中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

马润泽

南京大学, 南京 210023

E-mail: 1433638143@qq.com

摘要 本文为南京大学 2021 秋季学期高级 Java 程序设计大作业及结课报告，其内容主要有大作业的开发目标、设计理念、技术问题、工程问题及课程感想。

关键词 Java, 程序设计, 问题与解决，课程报告

1 开发目标

1.1 基本介绍

我所设计的游戏为一个 2D 射击小游戏，具有单人模式和多人模式两种游戏方式。单人模式为 PVE（玩家对抗电脑），玩家在怪物的阻挠下通过迷宫即获得胜利。多人模式为 PVP（玩家对抗玩家），玩家需要在其他玩家和怪物的攻击下存活下来，活到最后的玩家即为胜利者。

两个模式下的操作方式相同。玩家通过 WSAD 键分别控制人物进行上下左右的移动，通过 J 键让人物向当前方向发射一颗子弹。按下 ESC 键则可以使得游戏暂停并进入暂停界面，在其中可以选则保存当前游戏地图或者游戏存档。在暂停界面按下 ESC 键或选择 Back 选项返回游戏界面，在游戏界面按下 ESC 键继续当前游戏。

图 1 单人模式

Figure 1 Single Player

图 2 多人模式

Figure 2 Multi Player

1.2 灵感来源

游戏灵感来源于 4399 游戏上的一款 flash 小游戏坦克动荡，其玩法为多个玩家每人操控一辆坦克，在迷宫中通过各种各样的道具及子弹与墙体间的弹射效果，剿灭其他玩家从而活到最后，成为胜利者。

我开发的游戏由于地图的基本构造为二维数组，无法实现任意角度的子弹发射及弹射的效果，因此只能退而求其次，使子弹在两个方向上弹射。为了更好的显现出这一点，我将迷宫从原来的一格宽改为现在的两格宽，使得子弹弹射更加明显，也增加了躲避的机会。由于地图的每一格只能存储一个对象，因此借鉴了坦克大战的玩法，添加了子弹碰撞后相互抵消的机制。道具的机制依旧存在，但只实现了回复生命值与增加攻击力两种，且道具需要击杀怪物才能有几率掉落道具。在多人模式中，你可以通过击杀怪物来获得强化自身的道具，但也要背负被怪物击伤甚至击杀的风险，还有提防别的玩家，不是很有趣吗。

2 设计理念

总体设计上，最终的大作业沿袭 jw05 的框架，按照包分为默认包，世界包 world，屏幕包 screen，生物包 creature，线程包 thread，迷宫生成包 maze 以及提供的图形化工具包 asciipanel，在多人游戏的实现中又增加了一个网络包 net。这样的分包使得项目文件结构清晰明了。

