“高级战争”游戏

Ruosen Lee | 北京市海淀区上园村3号北京交通大学

项目报告

李若森

2016

“高级战争”游戏项目报告

目录

[“高级战争”游戏项目报告 1](#_Toc450114200)

[一、 需求分析 3](#_Toc450114201)

[1. 产品描述 3](#_Toc450114202)

[2. 产品功能 3](#_Toc450114203)

[3. 目标界面 3](#_Toc450114204)

[二、 项目设计 4](#_Toc450114205)

[1. 总体设计 4](#_Toc450114206)

[2. 功能模块设计 4](#_Toc450114207)

[1) 火力单元模块 4](#_Toc450114208)

[2) 对象池模块 8](#_Toc450114209)

[3) 工厂模块 9](#_Toc450114210)

[4) 地图模块 10](#_Toc450114211)

[5) 备忘录模块 10](#_Toc450114212)

[6) 广播模块 11](#_Toc450114213)

[7) 显示模块 12](#_Toc450114214)

[8) 全局模块 12](#_Toc450114215)

[3. 业务流程设计 14](#_Toc450114216)

[三、 项目开发计划 15](#_Toc450114217)

[1. 项目管理工具 15](#_Toc450114218)

[2. 项目开发环境 15](#_Toc450114219)

[3. 项目开发计划 15](#_Toc450114220)

[四、 项目测试 16](#_Toc450114221)

[1. 显示测试 16](#_Toc450114222)

[2. 基础操作测试 16](#_Toc450114223)

[1) 移动测试 16](#_Toc450114224)

[2) 战斗测试 18](#_Toc450114225)

[3) 撤回操作测试 19](#_Toc450114226)

[4) 待命操作测试 20](#_Toc450114227)

[5) 轮换操作测试 21](#_Toc450114228)

[6) 获胜方测试 22](#_Toc450114229)

[3. 地图更换测试 22](#_Toc450114230)

[附件： 23](#_Toc450114231)

[1. 完整UML类图 23](#_Toc450114232)

# 引言

本文章为战略策略游戏“高级战争”的项目报告，也是设计模式课程大作业的课程报告。

“高级战争”游戏使用JAVA语言进行开发，使用JDK版本为JDK8。

设计模式是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。在未使用设计模式时，代码的编写完全依据程序员的想法及经验，几乎没有规律可循，因此大大增加了工程师后期维护的复杂度和难度。使用设计模式开发工程项目时可能会增加代码量、加大程序员的工作量，但是在后期维护时可以极大的减少程序员的工作量及维护复杂度和难度。因此设计模式在当今工程项目的开发中起到了不可替代的作用。

“高级战争”游戏使用了11种设计模式：抽象工厂模式、原型模式、单例模式、装饰模式、享元模式、迭代器模式、备忘录模式、观察者模式、状态模式、策略模式、对象池模式。

# 需求分析

## 产品描述

本游戏是一款回合制战略策略类游戏(SLG)，是一款单机游戏，支持多人对战。

本游戏灵感来源于GBA上任天堂曾发布的一款战略策略类游戏“高级战争”。本游戏仿照“高级战争”进行开发，采用的游戏规则有一些不同。

## 产品功能

本游戏支持移动、对战、撤回上一步三种操作。游戏规则为多人对战，每位玩家控制一个军队进行对战，最终控制战场上留存的唯一的军队的玩家获胜。

## 目标界面

|  |
| --- |
|  |

# 项目设计

## 总体设计

整个项目分为三层。第一层为外部UI，这一层采用JAVAFX向外展示并接受鼠标的点击等操作，这一层全部由JAVAFX处理。第二层为游戏逻辑层，从JAVAFX中接受信息并做处理传递给第三层和从第三层接受信息更新并传递给JAVAFX显示。第三层为后端数据层，所有关于数据的处理都在这一层展开。

整个项目将用到11种设计模式：抽象工厂模式、原型模式、单例模式、装饰模式、享元模式、迭代器模式、备忘录模式、观察者模式、状态模式、策略模式、对象池模式。所有11种设计模式皆用于第三层开发。

## 功能模块设计

整个项目分为8个模块：火力单元模块、对象池模块、工厂模块、地图模块、备忘录模块、广播模块、显示模块、全局模块。其中显示模块和菜单模块为第二层模块。其余六个模块皆为第三层模块。

