中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

郭云昊 191220035[1]

E-mail: 1076626296@qq.com

收稿日期: 2021-01-16; 接受日期: 2021-01-16

Java 高级程序设计

摘要 本文是关于 Java 高级程序设计课程中的课设报告，主要报告了课设游戏的开发目标，设计理念，技术问题，工程问题，课程感言等方面的内容，通过插入图片，插入代码，文字阐述的方式进行报告

关键词 Java, 多线程, 网络通信, NIO

1 开发目标

1.1 游戏设计

以葫芦娃救爷爷为背景，设计一个葫芦娃与妖精之间的对战游戏，游戏支持离线游玩和联机对战。游戏中具有两个阵营，分别是葫芦娃阵营（红色）与妖精阵营（蓝色）。在离线模式中，玩家将扮演葫芦娃一方，通过在商店的金钱去购买你的葫芦娃战士，通过合理的排兵布阵打败妖精阵营。联机对战支持两个玩家游玩，分别扮演葫芦娃阵营和妖精阵营，使用相同的初始金钱，来摆兵布阵，通过阵容的合理性击败对方玩家。

游戏的胜负机制很简单，场上最后存在的角色是哪一个阵营的，哪一个阵营就取得胜利。对于双方同归于尽的问题，则依据葫芦娃阵营先获胜来判定胜负。

在游戏过程中，玩家可以通过键盘选择自己要操控的角色，可以控制角色的移动与攻击，离线模式中可谓是制胜法宝。在网络对战中，你的操控也会影响战局的胜败。

目前游戏还在开发阶段，支持两种角色，分别是一个高攻高防的近战角色和一个灵活的远程角色。其中每个角色的售价都为 100，在离线游戏中，游戏开始阶段玩家拥有 500 的初始资金，对方则有 10 个敌人，所以需要通过合理的布阵，可以让两种角色发挥最大作用，并通过玩家自己的操控实现扭转战局的操作。

游戏支持暂停，存档，读档，回放功能。

图片如1所示.