2.1 default package

其中只有一个 ApplicationMain 类，其中包含主函数，为整个程序的入口，负责一些初始化的工作，例如窗口的创建，屏幕刷新线程的启动，初始屏幕的创建等。

2.2 world

world 包下的 UML 图如下：

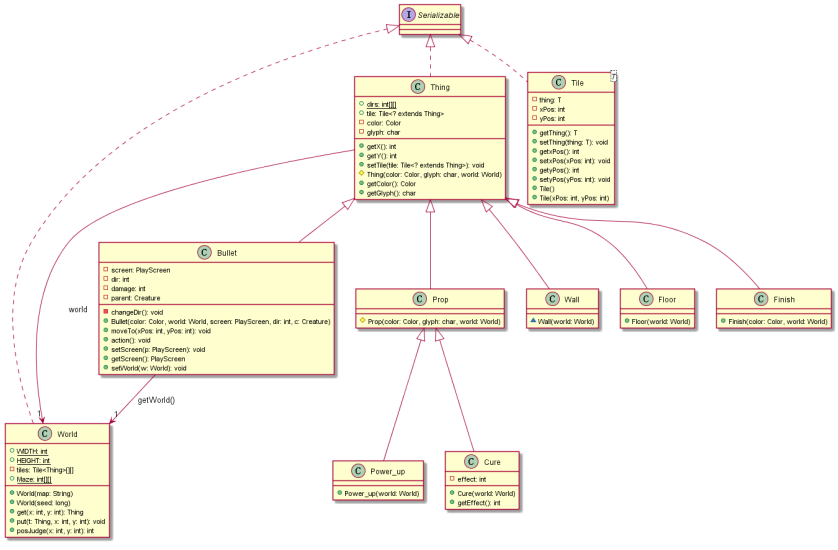


图 3 世界

Figure 3 world

world 包下最基本的三个类为 Thing 类，Tile 类和 World 类。游戏中任何一个物体，不管是生物还是诸如地板、墙体之类的地形，都是 Thing 类的子类，其规定了该物体的颜色、字符、所在 World 以及所对应 Tile。而 Tile 类可以理解为格子，每个物体都需要在地图中占据一个格子，就像在植物大战僵尸中每个植物都种植在一格上，Tile 类主要规定物体的坐标。World 类则就是地图，为一个 40*40 的 Tile 类二维数组，首先通过迷宫生成包生成仅有墙体和地面的地图，再在后面游戏的初始化与运行中不断加入与删去其他物体，World 类最重要的就是 posJudge 方法，其通过输入 x，y 参数可以返回不同的整型值来告诉我们位于坐标 (x，y) 的 Tile 中是什么类型。

剩余的类，Wall 类即为墙体类，Floor 类即为地面类，这两个类就不再赘述。Finish 类即为终点类，在单人模式中玩家需要通过迷宫并移动至该类所在点来获得胜利。Prop 类即为道具类，为治愈道具类 Cure 和力量增强道具类 Power-up 的父类。Bullet 类为子弹类，其中有移动、碰撞检测等方法。

