中国科学: 信息科学

SCIENTIA SINICA Informationis

JAVA REPORT

Developing a Java Game from Scratch

梁安泽

NJUCS, 南京 210000
E-mail: 1468623593@qq.com

摘要 综合 JW04-JW07 的所有要求，完成一个完整的图形化网络对战游戏，其中包括了开发目标、设计理念、技术问题等方面的阐述.

关键词 JAVA, 网络对战, 面向对象, 多人联机

1 开发目标

1.1 设计思路

回顾整个开发的过程，其实一开始我并没有很明确的开发思路。我最早的想法是基于 JW04 的迷宫地图进行修改，让玩家能够控制角色走出迷宫，在这个过程中，需要玩家不断避开敌人以免受伤害，同时还可以拾取奖励来增加生命值和得分。最开始我确实这么做了，差不多完成的时候，才发现群里有人把“葫芦娃和妖怪的对战游戏”拓展成了坦克大战..... 于是我便放弃了原先的玩法（实在是太无聊，界面也太普通了），1.0 版本到此结束。

为了修改游戏的玩法，我决定将其改为类似 2D FPS 之类的游戏。即为玩家提供射击的技能，让玩家去达成某种目标；同时为怪物提供不同的攻击手段，以抵御或击伤玩家。因为 1.0 版本中已经实现了远战怪和近战怪，以及各个角色之间的攻击、得分机制，因此只需要重新修改地图，并将角色放置到合适的位置即可。这便是 2.0 版本的基本思路。

顺利完成 JW05 以后，还需要进一步修改游戏，使之具备录像功能，以及后续的联网对战功能。出于时间限制（月底前四天要考完五科，真的挤不出时间了），我觉得直接修改 JW05 的游戏，不增加额外的游戏元素，直接在其上实现作业要求。这便是 3.0 版本的设计思路。

1.2 核心玩法

有了上述的基本设计思路，只需要确定最基本的核心玩法即可。作为类 2D FPS 游戏，玩家可以向上下左右四个方向射击。当所有的敌人被消灭的时候，玩家阵营胜利，否则玩家阵营失败。同时在对战过程中，可以通过拾取奖励来增加生命值以及得分。对于单人游戏而言，玩家需要躲避敌人的追踪以及攻击——敌人颜色一旦变深，则表明玩家已被发现，随时会遭到攻击。

1.3 BOT 逻辑

进行单人游戏的时候，BOT 需要具备一些基本的移动和攻击逻辑。我将他们的逻辑设定如下：

1.3.1 近战 BOT 逻辑

近战 BOT 需要较大的活动区域，即它应当在一定区域具备防守能力。同时应当具备一定的可视范围，当玩家进入可视范围的时候，它就会主动出击，寻找最短的路线去攻击玩家。因为它的攻击范围较小，因此需要将其放置在障碍物较多的地方，同时赋予更高的移动速度。

