中国科学 : 信息科学

SCIENTIA SINICA Informationis

java 作业总结

Developing a Java Game from Scratch

陈致远

南京大学, 江苏南京

E-mail: 3067887178@qq.com

摘要 本实验为南京大学 Java 高级程序设计课课程作业，采用 java 进行面向对象编程，实现了一个具有保存功能、联机游戏的图形化对战游戏。

关键词 Java, 作业, 面向对象, 多人联机, 图形化游戏

1 开发目标

实现一个在二维平面中，玩家在地块上移动并射击敌人的多人联机合作游戏。本游戏地图由自编算法随机生成，符合 rouge like 的基本特点。玩家可以自由移动、射击、存档读档，且不会出现线程冲突。

操作方式：wasd 移动，↑ ↓ ←→射击，r 键重新开始游戏，按 1 存档，按 2 读档

游戏灵感来源主要源于经典游戏坦克大战（基于地块的移动射击玩法），以及著名独立游戏《星露谷物语》内置的本地多人射击小游戏。

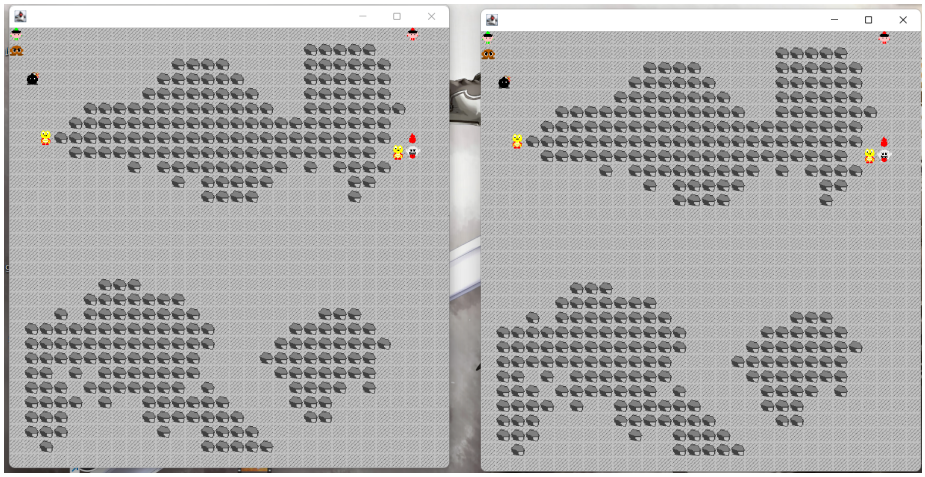

图 1 展示图

2 设计理念

大体上采用面向对象设计方法，尽可能做到贴近现实的对象之间的交互方式。

基本框架源于 jw04 的 Screen 以及 WorldScreen 类的利用 AsciiPanel 的屏幕显示方法，在此基础上进行面向对象的逻辑设计。游戏中使用的玩家、地块、npc 等类都是框架代码中 Thing 类的子类，并且一一对应游戏中每一个 Tile。最重要的 Calabash 以及 Monster 类（即玩家和敌人）都是 Creature 类的子类，具有血量、攻击力等基本属性。因为玩家采用远程攻击方式，因此增加 Bullet 类，用于控制发射的子弹。这些基类为 Thing 的类对象，都是在 World 类中的 Tiles 中显示的，其之间不能直接通信，必须通过 World 来进行上层交互。

World 和 WorldScreen 存储世界信息和屏幕显示内容，WorldScreen 同时也直接受到主 Main 函数中的屏幕刷新函数、接收键盘输入函数的控制，进行相应的操作。WorldScreen 类具有一个线程池，在游戏初始化时为每一个生物创建一个线程，并放入线程池。随后，每个生物线程池依照一定的速度移动、攻击，WorldScreen 类中必须控制相应的数据访问，以防止读写冲突。游戏中的敌人完全根据时序自由移动，并采用 BFS 算法追踪玩家。

而最顶层的 Main 函数，本质上相当于一个计算中心 + 服务器。通过 Channel 实现的 non-block 通信方式实现的一对多效果，可以同时与多个客户端实现通信。在收到客户端发送的操作信息后，给予相应的回复，或是将操作传达给 WorldScreen。与服务器相应的，也有一个用于实现客户端功能的 Client 类。相应的，也有一个类似 WorldScreen 类的 ClientScreen 类，但仅具有显示屏幕内容的功能。因此，Client 具有三种不同的操作：
1、发送创建玩家请求，让服务器加入新玩家，并等待服务器回复自己的玩家 ID；
2、发送获取地图信息请求，来获取最新的地图信息；
3、发送操作请求（同时附加自己的玩家 ID），来请求服务器进行远程操作。

另外，游戏具有开始新游戏、存档、读档的功能，基本原理就是加载新的地图和生物，并且关闭现有线程池以及创建新的线程池。在存档文件中，地图采用简易的编号方式存储（0 代表 Floor，1 代表 Wall），同时所有生物另外用一文件存储，用于恢复游戏时创建线程池时还原生物信息。

