中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

汪皓晨[1]

1. 南京大学电子科学与工程学院, 南京 210023

E-mail: 191180122@smail.nju.edu.cn

摘要 Java 是一门面向对象编程语言，不仅吸收了 C++ 语言的各种优点，还摒弃了 C++ 里难以理解的多继承、指针等概念。本文主要针对一学期的高级 Java 程序设计做一个总结，包括对于课程作业的分析、对于面向对象编程方法的思考和个人对于本门课程的一些感悟

关键词 Java, 面向对象，程序设计，多线程

1 开发目标

1.1 游戏概述

从 jwo5-jw08，我完成的是一个像素风格的射击对战游戏，主要阵营为玩家和怪物。玩家和怪物均有属于自己的外观和颜色，且均有技能 “射击” 和 “移动”。在移动过后，会想所移动的方向发射一个子弹，子弹的颜色与怪物/玩家保持一致。子弹会一直顺着当前方向进行移动，直到碰撞其他生物和墙。

游戏地图是随机生成的，其中包含 WALL 和 FLOOR，所有生物均有可以破坏墙壁，破坏条件是生物和墙相距 1 个单位，此时生物向着墙的方向会首先破坏墙壁，但不发生移动。或者，所有生物发出的子弹的下一个位移方向为墙，此时会破坏墙同时销毁子弹。

本游戏创新之处在于不是所有的生物都可以互相伤害，首先每个怪物的颜色是不一致的，无法互相造成伤害。玩家起始颜色是白色，无法对其他怪物造成伤害，但是玩家有 “变色” 技能，通过按键 “1” “2” “3” “4” “5” 可以变换颜色，分别对应 “红色” “黄色” “绿色” “蓝色” “品红色”。只有双方颜色相同才能通过子弹造成伤害。但是当双方距离为 1 时，如果移动目标方向会相遇则会直接造成伤害，不管颜色，均可互相伤害，这也是鼓励头铁玩家直接冲，因为发射子弹想要射中，其实也不是很容易。

1.2 灵感来源

游戏的灵感来源是 “开心消消乐”，这个游戏在 35-50 年龄段的人群中非常流行，从 6 年前就对这个游戏印象非常深刻，我的身边也有一位痴迷且非常有水平的玩家。对应颜色才能消除在很多游戏中都可以看见，比如 “连连看”、“消消乐”、“俄罗斯方块” 等，在我的作业里，我将这一玩法加入射击对战中，探索出另一种对战机制。

类	作用
RefreshGUI	用来刷新屏幕的 Runable 接口
Bullet	实现生物发射的子弹，实现了可以一直移动的 Runable 接口
BulletFactory	子弹的制造工厂，用来创造子弹实例
Creature	实现生物类，包括了生物的特性和基本技能
CreatureAI	实现生物的基本操作
CreatureFactory	生物的制造工厂，用来创造生物实例
Monster	Creature 的派生类，主要实现怪物移动、攻击的 Runable 接口
MonsterAI	CreatureAI 的派生类，实现怪物的基本操作
PlayerAI	CreatureAI 的派生类，实现玩家的基本操作
Tile	enum 类，实现 World 中每个位置的地图实例
World	构造出世界，包括地图上每个坐标的环境，保存创造出来的生物、子弹实例
WorldBuilder	实现随机生成地图
ApplicationMain	实现应用程序
Screen 及其派生类	提供了不同界面的显示内容，响应按键信息

表 1 主要实现类

2 设计理念

2.1 面向对象的程序设计

游戏设计主要采用面向对象的程序设计理念，主要包括类及其作用见表 1。

2.2 多线程编程

在游戏中，每一个怪物和发生出来的子弹，都是一个独立的线程，通过实现 Runable 接口，使得每个对象都是可以独立于主程序运行的线程。同时使用线程池来管理线程。由于 Java 回收机制，当线程不运行后会自动杀死，所以不需要手动操作。

一个进程中可以并发多个线程，每条线程并行执行不同的任务。

2.3 综述

在本工程中沿用了 jw04 的大框架，每个 Creature 拥有其属性，例如血量、攻击力、颜色、形状、坐标、所属的大世界等，为了获取发射子弹的方向，还另外设置一个 movedirection 用来保存移动的方向。Creature 还拥有移动、攻击（修改血量）、发送消息等方法。其中，部分方法放在 CreatureAI 类中，每个 Creature 都有一个专属类。

由于 Monster 类和 Creature 类唯主要差别在于作为一个独立的线程运行，所以直接继承 Creature 类并实现 Runable 接口即可，这也是 Java 非常方便的一点。

