中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

刘京龙[1]

1. 南京大学仙林校区, 南京210033

E-mail: 191220066@smail.nju.edu.cn

摘要 本文是java结课作业：从头开始实现一个java游戏的报告。此Java游戏是坦克大战游戏。游戏中实现了多人联机、断线重连等功能，并编写了相应的测试用例。旨在通过开发游戏，进一步理解和掌握java Socket编程、输入输出、并发控制、单元测试等编程技巧，培养良好的编程风格。采用JavaFx图形界面，没有使用任何游戏引擎。

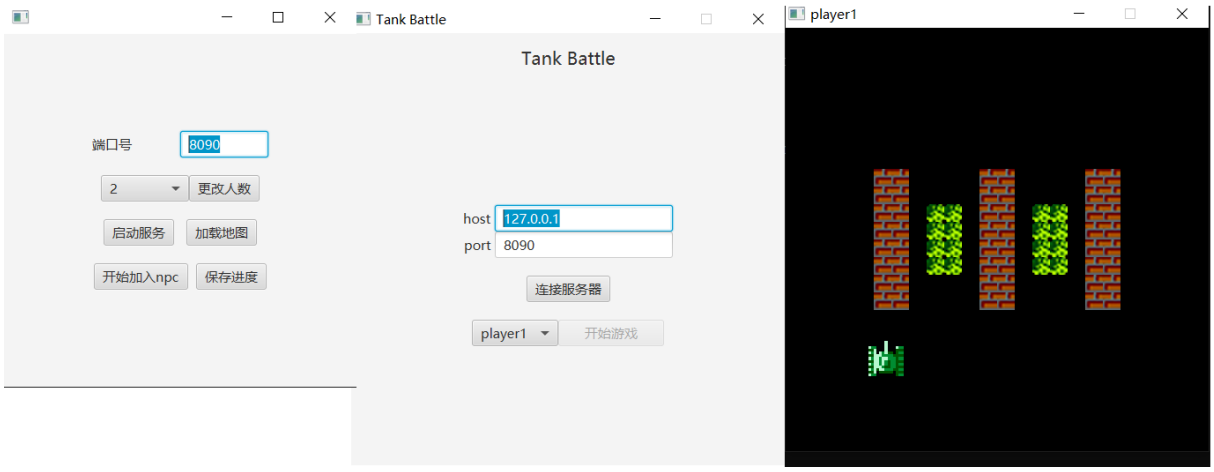
关键词 Java, JavaFx, 网络对战游戏, IO,C/S架构,Selector

1 开发目标

本次开发目标是完成一个可以多人联机的Roguelike网络对战游戏。结合作者经历和之前的开发经验，将具体游戏内容定为经典坦克大战游戏。灵感来源为4399 坦克大战

2 项目成果

完成了一个可多人联机、可保存进度的坦克大战游戏。



代码已经提交到https://github.com/jwork-2021/jw08-final-Jinglong-Liu

project目录为项目主目录

可执行jar包也已经发布。在project/target目录下，或者release分支下。

使用maven管理项目。在project目录下使用mvn package命令即可运行测试用例并将项目打包成两个包含依赖文件的jar包（client.jar,server.jar），并生成测试覆盖结果。

分为服务器和客户端两个程序。

首先运行服务器，启动服务，此时的地图为默认的地图。

然后启动(¿=2个)客户端，输入连接参数，点击连接按钮，完成连接。

然后选择player登录，登录成功即进入游戏。

按键ASDW移动，J发射子弹。子弹可以攻击别的玩家

玩家之间碰撞也会造成伤害。

玩家颜色随生命值变化。玩家生命值归零即告失败。当场上仅存活一个玩家时，该玩家即为胜者。

背景地图中，砖块可以被子弹或坦克撞击摧毁。

此外服务器可以设置或修改人数上限，且提供了短线重连和服务器维护功能。

1、当游戏进行中，客户端断开时，客户端可以快速重连，并以上一次的ID登录，继续未结束的游戏

2、服务器可以保存进度，服务器正常关闭也会自动保存。且当游戏进行中，服务器断线时。所有客户端收到提示。服务端再次启动时，可以选择加载地图，恢复到上一次的游戏进度，玩家登录继续游戏。

服务器端还可以增加NPC，增加可玩性。

游戏演示视频已经发布https://www.bilibili.com/video/BV1hY411p7mz?

