:

SCIENTIA SINICA Informationis

报告

# Java 高级程序设计开发报告

## 1 项目介绍

包含肉鸽要素的 RPG 游戏。游戏的地图，大部分普通敌人，特殊地形都是随机生成，保证游戏的随机性。人物控制角色有可以成长的属性，有灵活使用的技能，保证玩家的主观能动性。

### 1.1 单机玩法

玩家控制主要角色葫芦娃，通过击败敌人概率获得金币，花费金币可以提高自身的属性。探索地图，寻找 BOSS 并与之战斗，击败 BOSS，找到爷爷被关押的位置，救出爷爷，以达到游戏胜利。若在途中被击败则游戏失败。如果觉得某次随机地图还不错，或者某次挑战 BOSS 没有把握，可以选择进行存档，方便下次游玩。

### 1.2 联机玩法

玩家救出爷爷后，爷爷也可以参与战斗。联机游戏下，可以支持双人挑战，一位玩家控制葫芦娃，另一位玩家控制爷爷。在联机模式下怪物会源源不断的刷新，挑战自我，争取坚持的更久。

## 2 设计介绍

### 2.1 总体设计

程序的总体设计如图1所示。分为四个主要部分

- 用于实现图形化的 AsciiPanel。借用提供的图形化框架，对原有的框架稍加更改，可以绘制自定义的游戏图像。
- 用于维护游戏内各个体的 world 和 creature。world 负责对游戏地图进行随机生成和维护，记录所有已经生成的个体；creature 负责各类敌人、玩家的类定义，初始化和启动生成各类个体的工厂。
- 用于交互的 Screen。Screen 负责绘制出当前游戏进行的状态，同时响应玩家的键盘输入。
- 用于联机的 Server。通过 NIO selector 技术实现的网络通信功能，负责游戏的联机部分。

### 2.2 Screen 接口及各派生类

Screen 及其派生类的继承关系和主要成员变量、成员方法如图2所示。Screen 包括接口屏幕显示功能函数 displayOutput、响应键盘输入函数 respondToUserInput、respondToUserInput_released。

