中国科学：信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

蒋梓翔[1]

1. 南京大学, 南京

E-mail: 1214069454@qq.com

摘要 本文主要描述了 Java 作业 8 的开发目标，设计理念，技术问题，工程问题，课程感言以及完成编写之后的问题反思, 该项目使用了 JavaFx 作为 GUI 绘制，面向对象设计方法，通过 Nio 实现服务端和客户端的通信，文件读写实现游戏的读档和存档。

关键词 Java,JavaFx,Nio

1 开发目标

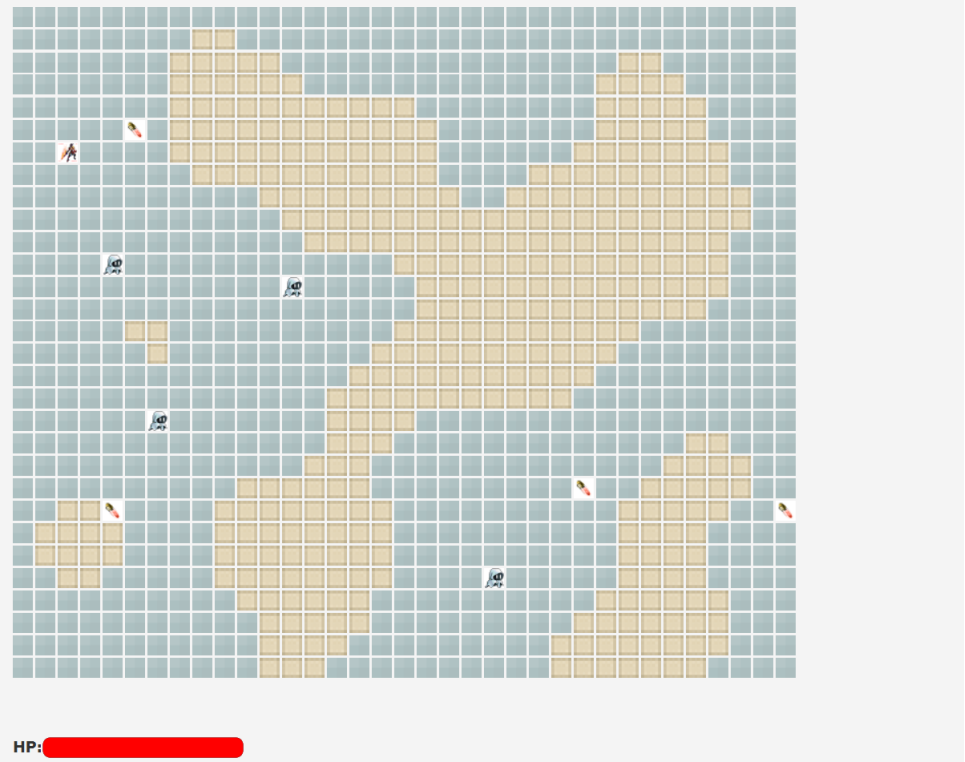
我的游戏是一个怎样的游戏?

就是那种劣质到不能算是小游戏的游戏，在单人模式下，玩家操控一个小人上下左右移动攻击地图中的怪物，可以存档和读档；在双人模式下，可以操控两个不同的角色进行对战。

游戏的灵感就是来源于作业四的 roguelike 分支，因为这样设计有很多优点，当你把 creature 的移动限制到格子的单位时，逻辑就会相应简单很多，双人模式下是想做出拳皇的那种感觉，设计了两个不同的角色，虽然很难看出来，但是大概定位成战士和法师，然后进行两个人的对战。

如上图所示，进入游戏后可以选择开始游戏以进行单人游戏，也可以点击读取存档继续上次单人游戏，每次单人游戏状态下结束游戏都会自行存档。

单人游戏如下图所示：



单人模式下玩家以一个人类英雄的形态通过上下左右移动，并可通过 WASDQE 等一系列技能攻击地图上的怪物，并可以拾取地图上的血包恢复血量。

多人游戏如下图所示:

玩家选择多人游戏后连接服务端（只可双人），并将两个玩家的操作同步直至一方血量为空分出胜负。

2 设计理念

我的代码没有什么设计理念，如果说有的话，基本都是参考了 roguelike 分支。这个游戏就是从 roguelike 扩展而来，因此基本延用了它原先的设计理念。当然，因为我自己的问题一定程度上破坏原来框架的设计思想，最严重的是我使得它的 model 和 view 不分离，原来的代码在 PlayScreen 中通过 3 个 display 函数很清晰地把绘图搞定了，但是很遗憾，在修改中我的代码逻辑和绘图纠缠在一起，因此给后来的代码编写和扩展带来了诸多不变。

当然，oop 的设计理念自然不可少，我一直认为使用面向过程的方式很难完成一个较大的项目，因为耦合性太高必然要束手束脚。此外，在本次代码中还尝试了一点函数式的编程，其实就是将集合转化成流来进行一系列操作。

3 技术问题

3.1 通信效率

按照要求，通信使用了 nio，当时遇到了很多挫折，主要集中在对 channel 注册时的四种状态的理解：简单可以概括为 write 是不太能注册的，connect 需要 finish：其实是因为这两个属性都会对 channel 形成一定的阻塞。相比于 bio,nio 的优点是可以减少线程开启，通过轮询的方式照顾多个 channel。

