中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

张峻[1]

1. 南京大学计算机科学与技术系, 南京 210023
E-mail: 1319341822@qq.com

收稿日期: 2021-12-31; 接受日期: 2021-12-31

南京大学计算机科学与技术系 2021 秋 Java 高级程序设计课程

摘要 在使用 Java 语言从零开始实现一个图形化、具有单机和网络对战模式的 Roguelike 游戏的过程中，开发目标、总体设计理念、遇到的技术问题、工程问题及其优化和解决办法。此外，附上了学习 JAVA 高级程序设计课程的收获感言、对课程的意见和建议。

关键词 Java, 游戏开发, Maven 自动构建, JUnit 单元测试, 网络编程

1 开发目标

1.1 我开发的是一个什么样的游戏？

我开发的是一个 *Roguelike* 风格的**打怪通关**小游戏，如图1所示。

游戏的**基本机制**为：玩家操控角色，在攻击怪物的同时，躲避来自怪物的攻击。消灭地图中所有的怪物后，地图上会出现通向下一关的门，进入门中即可挑战下一关。玩家遭受怪物攻击至生命值 (HP) 耗尽，则玩家死亡，挑战失败。

在网络对战模式下，每个玩家控制一个角色，组队消灭怪物。

游戏还有如下机制值得说明：

1. 玩家在每一关开始前，均可以选择任意一个角色来进行该关的战斗，这样的机制能够使玩家针对每个关卡的怪物种类和数目，自由选择合适的角色通关，增强游戏体验。如图2所示。