Bullet 对象是由 Creature 对象产生发射的子弹，其属性和方法和 Creature 类似，都有自身的形状、颜色、坐标等，方法也有属于自己的攻击、移动方法。

由于采用面向对象设计的方法，各个对象之间的关系是非常紧密的，每一个 Creature 及其派生类一定是有一个与之对应的 AI，且可以指明这些 Creature 属于哪一个大世界，这也很容易理解，就像我们每个人一定有一个属性就是存在在地球这个大环境中，这也方便各个对象之间交互，不仅可以互相联系，还可以对和周围环境交互。每个子弹是由一个 BulletFactory 创造的，同一个工厂创造的子弹的形状和颜色一定是一样的，这也要求，每个 Creature 一定有一个属于自己的 BulletFactory，同样创造出来的子弹也一定要明确他属于哪个世界、是由谁所创造，在每个子弹中一定有 World 和 Creature 这两个属性。

Tile 是一个枚举类，其中包括 WALL、FLOOR 和 BOUNDS。每一个 World 类就相当于一个大环境，其中利用一个二维数组保存 Tile，用来表示每一个坐标是墙还是地，如果要对环境进行操作，直接修改 World 中的 tiles[][] 即可，而面向对象设计中紧密的联系使得无论是 Bullet 还是 Creature 及其衍生类都可以操作环境。在 World 中还有两个 ArrayList 用来保存当前世界的所有 Creature 和 Bullet，这也意味着，最后在图形化显示的时候，关键的数据只有 tiles[][] 中每个位置对应的 tile 是枚举类中的哪一个、保存 Creature 及其派生类对象的 ArrayList 和保存生成子弹的 ArrayList。

无论是 Creature 派生类还是 Bullet 都涉及 move 和 attack 方法，在移动方法中，首先需要判断目标位置是 creature 及其派生类对象还是 tile 对象，这就需要我们能够根据目标位置坐标在 world 中保存 Creature 的 ArrayList 中检索，这时就可以根据需要补充方法，如果目标位置没有被 creature，那么就可以使用移动算法，否则就要使用攻击算法。子弹的攻击算法需要比较子弹的颜色和 other(creature) 的颜色，如果一致则造成伤害，如果不一样那么停止子弹线程，关于停止线程，利用 flag 标志控制 while 循环，同时将该子弹从 world 中保存 bullet 的 ArrayList 中删除。当使用移动算法准备移动时还要判断目标位置 tile 是 WALL 还是 FLOOR，如果是 WALL，就需要执行 dig 算法，否则直接移动即可。能实现这一系列复杂的判断都要归功于面向对象的程序设计，高内聚，低耦合，可以很方便操作各个对象。

在创造出来 creature 及其派生类和 bullet 时需要将其放到地图中，creature 及其派生类比较容易，只要放置的地方是 FLOOR 即可。而 bullet 的放置带有一定的指定性，需要想移动一样判断距离为 1 的地方有无 creature 及其派生类或者是否为 WALL。

Screen 及其派生类实现了一系列的屏幕，可以把 ApplicationMain 理解成应用程序的框架，而框架里显示的内容则是 Screen 及其派生类。Screen 作为一个接口，其他 Screen 实现其接口，在实现接口的类中需要完成对 world，creature 等的初始化、界面的刷新和按键的监听。同时实现了读取制定 txt 文件生成地图。

3 技术问题

3.1 并发控制

在本游戏中，每一个 Creature 和 Bullet 都可以独立于主线程自由移动，所以需要采用多线程编程，实现 Runable 接口，然后设置一个循环，可以独立处理自己的消息。

为了防止很多线程对同一块内存区域修改而造成混乱，涉及线程的方法均加入了同步锁 synchronized，使用 public synchronized void method() 锁定一个对象，当 method() 方法被调用时，调用线程首先必须获得当前对象所，若当前对象锁被其他线程持有，这调用线程会等待，方法法结束后，对象锁会被释放。

现在想想，只要想让一段代码单独执行，均可以使用线程，在本游戏中还使用了线程池来管理创造的怪物。

```
@Override
    public void run() {
        // TODO Auto-generated method stub
        while(flag) {
            if(this.hp()<1)
            {
                world.remove(this);
                flag=false;
                break;
            }
            direction = rand.nextInt(4)+1;
            switch(direction) {
                case 1:
                    if(this.x()>0){
                        this.setDirection(1);
                        new Thread(this.bulletFactory.newBullet((char) 250,
                    this.color())).start();
                        this.moveBy(-1, 0);
                    }
                    break;
                    //......
            }
            try {
                TimeUnit.MILLISECONDS.sleep(400);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
        }
    }
```