但其实通信的问题并不在这里, 而在于传个什么? 我使用的是 javafx 进行 GUI 绘图的, 在 javafx 中所有的类都继承于 node, 就是将 object 封装了一层，但是这个 node 并没有实现序列化，或者说 javafx 的控件基本都没有实现序列化。当时想了很多方法，因为如果可以，我甚至想把整个游戏本身 application 序列化传了，后来又尝试了给控件 transient, 用自己写的类继承它们完成序列化，都因为太麻烦放弃了。最后在同学的提示下使用了一个数据结构包装了坐标和图片的 URL 传输，但是这样的话，一些需要专属的状态很难传输，而其实游戏是十分需要这种动态状态数据的，后来这里我做的非常丑陋，勉强能跑。

最后谈一下 reactor 模式，我觉得它实现方法的灵魂之处在于事件驱动模型，就和鼠标监听和键盘监听类似，这时候我各开一个线程一直去监听发生了什么不仅需要处理的情况很多还很浪费资源，吃力不讨好，相反通过事件触发放入队列之后回调处理函数既解决了问题又只需要更少的线程。

本项目并没有使用严谨的 reactor 模式，而是采用了最基础的 nio 代码，这样没有线程池的帮助还是较容易因为某个 handle 函数过于复杂导致阻塞，但考虑到我的代码中只有两个客户端和一个服务端，需要处理的连接并不多每次传输信息量也不大，单线程的 nio 已经满足需求。刚开始只有服务端拥有 selector，客户端会每隔一段时间读取信息，后来我给客户端也增加了 selector, 理论上讲可以进行比较完美的相互通信。

3.2 并发控制

其实并发控制部分较为容易，我认为在这样的构架下并发争夺资源的机会并不多，主要是两个 creature 都想向同一个格子里走，同一个 creature 受到了多个攻击，我很粗暴的在行走和攻击的方法里 synchronized 了需要使用的资源（当时在这里主要遇到了锁一个类还是锁一个对象的问题）。

较为复杂的问题是网络连接下两个玩家输入的保序性，但是这个问题在上面得到很好地解决。因为是单线程的类 reactor 模式，玩家的输入自然保序（当然可能会因此有一点延迟）：需要在队列中先处理前一个用户的输入才会处理后面的输入，甚至都不需要 synchronized。

3.3 输入输出

项目本身并未对运行过程中的事件进行输入输出的日志记录，仅在服务端控制台打印了两个客户端的输入。

在存档和读档中对文件的写入和读取时可以多使用 BufferInputstream 这种装饰好的子类来通过缓冲区提高读写效率。写这里时探究了序列化的含义，发现之前的理解十分浅薄，“所有保存到磁盘的对象都有一个序列化编码号，当程序试图序列化一个对象时，会先检查此对象是否已经序列化过，只有此对象从未（在此虚拟机）被序列化过，才会将此对象序列化为字节序列输出，如果此对象已经序列化过，则直接输出编号即可。”但是以上的说法又只在同一个流中成立，至今还对序列化心存敬畏，虽然本项目中涉及不到太多，仅仅 implements 序列化接口，使用 ObjectInputStream 读入即可。

3.4 面向对象

通过面向对象设计, 可以将一个比较大的项目肢解成各个部分再拼接起来, 我认为就在这个“归”和“并”的过程中，好和不好的代码就区分了开来，因为面向过程是平铺的描述，但是打碎重组就多了无限的可能，同时分解下来的部分又使我们不用面对整个复杂的逻辑，此外，继承、封装、多态的特性很自然地方便了代码的编写。

4 工程问题

从 roguelike 的代码中学到了很多：当时看到 creature 和 creatureAI 的设计是不太能理解的，每一个 creature 都需要有 CreatureAI，然后 AI 控制 creature 的高级行为，creature 负责掌控生物的基本特性，这样的设计有一点桥接的感觉，除了语义上“灵”与“肉”之外，creature 更是一个较为底层的类，这时候选择继承 creature 不如使用耦合较低的聚合，使得同样的 creature 搭载不同的 AI；还有整个代码中随处可见的责任链，工厂等等。

后来在我修改的代码中尽力尝试模仿 roguelike 的代码，虽然 roguelike 代码逻辑简单，但感觉其中使用的工程方法，设计方法自然而巧妙，因为对设计模式了解不深，自己的代码并没有有意识地去使用什么设计模式。

5 课程感言及问题反思

课程很棒，内容很丰富。

我的代码虽然基本完成了要求，但是其实内部构架比较乱，虽然我一开始是非常想按照代码规范来完成一个低耦合高内聚的项目，可是做作业的过程中逐渐暴躁，然后放飞自我，反正我自己写完了不想再看。我觉得这样的问题主要是在写代码之前没有对代码有一个大致的规划，出现了问题之后又不会去寻找合理的重构方式，而是挑选一个最轻松的解决方法。

参考文献

1 https://njuics.github.io/java2021/

2 https://medium.com/coderscorner/tale-of-client-server-and-socket-a6ef54a74763

3 https://www.cnblogs.com/crazymakercircle/p/9833847.html

4 https://www.cnblogs.com/9dragon/p/10901448.html

Developing a Java Game from Scratch

Jzx[1]

1. *Nanjing University , Nanjing , China*
E-mail: 1214069454@qq.com

Abstract This paper mainly describes the development objectives, design concepts, technical problems, engineering problems, course speeches and problem reflection after writing Java assignment 8. The project uses JavaFX as GUI drawing, object-oriented design method, realizes the communication between server and client through NiO, and realizes the file reading and archiving of the game through file reading and writing.

Keywords Java,JavaFx,Nio