中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

苏颖康[1]

1. 南京大学, 南京210000
E-mail: 1093139885@qq.com

摘要 从零开始开发Java小游戏，对代码结构的构思、实现，对面向对象的思考，以及开发过程中遇到的困难及其应对方法

关键词 Java, 游戏, 开发, 面向对象

1 开发目标

我写的游戏是一个简单的打飞机游戏。由于时间安排不当，故选择了较为简单的小游戏完成。灵感主要来源于小时候在邻居爷爷的按键手机上看见的《雷电20xx》游戏。游戏场景发生在太空，有各式各样的太空战机。需要玩家操控自己的飞机，躲过敌人们的枪林弹雨，发射子弹击毁敌机，获取分数。

由于是纵轴射击游戏，玩家的操作也就局限于通过w，a，s，d按键进行上下左右移动，通过j健发射子弹。敌机实现了一种基础的，可以发射子弹以及向前移动。敌机按波次出现，每一波出现固定数量，在全部都被消灭或是离开场地外，才会出现下一波。

可以联机，上限是四人联机。联机用的IP:端口目前是硬编码在代码之中。

也支持回放。在全部玩家退出或是被消灭时，游戏结束，同时在服务端生成回放文件。将回放文件路径作为打开客户端jar包的参数时，客户端就作为回放工具，播放回放文件内的内容。

2 设计理念

2.1 游戏逻辑部分

在明确游戏开发目标后，首先需要把设计到的个体抽象出来。那么涉及到的抽象个体有：玩家的飞机，敌方的飞机，飞机的子弹，整个游戏场地，游戏本体。在编码阶段没有思考清楚，我觉得应该把游戏本体和游戏场地区分开来，但是在动手编码时却将他们的功能都混在游戏本体内。

区分好后，需要提取他们的共同点，实现共同的方法，减少不必要的重复编码。玩家的飞机、敌机、子弹都是场地上的元素，可以被创造、消灭，可以移动。场地、游戏本体与其他不是同一类，故初步的类图如下（大致示意，不包含细节）：

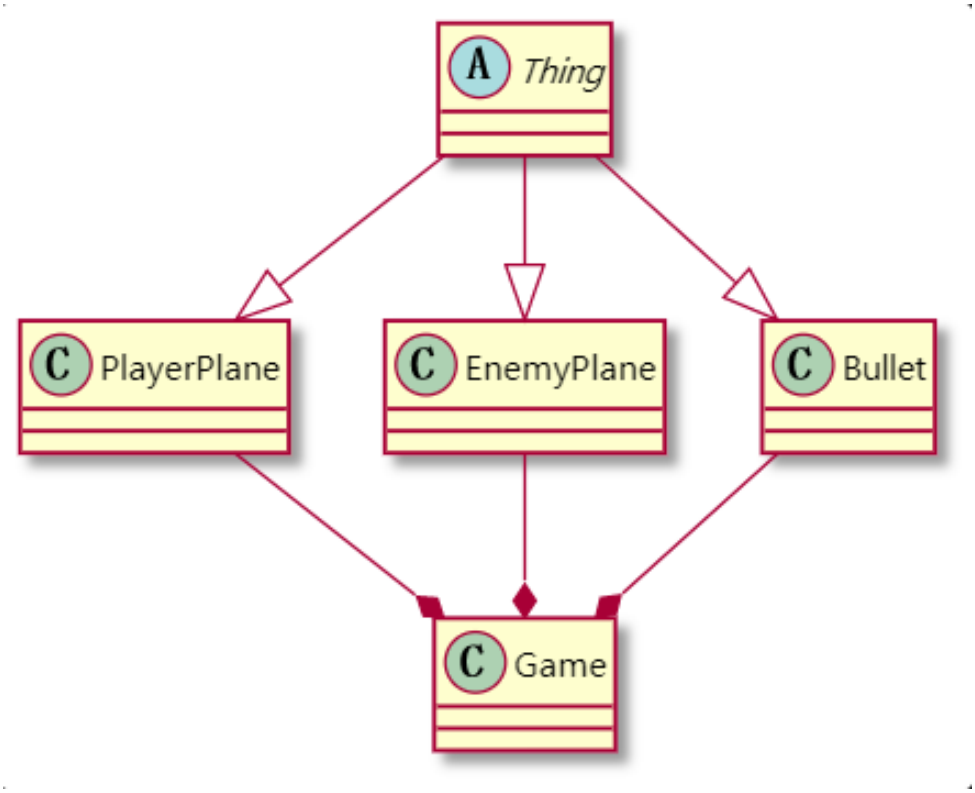

图 1　类图
Figure 1　Class diagram

分好类别，就要设计游戏的运作方式。在之前的Java作业中，完成过八所有生物实现为线程的功能。但是在之完成前的Java作业时，debug遇到了不小的困难。当时采取的方法是把每个生物当作线程，每过固定时间（如0.1s）就执行一遍生物逻辑，同时地图采用管程，每次只允许一个生物修改地图内容。由于线程多，而且线程之间算不上同步，故出bug时难以溯源。