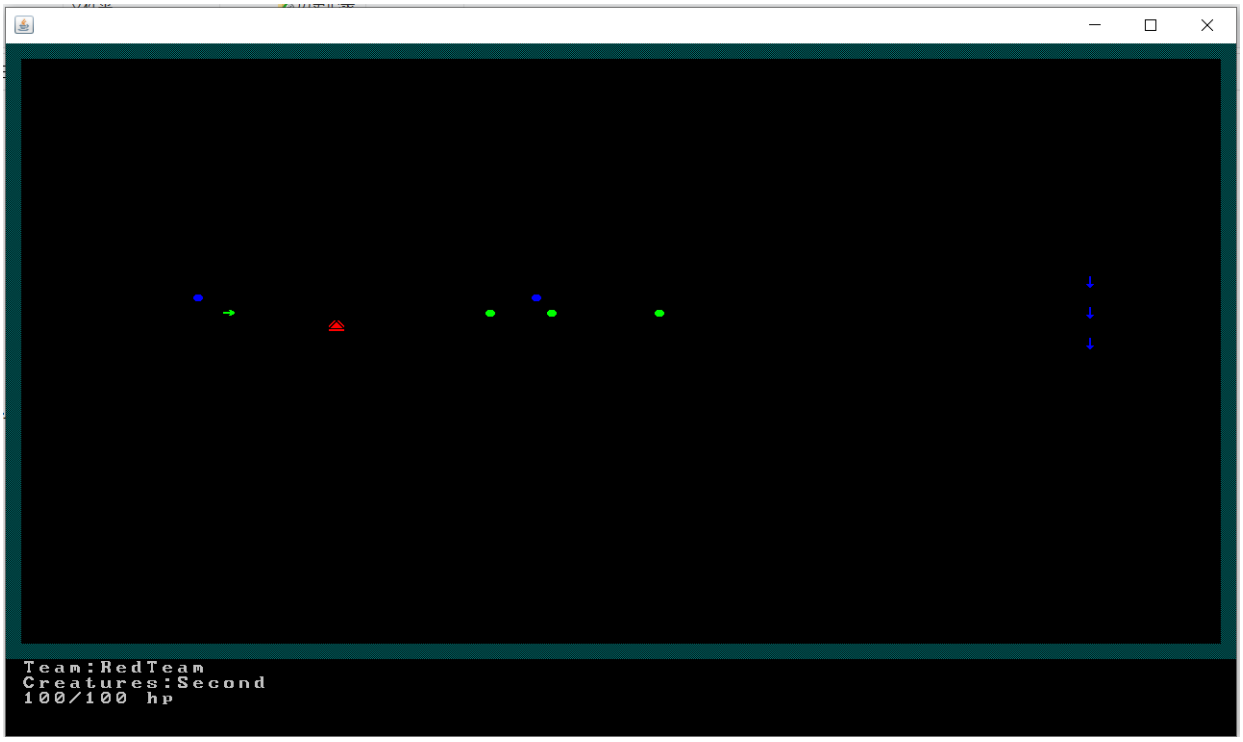

图 1 Game

1.2 游戏灵感

游戏灵感来自于一个沙雕游戏：全面战争模拟器。课程设计的初衷是实现葫芦娃与妖精之间的对战。双方对战让我想到了全面战争模拟器，既满足了双方对战的前提，也可以实现多种角色之间的战斗。并且游戏的可玩性比较强，所以我选择模仿它来实现我的课程设计。

2 设计理念

对于整个课设，已经提供了一套图形化工具，可以实现在指定区域输出图形，输出文本，设置指定区域的前景色与背景色，图像的清除与刷新，也实现了基于键盘的事件驱动。此外，在 jw04 已经提供了一种地图与地图上物品的设置方法。因此只需要在上述条件下进行设计即可。

2.1 总体设计

在这一部分，我打算分模块来介绍设计

2.1.1 物品设计

在物品设计中，首先是 Thing 类与 Tile 类之间的关系，其中 Tile 类是 Thing 类中的一个成员，通过这种方式，可以实现游戏地图上物品的储存。Thing 类还使用了一个接口：CreatureAttribute，接口内定义了一些生物类（Creature）所需的操作

是一些地图上的物品如地面（Floor 类），墙（Wall）类，以及生物类（Creature）都继承于 Thing 类。既可以复用 Thing 类诸如获取名称，获取横纵坐标的接口，又可以实现自己的图像设置以及其他一些接口。

对于游戏中的角色如 First 类，Second 类（原谅起名无能），以及子弹类（Bullet），都是继承于 Creature 类。

在 Creature 中使用了一个接口:Debug，内部是一些 boolean 型的变量，其主要作用就是控制程序插桩输出。

2.1.2 界面设计

在屏幕设计中，首先是定义一个接口为 Screen，然后其他界面如游戏界面（WorldScreen），开始界面（StartScreen），重新开始界面（RestartScreen），回放界面（ReplayScreen）都是对接口的完善。而对于其他界面如获胜界面（WinScreen），失败界面（LoseScreen）都是继承于重新开始界面。

由于游戏在开始时需要购买物品，所以使用引用类（Shop）实现了一个商店窗口，而商店类的使用则是嵌套在 WorldScreen 中。

游戏在开始界面或者暂停界面时都需要提供选项，为此设计了一个按钮界面，支持高亮显示与选择切换功能，也可以更改按钮颜色与文字。

此外，为了可以在游戏进行过程中显示记录，利用 Jframe 另开了一个窗口来显示游戏记录，同时在该窗口设计了一个保存选项，可以保存游戏记录，便于游戏回放时使用。

2.1.3 地图设计

游戏地图设计十分简单，就是通过长宽得到一个四周是围墙，中间是地面的区域，所以在这一部分的报告较为简单。

2.2 功能设计

在这一部分将对游戏的存档，读档，回放，联机对战功能进行报告

2.2.1 存档，读档功能

游戏的存档读档就是对游戏状态的记录与读取，基于对文本的读写操作，可以实现存档读档功能

具体操作：对地图的保存十分简单，由于地图生成的简单性，只需保存地图的长宽即可，而对于地图上的其他存在如角色，子弹，则需要分别记录他们的状态，如队伍，编号，血量，位置等，通过写文档可以完成对游戏的保存。而游戏的读取就是一个逆操作的过程，通过读取存档文件，获得地图的长度宽度，进而重新生成地图，然后根据存档中的信息，恢复所有角色的状态，然后开始游戏。

2.2.2 游戏回放功能

游戏的回放功能也是基于文档的读写操作。然而不同于存档读档，游戏的回放过程需要记录每一个角色的每一次行动，所以我选择在角色的动作代码中生成信息，在执行操作后将信息发送到游戏记录窗口，这样就可以实现游戏进程记录。