3 设计理念

代码采用maven进行管理，图形界面采用JavaFx。为方便起见，暂时不适用css优化界面，所有界面采用java 源码编写。项目采用C/S架构，设有1个服务器，n个客户端。服务器与客户端的界面和逻辑分开编写。而游戏的基础类信息(元素等信息，可以传递)两者共享，可以在服务器与客户端装箱-拆箱传递对象整个游戏不使用任何现成游戏引擎。

4 技术问题

4.1 面向对象设计

项目采用面向对象的方式编写，将地图、地图元素、玩家、客户端与服务器等事物设计成类和对象，达到对实际的模拟。基础抽象类Thing,表示地图中的一个元素。含有以下属性：位置(x,y),图片，大小(宽度和高度)：这些信息用于描述在地图中的状态，可以据此绘制元素。攻击、防御、生命、方向，地图的引用。这些信息用于描述在游戏中的状态，用于游戏中逻辑处理与状态判断。

主要含有以下方法：(略去getter-setter等方法) modifyHp(int):生命值变化，以及状态(图片，是否仍然存在)的变化。一般考虑到如果hp¡=0。就将自己从地图中移除。moveBy(double dx,double dy):描述对象有移动意愿时，发生的变化。默认实现为移动（dx,dy距离，在子类中考虑碰撞处理，出界问题）intersects（Thing thing)：用于判断两对象是否发生了碰撞（位置有交集）

Thing的子类主要有坦克（Tank）、子弹（Bullet)以及地图上的各种元素（砖块，草地等）。Thing的所有子类要重写setImage()方法，设置各自的图片，并至少在构造方法调用一次。坦克、子弹更需要根据当前的方向设置不同的图片。坦克类重写了Thing的moveBy方法，逻辑为如果目标点没有和别的对象碰撞，或者碰撞对象是草地，那么前往目标点，否则攻击一次目标。另外如果目标点出界，就产生动作。坦克类增加了shoot()方法，表示发射一颗子弹。玩家类在坦克类的基础上，增加了playerId属性，用于区分不同玩家；isOnline属性，判断是否在线。此外还增加了setColor方法，根据当前生命值，设置不同颜色的图片。在modifyHp（）中调用。子弹类重写attack()方法，与坦克类基本类似，区别在于子弹是一次性的，子弹攻击别人后，自己也销毁。还要重写moveBy，在出界时销毁。此外，子弹类还实现了Runnable接口，子弹一旦被创造，将独自开启一个线程运行，不受主程序或坦克控制；子弹被销毁时销毁线程。具体在[并发控制]部分介绍。

地图类World,保存所有上述的对象。用并发对象CopyOnWriteArrayList保存。绘制地图只有遍历这个列表，绘制所以对象即可。此外为调用方便还将所有Player单独设一个列表保存。World实现restart()方法，设置一个初始状态(默认地图，清空坦克和子弹)。每次重新开始时，调用这个方法即可。

4.2 网络通信

本游戏采用selector，nio实现服务器-客户端一对多异步通信。服务器启动时，在selector上注册为可接受,一旦有客户端连接，将频道(Channel)注册为可读，可写。此后检测到可读，读取信息并处理，并准备好待写的信息。注册可写；检测到可写，写入准备好的信息，注册为可读。可以同时异步处理多个客户端的信息。具体读写内容逻辑：所有要读写的对象实现Serializable接口用于序列化。因此上一部分讲到的World和World里的所有元素，都要实现Serializable接口。此外实现了辅助方法类ByteUtil，用于对象与ByteBuffer之间的转换。每次需要写入信息时，先封装为一个实现了Serializable接口的对象，转为ByteBuffer写入；每次读取信息时，先读入一个ByteBuffer，再转化为不同的Serializable对象，提取信息进行处理。

