中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

张久霖[1]

1. 南京大学, 南京, 210000

E-mail: 191220159@smail.nju.edu.cn

收稿日期: 2021-12-30; 接受日期: 2021-12-30; 网络出版日期: 2021-12-30

摘要 通过一学期在高级 java 程序设计课上的学习，用 Java 实现了坦克大战。

关键词 Java, 坦克大战

引用格式: 张久霖. Developing a Java Game from Scratch. 中国科学: 信息科学, 在审文章
zhang jiulin. Title for citation (in Chinese). Sci Sin Inform, for review

1 开发目标

我所写的 java 游戏的游戏原型和灵感来源是经典坦克大战 (如图1, 图2)。

图 1 坦克大战游戏界面

图 2 坦克大战开始界面

但糟糕的是我自己有点眼高手低，最终还原实现的内容非常有限。我自己实现的 java 小游戏，游戏的胜利条件是消灭除自己以外的所有玩家，同时保证自己不被玩家和怪物消灭。目前游戏只实现了很基础的内容：在游戏行为上，实现了玩家的移动和射击子弹；在游戏性上，实现了特殊道具红桃和太阳：吃掉红桃可以增加生命值，吃掉太阳可以获得五次射击的强化子弹，强化子弹可以摧毁墙壁，且会造成更大伤害，道具会被子弹销毁；实现了怪物，怪物会自动寻找并接近玩家，到达玩家身边时，玩家会持续扣除生命值到 1 点（不会死亡），但目前的怪物的行为逻辑实现的还不够好，后续还会修改；此外，预计还会实现视野范围（这部分代码在单机的时候写好了但还没来得及写入联机中）以及地图加载（同原因）。

2 设计理念

代码的总体设计是将基于 Selector 的 NIO 网络通信与游戏内容的基本实现分割开，并将多人游戏的运算逻辑全部交由 Server 端进行，Client 只负责读取数据以及显示图像。这样设计的好处，一方面可以将单机联机的部分分割开，便于各自实现修改；另一方面，基于 Selector 的 NIO 通信可以很好的解决多用户之间传信息使得线程低效率的问题，只用一个 Selector 线程监听多个信息流量很小的 Channel，提高了总体效率。

最后游戏效果如图3所示。

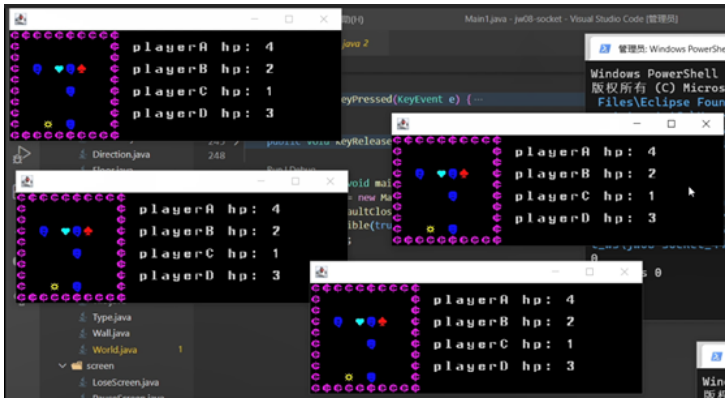

图 3 坦克大战联机对战展示

3 技术问题

代码的技术问题可以分为以下几个方面来看：

3.1 输入输出

首先是输入输出方面，输入方面实现的很简单，只对键盘的按键操作进行相应响应和设计；输出方面由于要实现图形化界面，就需要 Java Swing 的一些知识来实现界面设计。其实这里我在第五次作业用老师提供的 AsciiPanel 面板（图4) 的时候就像自己实现图形面板来写入自己的图片，但后来由于拖延症一直没有做成，直到 12 月的下旬才开始动笔打算重新写第五次作业，并把自定义图片也实现，但忙活了几天，就只是重新调整了之前的一些结构，看了 Java Swing 的教程，但没能写出来。希望自己假期的时候可以弥补上这里的遗憾。

3.2 网络通信

然后是网络通信方面，课程作业的要求就是实现基于 Selector 的 NIO 网络通信（图5，图6），这部分在课堂上没有听明白，实现的时候遇到了很大阻力。后来在网上找到了一些基于 Selec tor 的 NIO 网络通信聊天室，在大概理解之后将其改装成了现在的游戏网络通信。游戏中需要传输的数据是实时的地图信息和玩家状态，将这些数据转换为 string 后进行传输，其实与网络聊天室的差别不