2.3 screen

screen 包下的 UML 图如下：

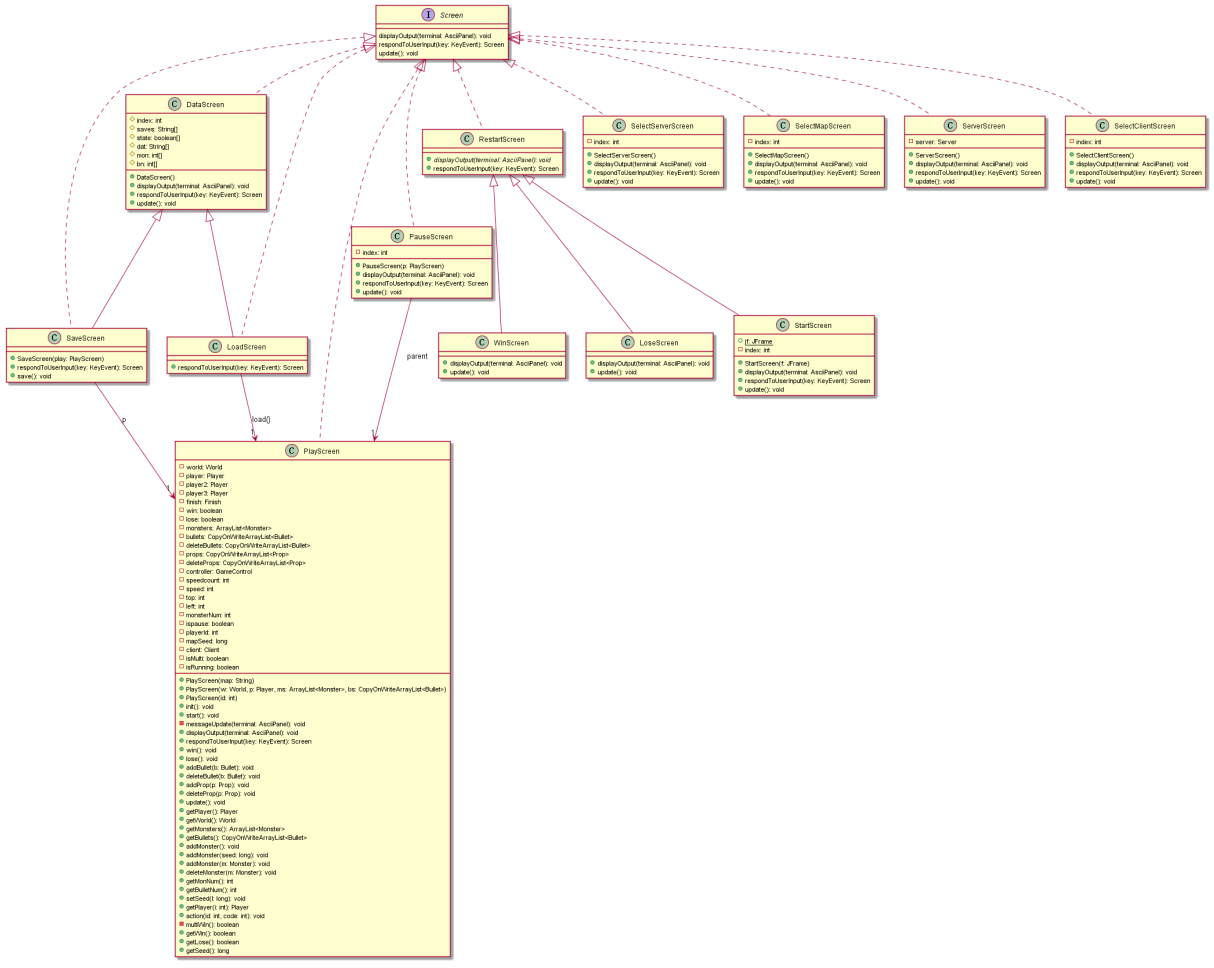


图 4 屏幕

Figure 4 screen

在原先的框架下我又增加了很多屏幕，这样虽然麻烦，但是使得整个游戏在选项方面更加的富有条理，例如增加 LoadScreen 类和 SaveScreen 类使你可以选择哪个存档位进行加载或存储，增加 PauseScreen 类作为暂停界面使你可以选择在暂停时是进行存储地图还是存储游戏，增加 SelectServerScreen 类使你可以选择是作为房主开服务器还是作为客户端接入服务器，增加 SelectServerScreen 类使你可以选择是作为几号玩家接入服务器。

2.4 creature

creature 包下的 UML 图如下：

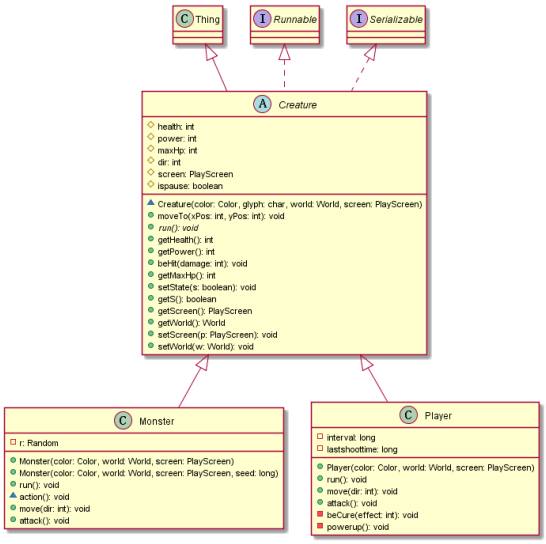


图 5　生物

Figure 5　creature

由于整个游戏中根本用不到 Creature 类而只使用其子类 Player 类和 Monster 类，我将 Creature 类设置为虚类。这样的设计使得后序开发很容易再添加一些别的种类的生物，也更加符合面对对象的设计思想。Player 类和 Monster 类中都包含一些基础的方法，例如移动、攻击等等，但两者方法又不同，例如玩家和怪物对道具有不同的反应方式。

2.5 thread

thread 包下的 UML 图如下：

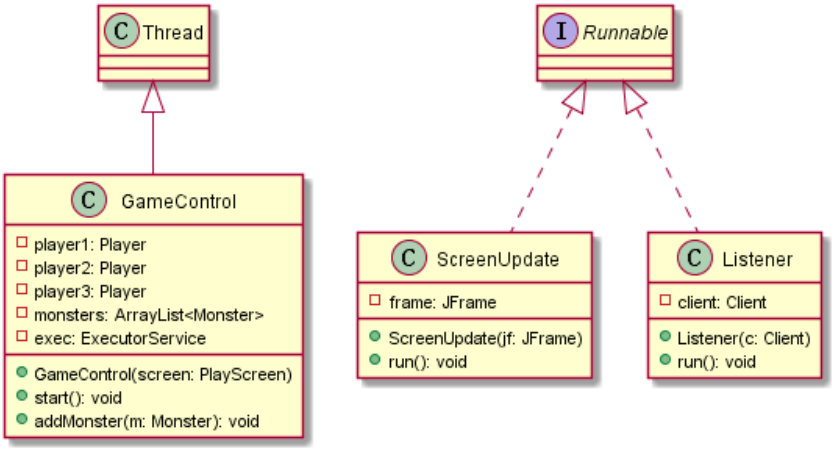


图 6　线程

Figure 6　thread

线程包包括 Gamecontrol 类，ScreenUpdate 类和 Listener 类。

GameControl 类为游戏运行中控制各个线程包括玩家与怪物的类，使用线程池机制来管理线程，使用线程池为 newCachedThreadPool，即可以无限扩大的线程池。尽管线程池可以无限扩大，但运行线程不宜过多，因此我没有将每一颗子弹也都作为一个线程，而是将所有的子弹在地图刷新时一起刷新，如此大大减轻了 CPU 的负担。