在本工程的设计中，为了较好的实现同步，采用瘦客户端的方式，也即客户端通过键盘鼠标传递请求给服务器，服务器接受不同客户端信息，进行处理，并将新地图和状态传回客户端，客户端根据收到的信息重绘地图。也即采用状态同步，不采用帧同步。如此逻辑较为清晰，并发控制只要在服务器端处理，客户端不处理逻辑，只负责发送请求和绘制地图。

游戏逻辑如下：客户端选择id，发送登录请求；服务器判断是否已经登录。如果已经登录，发回错误信息，不然设置登录状态，发回成功信息。客户端收到成功信息，进入游戏界面。此时启动一个线程，每隔一段时间发送状态请求。服务器每收到一个状态请求，发回一个地图。客户端收到地图，更新地图。另有一个线程负责绘制地图，每秒30帧避免卡顿。客户端按下键盘，发送信息给服务器；服务器收到按键，对玩家进行处理更新，表现在地图的更新上。由于客户端以较高的频率发送状态请求和绘制地图，客户端，服务器可以快速通信，效果可以很快体现在地图上，就好像在客户端本地进行单机游戏的效果一样。

服务器一旦检测到玩家死亡，发回失败信息；检测到只有一个玩家存活（至少有两个玩家参与过），发回胜利信息并重置地图。客户端根据胜利/失败状态信息给出界面提示。客户端收到胜利/失败信息，断开连接。一旦一局游戏产生胜者（其余为败者）服务器重置，可以进行下一轮游戏。

4.3 输入输出

本游戏实现了客户端断线重连和服务器端保存进度。由于客户端仅用于发送请求和绘制，信息全部保存在服务器端。一旦客户端掉线，服务器端不删除对应的玩家，而是将其设置为掉线(setOnline(false))，可以重新登录。而玩家对应的位置，生命值等属性，都不会发生变化。同一个玩家（以id区别）重登后，可以发现在上一次掉线的位置。当然在别的玩家眼里，该玩家掉线期间没有任何行动，重新上线后恢复行动。如果玩家在掉线期间被击败，重新登录后显示失败信息。到此为止非常容易实现，不需要考虑输入输出。

下面考虑模拟"停服维护"的情况：服务器关闭重启，应当可以重新恢复进度。由于所有游戏信息保存在一个World对象中，因此只要将World保存，读取即可。为此实现两个工具类SaveUtil和FetchUtil，用户保存和读取地图。World需要实现Serializable接口，这在上文已经提到过。SaveUtil类实现私有函数boolean save(Serializable o,String url)，表示将一个Serializable保存到文件名url中; 实现public静态方法saveWorld(String url)保存World,而在保存之前先清空现有的子弹和npc，并将所有玩家的在线状态online设置为false。客户端检测到连接关闭，会弹出提示框。此后玩家任何操作不被接收，地图不会变化，只能关闭游戏重来。每次服务器关闭时，会执行saveWorld方法，将现有进度保存在磁盘中。而每次重启，都可以选择恢复进度，执行fetchWorld（当然也可以选择重新开始）。然后，状态便恢复

到”服务器正常但是客户端都掉线”的状态。客户端重新连上即可从上次进度开始继续游戏。

4.4 并发控制

用selector 方式连接通信，服务器对客户端的信息异步处理，考虑到竞争冲突，将关键的信息:World的things属性设为并发对象CopyOnWriteArrayList,避免了冲突。由于信息只在服务器端处理，因此每次发给客户端的都是一样的信息，不存在不同客户端状态不同步的问题。服务器需要多线程处理，因此设置一个线程池工具类ThreadPoolUtil，维护一个线程池。每次启动Runnable 对象，都通过线程池调用。每个子弹都是一个线程。子弹被发出后，定时按一个方向执行moveBy()即可。当发生碰撞/出界时，销毁线程。此外，服务器端还有增加npc的选项，每个npc(NPTank) 也是一个线程，每隔一定时间随机游走，并有一定概率发射子弹。这样多线程的设计，更好地模拟了显示的状况。服务器在处理登录信息时需要加同步锁。避免同一时间多个ip以同一个id登录产生的问题。因为登录操作是相对较少的，且性能问题主要在于网络io，因此这里不大关注性能。

