中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

时欣[1]

1. 南京大学, 南京210000

E-mail: 1364195528@qq.com

摘要 本文主要介绍我用java语言编写的吃豆子小游戏，包括我的游戏设计想法，遇到的问题和解决方案，以及我上完这学期的java课程后一些自己的感想。

关键词 开发目标, 设计理念, 技术问题, 课程感言

1 开发目标

1.1 游戏玩法

我写的游戏简单来说是一个吃豆子小游戏，具体玩法为：

1. 每次开始新游戏会随机生成一张地图，地图上有“墙”和“空地”，其中“墙”不能进入,“空地”可以移动进入；
2. 并且除了上述的两种，还会有“果子”，即玩家需要走到有“果子”的地块才能拿分，每吃到一个“果子”拿一分；
3. 地图上还会产生随机数量的妖怪在随机位置，妖怪的目标也是吃到“果子”，所有妖怪吃到的“果子”总数是妖怪的总得分；
4. 如果是多玩家对战，则所有玩家吃到的“果子”总数为其共同得分；
5. 当地图上的“果子”被全部吃完以后，如果玩家的得分更高则胜利，否则失败；
6. 如果玩家和任意一个妖怪走到了同一个地方则失败。

1.2 游戏灵感

我的游戏灵感来自于小时候玩的一款4399小游戏叫“狂吃方便面”，即：

1. 一个学生（玩家）在上课时间偷溜到操场上，操场为一个固定地图，有固定的道路可以走；
2. 地图上会不定时产生一些“方便面”，学生走到有“方便面”的地方则默认吃到它，记为得到一分；

3. 每过一段时间会有一个老师从固定点出来巡查，在场最大的老师数量为4；
4. 如果学生无路可走和老师走到了同一个地方则游戏失败；
5. 偶尔会产生一些特殊“方便面”，吃下后学生可以变得更强壮，遇到老师时可以将其撞回教学楼而不会失败，一定时间后该能力失效，学生变回普通学生；
6. 最终胜利条件是吃满100 袋“方便面”。

我在这款游戏的基础上增加了随机地图，随机的NPC生成位置和数量以及多玩家共同游戏，并且修改了下游戏规则，即：在游戏开始就给出固定数量的得分点，而NPC也会抢占得分点，最终胜利的判别条件是双方得分的大小关系。