因此，在这次作业我采取了比较偷懒的方式，就是游戏逻辑部分只有一个线程，这个线程由Game类掌握。每过一定时间（如0.1s），该线程就会把游戏内所有实体的逻辑执行一遍。这样做有悖于面向对象的思想了，因为每个实体的时间（时钟）实际上应当是被实体自己掌控的，而不是被游戏掌控。做个类比，一个人的跑步速度应当是取决于自生的，和场地关系不大。但是这种偷懒做法也给编码和调试带来不少方便。一个是不在需要关注每个实体线程之间的同步问题，总线程在某一时刻只会在执行某一实体的逻辑而不会出现冲突，也就是不需要对公共资源添加管程以同步；还有就是调试时对于实体状态都更清楚。什么时候什么对象是什么状态都是确定的，在调试时也很容易就能看见其他对象的状态是否正确，方便调试；三是实体的时钟都能同步。在每个实体都是一个线程时，线程们调度顺序并不是确定的，在一轮时间内，可能有某一线程少被调度一次，也就是在1s的时间内，并非所有线程都能跑满十轮，可能有的多跑有的少跑。在本次作业的设计内，不会出现这种不公平的现象，不会出现多被调度或是少被调度的情况。

2.2 玩家控制指令

玩家的指令输入和游戏运行是没有关联的，所以同样需要设计好指令输入和游戏反馈的交互。首先指令的读入、存取应该由一个独立的线程控制的，但是指令的执行有两种形式。一是异步形式，什么时候读取，什么时候处理，二是同步形式，读取时先存储，在游戏线程运行到玩家实体的逻辑时，将其取出并处理。

那么这两者各有什么利弊呢？采取异步的形式，好处是反馈及时，按下按键就能马上为玩家处理，处理够快就不会有卡顿。也不需要设计指令的存取模块。坏处是这样需要设计好资源冲突的处理。由于要设计成联机的形式，故不同玩家的按键不可能一个个来，有可能两个指令同时发起，就会发生对地图资源的竞争，就有可能引起错误。故需要锁或者管程配合使用。若采取同步的形式，缺点是指令会有延迟，和游戏本体的时钟周期有关，同时需要统一好指令存储、读取的协议，做好统一。优点是指令的处理在游戏线程内，就不需要对资源进行限制。

在考虑到这次作业的结构，我决定采用同步的形式。同步带来的延迟感，可以通过缩短游戏时钟周期，加快时钟频率来达到。只要读取并处理指令的速度够快，就不会感到延迟带来的违和感。其次是小游戏本身的指令并不多，所以指令的储存、读取的协议很简单，只需要用键盘对应的字符代表对应按键被按下即可。在按下键盘时存储对应字符，之后读取指令时根据对应字符做出对应反应即可。只是这个指令的存取涉及到两个线程之间的同步，就是生产者、消费者的关系，需要加锁或者管程帮助同步管理。