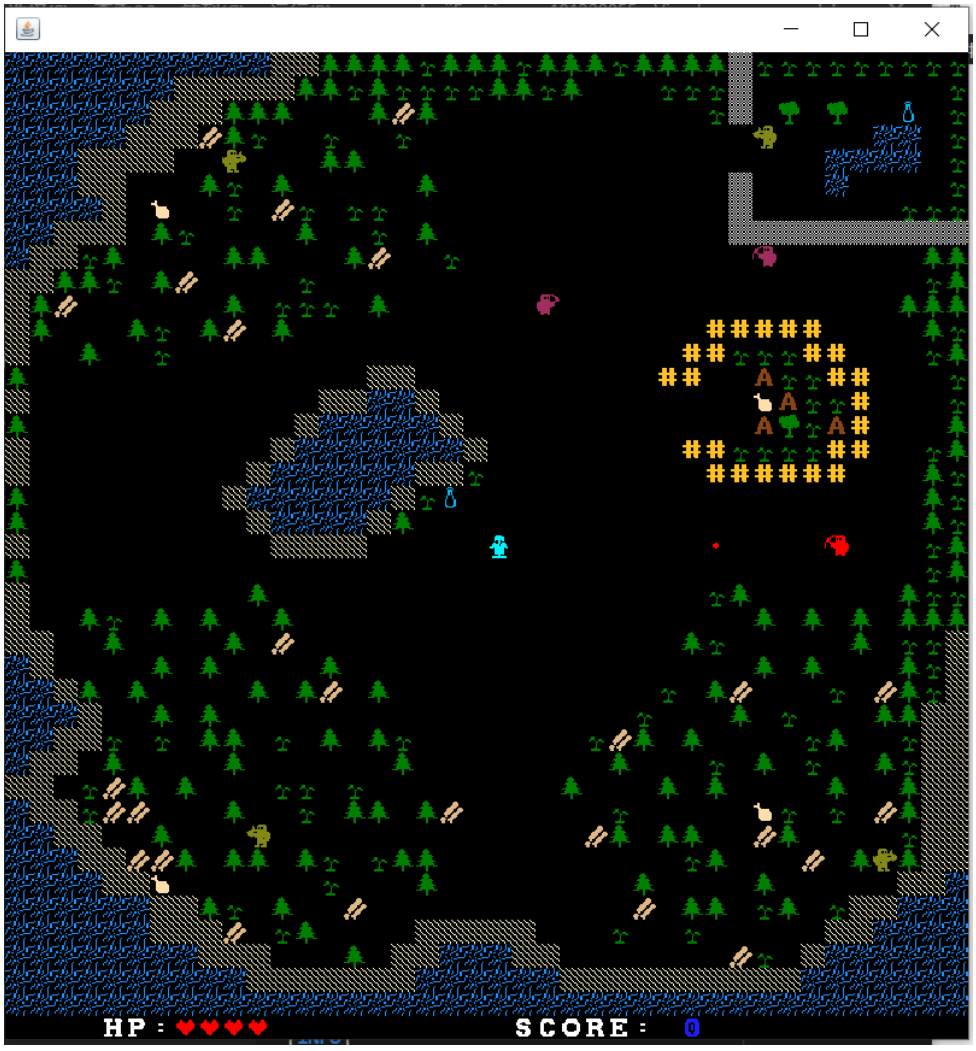
1.3.2 远战 BOT 逻辑

远战 BOT 因为具备远程攻击的优势，因此它的活动区域和移动速度较小，同时也应当活动在较为开阔的地方。当玩家进入它的攻击十字范围的时候，它便会开火。

1.4 UI 设计

UI 设计这部分是最让人头疼的。起初我直接使用的是 JW04 的 UI，但是觉得不够美观，边打算自己设计图案并整合出一张字符表，然后最终还是放弃了。后来在网上搜索一番之后，发现 Roguelike 游戏的贴图貌似十分的丰富，最终我选定了下面这一张：

之所以选中这一张，是因为其中包含了若干角色，以及树木、墙体、水流等地图元素，只需要确定具体的颜色，便能实现美观的 UI。因为起背景颜色以及字符颜色与代码框架不符，因此我通过 Python 脚本将其转换成了黑白的版本。这便是 UI 最基本的来源。以下是最终确定好的地图效果 (略丑)。



2 设计理念

首先在 JW06 中就有类似记录游戏状态的要求，因此需要设计一套统一的状态机制，它不仅可以起到记录作用，同时也可以对后续网络对战的实现起到通信作用。事实上类似的思想我在 JW05 最初设计的时候就已经做出来的，它依赖于一个全局的、对地图起到管理作用的类——地图类 Map。在这个类当中，我对地图上物体状态的改变统一为以下两个方法：

2.1 物体放置

这一方法主要用于地图的初始化，将玩家、BOT 和奖励物品放置到地图当中，以及发射炮弹时使用。因为地图上一个位置规定只能放置一个物品，因此使用这一方法将会判定放置的合法性，并返回到调用者当中。同时，如果放置的是炮弹，而位置上已经存在玩家或者敌人，则会扣除对方的生命值。

```
    public synchronized boolean setThing(Tuple<Integer, Integer> pos,
                                         int type, Thing t) {
    boolean res = false;
    lock.lock();
    if (map[pos.first][pos.second] == 0) { // 位置为空
        ...
    } else { // 位置不为空
        if (t.getType().equals("cannonball")) { //放置的是炮弹
            ...
        } else {
            ...
        }
    }
    lock.unlock();
    return res;
}
```

2.2 物体移动

这一方法主要用于物品的移动，包含玩家和 BOT 的移动以及炮弹的移动。当移动的是玩家时，会对目标位置进行判断，若为奖励物品则会增加玩家的生命值和得分；如果移动的是炮弹，则会判断炮弹是否命中目标，若是则会扣除对方的生命值。

```
    public synchronized boolean moveThing(Tuple<Integer, Integer> beginPos,
                                          Tuple<Integer, Integer> destPos) {
        boolean res = false;
        lock.lock();
        Thing t = world.get(beginPos.first, beginPos.second);
        String type = t.getType();

        if (map[destPos.first][destPos.second] == 0) {// 允许移动
            ...
            res = true;
        } else { // 该位置不可移动
            if (type.equals("cannonball")) { // 该物体为炮弹
                ...
                res = false;
            } else if (type.equals("player")) {// 该物体为玩家
                res = true;
```

```
            }

        }
        lock.unlock();
        return res;
    }
```