2. 每一关通过之后，游戏进度会自动保存。在某一关的游戏过程中，游戏进度无法保存。这样设计是参考了 Roguelike 游戏的常见存档模式。

3. 游戏为玩家提供了三个存档，进入游戏后可任选一个存档继续游戏，或者任选一个存档，覆盖存档开始新游戏。如图3所示

4. 游戏关卡的怪物设置为：每三关中的前两关只有小怪，第三关在出现小怪的同时还会出现 *Boss*。

5. 游戏具有 Roguelike 必备的随机性机制，每一关的地图地形（即地图中的障碍物分布）以及怪物的数量、种类、分布位置都是随机的。

图 1　游戏界面
Figure 1　GameInterface

图 2　选择角色界面
Figure 2　Character Choosing

图 3　选择存档界面
Figure 3　Archive Choosing

1.1.1　游戏视频链接

南京大学计算机系 2021 秋 JAVA 高级程序设计课程 JW08

1.2　灵感来源

游戏没有特定的某一个游戏作为原型，非要说的话，灵感来源于我从小到大玩过的各个游戏的综合。包括但不限于：地下城与勇士、以撒的结合、原神、坦克大战、元气骑士。

2 单机游戏版本的总体设计理念

面向对象程序设计的本质是通过类和对象对现实进行抽象和模拟, 因此, 我依照这样的思想, 依照图4和5所示设计了游戏的类结构。

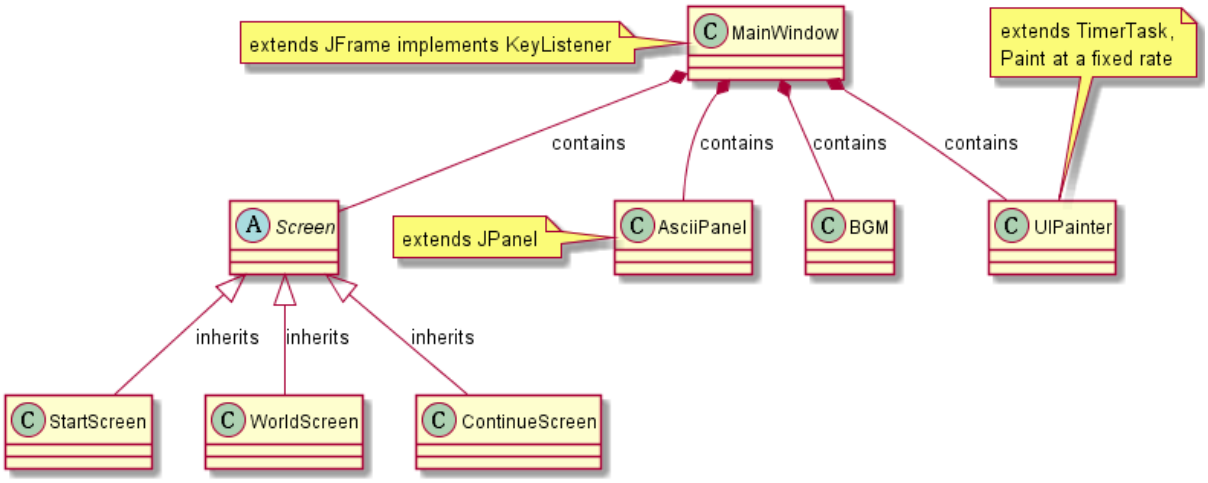

图 4 总体类结构，游戏与玩家进行交互的各功能模块

Figure 4 Classes1

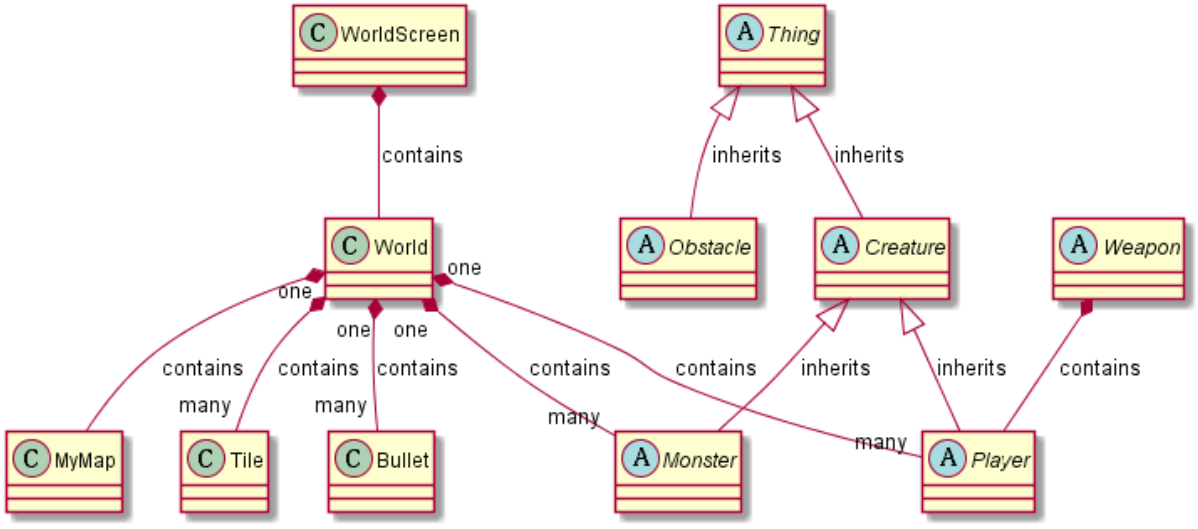

图 5 WorldScreen 类内部的类结构，实现游戏内部逻辑

Figure 5 Classes2

2.1 总体类结构设计，与玩家进行交互的各功能模块

图4展示了与玩家进行交互的各功能模块。最上层的 MainWindow 类即为整个工程的主类，继承 JFrame 类，实现了与玩家直接进行交互的主窗口程序。与玩家交互的通信过程如图6所示。

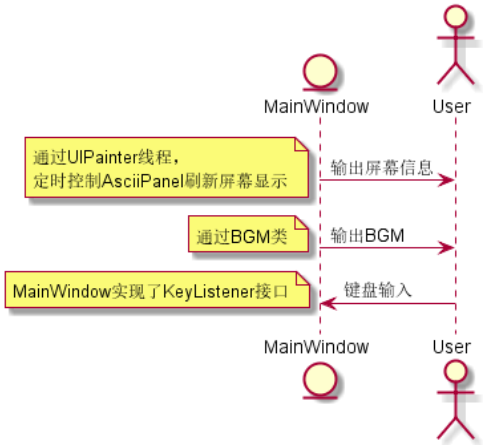

图 6 程序与玩家交互
Figure 6 interact with user

UIPainter 类实现了 TimerTask 接口，作为一个独立的线程定时控制 AsciiPanel 刷新显示内容，实现了 KeyListener 接口接受玩家的键盘输入。