对于游戏的回放，可以通过读取游戏进程，然后执行进程操作即可。

2.2.3 联机对战功能

联机对战基于网络通信，需要一个服务端与两个客户端。由于游戏回放功能已经实现了游戏进程的记录与读取，所以在联机对战中，只需要服务端进行具体游戏计算并把游戏进程发送到两个客户端。客户端读取并解析游戏进程，并做到相应输出即可。

2.3 设计好处

对于一个多模块的，高复杂度的程序，利用继承来进行设计既可以实现大量代码的复用，也可以提高模块内的内聚性，降低模块间的耦合性，所以在设计过程中使用了大量的继承。

在设计中利用 CreatureAttribute 接口，目的在于既可以通过父类 Thing 来作为角色以及子弹的获取，可以在攻击以及被攻击函数中实现多态，又可以降低在 Thing 类的代码量由于该接口中的函数只在特定的对象中进行调用，所以完全可以设计成为默认类，只在特定对象中进行重写，而不是在实现类中重写，而且不需要担心其他函数误调用时报错。

在设计中利用 Debug 接口，我认为是十分有效地。在多线程的程序设计中，调试是十分困难的，而插桩法又会造成在中端产生大量输出，无法有效判断是哪里的输出。参考 C++ 中的宏用于插桩的方法，我选择利用一个接口，在其中定义 static 类型的 bool 变量，然后在程序中进行 if 判断，实现对特定位置进行插桩调试。

3 技术问题

3.1 通信效率

在网络通信中，服务端与客户端数据的传输是核心，所以通信效率就决定了联机效果是否良好。在改善通信效率这一方面，我使用了 NIO 来代替 BIO。NIO 的非阻塞读可以有效地保证在通信过程中不影响其他操作的进行。利用 Selector 来处理数据的到来，在客户端与服务端都可以第一时间获得数据。而且由于 NIO 对线程是安全的，所以可以使用多线程处理数据传输，这样可以做到读取并处理数据与其他游戏进程相分离，进一步实现通信效率的提高。

此外，相比于使用帧同步，我使用了状态同步的方法，服务器只在角色执行操作之后向客户端发送数据，而不是在一帧的结束发送整个游戏界面。因为游戏中的角色数量并不多，子弹数量也有限，而且我设计的帧数为 30 帧，所以状态同步传输的数据要比帧同步传输的数据要少，这样也提高了通信效率。

在读写方面，我选择使用两个 buffer 与一个 channel 进行操作。由于读操作与数据操作分离，所以可能出现读取数据结束，channel 空闲，但由于读 buffer 在数据操作中仍被使用，所以此时无法利用读 buffer 去向 channel 写数据，而当读 buffer 空闲时，channel 又有新的数据到来，这样会导致写延迟。所以选择写操作利用一个新建 buffer 来进行。这样也提高了通信效率。

3.2 并发控制

在整个实验过程中，多线程之间的同步是一个难点。锁加的多了、封锁范围大了、一个对象持有锁的时间长了，会提高死锁现象出现的频率，锁加的少了、范围小了、时间短了又会出现数据不同步现象，产生各种不可预知的问题。在并发控制方面，我选择尽可能细化加锁对象，比如当角色移动时，我会在角色进入某一个地图位置前访问该位置是否可进入，如果可进入，则在进入时对该地方加锁，在进入代码结束后释放锁，我认为相比于当角色进入位置加锁，离开位置释放锁，我的处理可以减少锁的持有时间。但对于一个角色被攻击时，则会在攻击开始前上锁，攻击完成后解锁。因为攻击代码的执行时间要远远比角色移动耗时少（因为在角色的两次移动之间存在数百毫秒的时间间隔），所以可以加大加锁时长，减少 bug 出现可能。

此外，游戏设置了暂停选项，但由于暂停时整个游戏停止计算，所以可以放心地对所有角色加锁，当取消暂停时释放所有角色上的锁，这样就可以实现游戏暂停时的并发控制。

3.3 输入输出

在游戏的输入输出有如下几种: 文件读写，事件读写，通道读写

3.3.1 文件读写

文件读写是最简单的，只需要打开文件进行读写即可。打开文件与读写文件是十分耗时的，所以我选择在游戏开始或者结束时统一进行读写，特别是在游戏回放过程，我会把所有游戏过程先一起读入内存，在通过处理内存中的信息来还原游戏，以提高读写效率。

