反射部分没有写到太多相关的东西。

uml类图也是这门课学到一个很好的对于对象的一个描述工具。

我也帮到了不少同学，帮他们debug，教他们怎么写多线程之类的，也算是增进了同学友谊。

最后希望老师这门课给下一届同学能做的更好！！！