5 工程问题

对象在初始化时有一定的规约，对象的创建统一在简单工厂Factory中。只有服务器才可以使用工厂创建对象。

设置一个UI辅助类。用于规格化输出错误提示。因为界面变化需要在主线程处理（Platform）设置了一个ByteBuffer辅助类，用于Object，ByteBuffer，byte[]互转，主要用于网络传输。

经验证，图片类(Image)不可简单的转成buffer传输，而每个对象需要用图片去绘制，这会产生问题。我采用的解决方案是，Image图片不保存在Thing对象中，而是保存一个字符串FileName，表示当前文件的字符串。修改图片时，更新此字符串即可。绘制图片时，根据字符串获取图片绘制。

逻辑层与显示层解耦。一个world信息映射为一张图。客户端接受到world，更新图形界面。

编写了一定规模的测试用例，在测试基础信息的同时，也简单模拟了一遍(无界面)游戏过程来测试网络通信和基础游戏逻辑。测试要求可以使用本机的某端口。

6 课程感言

非常满意。

虽然在上这门课前，我自以为“会写”java程序,但这门Advanced Java 还是给我很大的收获。覆盖面极广。通过这门课，我把之前仅仅“会写”的泛型、多线程包括最基础的面向对象等有了更加深入的理解。还了解了注解、类加载原理等平时不大使用的功能的奥妙。之前认为java不过又是一门语言罢了，看看手册查查API，直接可以上手（之前确实以这样的方式写过若干现在看起来非常stupid的代码，之后用到其他语言还是得这样，如果只是简单用用并无不妥）。上完这门课就发现不是这么简单的。虽然可以简单上手，但多了解一些原理和方法可以写出更加优雅、高质量的代码，并且更有底气。而不是停留在重复地堆砌，所谓看起来写了很多行，其实是一堆屎山。

课程作业让我们用过不少工具，比如uml，asciinema等。代码类作业使用github提交是非常有必要的。其他一些没有用github管理的课程作业，我也大都会建仓库管理。因为经常发生写了半天自以为很有道理的代码，跑起来并不是这样，被迫返工。如果不用分支管理，不用版本管理，将是非常可怕的后果。

作业量适中，自由度较大，利于不同水平的学生发挥。作业难度“适中”，尤其这学期不再另外布置大作业，又或者从另一个角度，作业5-8其实只有一个大作业，而且提供了兜底的asciipanel方案。我主要是在网络通讯耗费不少经历，其他包括界面，应该说都是不断反复地调试、查资料就可以解决的问题（当然还是花了不少时间）。之后各种优化又费了很多时间。我觉得对于本课程而言，游戏界面效果，游戏功能，界面这些不是最重要的，毕竟不大有人拿java 开发游戏。重要的是代码逻辑的设计。一学期之后感觉有不少的长进。当然了有一些同学已经做的非常出色，我还得多多学习。

最后感谢曹老师一学期的教授。

致谢 曹春

Developing a Java Game from Scratch

Liu Jinglong[1]

1. *Nanjing University Xianlin Campus, Nanjing* 210033*, China*
E-mail: 191220066@smail.nju.edu.cn

Abstract This article is a report on the final assignment in *Advanced Java* course:Developing a java game from scratch.This java game is a tank war game. In the game, the functions of multi person online, disconnection and reconnection are realized, and the corresponding test cases are written.It aims to further understand and master java socket programming, input and output, concurrency control, unit testing and other programming skills through game development, and cultivate a good programming style.JavaFX graphical interface is adopted without any game engine.

Keywords Java, JavaFx, online game, IO,Client/Server,Selector