各个模块功能划分以及相对应的UML类图如下：

（具体UML类图见附件，以下为简化版UML类图）

### 火力单元模块

【功能】

该模块描述了一个火力单元的基础属性（攻击属性、移动属性、防御属性）。

【设计模式】

该模块使用了6种设计模式，分别为：抽象工厂模式、享元模式、策略模式、装饰模式和状态模式。充分降低各个属性间的耦合性，提高可维护性。

|  |
| --- |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-FireUnit.png |
| FireUnit模块总UML类图 |

火力单元模块分为三个小模块：攻击模块、基础模块、健康模块。

#### 攻击模块

【功能】

描述火力单元的武器攻击能力。

【细节描述】

此模块中总共有7种武器：冲锋枪、重机枪、肩扛式火箭弹、步兵战车、坦克、远程火炮、远程火箭弹。特殊能力分为两种：将军的激励和人手不足，分别对应总攻击力加倍和减半。

【设计模式】

采用策略模式为每种攻击模式设置武器，采用装饰模式为每种攻击模式进行附加修饰，使其具有特殊能力。

|  |
| --- |
|  |
| 标准装饰模式UML类图 |
|  |
| 标准策略模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-FireUnit-AttackComponent.png |
| AttackComponent模块UML类图 |

每一个AttackAttrOfFireUnit中采用不同WeaponOfFireUnit中的武器作为具体策略，通过AttackComponentDecorator采用GeneralDecorator和WeaknessDecorator两个具体装饰进行装饰。

#### 基础模块

【功能】

描述火力单元的基础属性。

【细节描述】

火力单元基础模块有三种属性：基础攻击、基础移动、基础防御。共有三种基础攻击能力（机枪进攻、装甲进攻、远程进攻）、两种基础移动能力（步行移动、车辆移动）以及两种防御能力（士兵防御、装甲防御）。

【设计模式】

采用抽象工厂生成3类火力单元基本能力产品簇。运用享元模式将3类产品簇共享。享元工厂采用单例模式维护。

|  |
| --- |
|  |
| 标准享元模式UML类图 |
|  |
| 标准抽象工厂模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-FireUnit-BasicComponent.png |
| BasicComponent模块UML类图 |

由WayOf\*三个接口组成抽象产品，其下共7个具体实现类组成具体产品类。AbstructFactoryOfFireUnit作为抽象工厂类，FacotryOf\*三个抽象类组成具体工厂类。BasicComponent作为抽象享元角色，ConcreteBasicComponent作为具体享元角色同时也相对于抽象工厂作为客户端角色。FacotryOfBasicComponent作为享元工厂且采用单例模式维护。

#### 健康模块

【功能】

描述火力单元当前状态。

【细节描述】

健康模块中健康状态分为3中状态：健康、受伤、死亡。其中健康对应特殊能力“将军的激励”、受伤时为“人手不足”。

【设计模式】

采用状态模式描述当前火力单元的健康状态，3种状态作为具体状态角色。

|  |
| --- |
|  |
| 标准状态模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-FireUnit-HealthComponent.png |
| HealthComponent模块UML类图 |

HealthComponent作为抽象状态角色。Healthy、Death、Injured三个类作为具体状态角色，并且不同状态会影响到攻击模块的装饰。FireUnit作为上下文环境。

### 对象池模块

【功能】

该模块使用了4种设计模式：对象池模式、原型模式、迭代器模式、单例模式。

【细节描述】

此模块中存储着火力单元模块产生的火力单元。当产生新火力单元时通过原型模式进行克隆生成，其中原型管理器角色由对象池承担。对象池的遍历由迭代器模式承担。对象池本身采用单例模式维护。

【设计模式】

|  |
| --- |
|  |
| 标准登记形式原型模式UML类图 |
|  |
| 标准单例模式UML类图 |
|  |
| 标准对象池模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-ConnectionPool.png |
| ConnectionPool模块UML类图 |

本项目中对对象池模式进行了简化操作，仅保留了PoolManager和PoolItem但足以表达对象池模式的核心思想。PoolItem作为基础对象，PoolManager作为对象池管理者。在PoolManager中进行查找时采用迭代器模式进行查找，增加新成员时采用原型模式增加新成员。

### 工厂模块

【功能】

该模块使用了1种设计模式：单例模式。

【细节描述】

该模块负责调用对象池模块生成相应的火力单元。

【设计模式】

|  |
| --- |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-FactoryOfFireUnit.png |
| FactoryOfFireUnit模块UML类图 |

FactoryOfFireUnit采用单例模式保证工厂唯一，利于后期管理。

### 地图模块

【功能】

该模块使用了1种设计模式：单例模式。

【细节描述】

该模块负责调用所有第三层资源并生成地图信息呈现给第二层，并且负责移动操作、战斗操作、撤回操作和回合转换操作。

【设计模式】

|  |
| --- |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Battlefield.png |
| Battlefield模块UML类图 |