那么游戏的本体逻辑大致如下图所示：

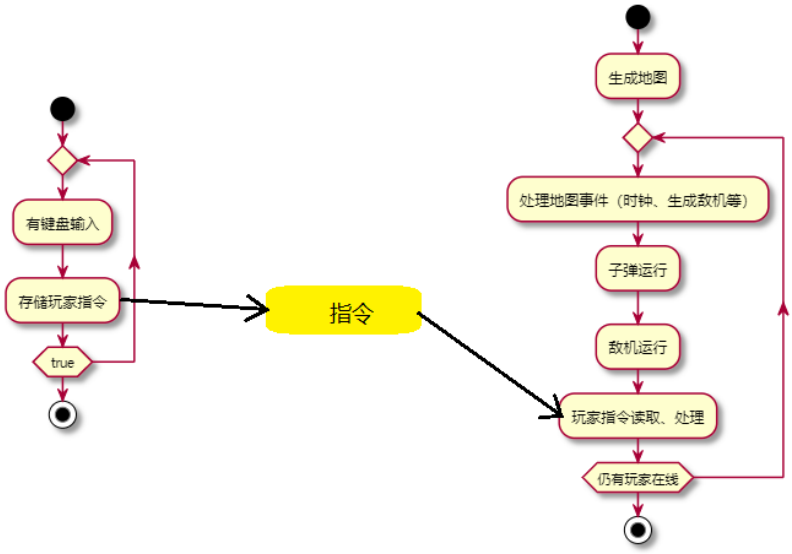

图 2 线程逻辑示意图

Figure 2 thread

2.3 网络

网络按要求需要做成selector和nio的形式，而且还要完成联机功能，所以需要考虑连接模式。首先，网络上的连接方式我只了解过peer to peer以及server/client模式。peer to peer既需要实现server功能也需要实现client功能，不同端之间也要完成信息的同步，实现起来较为麻烦。所以我选择server/client模式。

那么server/client模式要做成什么样的呢？我在暑假期间和小伙伴玩MineCraft的时候，鼓捣过mc的联机服务器，对mc的运作有部分了解，恰巧mc也是由Java编写的，故我就从mc的模式出发思考。mc采取的是C/S模式，客户端、服务端的形式。但是mc既有单机模式，也有多人联机，那么单机时就不用C/S模式了吗？并不是。在平时，单人模式的情况下，也有server端和client端，不过这俩都同时在自己的电脑上运行。当房主开启多人时，server端就会对外开放，此时其他玩家只需要启动client的功能，就能连入房主的server，一起游戏。在了解mc的mod工作时，我也了解到mc的server和client端都持有数据模型，也就是地图、人物、物品的数据一式多份，server和client就是通过交换数据模型实现同步，server利用数据模型实现游戏逻辑，而client通过数据模型完成画面的渲染。

因此，在设计本次作业时，首先采取C/S模式，区别是，MC有复杂的实现，可以把服务端和客户端打包成一个软件。自己开发软件，首先是要完成一个可运行的jar包，只允许一个进程（大概？）。如果把服务端和客户端的功能在两个线程内实现又会带来debug上的麻烦。故把server和client分开实现。server用selector和nio的方式连接和传输数据，client只需要和server连接即可。再然后是游戏内数据模型的同步。一开始我也想设计成server和client都持有数据模型，然后client只负责接受数据模型的更改，再把数据模型对应的画面表示出来。但是不同玩家接入服务端的时间并不相同，也就是在接入时还需要给客户端的数据模型进行初始化操作，先同步到和server端一致。也就是需要更加复杂的通信协议。而且我还想实现回放功能，更重要的是时间并没有分配好，所以我把client做成了一个单纯的播放器+按键事件转发器。server端只需要传达当前的画面（可以传向client，也可以写入回放文件），client只需要接受画面数据并且显示出来（可以来自server，也可以来自回放文件）。