Screen 类的作用：UIPainter 告诉 AsciiPanel" 什么时候该画", 而 AsciilPanel 在 " 每个位置上该画什么 " 是由 Screen 类来控制的。Screen 类保存了屏幕上的每个位置有什么元素，AscillPanel 保存了屏幕上每个位置显示的内容对应的 *char* 值，前者控制游戏内部真正发生的事情，后者控制玩家看到的是什么。

MainWindow 类接受到的玩家键盘输入，也被传递给 Screen，作相应的处理。

不同 Screen 间的切换:Screen 的切换顺序为 StartScreen->WorldScreen->ContinueScreen->Wor -ldScreen->... 当 Screen 内部判断要切换至下一个 Screen 时,会调用上层类 MainWindow 的 SetScreen() 函数来切换 MainWindow 中包含的 Screen 对象。

BGM 类负责播放背景音乐。

2.1.1 这样设计的好处

1. 使用单独的 UI 线程，便于调试以确定合适的刷新率，同时保证游戏过程中刷新率恒定。

2. 将游戏内部逻辑和游戏显示两个模块分离，降低了模块之间的耦合度，极大地方便了游戏的开发。三个模块各自的功能分工非常明确:

Screen 的功能：控制游戏逻辑之外，只需提供一个方法给 UIPainter 调用，该方法根据 *Screen* 中每个位置的信息，将要显示的内容写入 *AsciiPanel* 中。

AsciiPanel 的功能：根据保存的信息，显示出正确的内容。

UIPainter 的功能: 定时调用 Scrren 提供的方法，将要显示的信息写入 AsciiPanel 中，再调用 AsciiPanel 的 Paint 函数绘制出相应信息 (通过调用 MainWindow 的 repaint 方法来隐式调用)。

2.2 WorldScreen 类内部的类结构，实现游戏内部逻辑

Screen 的三种子类中，WorldScreen 实现了游戏进行时的内部逻辑，图5展示了其内部的类结构。

World 类抽象了游戏中的整个世界，包含地块类 Tile，子弹类 Bullet，玩家类 Player，怪物类 Monster、武器类 Weapon 等。其中后四者均为抽象类，游戏中的各种子弹、角色和怪物由其子类实现。

MyMap 类是专门负责生成每一关的地图的功能模块，其功能包括了随机生成每一关的障碍物位置、怪物的数量和种类、怪物和玩家的初始位置。World 类只需在构造器中根据 MyMap 提供的相关信息构造相应对象即可。

3 网络对战版本的总体设计理念

课程群里群友说的一句话，虽然是在我已经完成了网络对战之后听到的，但我听了之后豁然开朗: 网络本质上是 *server* 和 *client* 间的 *IO(*数据交换*)*, 我们只需设计一个 *"protocol"* 并依照 *protocol* 进行通信即可。

我设计的"protocol" 如图7所示。

图 7 网络协议

Figure 7 protocol

Client 端将用户输入的按键的键码 (KeyEvent 对象的 getKeyCode() 方法获得) 以 int 的形式传给 Server 端，Server 端接收后进行与" 单机对战部分" 相同的处理。

Server 端控制刷新显示的 UI 线程 (UIPainter 类) 由定时刷新显示改为定时将显示信息传给 *Client* 端。该信息的大小为 AsciiPanel 二维数组的大小，长度固定。Client 端接受了每次传时的全部显示信息后，进行刷新显示。

