

­­

**设计模式课程­设计**

2016.11.16~2016.12.14

学院： 软件学院

小组成员：陈冉冉 1452685

王冠淞 1452693

李威 1452714

目录

[一．设计背景 3](#_Toc469516324)

[1.项目简介 3](#_Toc469516325)

[2. 参考书目 3](#_Toc469516326)

[二．开发环境 3](#_Toc469516327)

[1. 开发工具： 3](#_Toc469516328)

[2. 开发语言 3](#_Toc469516329)

[3. 文件依赖 3](#_Toc469516330)

[三．设计模式 4](#_Toc469516331)

[1. 工厂模式 4](#_Toc469516332)

[2. 装饰者模式 5](#_Toc469516333)

[3. 策略模式 5](#_Toc469516334)

[4. 委托模式 6](#_Toc469516335)

[5. 单例模式 6](#_Toc469516336)

[6. 外观模式 7](#_Toc469516337)

[7. 备忘录模式 7](#_Toc469516338)

[四．类图 8](#_Toc469516339)

[五. 总结与反思 10](#_Toc469516340)

# 一．设计背景

1.项目简介：

我们的项目是一个在cocos-2dx平台开发的塔防游戏。这个项目是我们在大一下c++课程项目，由于当时没有学习过设计模式等先进理念，开发的项目可读性差、耦合性高、维护性差，此次借设计模式学到的知识，改进维护项目，提高自己。

2. 参考书目：《cocos-2dx游戏教程》

# 二．开发环境

1. 开发工具：Visual Studio 2015。

2. 开发语言：C++。

3. 文件依赖：cocos-2dx库文件。

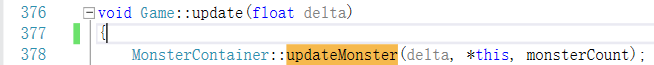
# 三．设计模式

### **1.工厂模式**

游戏中有多个怪物，在旧版中我们是直接在game主函数中判断怪物类别并创建。

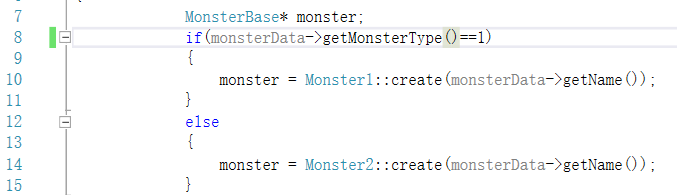
在新版中采用了简单工厂模式。在MosterBase中定义怪物的基本属性和操作，Monster1和Monster2继承MonsterBase，并实现各自不同的操作，如击中后掉血。当需要创建怪物时，game给MongsterFactory发送命令，工厂根据配置文件中的内容创建不同的怪物，放到MonsterContainer中封装起来，一起传回给主函数。

在game主函数中调用MonsterCantainer



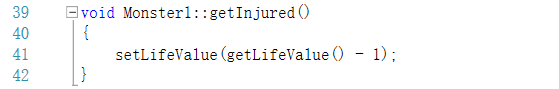
在MonsterContainer中调用工厂创建怪物

在MonsterFactory中创建不同怪物

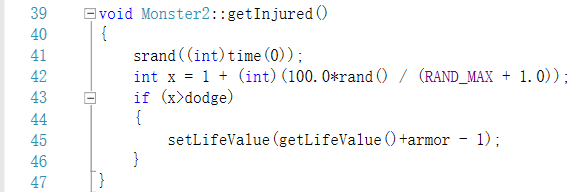


Monster1和Mosnter2的不同之处有：Monster1被击中后直接掉血，Monster2有40%的概率不掉血。

Monster1：



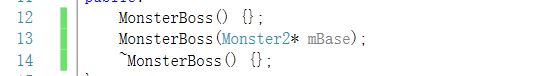
Monster2



## 2. 装饰者模式

在游戏结束前，我们有一个Boss，在旧版本中我们直接创建。

在新版本中，考虑到Boss是在Monster2的基础上，所以我们采用装饰者模式，直接在Monster2的基础上改变属性值，创建并返回。

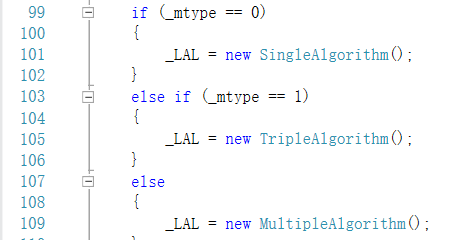


## 3.策略模式

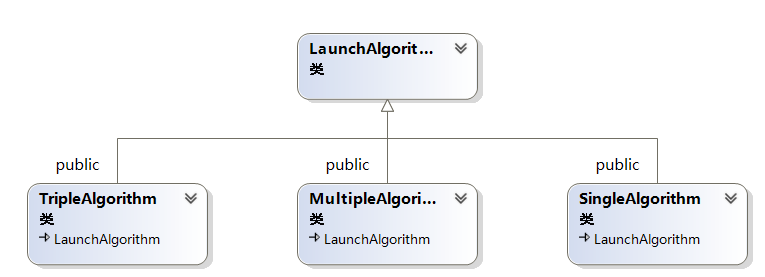
我们的炮塔有三种不同的发射子弹的方法，一种是发射一个子弹，一种是发射三个子弹，一种是发射一圈子弹。在旧版本中是在不同塔内分别实现发射算法。

