中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

曾庆扬[1]

1. 南京大学仙林校区，南京 210023
E-mail: 191220004@smail.nju.edu.cn

摘要 Java 高级程序设计是南京大学计算机科学与技术系面向高年级本科生开设的课程，课程大作业要求完成一个完整的图形化网络对战游戏. 本文系统阐述作者完成此大作业的全过程，包括开发目标、设计理念、技术问题、工程问题和课程感言等.

关键词 Java, Netty, NIO Selector, Java Swing, Thread, Java IO

1 开发目标

1.1 游戏内容

本项目以游戏 QQ 堂为参考，实现多玩家对战的游戏场景.

首先启动游戏服务端，然后启动客户端. 游戏房间要求达到四个玩家才能开始游戏，当房间人数未达到要求时，客户端界面显示”Waiting Connection”，若房间人数已满，则尝试连接服务器时会提示连接失败. 游戏开始时四个玩家分别位于竞技场的四个角落. 玩家对应的游戏人物根据玩家连接服务器成功的顺序给出，默认第一个玩家位于地图左上角，第二个玩家位于右上角，第三个是右下角，第四个是左下角. 玩家通过键盘的 ↑↓←→ 键操纵人物移动，空格键可放置炮弹. 竞技规则为 2V2，默认竞技场中左上角与右下角的玩家为一队，左下角与右上角的玩家为一队，玩家的目标是把对手都炸死，最后剩下的玩家所在的队伍胜利. 每个玩家有两个炮弹，对炸弹的使用权为玩家放置的炮弹爆炸后，玩家的炸弹数就得到恢复.

1.2 灵感来源

此项目的前身为 jw05 的炸弹人游戏，在写 jw05 时，看到有很多同学都在写射击类型的游戏如坦克大战等，但作者早在两年前的 C++ 高级程序设计课上完成了坦克大战的项目设计，才思枯竭之时想着换个法子实现炮弹，然后就想到了经典的炸弹人游戏，顿感才思泉涌，于是 jw05 完成了单机游戏炸弹人 FC.

引用格式: 曾庆扬. Developing a Java Game from Scratch. 中国科学: 信息科学, 在审文章
Zeng Qingyang. Developing a Java Game from Scratch. Sci Sin Inform, for review

而 jw08 需要图形化和网络对战，与单机版的炸弹人相比，QQ 堂的游戏逻辑和游戏素材显然更加符合要求，二者的游戏内容也有大部分的相同之处，使代码原框架能够尽可能保留，因此 jw08 改为实现 QQ 堂.