上述设计理念的优点是显而易见的，它简单且高效。当我在实现 JW06 中的录制功能时，可以在上述方法中输出一条 Log 信息，它指明了是放置物品还是移动物品；当我在 JW08 中实现多人对战时，可以通过网络传输一条 Log 字符串来进行通信。这一设计理念贯穿了前后四个实验，为其余功能的设计带来巨大的便利。

3 技术问题

在整个游戏的开发过程中，主要涉及的是多线程的同步以及网络通信的问题。多线程同步得益于上述设计理念，较快地得到了完成；而网络通信方面，我主要采用了状态同步的方法。

状态同步的方法非常简单：服务器端统一管理所有玩家的状态，当有玩家更新状态的时候，只需要发送给服务器；如果新状态是合法的，则服务器会将其广播给所有玩家，玩家收到之后更新本地的状态即可。为此，我设计了服务器类和客户端类，以及他们之间的通信信息类。

3.1 服务器类

服务器类用于接受玩家发送的信息，然后对信息进行处理，最后将合法的信息广播给所有玩家。它包含了以下几个主要的方法。

```
// 接受连接请求，增加一名玩家
private void accept(SelectionKey key) throws IOException
// 读取一条信息
private void read(SelectionKey key) throws IOException
// 处理一条传入的信息
private void handleInputFromClient(SelectionKey key, String s)
//向某通道写入一条信息
private void write(SelectionKey key, String s)
// 向所有玩家广播一条信息
private void broadcastToAllClient(String s, SocketAddress exception)
```

因为服务器端需要与多个客户端进行通信，故服务器端采用的是 NIO 的通信方式。

3.2 客户端类

客户端其实也分为两种，一种是依附于服务器端的客户端，即” 创建房间的玩家”，另一种是独立的玩家。因为这两类客户端的重合度较高，因此我将他们并入一个类当中进行实现。它主要包含

了以下几个主要的方法。

```
// 建立与服务器的连接，用于初始化本类时调用
private void establishConnection()
// 开始从服务器端读取数据
private void startReadFromServer()
//处理服务器端的输入
public void handleInputFromServer(String[] infoFromServer)
//处理本地的按键输入
public void handleKeyEvent(KeyEvent key)
//向服务器端发送一条数据
private void writeToServer(String s)
```

对于客户端而言，因为仅有服务器端一个网络输入，故单独开启一个线程，并使用阻塞的 IO 进行监听。

3.3 通信信息类

该类的作用非常简单：构建一条通信信息，并将其转换为字符串供给服务器和客户端通信。它规定了服务器和客户端之间的通信只能使用以下几种信息：

```
% 关于地图上物品的放置和移动
setThing //放置物品，需要指定物品的位置、类型、颜色等信息
moveThing // 移动物品，需要指定物品的起始位置和结束位置
launchCannonball  // 发射炮弹，需要指定发射的起始位置和方向

% 关于建立玩家和服务器端的连接
playerJoin // 新玩家申请加入
refuseToJoin //拒绝玩家加入
admitToJoin // 同意玩家加入，为玩家分配初始信息
playerLeave // 玩家离开游戏，需要指定玩家的ID

% 关于游戏状态的控制
startGameRequest //开始游戏的请求
startGame //开始游戏，服务器向所有玩家广播该消息
addScore  // 为某一位玩家增加得分，需要指定玩家的ID
gameOver // 游戏结束，服务器向所有玩家广播该消息
resetGame //重置游戏，开启下一轮对战
```

4 课程感言

我认为这次 JAVA 学习之旅是收获满满的，不仅收获了 JAVA 的代码经验，并且加深了对面向对象编程的理解。JAVA 语言最令我难忘的是垃圾回收机制，它意味着我们不需要手动地回收资源，有效地简化了代码地设计。

对于课程的设计，首先我认为上课的质量非常高。相较于传统的老师念 PPT、学生做笔记的方式，本次 JAVA 课程都有一定的突破——第一，PPT 详略得当，重点突出。与之相反的课程有“操作系统”和“计算机图形学”，PPT 又长又说不清楚，甚至错漏百出。在 PPT 这点上，我要给曹老师打个满分。第二，讲课方式上，曹老师不仅讲授基本的知识，还利用一些简单的 DEMO 来辅助讲解，我认为这也是一大亮点。

其次是作业上，作业不仅抛弃了传统的 OJ 评判，让学生自己发挥想象去实现，同时还做得非常 Fancy，这一点是以前所有编程课都做不到的。当然我在这里有一点点想抱怨的：对于我的专业方向而言，十二月底四天时间要考五门科目，因此可用的时间少之又少，和 JAVA 的大作业产生了冲突。如果可以适当延后作业的截止时间，比如考试周结束后一两天，我认为这将对作业质量产生很大的帮助。