在此设计框架下，游戏的基本方式即为：
1、启动服务器，生成地图和敌人；
2、启动一个客户端，使得新玩家加入游戏；
3、玩家通过对自身客户端窗口进行键盘输入并发送给服务器，服务器根据玩家输入来控制葫芦娃进行相应行动；
4、新玩家加入，重复 2、3 步骤。

3 技术问题

3.1 屏幕刷新问题

在 jw04 的框架代码中，屏幕的刷新操作是在有新的屏幕输入时进行的，也就是“一按一动”的模式。很显然，这样的逻辑在游戏中并不适用，因此我们需要一个稳定的屏幕刷新速率。我采用的方法是，在 Main 函数中创建一个单独的线程，每 sleep 50ms 便请求一次屏幕刷新操作。尽管在利用读写锁的情况下，这样的机制用于绘制并不复杂的像素窗口绰绰有余，并不会卡顿，但是很显然，如果用于更为细致的游戏图像绘制，我所采用的这种逻辑无疑相当低效。

大致程序实现如下：

```
public class repaintTimer implements Runnable {
    public void run() {
        while (true) {
            try {
                repaint();
                TimeUnit.MILLISECONDS.sleep(20);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

图 2　屏幕刷新线程的操作

在我较熟悉的经典游戏引擎 GameMaker 的屏幕绘制流程中，是通过严格顺序执行的帧同步来实现的。在每一帧中，顺序执行各个对象的诸多操作，在全部执行完成后，在统一按照对象顺序进行绘制。而在我的程序中，对象本身并没有帧同步这一操作，因此很可能会占用屏幕绘制线程，造成屏幕输出不稳定（虽然在低帧率下感觉不出来）。在更高级的游戏引擎中（如 godot 或是 unity），屏幕绘制帧和物理帧并不绑定，但在发生冲突的情况下，依然是在执行完一个物理帧中的所有操作之后再进行屏幕刷新绘制。这样的设计思路虽然未能在我的程序中体现出来，但在老师提点下我也进行了思考，在日后采用游戏引擎进行游戏开发的过程中，这样的知识无疑很有用。

3.2 输入输出问题

3.2.1 输入

本游戏所有的操作都是通过键盘输入来实现的，同时玩家也有固定的行动速度。因此，采用“输入一次行动一次”的逻辑无疑不合适。因此，我采用的方式是记录玩家最后输入的键盘键码，而玩家线程以一定的帧率来处理这些键盘输入。如此便可以实现较为稳定的键盘输入处理操作。在实际测试的过程中，这样的效果也较为理想。

但是，这样的“输入 - 行动”模式并不是最好的，依然有改进的空间。虽然在我的游戏中，我采用的模式足以进行流畅的游戏，但是频繁响应并处理输入操作依然会有隐患。我认为可以改进的方式，是将输入内容的处理留给本地客户端，根据定好的玩家状态机逻辑，仅在状态机发生变化时给服务器发送操作请求。这样的模式即使是在最简单小游戏中也可以见到，可以说是最为通用的利用键盘输入来进行流畅操作的方式了。

3.2.2 输出

在输出方面，我采用的设计逻辑如下：
1、客户端以一定的速率（即屏幕刷新速率）向服务器发送回显请求；
2、服务器收到客户端的请求后，将当前屏幕各个格子的信息简化为图形编号（AsciiPanle 上的编号）以及颜色回复给客户端（在使用图形化界面后，或许可以不发送颜色？）；
3、客户端根据收到的屏幕信息，刷新本地的屏幕内容。