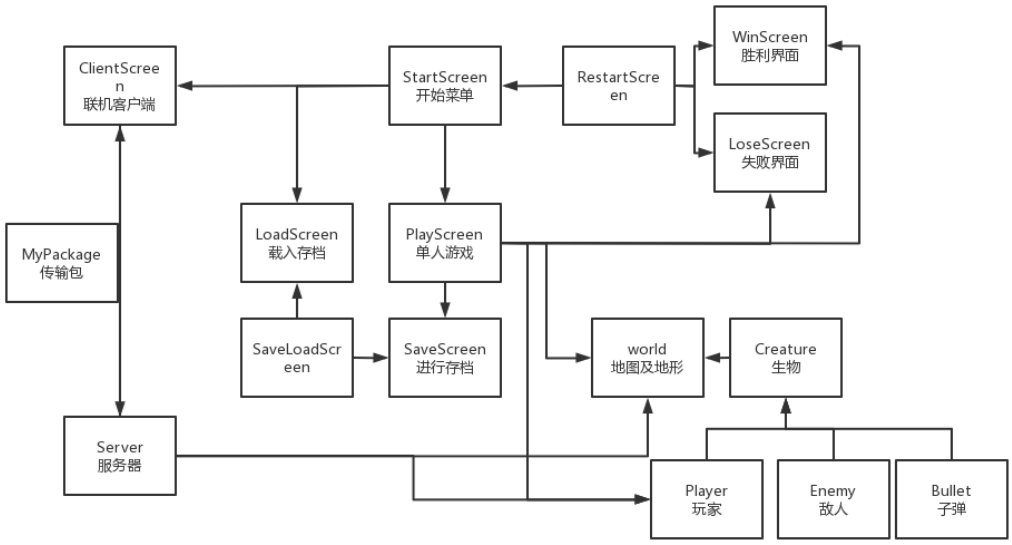


**图 1** 总体设计

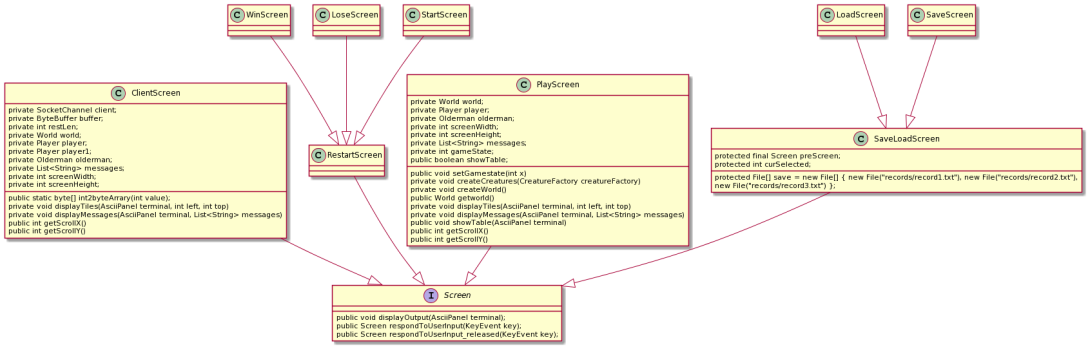


**图 2** Screen 及其派生类

### 2.2.1 StartScreen 类

游戏的开始界面，其对键盘的响应结果是创建并进入不同的界面。玩家根据提示键入对应操作后即可进入所需功能的界面。

### 2.2.2 RestartScreen 及其派生类

RestartScreen 类作为重新开始游戏的界面，在接收到回车后重新加载 StartScreen。在需要进行重新开始游戏的场合使用其派生类，如 WinScreen、LoseScreen 等。它们屏显的内容不同，但都有相同的 respondToUserInput。

**图 3** Creature 及其派生类

### 2.2.3 PlayScreen 类

PlayScreen 作为游戏的交互主体，最为复杂。首先它维护当前游戏的 world，对于 world 中的每一个 creature 和每一块 tile 都做跟踪。其次它也维护当前玩家控制的 Player，它的 respondToUserInput 响应的是玩家正常游戏过程中的键盘输入，对输入做简单的判断（如是否开启存档功能、是否显示帮助界面），然后将键入的键码值传给 Player 的内置 respondToUserInput 函数进行后续的操作。同时它负责游戏过程中的细化屏幕显示，对每一块 Tile、每一个 creature 进行处理（是否可视等）后在终端游戏界面绘制。

### 2.2.4 SaveAndLoad 派生类

特化的 Screen 类负责的是游戏的存档读档功能。SaveLoadScreen 绘制了存档界面，为用户存档的输入做出响应。PlayScreen 拥有进入它的入口，玩家在游戏途中进入该界面进行存档，该类有 file[] 的成员，其中存放了作为存档载体的文件。玩家存档时选择某个文件名进行存档，即新建存档文件或覆盖已有的存档文件。存档功能由 SaveScreen 类实现，存档的实现方式比较粗暴，前文提到的 PlayScreen 设置为可序列化，存档即通过将当前游戏的 PlayScreen 序列化并写入对应的存档文件来实现的。由于 PlayScreen 本身维护了包括 world 和 Player 在内的几乎所有游戏信息，这样粗暴的存档实现方式虽然有较大的开销，但是在本地进行的 IO 操作下可以接受。

### 2.2.5 ClientScreen 类

特化的 Screen 类负责的是联机游戏的内容，详见后文 Server 的介绍。

## 2.3 World 和 Creature

Creature 及其派生类的继承关系和主要成员变量、成员方法如图3所示。

### 2.3.1 World、地图

WorldBulid 类用于生成随机地图，通过调用 MazeGenerator 随机生成地图算法，再进行一些处理，包括地形成块、添加特殊地形，生成一个二维 Tile 类数组表示地图。World 类作为整个游戏的“容器”，worldbuild 生成的地图信息由它存储，PlayScreen 再从它取出并绘制，同时每一个 creature 生成后也由他存储，PlayScreen 再从它取出并绘制。

### 2.3.2 Creature 及其派生类

Creature 类包含了生物体（包括敌人和玩家）的所有状态信息, 如:

- 坐标值 x 和 y
- 生命值 HP 法力值 SP
- 攻击力 attackValue 防御力 defenseValue
- 可视范围 visionRadius
- 金钱 money

也定义了许多行为函数，如:

- 攻击 attack()
- 相遇 meet()
- 移动 moveby()

分为四类：玩家、敌人、NPC、子弹（设计失误，将子弹也归为了 creature）
对于玩家，在 Creature 的基础上，另外添加了玩家所特有的状态信息和行为函数:

- shoot() 射击以及子弹威力、蓝耗
- treated() 恢复以及回复量、蓝耗
- steal() 偷窃
- dig() 挖掘
- respondToUserInput() 响应玩家键盘输入，由 PlayScreen 调用

对于敌人，在 Creature 的基础上，各个敌人有各自特殊的状态信息和行为函数:

- shoot() 射击以及子弹威力、射程，蝙蝠怪特有
- walk() 巡逻以及巡逻路线，青蛙怪特有
- flash()、rush() 等 BOSS 特有的技能