那么，客户端和服务端就可以想下图所示进行工作：

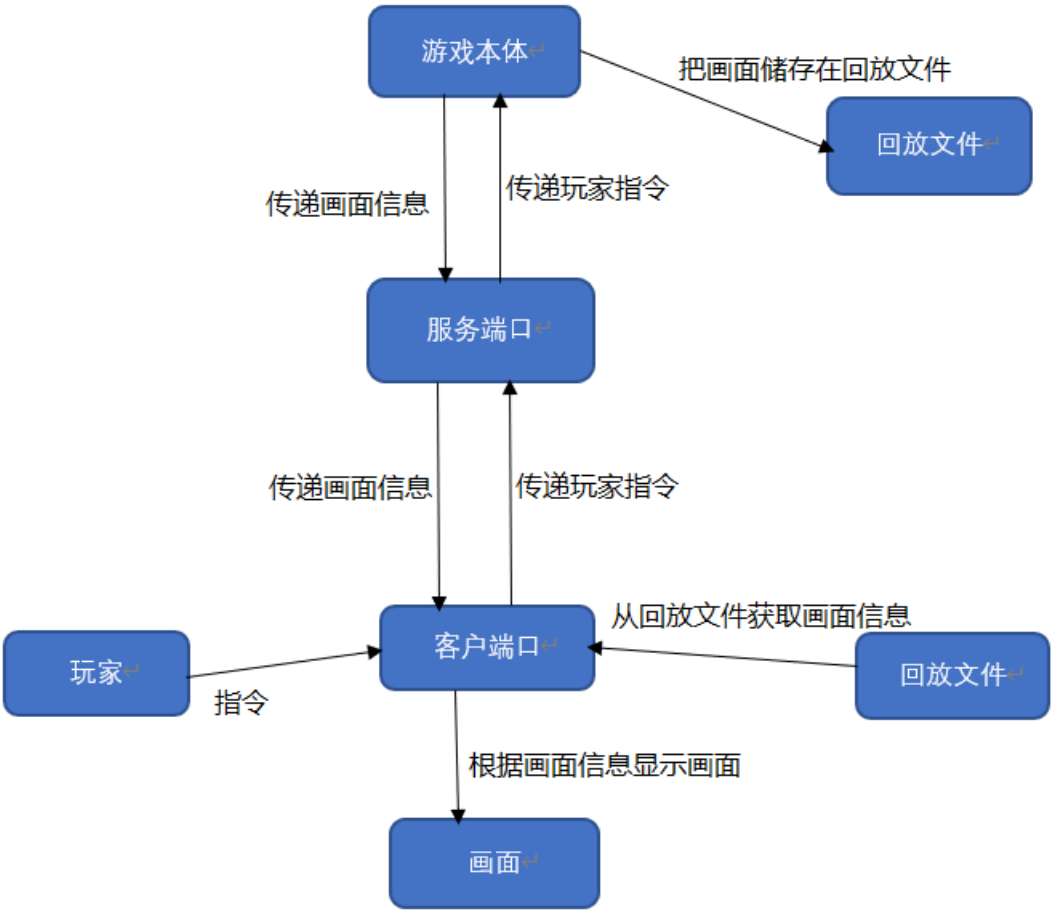

图 3　网络连接示意

Figure 3　net

3　技术问题

3.1　通信相关

通信设计正如上文所述，舍弃了通信效率而换来了设计、编码上的精简。如果再实现一次的话，我会按照我最初的想法，在客户端和服务端都做好数据模型。在客户端连上服务端时，服务端给客户端发送当前的数据模型。之后的通信就可以简化为（实体）（编号）（动作）（补充内容），以减少通信内容，提高效率。同时也可以将这个模式同步到回放功能，减少回放文件大小。而且还可以实现客户端记录对局回放，不再需要从服务端获取回放文件，更贴近真实游戏的回放系统。

3.2 并发控制、输入输出

关于并发控制还有网络io、文件io相关我在上文已描述完毕，不再赘述。

3.3 面向对象

关于面向对象，我主要和面向过程比较，因为向函数式编程等等我并没有真正上手实践过，没有发言权，因此之和面向过程对比。

我认为面向对象的思想给我开发带来的好处，最主要就是一点：可以清晰明确地划分模块。就讲游戏逻辑，如果要用面向过程地方法实现，那么我首先要规划好变量。比如放一个二维数组表示地图，一个数组表示玩家，一个数组表示子弹，一个队列表示玩家指令等等。再去完成基础部分的函数。比如子弹撞击函数，需要子弹变量和被撞击对象变量作为参数，在函数内完成被撞击对象的血量扣除，子弹删除等（其实应该叫方法而非函数）。在基础函数的基础上，实现更深一级的调用，比如敌机的攻击函数会用到子弹的生成函数。总之一步步聚集，把功能精简为最终函数run，在主线程循环调度run函数，完成游戏设计。这个思路理下去，会发觉到这个需要做到更复杂的规划，规划好什么函数需要完成，它们之间的调度关系等等。而如果用的是面向对象的方法，那么可以把变量、方法拆封到各个模块内，而且模块的划分可以依据现实，更加直接直观。同时什么方法由哪个模块实现也很明确。比如敌机的设计方法就应该在敌机的模块内，不会跑到子弹的模块内。每个模块分工明确复杂多样的内容拆分到各个模块，在编码前的规划阶段就比较轻松。

其次是面向对象减少了代码耦合度。面向过程中，代码的运作以函数之间互相调用实现，耦合程度很高，不全部完成就很难开始调试，调试时需要注意的内容也偏多。面向对象，代码耦合度较低，比如在完成Game模块和Player模块时就可以测试画面能否正常现实，按键能否正常反馈，也可以调试player和game的某些功能。同时，其他模块的调试完毕也可以帮助后来模块的调试，寻找bug不再大海捞针似的全局找。虽然这么说，在编写测试的时候还是感觉到自己代码的耦合度太高，模块之间耦合度较高而模块内部的方法耦合度尤为高，有不少方法是纯副作用或是返回值不痛不痒。在模块、方法设计上仍需改进。

4 工程问题，课程感言

暂无。

YingKang Su[1]

1. *NJU, NanJing* 210000*, China*
E-mail: 1093139885@qq.com