1.3 效果预览

游戏录屏发布在 B 站上,可点击右方链接跳转:https://www.bilibili.com/video/BV1FD4y1w7w4

部分相关截屏如下

图 1 对战场景

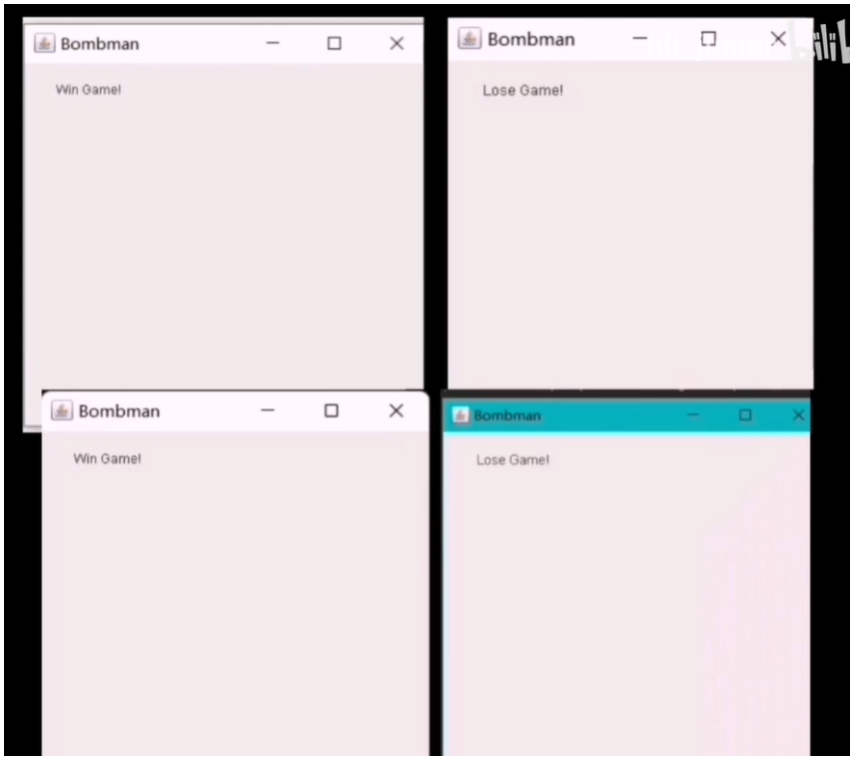

图 2 游戏结果

2 设计理念

2.1 项目框架

src/main/java 中的代码架构

1. client: 客户端
   - Client: 实现客户端的框架
   - ClientHandler: 实现客户端管道的初始化，信息处理和异常捕获
2. server: 服务端
   - Server: 实现服务端的框架
   - ServerHandler: 实现服务端管道的初始化，信息处理和异常捕获
   - ServerMain: 服务端的 main 函数所在的类，启动游戏服务器，控制游戏总流程
3. marshalling: 编码器和解码器
   - MarshallingCodeFactory: 创建 Jboss Marshalling 解码器和编码器，用于通信
4. request: 各种类型的请求消息实体

- LoginRequest: 请求登录
- ResourcesRequest: 请求更新地图资源
- PlayerMoveRequest: 请求移动
- BombRequest: 请求放置炸弹
- CloseRequest: 请求关闭客户端

5. response: 各种类型的响应消息实体
   - LoginResponse: 登录请求响应
   - StartGameResponse: 游戏开始响应
   - ResourcesResonse: 游戏资源响应
   - LoseGameResponse: 游戏失败响应
   - WinGameResponse: 游戏胜利响应
6. screen: 客户端的游戏界面
   - GameFrame: 游戏主框架，客户端 main 函数所在的类，控制玩家游戏流程
   - MyPanel: 游戏 UI 绘制更新
7. world: 游戏世界中的实体
   - Bomb: 控制炸弹的爆炸和对玩家的伤害等
   - Creature: 游戏角色的基类，各种特殊的角色可以在此之上创建 (但此次本项目只实现了一种类型的角色，故没有进行继承)
   - Dir: 枚举类，定义上下左右四个方向
   - Factory: 工厂类，用于生成新玩家和新炸弹
   - Tile: 枚举类，定义地图中的地砖、墙壁、边界等
   - World: 游戏资源的载体，存储地图、人物和炸弹等的坐标信息
   - WorldBuilder: 创建游戏初始地图

2.2 设计优势

- 代码复用: 在 jw05 的框架上进行修改, 代码复用度高, 且 jw05 的框架来自于 jw04 的 Rougelike 框架，保留了该框架原有的高质量代码设计.
- 高内聚度，低耦合度: 基于面向对象设计的想法，在设计项目时准确划分各个 package 模块，保证各个模块下的类有较高的聚合度，而各个模块之间的耦合度尽可能低，对编写单元测试也带来了方便.

3 技术分析

3.1 设计模式

3.1.1 单例模式

类 ServerMain 和类 GameFrame 分别作为服务端和客户端的主框架，在生成这两个类时使用了单例模式，目的是为了保证在一个进程中，该类只能有且仅有一个实例。在类的内部用静态字段来引用唯一创建的实例，赋予 public 属性则可以让外部调用方获得这个唯一实例.

```
1    //在GameFrame的构造函数中生成单例对象
2   public static final GameFrame INSTANCE = new GameFrame(320,320,32);
3   //在ServerMain的构造函数中生成单例对象
4   public static final ServerMain INSTANCE = new ServerMain();
```

3.1.2 工厂模式

在创建游戏角色、炸弹等实体的对象时，World 类的相关函数调用了工厂类 Factory 中创建相关实体对象的函数. 使用工厂模式的好处是在创建对象时不会对客户端暴露逻辑，并且是通过使用一个共同的接口来指向新创建的对象.

```
1   /**  World类中的addPlayer函数
2    * pram:id 玩家的标识id
3    */
4   public void addPlayer(int id){
5       Creature player = factory.createCreature(id);
6       this.players.put(id,player);//players为存储player对象的集合
7    }
```

