中国科学：信息科学

SCIENTIA SINICA Informationis

论文

# Developing a Java Game from Scratch

赖烨文[1]

1. 南京大学, 南京
E-mail: 1808479906@qq.com

**摘要** 本文主要描述了 Java 作业 8 的开发目标，设计理念，技术问题，工程问题，课程感言以及完成编写之后的问题反思。该项目使用了 JavaFx 进行图形界面绘制，通过 Nio 实现服务端和客户端的网络通信，文件读写实现游戏的回放功能。

**关键词** Java,JavaFx,NIO

## 1 开发目标（我写的游戏是个什么样的游戏，灵感来源是什么）

### 1.1 什么是 Roguelike 游戏？

首先介绍一下 Roguelike 游戏。Roguelike 是欧美国家对一类游戏的统称，是角色扮演游戏（RPG）的一个子类（Roguelike-RPG）。Roguelike 类型游戏在 2008 年的国际 Roguelike 发展会议上进行了明确的定义，并被命名为“柏林诠释”。

在命名并定义的过程中，其中出现频率较高的特点包括：

1、生成随机性。

2、进程单向性。

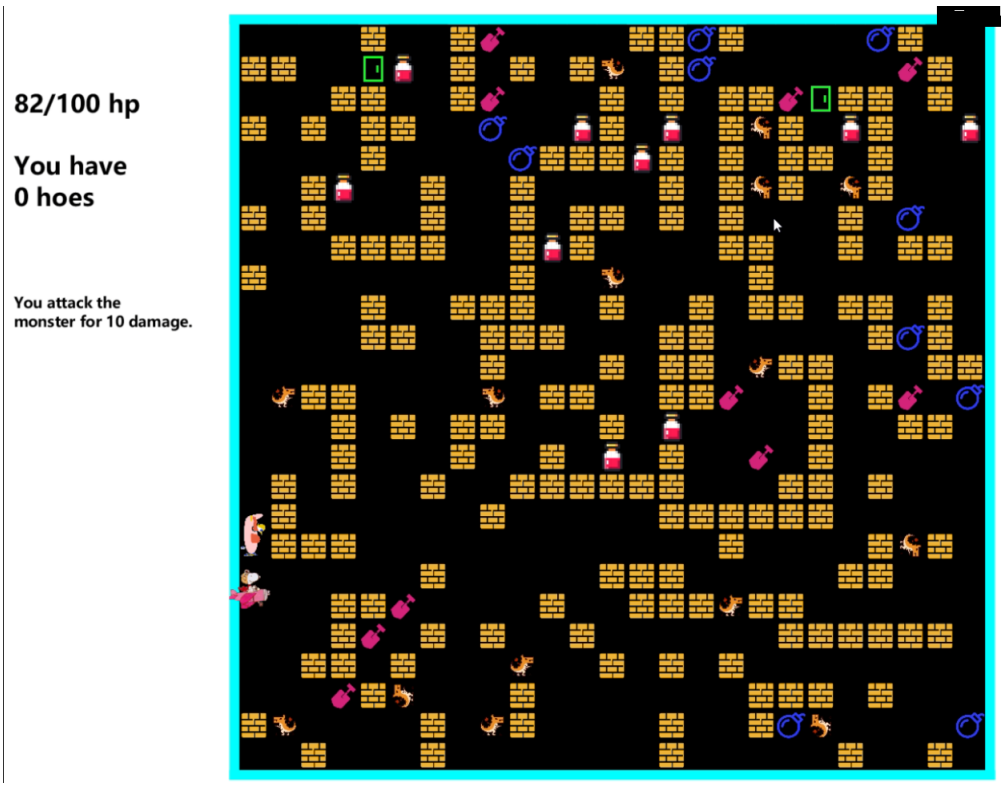
3、不可挽回性。

4、游戏非线性。

5、画面朴素性。

6、系统复杂性。

本次实验我开发了一款 Roguelike 游戏，灵感主要来源于泡泡堂及元气骑士。在游戏中，双方玩家各自操控一个角色，在一个迷宫地图中对战。在对战的过程中，地图中会随机生成不同的敌人与各类物品，玩家可以通过拾取物品获得对应的技能，也会被敌人攻击而失去血量。