这样的“请求 - 回显”模式是常用的页游显示逻辑的简化版，将所有的运算处理操作留给服务器，客户端仅需要发送简单的请求即可。

而图形化的过程，我选择手绘了一些简单的贴图，借鉴了任天堂的《超级马里奥系列》中经典的怪物形象，并利用 AsciiPanel 的框架来实现显示。为了区分玩家，我也绘制了不同颜色的玩家和子弹。贴图如下图所示：

图 3　手绘像素贴图

在一些今年来的单机玩法为基础、但又有联机内容的游戏（或是单机内容相对复杂的游戏）中，客户端和服务器的逻辑恰恰相反，大部分的模型加载、显示内容等非交互性处理都是在客户端本地完成的，服务器基本只用于交互内容的处理。这样的处理模式在现实的游戏开发中一般更为常见。但是，我认为在本次作业的游戏项目中，我所采用的服务器逻辑足以处理所有玩家操作和回显请求，在用户的客户端本地没有太多需要进行处理的内容。

4 工程问题

4.1 maven 项目管理

在使用 maven 进行项目管理后，测试和项目集成都非常轻松。如图所示，在使用 maven 将项目打包输出后，仅需要一个 jar 包即可运行游戏（bat 文件用于带参数启动客户端）。

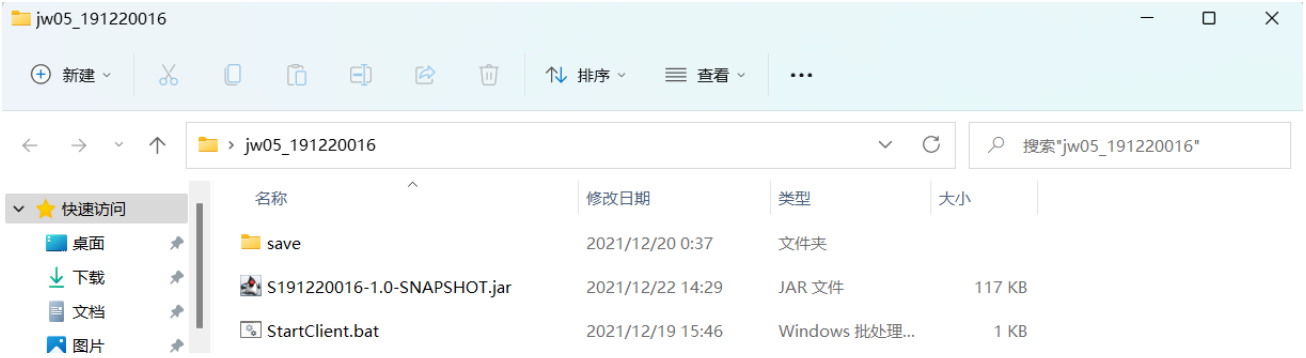

图 4 打包可执行程序

4.2 测试问题

在测试问题上，虽然 maven 提供了非常方便的一键测试功能，但逻辑并不完美的面向对象程序测试难度较高。由于地图随机生成且敌人位置随机，很难稳定测试子弹击中敌人的情况。但除此之外，在测试中我尽可能模拟了键盘输入，控制玩家进行简单的操作，并测试通信过程是否稳定。最终经过测试的代码，覆盖率达到约 70%。即使最后的结果能满足作业要求，我依然意识到我在软件测试方面

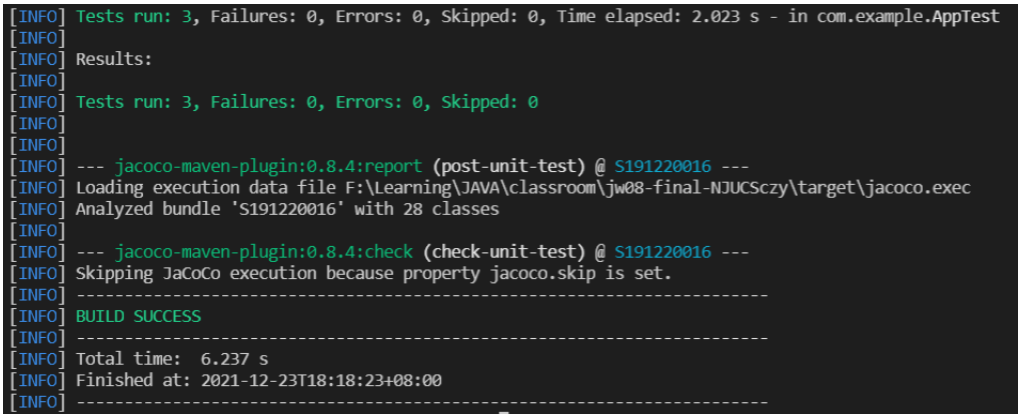

图 5 测试结果

知识的不足。这种有着较为复杂机制的程序，特别是模块化工作做的不太好（比如我自己写的代码）的情况下，想要进行全面而详尽的测试尤为困难。软件测试也给编程人员本身提出了一定的要求，这些能力我还得在日后继续磨练。

5 课程感言

这个学期我在尝试自己开发游戏（godot 引擎 2D）并参加了游戏比赛，也拿到了不错的成绩。正是在这样一种充满自我肯定的氛围中，java 课可以说是不断给我敲响警钟：还差的远呢!

图 6　我这学期自主开发的游戏，许多设计思路也受到 java 课的启发

java 这门课的难度可以说是相当高，即使是有一些游戏开发经验的我也经常会因为新的需求而抓耳挠腮。如果是自己做游戏是发挥想象的创作，那么 java 课学到的这些相对基础的东西便可以说是更为全面的客户需求。知其然也得知其所以然，这部分恰恰是我所缺乏的，这个不断挑战自我的过程也让我收获颇多。每次老曹在群里批评一些编程习惯或是生活方式时，我也会对自己平时的开发历程、设计思路等等进行反思，而且往往是在我最沉浸在自我满足的时候（笑）。毫不意外的，每次我都会被泼一盆凉水：还差得远呢！我把老曹的规范套在自己身上难免会得出这样的结论。

也正是这样的不断 push 的过程，让我了解到了更多编程本身的艺术，对相对底层的程序设计有了更强的探索欲和一些敬畏。许多我们在高级语言或是引擎中简单调用的功能背后，蕴藏着多么精妙的逻辑设计，在学习了这门课之前我知之甚少。

或许我这门课的成绩未必非常出色，但是这一路走过来，不得不说我许多的成长都是拜这门课的 push 所赐（也有隔壁 IOS 课的功劳，不一样的方面）。这确实是一门能让我戒骄戒躁，虽然没能完全学明白但是依然收获颇丰的课程吧。