图 4 AsciiPanel 面板

大，所以就这么实现了，Server 负责不断地向注册到 Sel ector 的 Channel 广播地图信息和玩家状态（全局信息），并只用一个 selector 线程监听所有 Channel 并对响应他们中自己感兴趣的状态；Client 则在用户进行键盘操作后将用户名和对应操作传回给 Server，这里 NIO 的优势就体现出来了，传统的阻塞式 IO 需要使用多个线程来监听这些操作，而且当线程调用 read() 或 write() 时，该线程被阻塞，直到有一些数据被读取，或者数据完全写入，在此期间线程不会做其他事而一直被占用，而 NIO 只需要这一个线程就可以处理，避免了线程上下文切换带来的开销，并且由于 NIO 面向缓冲区的特点，有线程请求写入一些数据到通道时，不需要等待它完全写去，该线程可以同时做别的事情。虽然这种模式的代价是解析数据会更加复杂，但我的通信数据数据量很小，这带来的代价没有很大。

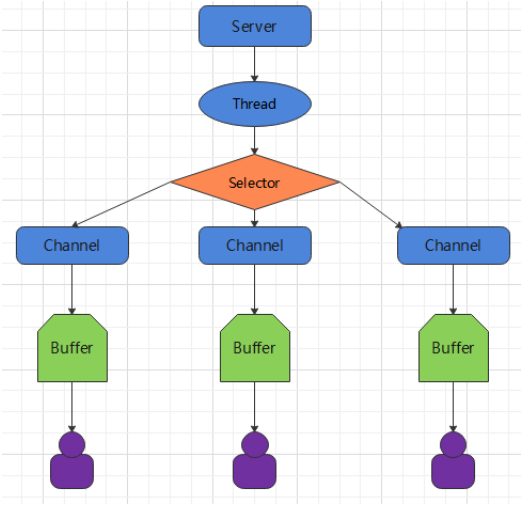

图 5 基于 Selector 的 NIO 网络通信结构图

3.3 并发控制

至于并发控制方面，实践上在实现生物类移动的时候，应该通过进程间通信来实现并发不会抢占的，但由于对并发编程不是很熟悉，就没有实现需要实现的加锁等同步，如果可以的话希望寒假

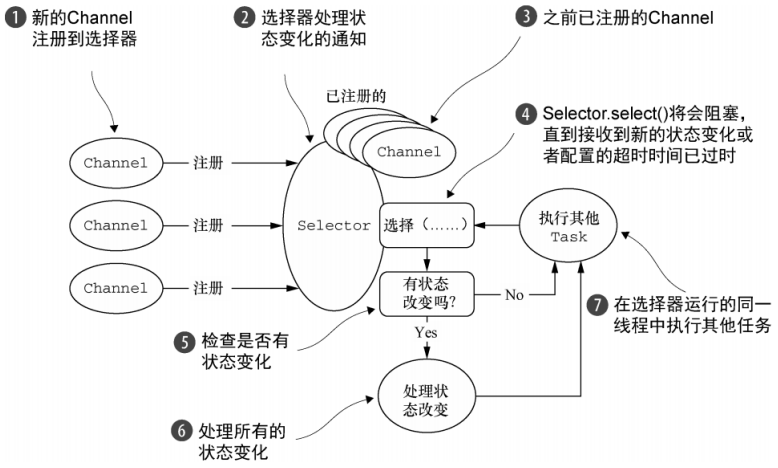

图 6 基于 Selector 的 NIO 网络通信流程图

的时候进行补充。

相关进程如图7所示。

```
private class UpdateMap implements Runnable{…

private class MonsPlant implements Runnable{
    @Override
    public void run() {
        while(true){
            try{
                Thread.sleep(10000);
            }catch(Exception e){
                e.printStackTrace();
            }
            Random r = new Random();
            int posX = r.nextInt(World.WIDTH-2);
            int posY = r.nextInt(World.HEIGHT-2);
            if(world.checkType(posX+2, posY+2) == Type.FLOOR )
            {
                monster = new Monster(world);
                world.put(monster, posX+2, posY+2);
                new Thread(monster).start();
            }
        }
    }
}

private class StarPlant implements Runnable{…

private class HeartPlant implements Runnable{…
```

图 7 坦克大战进程

这里还想补充的一点是，进程的实现；因为其实是在学 Java 之前对进程同步异步的实现没有任何实际操作的经验（还没学高程），实现起来的时候很生疏，缺少总体上的把握和设计，基本上觉得哪里需要线程（比如刷新界面，持续发送信息等）就添加上一个，而且也没用到 Lamda 表达式，就导致代码结构有些地方不大合理，当然其实问题的根本原因还是出在自己开始太慢太拖延。