ScreenUpdate 类为屏幕刷新类。之前在 QQ 群中受老师启发，才明白游戏屏幕刷新的实际原理。因此将屏幕刷新独立成一个线程，在程序入口就开始运行，并已 20fps 的速率不断刷新屏幕。

Listener 类为多人游戏中监听服务器发送来的信息的类，在多人游戏中不断令客户端进行读取客户端信息并执行，以此来实现多人之间的同步。

2.6　maze

迷宫生成包，即在 jw04 中所提供的迷宫生成算法。在其基础上增加了固定随机数种子的生成方式（网络通信中帧同步的需要），并将迷宫从一格宽拓展至两个宽，除此外没有什么变化，就不再赘述。

2.7　asciipanel

图形化工具包，为 jw04 中提供的图形化工具，其具体原理为通过将图片分割为一格一格的图标，并将它们与字符对应起来，以此来达到通过字符显示对应图像的效果。

2.8　net

net 包下的 UML 图如下：

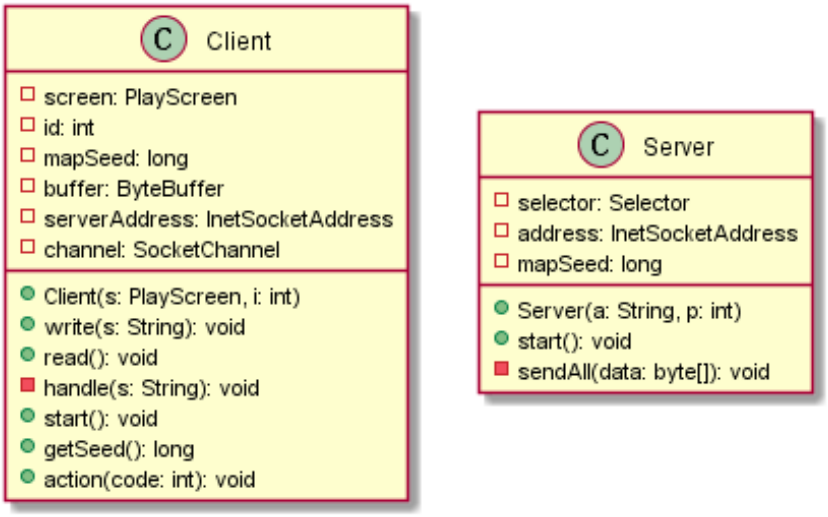


图 7　网络

Figure 7　net

网络通信方面采取帧同步的方式进行实现，由于帧同步中服务器不作任何逻辑处理操作，仅需转发客户端操作即可，因此帧同步的服务器压力小，效率高，流量小（最重要的是开发简单）。服务器 Server 类和客户端 Client 类使用 NIO Selector 机制建立通道进行通信。为了实现帧同步，在服务器向其他客户端转发操作的同时，也需要保证一些初始化的相同，例如迷宫地图的生成以及一些随机问题，例如怪物有几率掉落道具的同步。因此在服务器中存储一个随机数种子，在客户端连接至服务器时立即向服务器请求该种子，得到种子后将其作为地图生成、道具掉落几率判断的随机数种子来实现帧同步。

当然帧同步也不是没有缺点，帧同步安全性比状态同步低，容易造成作弊行为，也就是常说的开外挂，其次在网络通信中状态同步的断线重连问题也要比帧同步简单许多。

3　技术问题

3.1　通信效率

老实说网络通信这块确实很令人头疼，花费了我大量的时间，其中最坑的是我如果处于科学上网的状态，客户端会有概率卡死连接不上服务器，我当时排查查了很久。

在提高通信效率这方面，尽管为了实现存档功能，Player 类和 World 类都已实现了 Serializable 接口，使得它们可以序列化来在网络上传输，但不得不说这数据量太大了。我完全不需要让服务器广播一张地图或是广播一个玩家对象来进行帧同步，我只需要将其状态的改变提取出来，将其发送出去，并在读取时加上对应处理即可。例如让三个客户端生成同样的地图，我不用广播一整张地图，仅需广播其地图生成的种子即可，而对于其中某一个玩家移动的状态改变，例如玩家 1 向下移动了一格，那么我就广播一个“action 1 83”的字符串，用空格分隔，action 表示玩家的行动，1 表示玩家 id 为 1，83 则表示行动对应的键盘码为 S 键，这样，在其他客户端接收到这样一条字符串时，只需将其解码并执行对应操作即可。

