SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

李广源[1]

1. 南京大学, 南京191220048
E-mail: 1311590726@qq.coom

摘要 本项目是设计一个冒险岛类型小游戏，在设计过程中主要使用了面向对象，多线程等技术，采用文件写入读出来保存游戏进度，使用Socket来进行多人对战。.

锟截硷拷锟斤拷 开发目标, 设计理念, 技术问题, 工程问题, 课程感言

1 开发目标

1.1 游戏介绍

我写的是一个冒险岛类型小游戏，核心玩法是玩家通过射击清除屏幕中的所有怪物即为成功。我写这个类型的游戏的灵感最初来自于我以前玩过的一款小众的2D横板闯关类游戏iwanna，iwanna游戏主角kid需要通过重重机关并且打败许多boss才能通过。但由于自身能力、时间有限，废除了繁重的机关设计，只保留了2D横板打小怪的风格，也就是完成版冒险岛类型游戏。随着课程进行和游戏功能的完善，游戏公开发了两个个版本。最初始版本为单人游戏，玩家操控主角清除完全部的怪物即可获胜，在这之后添加了存档读档功能，在单人游戏下玩家可以随时通过esc键存档并退出，通过load from save.txt 选项读取存档；第二个版本添加了多人对战系统，玩法与单人游戏玩法不变，同样是清除完所有怪物即为获胜，但如果任何一个玩家死亡，则游戏失败。

1.2 详细玩法讲解

1.2.1 单人游戏

开始界面选择Single game，玩家键盘控制射击方向（W向上，A向左，S向下，D向右）和移动方向（A向左移动，D向右移动），J射击，K跳跃。玩家控制主角击杀地图内全部怪物即可获得胜利，相反，若被怪物击杀则游戏失败。怪物在玩家靠近时会主动追击玩家，攻击方式为接触攻击，玩家与怪物接触就会损失生命值；如果怪物周围没有玩家，其行动路径随机。

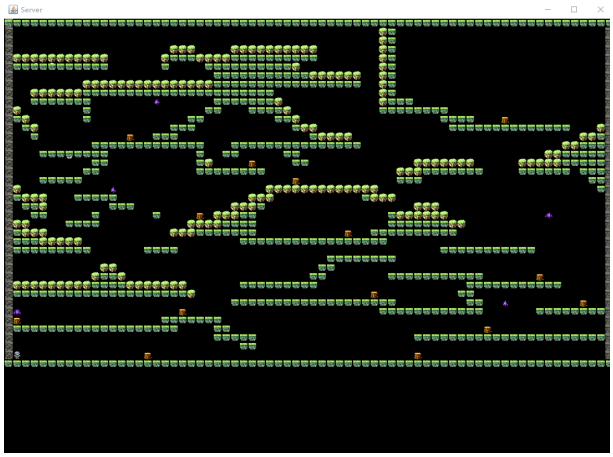

图 1 单人游戏

单人游戏有独特的存档、读档机制，在游戏过程中按下ESC键可以实现保存游戏并退出，另外提醒，通过直接关闭游戏等非正常退出是不会保留游戏数据的。存档位置只有一个，如果进行了多次存档，游戏仅会保留最近一次存档数据。若要读取上一次存档进行游戏，在开始界面选择load from save.txt选项即可。游戏存档的实现是通过将游戏数据写入save.txt中，在读取存档时对save.txt 里的内容进行解析化为游戏数据。

1.2.2 联机游戏

连接双方需要处于同一局域网下，房主一方启动server，另一方以房主ip地址为参数启动client，二者启动先后顺序不同会有一些差异。若client 在server之前启动，client会尝试10次连接（大约10s），如果没有连接到server，client会自动关闭。游戏最多支持两名玩家，但一个client连接上server时，自动开启游戏。游戏玩法与单人游戏玩法规则类似，玩家们清除完地图上的全部怪物即为游戏胜利，若玩家中任意一员死亡，则游戏失败。