最先击败另一玩家，即使对手血量降为 0 的一方即可获得游戏胜利。

## 2 设计理念（代码总体设计是什么？这样设计的好处是什么？）

### 2.1 总体设计

本次实验我将代码分为服务端与客户端两类。

整体思路为：由服务端代码向两个客户端推送地图信息，客户端只负责显示游戏画面及接收用户的键盘输入，并将其发送给服务端进行计算。在服务端，开启多个线程控制地图上每个单位的行动，并读取客户端发来的用户输入，进行相应的计算。每隔固定的时间，服务端就向客户端发送当前的游戏画面，由客户端显示给玩家。

游戏地图为一个网格地图，通过在每个格子中加入贴图来形成游戏画面。其中，每次画面刷新时，服务端都向客户端发送一个可序列化的 POJO 类，其中记录了当期地图中所有元素的位置信息（即横纵坐标 x 和 y）以及该元素对应的图片名称。为了缩减传输数据的大小，我将游戏资源文件放在客户端本地，这样服务端在解析数据时，只需根据图片名称生成 ImageView 并放置到对应位置即可。

同时，发送的 POJO 类中还会存放游戏状态信息，包括游戏中、游戏失败即游戏胜利三个状态。客户端解析数据时，可以根据不同游戏状态展示不同的画面，以展示当前游戏进程。

在网络连接方面，服务端与客户端都采用了 NIO 进行网络通信。在服务端中开启一个线程，在一个死循环中反复检测 selector 的状态，如果监听到有通道的连接，则为其注册一个监听写入的通道并保存。这样，每当客户端向服务端发送信息，都可以由该通道取出并解析，而服务端也可以直接通过该通道发送数据。在客户端则监听服务端通道的可读取信息，若通道中存在数据则取出并解析，以此刷新游戏画面。

### 2.2 设计优势

1、将所有计算都放置在服务端，可以保证多个客户端在同一时间接收到的数据是相同的。也就是说，确保了玩家看到画面的一致性。并且，如果需要对游戏内容进行简单调整，如数值调整，则只需修改服务端代码，减少了代码维护成本。

2、将资源文件保存在客户端本地，减少了网络通信传递数据的大小，服务端可以传输更多有效数据给客户端，可以加快数据传输效率，减少网络延迟带来的影响。

3、使用 NIO selector 实现网络通信，可以避免线程开销带来的性能问题，节省了计算机资源。同时，NIO 使得线程在没有数据可用的情况下不会一直保持阻塞状态等待数据，而是可以利用没有可用数据的空余时间进行其他操作，解放了 cpu。

## 3 技术问题（通信效率、并发控制、输入输出等问题我是怎么解决和优化的，面向对象设计方法带来了什么好处，等）

### 3.1 画面展示

首先遇到的问题是游戏画面的展示问题。由于之前实验中使用的 asciipanel 具有一定局限性，难以使用自定义的图形，不能很好满足个性化需求，因此决定采用 javafx 实现游戏画面展示。这样可

以使用更多不同图形，给画面带来更多可能，使游戏内容更加丰富。

### 3.2 网络连接

最初设计中，我在服务端为每个连接的客户端都单独开启一个线程，在线程中使用循环向客户端发送数据。这样带来的问题是当线程处于等待状态时，部分 cpu 资源会被浪费。虽然本次实验中最大客户端连接数只有两个，但若连接数过多，容易造成性能上的损失。因此，最后使用 NIO 来实现。NIO 一个重要的特点是：socket 主要的读、写、注册和接收函数，在等待就绪阶段都是非阻塞的，真正的 I/O 操作是同步阻塞的（消耗 CPU 但性能非常高）。NIO 在单线程中使用轮询方式，可以减少线程开销，提高效率。