3.2 输入输出

本游戏中实现了从文件中加载和保存地图，每次随机生成地图后，都会保存文件，按键 F 用来监听从文件中读取地图指令。由于保存文件读取为 char 型，所以需要根据 ascii 码来判断，部分代码如下:

```
public WorldBuilder worldInput() throws IOException
{

    try {
        inputStream = new FileReader("worldinput.txt");
        int c=0;
        for (int i=0;i<width;i++)
        {
            for(int j=0;j<height;j++)
        {
            c = inputStream.read();
            while(c!=48 && c!=49)
            {
                c = inputStream.read();
            }

            if(c==48)
                tiles[i][j]=Tile.FLOOR;
            else if(c==49)
                tiles[i][j]=Tile.WALL;
        }
        }

        } finally {
        if (inputStream != null) {
        inputStream.close();
        }
        }
```

```
        return this;
}
```

3.3 面向对象优点

1. 可维护性强: 采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是在局部模块，所以维护起来是很方便和成本较低。
2. 抽象性强: 万物皆对象，根据设计的需要对现实世界的事物进行抽象，产生类。利用现实世界的思维解决问题。例如在本游戏中，构造一个大世界，大世界中有 tiles 环境、排列所有生物的队伍等。
3. 拓展性强: 高内聚，低耦合，封装、多态、继承等特点。对对象的操作不需要知道内部的实现方法，只需要进行适时调用即可。

4 工程问题

4.1 工厂模式

在 Java 中，万物皆对象，有很多多想需要创建如果创建一个对象就 new 该对象，会对该对象耦合严重，如果需要更换该对象，就要把用到该对象的地方全都更改，使用工厂模式，在更换对象的时候只需要在生产对象的工厂中做更改就可以了，降低了对象耦合性。

```
public class BulletFactory { //每一个creature拥有一个子弹生成的工厂

    private World world;
    private Creature owner;

    public BulletFactory(World world, Creature owner)
    {
        this.world = world;
        this.owner = owner;
    }

    public synchronized Bullet newBullet(char style,Color color)
    {
        Bullet bullet = new Bullet(this.owner,this.world,style,color, 2,this.owner.
        world.addBulletAt(bullet);
        return bullet;
    }
}
```

除了子弹工厂外，还有 CreatureFactory 用来生成 Creature 及其派生类。

4.2 建造者模式

在生成地图的时候，使用了 WorldBuilder 进行管理，当然这个类是大框架自带的，我能做的只是学习这一设计模式，在复杂对象的构建中可以简化过程。

4.3 OCP 开放封闭原则

本游戏中，通过 Creature 从基类派生出了 Monster, 可以对方法进行覆写，但是对于修改封闭。

5 课程感言

5.1 个人感受

这学期终于结束了（完结撒花），这门课真的好高级，从面向对象到多线程到网络编程到设计方法。我的感受只能是基础不牢、地洞山摇，在 jw05 的时候，好不容易了解了多线程的工作机制，但是 Socket 编程又破防。没有学过计算机网络真的哭了，对于通信机制很不了解后寸步难行。在这里也反思一下，在网络通信中，首先尝试新开一个命令行作为一个服务器转发消息，其他 ApplicationMain 作为 Client 客户端，但很不幸，失败了。然后我又尝试将一个 ApplicationMain 作为服务器，转发消息的同时也独立运行，但是很不幸又失败了。我也在反思，我深刻认识到这是由于计算机网络知识了解太浅薄（我为什么要跨专业选课？哦为了学 Java! 怎么还要学计算机网络呜呜呜）。

如果问我后悔选课吗？我可以坚定地说，不后悔! 因为如果不选课，我相信我课后是不会花时间学习的。虽然历尽挫折，大作业并没有做的多么好，但是 jw01-jw04 还是做得非常快乐，同时很自豪的，jw05 也是尽力了。我觉得最大的收获还是开拓了眼界，也对面向对象编程有了自己的理解，在 9 月前我对于面向对象一无所知，原来计算机领域还有这么多新奇的玩意，还有这么多可以学习的，现在，不是我学习 Java、面向对象编程的终点。我也深刻认识到自己有多菜，今后一定再学 Java，不辜负老曹的教诲。我也深刻认识数据结构、算法等基础的重要，今后一定会继续学习。

5.2 课程建议

老曹很和蔼、老曹很温柔、老曹很包容、老曹学识渊博，下学期还来。

如果 b 站有老曹同学的马甲线-网络编程视频，一定一键三连!!!

最后感谢老师录制的视频、精心设计的作业、课堂上的讲授和 Java-2021 的水群!