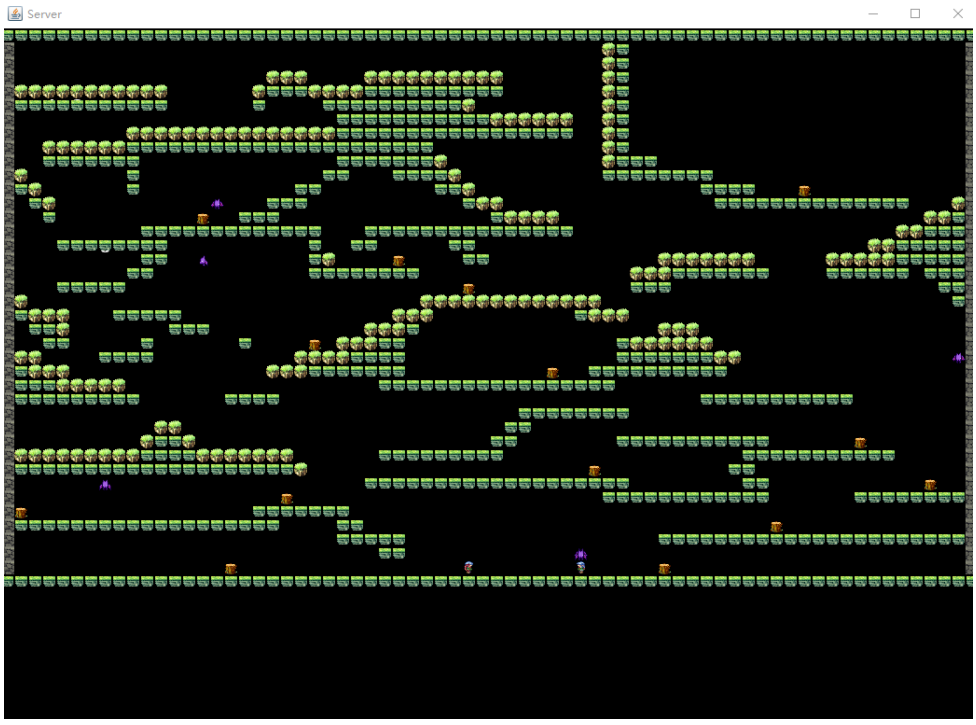

图 2 多人对战

玩家2死亡后，玩家1和玩家2画面显示：

图 3 游戏失败

1.2.3 优化方向

整个游戏是一个较为简单基础的部分，在游戏玩法上并没有很大的创新。首先可以增加图形化的葫芦娃，豆子和地面。现在仅使用基础图形，这个方面可以进行优化。其次在玩法上可以模仿市场上的贪吃蛇大作战等多人游戏，死亡之后尸体会形成豆子，这样可以大大加快游戏节奏。此外还可以创新玩法，比如没有豆子的存在，贪吃蛇的变长是依据时间的，在相同时间内每个玩家的贪吃蛇都会增加相同的长度，而且运动时间随着长度增加会不断加快，蛇头碰撞到别人或者自己都会死亡，死亡即淘汰。另外整体面向对象的结构也有一些紊乱，更加条理化，增加测试也是优化的方向。整体来说，还有很大的进步空间，预计在寒假中继续完成。

整个游戏仅包含打怪部分，在游戏玩法上并没有太大的创新，如前面所言那样自身能力不足不足以完成整个游戏设计。但游戏基础部分是完成的，后续可以添加更多功能优化完善游戏。如，增加怪物种类，目前怪物仅有一种且只能接触攻击，可以添加弹幕攻击类型怪物、自我增殖类型怪物、提供增益或负面效果的怪物等，来增加游戏难度。除此外，还可以添加道具设定，玩家通过宝箱或者是打怪获得增益道具，提升游戏可玩性。或者是添加关卡设定，不同关卡有不同的环境、怪物，让玩家对地图保持与新鲜感，同时不同关卡的探索能保证游戏的趣味度等。总体而言，目前游戏基础功能完善，具有向上提升空间。但游戏面向对象的结构设计不够合理，代码过于臃肿，极大影响了运行效率。完善对象设计、改进底层运行逻辑也是未来优化代码的方向之一，我也会在之后保持游戏更新，逐步完善游戏。

2 设计理念

2.1 代码整体布局

由于联机游戏功能的加入，游戏整体设计相较于jw05出现了大的变动。首先，代码重新分为两个相互独立的部分，一部分是服务器server，另一部分是客户端client，server负责游戏本体GameBody的运行和GameBody信息与本地展示Display类和client的之间的传递；client负责接受server发送的数据并解析传递到本地Display类进行展示。

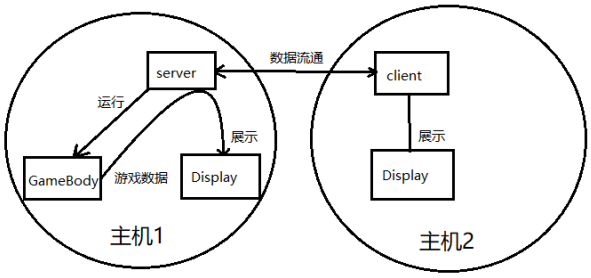

图 4 代码整体布局

2.2 主要类介绍

1.GameBody类，由jw04的PlayScreen类演变而来，处理游戏内实体交互的结果，不同的是删除了将交互结果（屏幕数据）展示的部分，而是将其转化为屏幕数据发送给Server类，然后在Display上展示。