在新版本中，我们将算法抽象出来，使用策略模式，根据塔的种类将发射算法的指针指向不同的算法。

塔：



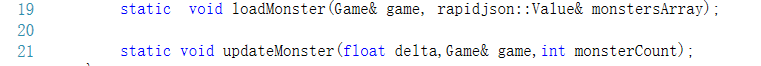
发射算法：



## 4. 委托模式

委托模式在这里是Game类和MonsterContainer类协同解决怪物数组的建立和更新问题而存在，简单的说就是把原本属于委托者Game类做的事情交给被委托者MonsterContainer来做。

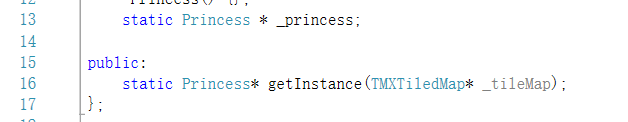
比如从文件中读取数据并建立Monster数组和每次更新Monster数组：



## 5. 单例模式

在旧版本中，公主是写在主函数中的一个static的变量。

在新版本中 ,利用单例模式，创建一个公主的唯一全局变量。



## 6.外观模式

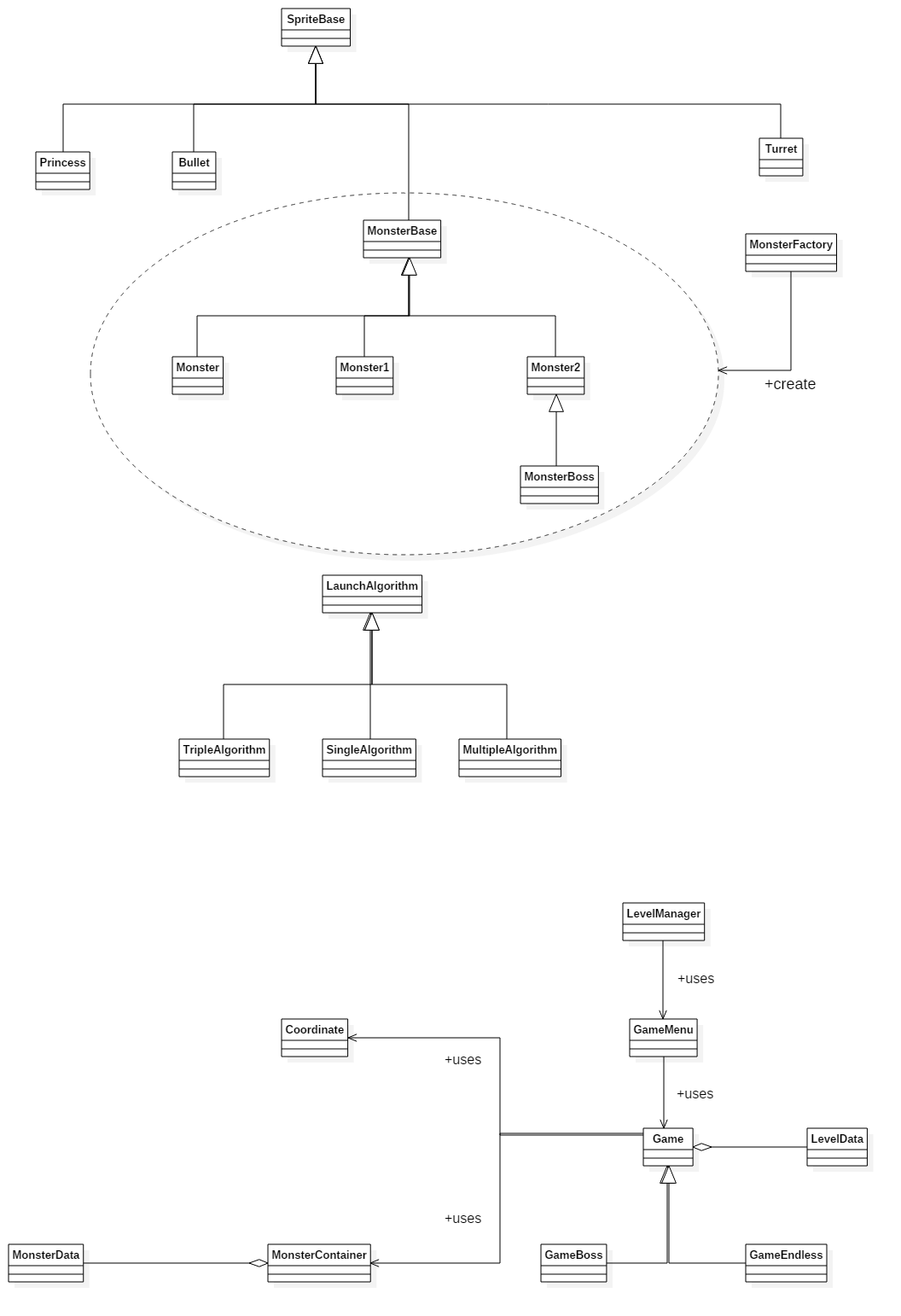
在旧版本中，我们的game主函数有将近1000行。

在新版本中，我们通过将函数解耦，是函数减少了很多，并且将各个功能点分散到不同类中，提高了monster和turret的内聚性。

## 7.备忘录模式

通过记录游戏中金币、关卡等数据，实现了游戏的存档功能。

# 四．类图



# 五. 总结与反思

通过此次设计模式项目，我们对各种设计模式有了更深的理解，之后写项目时我们要提前设计好各个功能点，做到高内聚、低耦合，保证项目的可读性和可维护性。