3.3.2 事件读写

游戏提供了键盘的事件驱动，但是由于需要在商店购买小兵并选择位置进行安放，利用键盘对上述操作进行实现比较对用户不友好，所以我又添加了对鼠标事件的响应，通过读取鼠标的点击位置，转化为在窗口上的坐标，进而判断此时点击的是商店还是地图，通过这样就可以实现利用鼠标完成排兵布阵。

3.3.3 通道读写

通道读写存在于服务端与客户端。在服务端读到的数据是客户端发来的按键输入，鼠标输入信息，在客户端读到的是服务端发来的游戏过程。此外，还有一些信息如玩家进入，玩家准备等信息也会存在于服务端与客户端的通道中。

不论信息是什么，所有的信息都是以行为单位进行传输的，所以当一端读到数据时，首要任务就是将其从 bytebuffer 转化为若干行数据。在这里，我先利用了 bytebuffer 与 charbuffer 相互转换实现二进制串向字符的转变，然后通过拆分换行符得到若干行信息，就实现了信息的读取。但在实际操作过程中会出现如下几个问题:

1. bytebuffer 中的内容有时候可能不足一行，此时就需要通过最后一个换行符的位置，来设置 buffer 的 limit 与 position 的位置，等待该行信息彻底传完之后再读取完整信息。

2. bttebuffer 中的内容有时候可能不止一行，此时就需要利用一个 list 储存所有已经成行的信息并转向处理，将未成行的数据留在 buffer 中，等待下一次读入。

3.4 面向对象的好处

面向对象的好处有如下几点:

3.4.1 过程抽象

面向对象设计将过程的具体实现抽象成了一个个接口函数，外界的调用无需在意过程的实现，只需要提供所需数据即可返回目标结果。这种设计降低了不同模块之间的耦合性，便于开发。

3.4.2 成员封装

面向对象设计可以将一些不希望外部改变的数据封装，只提供一个操作的接口，而不是直接对成员进行操作，这样可以保证数据的安全与稳定。

3.4.3 继承

面向对象设计提供了一个十分好用的机制就是继承。通过继承，可以实现代码的复用。通过继承，可以实现对象的多态，一个父类对象可以指向多个子类对象，而子类对象通过重写父类对象的接口函数，可以实现对于同一个操作，不同子类对象给出不同结果。

3.4.4 与现实的交互

面向对象设计提供了一个代码世界与现实世界进行交互的方法：将现实世界的问题抽象成几个对象之间的交互，比如下棋，可以抽象为人指挥棋子，棋子自己进行移动，棋子攻击棋子，进而可以使用代码重构这些对象，并创造相应的动作接口。好像现实世界的主语，谓语，宾语，可以转化为代码世界的对象甲，接口函数，对象乙。

4 工程问题

4.1 设计方法

如在第二节描述的，我使用了 Debug 接口进行插桩，这提高了调试的效率，也提高了代码的准确性。进而提高了开发效率和代码质量。

具体实现如2:

```
public interface Debug {
    boolean DebugFirstMove = false;
    boolean DebugEnemyMove = false;
    boolean DebugBulletMove = false;
    boolean DebugBulletMoveDirect = false;

    boolean DebugPlayerBeAttacked = false;
    boolean DebugFirstBeAttacked = false;
    boolean DebugBeAttacked = false;
```

图 2 Debug

4.2 工程方法

除了插桩调试，我也编写了一部分的测试脚本来测试代码执行的正确与否。

测试覆盖率（网络对战前）如下:

Element	Class, %	Method, %	Line, %
com	80% (29/36)	69% (214/310)	51% (931/179...

图 3 Coverage

5 课程感言

在 Java 高级程序设计这门课上，我学到了很多东西。除了最基础的 Java 语法，更多的是对 Java 语言在编译，执行时的一些理解，还有对面向对象设计的进一步认识。课程在后期的多线程与网络对战虽然难度较大，但是十分的有意思。除此之外，我还学到了一些编程语言之外的内容如 uml 的编写，latex 的编写等等。我认为我在 Java 课上学到最多的，是如何高效地使用手册，高效地在网络上查找并学习自己没有接触的一些内容。

课程难度还是比较大的，但老曹很好，讲课很认真负责，除了我没有听懂（这是我自己的问题），上次遇到这么好的老师还是在上次。考试难度一言难尽，让我体会到什么叫就算开卷也不一定能及格的滋味。

总之一学期的学习下来，受益匪浅，感谢老师，也感谢所有同学。