2.Wold类，用于加载地图和处理地图内子弹、怪物、玩家间的交互。

3.CreatureFactory类，构建creature的工厂，创建玩家PlayerAI和怪物CommonMonster。

4.PlayerAI类，玩家控制的对象，通过键盘实现player移动、跳跃、射击。

5.CommonMonster类，控制怪物的移动、攻击。

6.Creature类，实体类，由于构建记录玩家、怪物、子弹等的信息。

7.Tile，地图元素的枚举类型，用于地图构建。

8.Screen类，将屏幕数据显示到terminal。

9.Display类，由ApplicationMain演变而来，仅保留了选择控制不同的Screen功能。

2.3 设计的思路

游戏是在在jw04原有的代码逻辑上，基于jw04框架下修改添加的。word在world里对所有实体creature进行集中管理，负责creature的添加、删除、修改。有world管理的实体包括player、button、monster以及地图元素floor、wall等，而player、button、monster需要有自己的UI独立运行，是实体集中的特殊元素，它们的UI则通过多线程运行，针对三者设计相似但不同的CreatureAI。GameBody负责调整player、monster的运行并将交互结果转化为屏幕数据，传递给server。之后就是server、client、display之间关于屏幕数据的传递、显示。

3 技术问题

3.1 面向对象

该游戏对于java对面对象的需求很高，本项目采用面向对象的方法，但是在设计对象性质和交互范围较为混乱，对象内资源相互调用十分复杂，这也是未来需要改进的地方之一。

3.2 并发编程

本项目采用并发编程，在单机游戏中，每一个可以运动的实体都需要开辟一个线程来执行该实体逻辑，如player，monster，button。除此外，游戏本体运行（GameBody获取游戏数据）、屏幕刷新（Screen对象显示屏幕数据）都需要开辟线程执行；而在联机游戏中，server的接收端、发送端、client的接受端、client发送端此类需要时刻进行网络通信的也需要开辟线程执行。总体而言，游戏开辟了许多线程执行，而线程之间对于同一资源修改可能会引发意想不到的错误，在游戏中，我没有对多个进程访问同一数据资源进行限制，但对访问后导致的结果进行了控制，对多进程使用的资源在使用前进行检查，防止程序终止类错误。

3.3 文件读写

存档读档：存档即将游戏数据进行合理包装后转化为String写入save.txt文件中，读档即从save.txt读取String进行与存档包装相对应的解析获得游戏数据，将其加载进游戏中。socket通信：原本打算使用ObjectStream来进行主机间数据流处理，可由于client获取的对象在传递中因不知名因素发生了改变，故采用字符流的方法。将游戏数据转化为String字符流发送给client，client在把String解析为游戏数据。

3.4 网络对战

采用socket来实现网络通信。首先要打开server，server会在主机特定端口等待连接建立。然后打开client，client会向特定ip地址、端口发出连接请求。然后client 就可以与server 建立连接。游戏运行本体GameBody是由server控制完成，本地（主机1）上，server接受主机1上的键盘信号并传给GameBody处理，在网络通信上，server从端口接受client发来的键盘数据也交给GameBody处理。处理完成后，server获取GameBody的屏幕数据传给本地Display类显示，同时server也将屏幕数据从端口发送给client，client将接受到的屏幕数据传递给本地Display类进行显示。

4 工程问题

很抱歉，并没有采用自动构建工具进行工程管理。

5 课程感言

高级java这门课让我学到了很多东西，特别是多线程执行与网络通信方面，我感到获益匪浅。说起来惭愧，这是我第一个尝试使用并发编程来完成项目，而不是通过while(true)循环运行整个项目逻辑，同时也是我第一次正式学习并发编程。同时，高级java也运用了许多启发式学习方法训练我，它作为一门拥有这大作业的课程，在将课堂上学到的知识运用到复杂的作业项目中，难免会遇到许多问题。这时就需要我去查阅资料、研究调试代码，也提高了我独自动手解决问题的能力。但惭愧的是，即使老师你一而再、再而三地强调了拖延症问题，我还是受拖延症影响没能按时完最后一次作业，真是非常抱歉！总体来说，我非常感谢曹春老师的教导，让我第一次了解进行并发编程，第一次编写了通过网络通信进行多人对战的小游戏。在寒假，我还会继续完善我的java小游戏，希望能够实现游戏优化方面的内容。