3.1.3 Reactor 模式

Reactor 模式为本项目网络通信中使用的模式，为了提高网络通信效率，学习并使用了 Netty 框架，Netty 是一个典型的多线程的 Reactor 模式的使用，在这个模式里，mainReactor 只有一个，负责响应 client 的连接请求，并建立连接，使用一个 NIO Selector. subReactor 可以有一个或多个，每个 subReactor 都会在一个独立线程中执行，并且维护一个独立的 NIO Selector. 这种设计的好处是让 subReactor 执行一些比较耗时的 IO 操作，减少 IO 等待时间.

在 Server 端，bossGroup 线程池对应 mainReactor，workGroup 线程组对应 subReactor，再定义一个 DefaultChannelGroup 类的对象 clients 来存储所有的 Channel, 达到向所有客户端广播消息的目的.

客户端与服务端的信息交换流程为：客户端每次发送请求消息到服务端，则服务端收到并处理请求消息后，将响应消息广播给所有的客户端，客户端收到响应消息后可根据消息的标识和消息的

类型来决定是否需要处理或丢弃.

客户端负责发送请求消息，接受响应消息，除了自身的 id 号，在整个游戏过程中不存储任何游戏中的实体数据，这样做的想法是现实中某些玩家可能会对游戏的数据进行恶意修改，即开挂.

服务端负责接受请求消息，广播响应消息，游戏中的实体对象由服务端进行创建和管理.

具体的游戏通信流程为:

1. 玩家启动客户端: 发送请求登录消息到服务端.
2. 服务端为玩家分配 id 号 [1-4], 将 id 号传回给服务端.
3. 游戏开始后，服务端每隔一定的时间 (33ms，即一秒大约发送 30 帧) 就将更新的地图资源广播给各个客户端，地图资源包括游戏地图和所有玩家及所有炸弹的实时状态.
4. 客户端每次收到地图资源的更新消息，就重新绘制游戏屏幕.
5. 玩家每次按下键盘的按键准备操纵人物的移动和炸弹的放置时，客户端就会发送按键的信息和对应的玩家 id 给服务端，服务端对此类请求消息做处理，完成对玩家移动状态修改或新炸弹的创建.

3.2 并发控制

3.2.1 线程创建

服务端创建了一个定期更新游戏资源的线程，在游戏开始时生成，在游戏结束时销毁

对于游戏实体，每一个炸弹都是一个线程，而游戏人物由于和客户端的进程能够相对应，则不作为线程处理.

3.2.2 synchronized

对于每个人物的移动和每个炸弹，考虑线程 race condition.

首先是两个物体不能占据同一个 tile，物体每次准备移动到 tile 上时，会先访问 tile 是否空闲，该函数由 World 类管理，则对该函数加 synchronized 关键字.

其次是每个生物受到的多次伤害是有顺序的，虽然在现在的游戏中一个炸弹就能杀死一个游戏人物，但若设计了需要伤害几次才能杀死的生物，则可以对生物的 getAttack 函数加 synchronized 关键字，保证线程的同步执行.

3.2.3 volatile

每个炸弹在爆炸时，如果在它爆炸的范围内正好有其他炸弹，则多个炸弹应当同时爆炸，此处涉及到多个炸弹线程的通信，先爆炸的炸弹需要修改它引爆的炸弹的状态，让炸弹能够提前爆炸.

修改线程状态可以使用 volatile 关键字, 它能够使线程读取到最新的状态值, 因此, 将 Bomb 类中的私有布尔型变量 explode 加 volatile 关键字，explode 的初始值为 false, 若该定时炸弹完成倒计时，则 explode 为 true, 亦或被其他炸弹检测到时，其他炸弹会将该炸弹的 explode 状态改为 true，此时线程能够迅速检测到状态的修改，从而提前退出倒计时，完成爆炸.