Battlefield类采用单例模式避免出现地图重复加载的情况。

### 备忘录模块

【功能】

该模块使用了2种设计模式：备忘录模式、单例模式。

【细节描述】

该模块负责记录操作者上一步的操作，在撤回操作激发时恢复上一步状态。

【设计模式】

|  |
| --- |
|  |
| 标准备忘录模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Memeto.png |
| Memeto模块UML类图 |

Memeto作为备忘录角色，其中存储LastAction作为上一步动作的记录。MemetoCaretaker作为备忘录管理者角色。

### 广播模块

【功能】

该模块使用了2种设计模式：观察者模式、单例模式。

【细节描述】

该模块负责试试提醒玩家己方军队健康状况。其中观察者由玩家和火力单元共同承担。

【设计模式】

|  |
| --- |
|  |
| 标准观察者模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Observer.png |
| Observer模块UML类图 |

AllyControlCenter作为目标类，ConcreteAllyControlCenter作为具体目标类，Observer作为抽象观察者类，FireUnit作为具体观察者类。

### 显示模块

【功能】

将第三层的信息呈现给第一层，将第一层的信息处理后传递给第三层。

【设计模式】

该模块没有使用设计模式。

【细节描述】

该模块负责整个游戏的业务流程控制及界面显示以及军队状况实时播报。同时负责整个游戏的所有菜单显示，包括每个火力单元的控制菜单、属性菜单和控制菜单。

|  |  |
| --- | --- |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Canvas.png | C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Menu.png |
| Canvas模块UML类图 | Menu模块UML类图 |

### 全局模块

【功能】

管理全局类

【设计模式】

本模块实现1种设计模式：迭代器模式。

|  |
| --- |
|  |
| 标准迭代器模式UML类图 |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-Global.png |
| Global模块UML图 |

Iterator作为抽象迭代器，ConcreteIterator作为具体迭代器。Aggregate作为抽象聚合类，ConcreteAggregate作为具体迭代器。

## 业务流程设计

|  |
| --- |
| C:\Users\Ruosen\SkyDrive\GitHub\Design-Pattern-Homework\UML\UML-IDEA\UML-ALL-PACKAGE.png |
| 总UML类图 |
| 总业务流程图 |

# 项目开发计划

## 项目管理工具

本项目采用Git进行项目管理，平台为Github。

项目放到Github上：<https://github.com/lrscy/Design-Pattern-Homework/>

## 项目开发环境

项目采用语言：JAVA

项目开发IDE：Intellij IDEA 2016.1

地图编辑器：Tiled

## 项目开发计划

先进行第三层模块的编写，然后写第二层。每写一个模块进行一次单元测试，直至开发结束。

5天进行模块的设计，包括个模块的分UML类图以及程序流程控制图的初步设计。

5天完成第三层所有模块的开发，2天完成所有第三层模块的测试。

3天完成第二层所有模块的开发以及与第一层模块的链接。

2天进行总体调试检查bug。发布稳定版Version v1.0.0。

在开发过程中可修改部分设计以应对未知的问题。

# 项目测试

## 显示测试

|  |
| --- |
|  |
| 经测试，界面显示正常。 |

## 基础操作测试

### 移动测试

|  |
| --- |
|  |
| 移动准备 |
|  |
| 移动范围显示 |
|  |
| 移动成功 |
|  |
| 移动选项不可再次点击 |

### 战斗测试

|  |
| --- |
|  |
| 战斗准备 |
|  |
| 可攻击对象提示 |
|  |
| 攻击结果 |
|  |
| 攻击选项不可再次点击 |

### 撤回操作测试

|  |  |
| --- | --- |
| 移动撤销操作测试 | 攻击撤销操作测试 |
|  |  |
| 移动准备 | 战斗准备 |
|  |  |
| 移动结果 | 战斗结果 |
|  |  |
| 移动撤回后结果 | 战斗撤回后结果 |

### 待命操作测试

|  |
| --- |
|  |
| 待命准备操作 |
|  |
| 待命完成操作 |

### 轮换操作测试

|  |  |
| --- | --- |
| 红方士兵 | 蓝方士兵 |
| 红方操作 | |
|  |  |
| 蓝方操作 | |
|  |  |

### 获胜方测试

|  |
| --- |
|  |
| 获胜提示 |

## 地图更换测试

|  |
| --- |
|  |
| Battlefield\_02 |
|  |
| Battlefield\_01 |

# 附件：

## 完整UML类图

详见文件UML文件夹下UML-ALL.pdf及其他UML图。