### 3.3 通信效率

由于本次实验是实现网络对战，因此服务端与客户端之间的数据传输会收到网络速度的限制，理论上当传输的数据过多时会出现卡顿现象。因此，我将较大的资源文件都存放在客户端本地，在服务端发送的数据当中，服务端仅仅传输每个贴图的位置，以及对应图片的名字，这样客户端就可以根据收到的数据信息，将对应的贴图放置到正确的位置，从而实现场景的刷新。

当然，这样的传输方式仍存在不小的弊端。本次实验中我将游戏画面的刷新率固定在每 30ms 刷新一次，这意味着服务端每 30ms 就要将当前完整的地图信息发送给客户端，这是一个较快的刷新速度。因此，如果游戏画面尺寸较大，每次刷新传输的数据变大，就有可能造成客户端界面的卡顿。我的一个设想是在游戏一开始服务端向客户端发送整张地图的初始信息，然后每次地图刷新时，都只将地图的当前状态与上一个状态的差异信息发送给客户端，由客户端计算生成新的画面，这样可以继续缩减传输数据的大小，避免应网络延迟带来的卡顿或画面不一致现象。但由于时间及能力限制未加以实现，在此仅做设想。

### 3.4 并发控制

在并发控制方面，为了避免两个单位走到同一个 tile 上，我选择使用 java 锁机制。当一个单位想要进入某一个 tile 时，其会先查询该 tile 上是否有其他单位，而我在查询开始前先上锁，查询结束后释放锁，即可避免两个单位同时检测到一个 tile 上为空，并同时进入的情况。

在游戏进行过程中，还发现有另一个与多线程编程相关的 Exception。经查询后发现，这是由于一个线程在读取某个 List 时，另一个线程向其中写入数据而引起的并发错误。因此，每当对某一个 List 进行读写操作之前，我都会申请一把锁，确保读写过程不受其他线程行为的干扰。

### 3.5 面向对象设计方法的优势

1、面向对象方法是在相对于面向过程的，面向对象的方法是以对象作为核心，将数据和方法作为一个整体看待。尽可能接近人类习惯的抽象思维方法，并尽量一致地描述问题空间和解空间，从而自然而然地解决问题。

2、面向对象方法用对象模拟问题域中的实体，以对象间的联系刻画实体间联系。当系统的功能需求变化时，不会引起软件结构的整体变化，仅需做一些局部的修改。由于现实世界中的实体是相对稳定的，因此，以对象为中心构造的软件系统也会比较稳定。

3、面向对象方法具有的继承性和封装性支持软件复用，提高代码的可重用性。

4、由于面向对象的软件稳定性比较好，容易修改、容易理解、易于测试和调试，因而软件的可维护性就会比较好。

5、面向对象方法可以通过对象的”键”快速找到某个数据。当某个函数需要多个数据进行传参时，不再需要一个个传参，而是将这些数据放入一个对象中进行传参，这样就只相对于传了一个数据，提高信息的查找速度和传播速度。

## 4 工程问题（如何采用各种设计方法、工程方法来提高开发效率和代码质量）

在程序设计过程中，工厂模式被频繁使用。工厂模式有如下优点:

1、一个调用者想创建一个对象，只要知道其名称就可以了。

2、扩展性高，如果想增加一个产品，只要扩展一个工厂类就可以。

3、屏蔽产品的具体实现，调用者只关心产品的接口。

使用工厂模式后，代码的结构更加清晰，对象创建的方法被有效封装起来，降低了耦合度，减少了代码量。

## 5 课程感言（对课程形式、内容等方面提出具体的意见和建议）

本学期上课过程感觉课程安排都比较合理，各个作业间层层递进的关系也比较有趣，采用制作游戏的方式也能有效提高学生们的积极性。就是希望老曹可以展开讲讲 java 如今的使用情况？希望可以了解到当下 java 在实际开发环境下的使用，了解一下 java 主要的使用场景和方向。