3.1 类模块的重新划分

网络对战模式下，Server 端和 Client 对单机游戏版本下的功能模块进行了重新分配，如图8和图9所示。

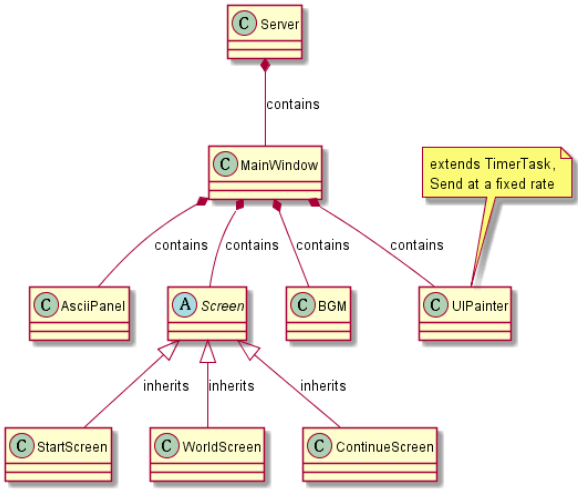

图 8 Server 端的类结构
Figure 8 Server

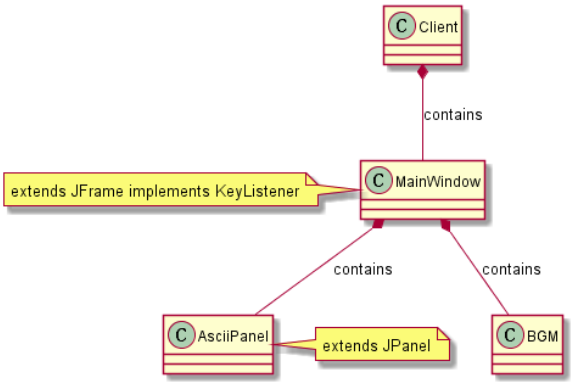

图 9 Client 端的类结构
Figure 9 Client

与单机版本相比，有了如下改变：

MainWindow 的功能中，接收键盘事件、作为程序窗口显示和播放 BGM 的功能（即与用户之间交互的功能）在 Client 端完成，剩余功能在 Server 端完成。

UIPainter 的功能变为：调用 Screen 提供的方法，获得显示信息，将显示信息传输给 Client 端。Client 端的 AsciiPanel 完成了单机游戏中 AsciiPanel 的全部功能，Server 端的 AsciiPanel 仅用于暂存 Screen 的显示信息，以便后续传输。

3.2 这样设计的好处

这样进行传输，传输的数据量较小，每次按键时传输一个 int 数据，而每次刷新显示时传输 1650 的 char 型数据，传输流量在可接受的范围内，且每次传输的数据的单位长度和总长度是固定的，使通信变得简单 (十分傻瓜的网络协议)。

单机模式中的功能全部可复用。在网络对战模式下，只需在 Client 和 Server 端进行功能的重新划分、稍作修改，并将代码逻辑改为多用户、多角色即可。改动的代码量极少，且单机模式的代码经过单元测试和人肉测试后基本排除了 bug，因此修改后的代码出现问题的概率也较小。

4 技术问题和技术细节

4.1 多线程的临界区管理，避免 Race Condition

每一个生物体对象 (属于 Creature 的子类的子类，即 Player 和 Monster 的所有子类) 均是一个单独的线程。因此，会出现如下几种 Race Condition：

1. 进行移动时，可能同时向同一个 Tile 移动，导致 Tile 的 Thing 对象被同时修改。

解决办法：将 Tile 类的 setThing 方法修改为如图10所示。通过 tryLock() 尝试获取锁，获取锁后才改变 Tile.thing 指向的 Thing 对象。若修改成功返回 true，无法获取锁则返回 false，这是用于在生物体的移动方法中，根据调用的 *setThing* 的返回值判断是否移动成功，成功之后，再将移动前的原 *Tile* 的 *thing* 对象清空。

```
public boolean setThing(T thing) {
    if (lock.tryLock()) {
        try {
            this.thing = thing;
            this.thing.setTile(this);
        } finally {
            lock.unlock();
        }
        return true;
    }
    else{
        return false;
    }
}
```