3.4 面向对象

最后谈一谈面向对象带来的好处。面向对象与面向过程相比，最重要的一点是，面向对象可以更好地对物理世界进行抽象，这对于游戏世界也是同理的。

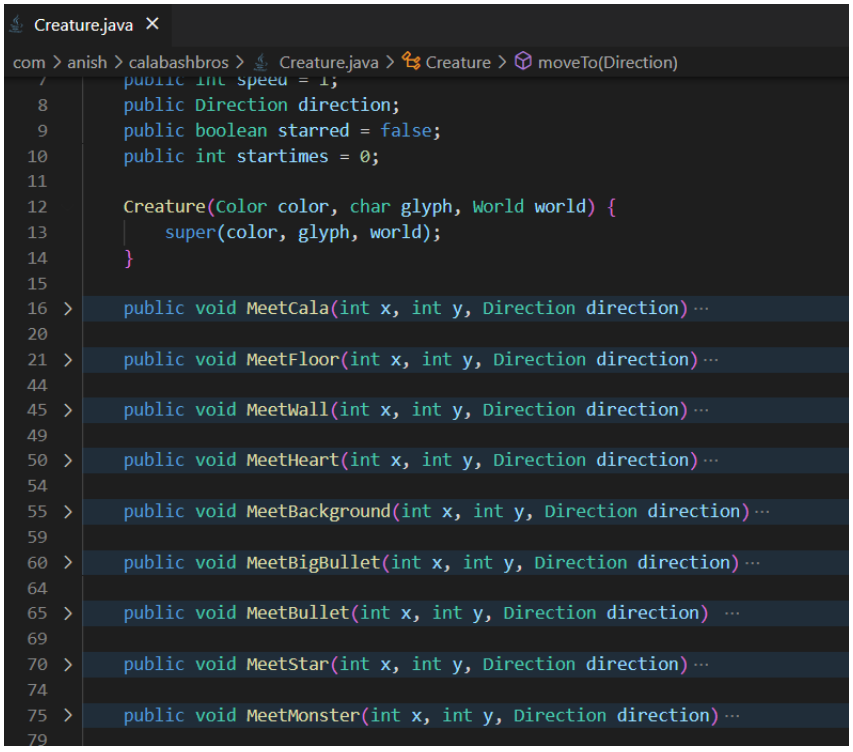

图 8 Creature 的移动 1

游戏中所有对象都是继承自原本框架代码中的 Thing 类，Tile 对应的是每个地块，不会变动的游戏对象，如 Wall、Floor、Background 就直接继承自 Thing，Creature 对应的是各种活动体，当然也是 Thing 的子类，但还要由此衍生出更多类型，他们具有的相同的属性和方法就可以先在 Creature 进行定义和实现，而在具体去实现每个类的候去添加新的方法或者重写原有的方法，举例来说（见图8，图9），在实现 Creature 的移动时，Creature 会遇到各种各样的对象，但不同 Creature 之间相遇（碰撞）的实现逻辑并不相同，这里我就广泛地运用了方法的 Override，这使得代码的可读性和易拓展性大大提高；此外，存档的实现主要依靠的就是 WorldScreen 的构造器重载，根据不同的参数类型加载不同的地图进度。

4 工程问题

糟糕的是，这个小游戏项目除了有用到 Maven（图10）来管理项目（但也是在半途才加入进来的）以外，并没有用到任何高级设计方法或者工程方法来提高开发效率和代码质量。在用 Maven 管理项目的时候，还按照实验要求进行了单元测试，测试了部分方法，检验了其正确性，在力所能及的情况下保证了代码的基本质量。

Creature.java

com > anish > calabashbros > Creature.java > Creature > moveTo(Direction)

```
80      public void moveTo(Direction direction) {
81          this.direction = direction;
82          int x = this.getX();
83          int y = this.getY();
84          switch(direction) {
85              case UP:
86                  if(this.world.checkType(x, y - speed) == Type.FLOOR)
87                  {
88                      MeetFloor(x,y,Direction.UP);
89                  }
90                  else if(this.world.checkType(x, y - speed) == Type.HEART)
91                  {
92                      MeetHeart(x, y, direction);
93                  }
94                  else if(this.world.checkType(x, y - speed) == Type.MONSTER)
95                  {
96                      MeetMonster(x, y, direction);
97                  }
98                  else if(this.world.checkType(x, y - speed) == Type.STAR)
99                  {
100                     MeetStar(x, y, direction);
101                 }
102                 else if(this.world.checkType(x, y - speed) == Type.WALL)
103                 {
104                     MeetWall(x, y, Direction.UP);
105                 }
106                 else if(this.world.checkType(x, y - speed) == Type.BACKGROUND)
107                 {
108                     MeetBackground(x, y, direction);
109                 }
110                 else if(this.world.checkType(x, y - speed) == Type.BULLET)
111                 {
112                     MeetBullet(x, y, direction);
113                 }
114                 else if(this.world.checkType(x, y - speed) == Type.BIGBULLET)
115                 {
116                     MeetBigBullet(x, y, direction);
```