3.2 并发控制

并发方面在前面设计介绍的线程包中已经介绍过，使用线程池来管理。对于玩家与怪物的移动与攻击操作，都需要获取当前地图上某一位置的情况，其中存在临界问题，因此将玩家与怪物的移动方法 move 和攻击方法 attack 都使用关键字 synchronized 来修饰，以此解决临界区问题。

3.3 输入与输出

地图与游戏的存取都涉及输入输出问题。对于地图的存取，无需使用序列化接口存储整个 World 类，地图归根结底只有墙体和地板，仅需将作为一个 01 矩阵存储在 txt 文件中。也不需要在为地图的存储槽位添加界面，可以直接使用文件对话框 FileDialog。对于游戏的存取，使用序列化接口存储整个对象，需要注意到是序列化一个对象需要使得该对象的所有属性都满足序列化的条件，而 Player 类和 World 类中都存有 PlayScreen 属性。将 PlayScreen 序列化，其一数据量过大，其二其属性 GameControl 中包含的线程池等不能序列化，显然弊大于利，因此 PlayScreen 不能序列化，我们在 Player 类和 World 类中对 PlayScreen 属性使用关键字 transient 进行修饰，使其在序列化过程中被忽略。这样就带来另一个问题，读取获得的对象也不会拥有 PlayScreen 这个属性，因此在读取存档时，读取 Player 类和 World 类后还需重新将他们与新的 PlayScreen 建立关联。

3.4 面对对象设计方法的好处

面对对象设计方法使得程序设计模块化，在开发中增加了代码复用，例如在实现了 Creature 这个虚类后，我可以将它作为模板创建很多不同的生物而不用再重复实现那些相同的方法。

面对对象设计方法还提高了程序的可维护性，例如使用 JUnit 进行单元测试，我们在测试代码时可以一个类一个类进行测试，甚至一个方法一个方法进行测试，而不是将程序整体运行一遍进行测试。

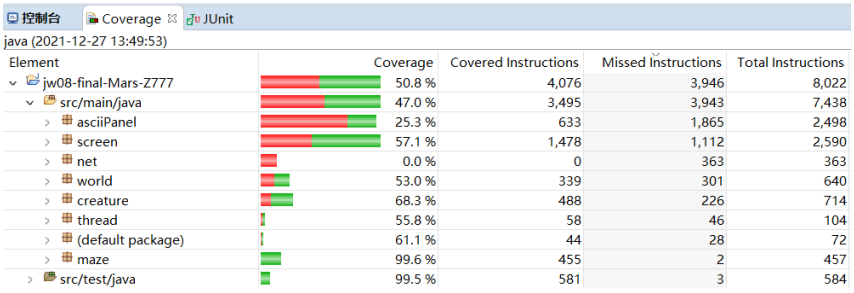
4 工程问题

4.1 依赖管理和自动化构建

使用 maven 来进行依赖管理和自动化构建。

4.2 单元测试

使用 junit4.12 来进行单元测试，工程总覆盖率为 50.8%。


控制台 Coverage JUnit

java (2021-12-27 13:49:53)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
jw08-final-Mars-Z777	50.8 %	4,076	3,946	8,022
src/main/java	47.0 %	3,495	3,943	7,438
asciiPanel	25.3 %	633	1,865	2,498
screen	57.1 %	1,478	1,112	2,590
net	0.0 %	0	363	363
world	53.0 %	339	301	640
creature	68.3 %	488	226	714
thread	55.8 %	58	46	104
(default package)	61.1 %	44	28	72
maze	99.6 %	455	2	457
src/test/java	99.5 %	581	3	584

图 8 单元测试

Figure 8 JUnit test

进行单元测试不仅对我所写的类进行了测试，同时也规范了我的代码，例如对于某个类中的私有属性，我只实现了如何改变这个属性却没有实现如何获取这个属性的值，这样的方法使得我很难进行单元测试用例的书写，到最后还是要实现获取属性的方法。

5 课程感言

这门课真的能学到很多，老曹也是一位很认真很负责的老师。学习一门语言真正的方法还是要去实践，虽然这门课的作业有些挑战，上课有时候会听的一知半解，但等到你亲自在作业中实现一遍后，你就算一开始不懂也会对它有一个大致的思路。还有，学的越深越觉得自己当初写的代码的丑陋。