图 10 Tile 类的 setThing 方法

Figure 10 setThing method in class Tile

2. 生物体受到攻击时，可能同时被多个敌人/子弹攻击，生命值 int HP 被同时修改。

解决办法：将生物体受到伤害、生命值减少的方法改为 synchronized 方法。

3.Player 对象有一个键盘输入缓冲区成员 KeyEventBuffer，Player 线程以一定频率取出 Buffer 中的键盘事件进行响应。可能监听到键盘事件加入 Buffer 的同时，Player 线程正在取出 Buffer 中的内容，造成同时访问。

解决办法：在 *synchronized* 块中访问 buffer

4.2 网络通信的实现

网络通信的实现中，考虑到效率问题，为了避免玩家较多时产生阻塞，使用了 NIO Selector 模式实现 Non-Blocking-IO.

参照的模板为 Medium 上教程Blocking I/O and non-blocking I/O的示例代码。

主要在例子中的最基本的 Client 和 Server 的基础上，做了如下修改：

Server 的行为由原来的在 *while(true)* 循环中完成所有的建立连接、从读 *Channel* 中读数据的操作，变为先一开始只处理建立连接的操作，与指定数量的玩家建立连接后，开始游戏，之后只处理读数据的操作。

此外，考虑到键盘相应和 UI 刷新都必须是无阻塞进行的，且查阅资料了解到 Channel 对于写操作应该是 Always Ready 的，因此，*Server* 和 *Client* 向 *channel* 中写数据的操作不在 *while* 循环中进行，而是在需要写时调用单独的 *write* 方法进行写数据操作。

4.3 面向对象设计的好处

1. 使代码变得易于设计、编写和维护。以对现实世界进行抽象的角度去设计各个类以及类之间的关系，使得设计代码总体架构和大部分实现细节时非常直观。代码设计者无需思考违背直观思维的设计，提高了效率，同时也避免了很多由于设计违背直观思维而产生的可能的 bug 和开发中遇到的困难。

2. 通过封装机制实现了类之间的分离和功能模块的分离。类与类之间的协作，是通过消息传递（即方法调用）实现的。在开发过程中，设计好类之间的发送消息机制后 (即某一个类要传给另一个类哪些信息，以什么格式传递，类似于一个通信协议)，只需关注某一个类的具体实现，在实现其内部逻辑的同时，正确地向其他类传递消息即可。方便了开发，也使 Bug 的定位变得容易。

3. 多态和动态绑定方便了代码的编写，无需穷举对象类型。怪物、玩家角色、障碍物、子弹、武器都有多种子类，在基类 (抽象类) 中定义好要实现的各个方法，在子类中进行重载，最后使用基类引用调用各个方法即可。

5 工程问题

5.1 工程开发的总体步骤

1. 确定总体设计目标 (写什么样的游戏)、大致完成模块划分和类的设计。

2. 第一轮开发：不关注 UI 图形化效果，先采用最简单的 UI 显示。不关注增加玩家体验的精美的角色选择、存档选择界面、游戏操作提示等，也不关注多种多样的怪物、角色，先专注于实现游戏最为基础的内部逻辑，实现最原始 *UI* 界面上、一种怪物、一种角色、一个关卡的原始版本游戏。

3. 后续开发：不断地迭代，增加新的怪物、角色种类等各种机制，增加多个关卡，优化 UI 界面，改善玩家交互流程的交互体验，增加存档等其他功能，调整游戏中的角色和怪物的各项属性值，以增加游戏的可玩性和平衡性。

5.2 单元测试和 Maven 构建

将项目改为 Maven 构建，使执行单元测试、打包为 jar 文件等操作更为方便。

单元测试部分，采用了 JaCoCo 插件生成单元测试报告，报告如图11所示,Method 覆盖率达到百分之 55(是我代码写的不够好，以及把能测试的都测了)。

Roguelike

Roguelike

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
roguelike.asciiPanel		30%		17%	161	180	235	360	38	52	1	3
roguelike.screen		40%		21%	51	65	121	193	16	24	3	5
roguelike.game.creature		58%		46%	65	113	98	227	17	43	0	8
roguelike.game.bullet		43%		25%	55	70	107	170	7	18	2	6
roguelike.game.weapon		41%		0%	19	22	20	33	6	9	0	3
roguelike.game		84%		75%	14	49	19	114	5	27	0	8
roguelike.mainWindow		64%		n/a	4	8	21	41	4	8	0	1
roguelike.music		88%		66%	2	9	3	24	0	6	0	1
roguelike.map		100%		98%	1	51	0	80	0	16	0	2
roguelike.ui		100%		n/a	0	3	0	7	0	3	0	1
Total	3,864 of 7,385	47%	440 of 683	35%	372	570	624	1,249	93	206	6	38