2 设计理念

2.1 代码总体设计

我的总体设计理念是模块化、低耦合、高内聚。总体设计中类的相互关系用uml图表示如下

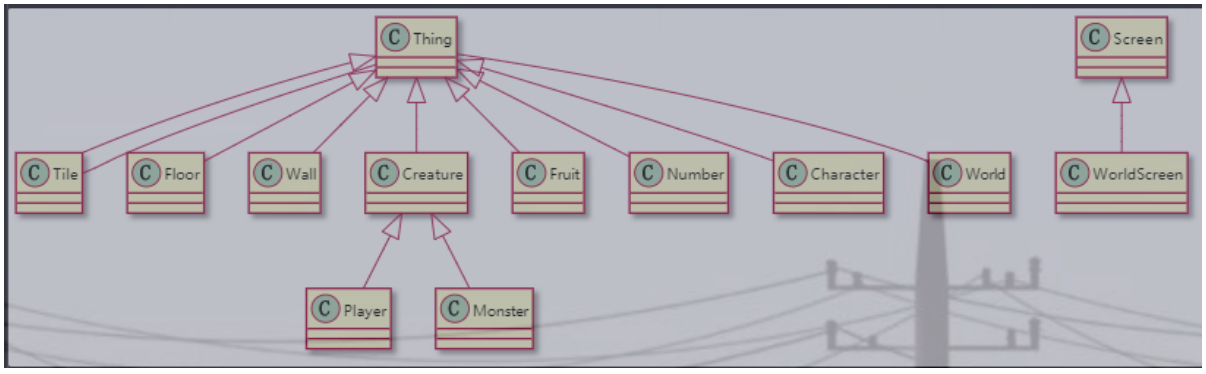

图 1 类间相互关系
Figure 1 Caption

而其中主要的各个类及其含义分别如下：

1. Thing: 所有含义为地图上元素的类的基类，有其共同属性为所在的地块、所在的地图（world）、颜色。其含义为所在地块上的元素内容。
2. Tile: 继承自Thing，含义为地块，有Thing类型的成员（表示当前地块上所放置的东西）。
3. Number: 继承自Thing，含义为游戏场景旁边的得分显示。
4. Wall: 继承自Thing，含义即上述的“墙”。
5. Creature: 继承自Thing，实现Runnable接口，含义是会在地图上进行移动的生物，包括玩家和妖怪。
6. Floor: 继承自Thing，含义为上述的“空地”。

7. Fruit: 继承自Thing，含义为上述的“果子”。

8. Player: 继承自Creature，含义为玩家，关键属性有id，用来区分多玩家对战中的不同玩家。

9. Monster: 继承自Creature，含义为妖怪。

10. World: 为关键的类，包含着所有地图上的元素信息以及游戏状态。含有一个Move类对象，负责对地图上的元素进行移动。

11. Move: 它的含义是一个动作即移动，成员有World类对象，因为它需要知道它要移动哪个world上的元素。

12. Save: 负责游戏进度的保存，成员有World类对象，因为它需要知道它要保存哪个world上的信息。

13. Continue: 负责游戏进度的恢复，成员有World类对象，因为它需要知道它要保存哪个world上的信息。

14. WorldScreen: 为关键的类，负责游戏关键的逻辑，成员有玩家线程数组、妖怪线程数组、Save类对象、Continue类对象和World类对象。

15. TestClient: 为了图形化显示，继承了JFrame；为了接收用户按键信息，实现了KeyListener接口。是多玩家对战中玩家逻辑的实现，包含了接收键盘信息的逻辑，与服务端通信的逻辑，以及为玩家显示游戏图形化界面的逻辑。

16. Main: 也就是游戏最开始的入口处，控制整个游戏的全局逻辑（游戏进展和网络通信）。有WorldScreen类的对象作为其成员。有EchoServer作为其内部类，负责和客户端进行网络通信，接收用户数据并且递交给WorldScreen处理。

在一个World对象被创建时，所有的游戏状态（分值、地图样式、“果子”的位置等）会被初始化。而当一个WorldScreen对象被创建时，所有的初始游戏状态就位，并且妖怪线程和玩家线程被启动。玩家通过控制键盘来操控自己的角色，所按下的按键信息被另外单独的函数接收，通过WorldScreen提供的接口传给它进行处理。

2.2 如此设计的好处

如此设计的好处即尽可能地吻合面向对象的“模块化、低耦合、高内聚”的编程风格。各个类之间的实现不会纠缠不清，影响效率或效果。将Save和Continue功能作为单独的类摘出来实现，不会影响前面写的类，而只需要在总逻辑中包含这个类的实例化对象，即可完成这个功能。

TestClient类只负责接收用户按键信息和给用户显示图形化游戏界面，并不涉及具体的游戏逻辑处理，而将其交予服务端处理。这样的好处是低冗余，并且减轻客户端处理压力。因为如果让客户端处理游戏逻辑，那么它需要处理好自己玩家的信息再给Server，还需要各个客户端信息统一一致，则可能有些非法操作因为彼此暂时看不到的原因而出现，逻辑乱成了一团。

Move类单独实现，并且将具体负责移动地图元素的那个函数给予Synchronized的限制。这样就防止了两个生物同时探查一个地块,发现是可以进入的，由此导致同时进入的错误。除此以外，实现了代码的高复用性，即Move是Player和Monster一致的行为，它们只需要用封装好的Move类中的移动接口进行移动即可。否则还需要在每个类中单独实现，既麻烦又难以解决上述可能的错误。