3.3 输入输出

3.3.1 地图保存和读取

服务端启动时会先尝试加载地图文件 World.data，若地图文件不存在，则新建一个地图并保存到 World.data 文件中，若文件存在，则直接读取. 想法是如果要继续优化这个游戏，则可为服务端提供多份不同的地图进行挑选. 这里使用的 IO 操作是 FileinputStream/FileOuputStream 和 ObjectInputStream/ObjectOuputStream, 可以将创建的 World 类的对象（成员变量为地图长度和宽度，各个 tile 的具体类型等）直接保存到 World.data 文件中，也可以从 World.data 中直接读取整个文件的数据并赋值给 world 对象.

3.3.2 Serializable

在客户端与服务端的网络通信中，发送方需要将准备发送的数据实体进行编码，接收方需要将接受到的数据进行解码，在编码和解码时使用了 java 的序列化，即对每个消息实体进行 Serializable 接口的实现.

3.4 图形化

使用 Java Swing 完成简单的图形化，首先准备游戏素材，本次项目的 2d 图片素材均来自https://www.aigei.com, 像素控制在 32px*32px.

Swing 组件的 JFrame 用来设计界面，JPanel 是界面中的面板，Graphics 是面板中的画笔.

项目中的 GameFrame 类继承了 JFrame, 在初始化时就创建了游戏窗口，MyPanel 类继承了 JPanel, 并添加到 GameFrame 的窗口中进行绑定，当 MyPanel 每次调用 paint 函数时，会使用 Graphics 的 draw 函数来进行绘制.

4 工程问题

4.1 项目构建

项目使用 MAVEN 作为自动构建工具，添加了 netty 和 marshalling 依赖包.

在打包项目导出 jar 包时，亲测 marshalling 依赖包需要添加到工程的 jar 包中才能使 jar 包正常运行，而 maven 自动生成的 marshalling 会失效（错误原因不详）.github 上提交的项目代码中，out/artifacts/Bombbman.jar 是按要求打包好的 jar 包，可以在 java8 及以上的环境下使用.

4.2 单元测试

使用 JUnit 进行单元测试，主要测试 world package 下各个游戏实体的函数.

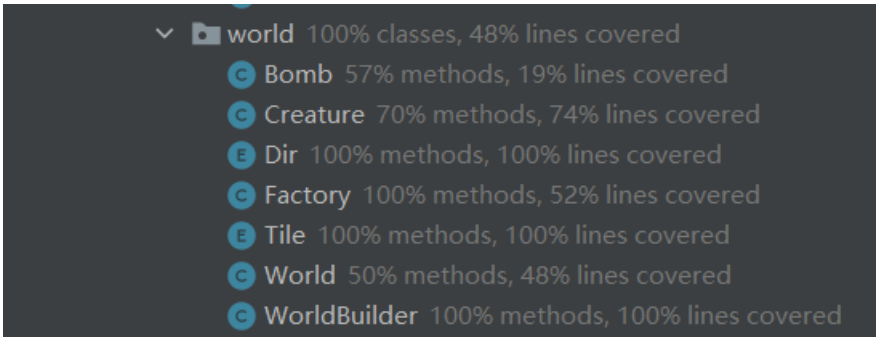

图 3 覆盖度测试

5 项目优化

- 添加更多的游戏元素：不同人物的设计、游戏地图的改进、增加更多道具等，增强可玩性.
- 改进用户界面友好度，现在的服务器 ip 地址是写死的，若要修改 ip 地址则需要修改源码，更合理的方式应是让客户端能够在界面上更改要连接的 ip 地址.
- jw08 不是在 jw05 的项目里优化生成的，而是一个借鉴 jw05 并从零做起的项目，如果能够把单机的炸弹人和联机的 QQ 堂二者融合到一起应该是更好的，游戏内容也会更为丰富.

6 结语

JAVA 高级程序设计是一门内容丰富而具有挑战性的课程，曹老师上课讲解的很多知识其实不仅仅是在传统的 Java 层面，让我对程序设计和程序开发有更多新的认识和掌握. 做项目时则更加能感受到只有亲自动手才能熟练掌握 Java 中的各种语法和机制，在调试 bug 时才能对很多复杂的问题有深入的了解，甚至是最后的项目报告的编写，也要花一定的时间了解没有使用过的 latex 语法等等，也许优秀开发者的素养就是在这种略为痛苦的过程一点点累积起来的吧.

致谢 曹春