图 11　JUnit 单元测试

Figure 11　JUnit Test

5.2.1　编写单元测试的体会

编写单元测试时，我深刻体会到了自己代码的不足，类中有大量的 void 方法，纯粹的 side effect（只改变了对象的状态，无返回值），使得单元测试的编写变得极为困难，只能在调用要测试的方法后，查看对象的相关属性是否发生变化，来进行测试。测试过程十分困难，大量代码的正确性难以测试。

此外，我的代码对于大量可能的异常情况都没有进行合适而完备的异常处理。想要编写具有 @expected 属性的单元测试来测试是否抛出相应异常，也无法编写有效的测试用例。

分析原因，我的面向对象设计能力还有待提高，部分代码还是带有了过程式编程的思维习惯写的，没有很好地遵循 SOLID 设计原则，一些类之间的耦合度过高，导致测试困难。

6　课程感言

6.1　我的收获

1. 对于面向对象程序设计，以及”程序设计”概念本身的理解上有了层次上的提高。

上学期学完《高级程序设计》，我仅能按部就班、较为熟练地运用 C++ 进行面向对象编程，但我对面向对象的理解层次很低，没有真正地理解面向对象的各种特性，而仅仅是”知道”罢了。本学期的课程中，我真正体会到了面向对象的博大精深，也体会到真正写出”好的代码”和”好的面向对象代码”是一件很难的事情，，真正的”程序设计高手”和会写代码之间还有很长的距离，我的境界还差的很远，需要更多的修炼。

2. 我在如何有效并高效地寻找自己需要的知识，并进行自主学习以及将学到的知识化为己用，运用到代码实践中，这两方面的能力得到了极大的锻炼，而我相信，这也是最重要、最应该学习的能力。

本课程的内容很多、有一定难度，并且有大量的可自行拓展学习的东西，老师上课时做一个介

绍和引入，让我们明白基础知识和总体的知识框架，课后，我们通过书本、老师给出的补充阅读材料以及自行上网搜寻资料，进行课外的拓展学习。这样的方式极大地锻炼了我，让我能够高效地完成” 需要学习什么知识，准确地搜索到需要的资料，学习掌握知识，运用于代码中” 的过程。

3. 当然，我对 Java 语言本身的语言特性也有了一个初步的了解，最后的大作业也证明了我能用它实现一个效果不错的小游戏，还算是对 Java 有了一定的初步掌握。我真实地体会到了 Java 这门语言的魅力，的确是一门非常好用而博大精深的语言，在我心中，其地位已经远超 C++ 了 (滑稽)。

6.2 对课程的意见和建议

上课时，老师您常说有一些” 较为基础” 的内容，讲的速度较快，我们通过自行学习可以学会。我的体会是确实如此，但如果不通过预习的方式先弄明白 “基础部分”，上课时会出现基础部分老师讲的较快，还没完全搞明白，而讲到难的部分的时候因为基础没有完全搞懂而听不懂。因此，我认为这门课程需要课前预习和课后自主拓展学习双管齐下。因此：能否在课前几天将课件早一些放出，以便大家预习。如果能告知一下哪些部分最好能够先自己掌握就更好了。

致谢 感谢曹春老师布置了这么精彩的作业，让我学到了很多东西。感谢在 jw05-Jjw08 的开发过程中与我 (在设计理念、思路方面，不涉及具体代码实现) 进行了许多讨论，给我提供了许多建议的许嘉禾同学和王 jun 同学 (名字显示不出来)。

参考文献

1 www.baeldung.com
2 www.javatpoint.com