3 技术问题

3.1 通信效率

通信效率我采用了NIO即非阻塞的方式，这样就不会出现因为一个信息未处理完而导致另外的等待卡死的情况。

而对于Server和TestClient端之间到底互传什么信息，一开始我并没有想的这么清楚。我在想客户端接收到按键信息后该如何操作将各个客户端的动作统一起来不发生错乱，这就导致了整个思路的混乱。随后考虑到TestClient的整体含义，它只是接收按键信息并且给用户展示游戏界面的，因此实际上我也只需要让它具备这两个行为即可，其它的就交给Server端处理就好。

所以就像上面“如此设计的好处”中所说的那样，采取了这样的设计以后，我所需要传递的所有信息只有：Server端给TestClient端发送当前表示地图情况的数组以及几个表示游戏状态的int型数据；TestClient向Server端注册自己、或向Server端发送玩家输入的按键信息、或向Server 端索要当前游戏信息。以上需要传递的数据是很少的，因此通信效率较高。

3.2 并发控制

其实我需要并发控制的只有两个生物不能移动到同一个地块而已。这样考虑两种加锁方式：

第一种是对地块加锁，即一个地块只能被占领一次；

第二种是对移动这个动作加锁，即同一时刻只能有一个生物在移动。

考虑以上两种方式显然后者更为简单，只需要让所有生物用同一个加锁的移动类对象，在每次移动的时候判断当前地块是否有东西即可。而如果采用第一种，则需要对每一个地块加锁，地图越大，加锁的就越多，极难实现。因此我采取的是第二种，方便简洁。

3.3 输入输出

输入输出是一个简单的文件读写问题。在此游戏中主要应用于进度的保存与读取。在本游戏中每次按“S”键会将当前游戏进度保存，并且仅有一个存档位，永远保存最新的存档。而关闭游戏后再启动可以选择“N”（开始新游戏）或者“C”（继续上一次存档处）。因为这个动作与其它类并没有什么关系，所以也采取把其作为一个单独类实现的方法。只要注意一下，如果需要读取存档，必须判断是否存在存档文件。如果以前从未存档，则按“C”默认开始新游戏。这是一个边界条件的考虑问题。

4 课程感言

本学期是我第一次使用java语言编程，这样生动有趣的课堂不同于以往的编程课，令我收获了很多。但编程仍需实践。实际上课上仿佛“听懂”以后，真正写代码仍然会遇到很多问题。其中最大的难题不在于java的语法，而是各个模块间究竟如何划分、以及如何灵活使用java这样一种面向对象语言的特性，才能实现良好的面向对象效果。

根据经验，我发现，应该更多地使用“面向对象”思路去分解问题，即

1. 整个逻辑中有哪些主体，各自有哪些行为；
2. 是否有一些共同的行为可以进行提炼，使得逻辑更清晰、代码的复用性更高；
3. 有一些行为究竟与某个类关系大不大，是否需要依附于这个类存在；

4. 并发控制的根本矛盾点存在于哪一个主体（包括表示某个具体事物的类或表示某个具体行为的类），以怎样的粒度控制；

5. 网络通信时，类之间真正需要传递的信息是什么－－－－哪些信息可以由这个主体自己处理，哪些可以由另外的主体处理会更好等等。

这些问题都需要在编程之前想清楚再写，否则写着写着发现不对，早已耗费了大量精力而且打乱了思路。

本学期前面的任务较松，难度小。而后面稍显紧急，并且相比较前面有一定难度。拥有马甲线的老曹同学或许可以考虑之后把整体进度再往前挪一挪。

虽然编程和debug的过程中多有痛苦，但写完之后还是有点成就感的。相比较上学期用C++进行面向对象编程，感觉通过本学期的学习对面向对象有了更深层次的了解，也接触到了以前只是听说过的并发控制和网络编程，学习到了java一些特有的语言机制，明白了看文档的重要性。以后也会对java语言进行更多的学习。

致谢 完结撒花！感谢老师和帮助过我的同学们！