图 9 Creature 的移动 2

5 课程感言

回想起来，最开始的时候，我仅仅是抱着多认识一门语言的念头来选修这门课，但没想到课程最后竟然写出了这样一个小游戏（好吧虽然我的这个真的太简陋了 qwq），但又一次见识到了很多大佬的非常 AMAZING 的课设作业，真的非常震撼和佩服，当然更多的是自惭形秽。这门课程的课程作业一直都很有趣，从最开始的图片可视化排序、用隐写术图读类，再到后面的迷宫小游戏和 Rougelike 游戏。这么优秀的课竟然不是必修课我不理解!! 但由于自己的拖延和懒惰，本来分担到五周的 jw06-jw08 竟硬是被我压缩到了一周多，而且还非常不自量力地想重写前面的代码。结果就是原本想优化的地方没优化多少，新完成的需要优化调整的地方又没能完成。在交过作业逃离期末之后应该还会再力所能及的补充一些逻辑和实现细节，希望多少可以弥补些许遗憾。

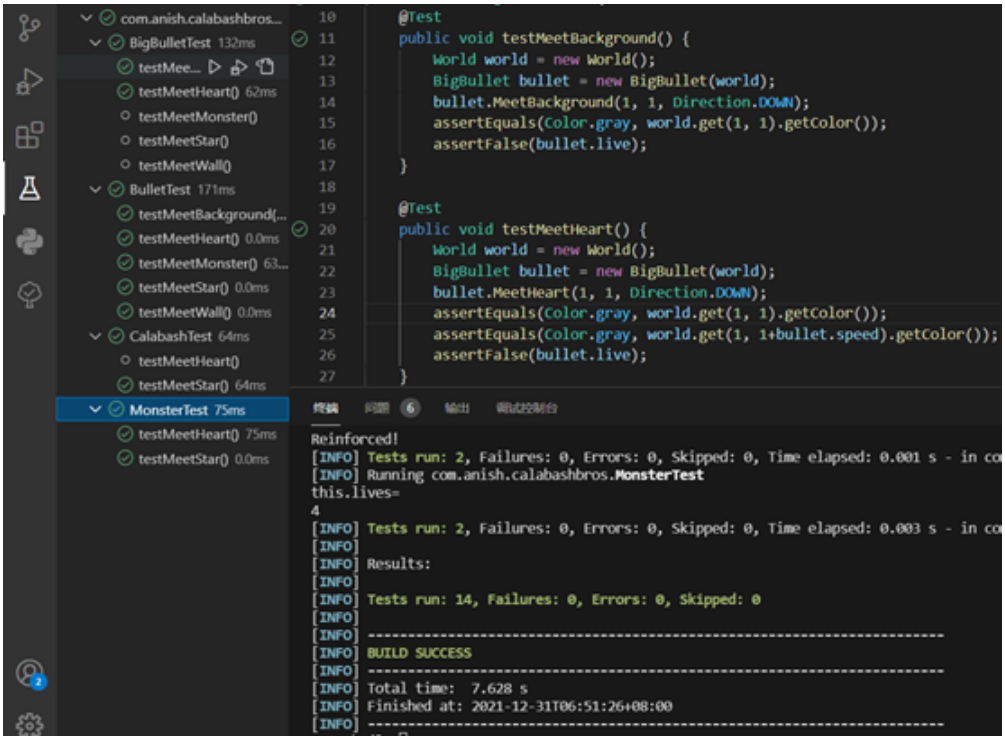
com.anish.calabashbros...
BigBulletTest 132ms
testMee...
testMeetHeart() 62ms
testMeetMonster()
testMeetStar()
testMeetWall()
BulletTest 171ms
testMeetBackground(...
testMeetHeart() 0.0ms
testMeetMonster() 63...
testMeetStar() 0.0ms
testMeetWall() 0.0ms
CalabashTest 64ms
testMeetHeart()
testMeetStar() 64ms
MonsterTest 75ms
testMeetHeart() 75ms
testMeetStar() 0.0ms
@Test
public void testMeetBackground() {
World world = new World();
BigBullet bullet = new BigBullet(world);
bullet.MeetBackground(1, 1, Direction.DOWN);
assertEquals(Color.gray, world.get(1, 1).getColor());
assertFalse(bullet.live);
}
@Test
public void testMeetHeart() {
World world = new World();
BigBullet bullet = new BigBullet(world);
bullet.MeetHeart(1, 1, Direction.DOWN);
assertEquals(Color.gray, world.get(1, 1).getColor());
assertEquals(Color.gray, world.get(1, 1+bullet.speed).getColor());
assertFalse(bullet.live);
}
Reinforced!
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s - in co
[INFO] Running com.anish.calabashbros.MonsterTest
this.lives=
4
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in co
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 14, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 7.628 s
[INFO] Finished at: 2021-12-31T06:51:26+08:00
[INFO] --

图 10 Maven

Title

zhang jiulin[1]

1. *NanJing University, NanJing* 210000, *China*
E-mail: 191220174@smail.nju.edu.cn