对于子弹，在 Creature 的基础上，子弹有自己的状态信息和行为函数:

- aim_type 目标，用来分辨敌我子弹
- fly() 子弹飞行函数
- hurt() 子弹击中函数

对于 NPC，也就是单人模式下爷爷角色，在 Creature 的基础上，当 meet() 识别到葫芦娃与爷爷相认时，判定游戏胜利，会返回 WinScreen。

### 2.3.3 CreatureFactory 类

作为生产 creatur 的工厂。在 World、PlayScreen 和各类 Creature 中充当桥梁，PlayScreen 拥有 CreatureFactory，通过调用 CreatureFactory 的方法生成 Creature，并将其加入到维护的 World 中。

## 2.4 Server 类和网络通信

网络通信功能涉及三个问题:

- 服务器维护游戏进程

- 客户端响应玩家操作
- 服务器和客户端的通信

与单人模式相比较，在 PlayScreen 中实现的游戏逻辑囊括了维护游戏进程和响应玩家操作这两个部分，设计联机模式就需要将这两部分功能拆开，分别由服务器和客户端实现。

### 2.4.1 服务器

Server 类中实现了一个简单的服务器。分为两个部分。其一：维护游戏进程，PlayScreen 中所有的维护 World、通过 CreatureFactory 的方法生成 Creature 的功能，Server 都有复用。但是碍于 PlayScreen 设计的太过粗放，为了维持网络通信的质量，Server 对 world 的规模、Creature 的数量都做了调整。其二：充当网络通信的服务端，与客户端建立连接并进行交互。

### 2.4.2 客户端

见上文提到的 ClientScreen 类，它就是网络通信功能中的客户端。同样分为两个部分。其一：响应玩家操作，ClientScreen 包括 Screen 接口，它的 respondToUserInput 函数简单直白，就是将接受到的玩家键入键码值传给服务器；它的 displayOutput 与 PlayScreen 功能类似，也是对游戏过程进行细化屏幕显示。不同之处在于，ClientScreen 的 world、player 等成员并不由自身维护，而是通过不断接受来自服务器的数据并转录为这些游戏信息。其二：充当网络通信的客户端，与服务端建立连接并进行交互。

### 2.4.3 通信细节

采用 NIO Selector 技术实现网络通信。服务器和客户端通过 Buffer 写入字节流进行通信。对于服务器来讲，它接受到的字节流应该为一个键码值，它传输出的字节流应该包括 world、player 等游戏信息；对于客户端对应相反。如何将这些内容准确无误的写入 Buffer 就成为主要的问题。客户端到服务器的键码值，通过位运算等操作，将其按每四个字节写入一个键码值写入 Buffer，服务器只需每四个字节将 Buffer 读取并强制转换为键码值，就可以根据键码值对 player 进行操作；服务器到客户端的游戏数据，则打包传输，包头写上传输包的长度，内容包括当前游戏的 world 信息，客户端所控制的 Player 信息以及队友信息。

## 3 困难及解决

1. 实现自定义图形化。涉及到对 AsciiPanel 的改造，困难在于研究 AsciiPanel 的代码逻辑，实现自定义图形化，就是改变 AsciiPanel 的绘制函数 paint()，将从图片资源读取的内容正确显示。
2. 多线程同步带来的问题。将每一个 Creature 通过单独线程控制带来了许多问题。如果不加以同步锁，很容易出现多个线程对同一个资源进行操作导致崩溃。
3. 没有良好的设计规划带来的一系列问题。因为没有好好的规划设计，导致太多功能的实现集中在了 PlayScreen 类中，使得代码冗余过多。由于开发过程天马行空，想到哪就添加一个函数，

导致子类继承没有体现出实际作用。另外许多功能实现都是纯副作用函数，编写测试更是无从下手。

4. 网络通信的质量较差。由于游戏数据设计的太冗余复杂，为了能在网络通信过程中不发生栈溢出等问题，联机模式不得不阉割了许多玩法功能，缩小了游戏体量。

## 4 课程总结

在选择这门课前，我的想法是多掌握一门语言。课程结束才发现，Java 语言之外，面向对象的高级程序设计模式更是课程内容的重点。Java 语言自身的优势，使得它作为学习面向对象程序设计的载体，能够提供许多复杂功能的实现框架。在之前学习高级程序设计课程时，大作业同样是开发一款简单游戏，但是没有 java 支持的网络通信框架、便利的图形化框架和多线程编程，体验就完全不一样。这门课对于我这样的新手来说，除了 Java 语言编程外还有很多赠品。诸如绘制 UML 图表示数据结构、攥写 Latex 格式的实验报告、配置各种各样软件和应用环境等等，总之受益良多。