中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

191220059林龙杰[1]

1. 南京大学, 计算机科学与技术系
E-mail:

摘要 本文主要介绍了本人在2021南京大学秋季学期JAVA高级程序设计课程进行的课程实验设计。实验的内容是用java语言编写一个ruguelike地牢游戏，并进行逐步优化，添加多线程，存档，测试，网络对战等功能。

关键词 JAVA语言程序设计, Ruguelike, From Scratch

1 开发目标

项目介绍，项目源代码下载，项目基本功能浏览，项目最终功能浏览
本项目的目标为用JAVA语言，开发一个ruguelike地牢冒险游戏—-《Grandpa saves gourd babies》。

2 设计理念

该游戏编写的主要理念就是用简明清晰的代码实现一个画面精美的小游戏。在代码编写的过程中，体现出java面向对象程序设计的逻辑性。框架上尽量能够做到代码简洁，架构清晰。在类的设计上尽量能够降低耦合度，提高可复用性，可维护性。

3 技术设计

3.1 并发问题

在游戏中，存在许多敌对势力不停巡逻，在玩家靠近时会追踪玩家进行攻击。在创建敌方时首先运用了多线程来提高程序效率。而在后续多人游戏开发过程中，涉及到一个随时接受信息的过程，这里也必须用线程来接收信息从而让游戏能够非阻塞地正常运行。

在我的程序中，并没有涉及到线程之间有太多相互依赖关系，所以想要实现多线程其实就非常简单，只要用基本的Thread类，Runnable接口，或者线程池来进行线程的启动即可。

比如网络通信中的接包线程如图一：

```java
public class InfoReceive implements Runnable{
    DatagramSocket socket=null;
    private int port;
    NetPlayScreen t;
    public InfoReceive (int port,NetPlayScreen t)
    {
        this.t=t;
        this.port=port;
        try{
            socket=new DatagramSocket(port);
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        while(true)
        {
            try {
                byte[] data=new byte[1024*60];
                DatagramPacket packet= new DatagramPacket(data,0,data.length);
                //接受包裹
                socket.receive(packet);
                //分析数据
                System.out.println(port+": 收到一个包");
                byte[] datas=packet.getData();
                int len=packet.getLength();

                ObjectInputStream ois=new ObjectInputStream(new BufferedInputStream(new ByteArrayInputStream(datas)));
                Object r=ois.readObject();
```

图 1　receiver

Figure 1　receiver

3.2　输入输出

本项目中几个重要功能运用了对象的序列化。具体来说，有两个地方用到了输入输出流。

1.存档功能：在存档功能的实现中，我的方法是在存档时直接将当前的游戏全局PlayScreen类利用IO给序列化到文件中，在读取存档时，再从对应文件中反序列化得到游戏全局加载到当前PlayScreen。利用对象的序列化，可以用非常简便的代码实现复杂类的保存，你需要做的就是将所有涉及到序列化的类全部实现Serializable接口。

2.网络通信功能：在网络通信功能实现中，与socket交互必不可少要涉及IO，我选择了UDP的Datagrampacket来发包，同样发送的是序列化的对象，这样可以在对象的角度来包装数据，接取数据。

3.3　通信问题

在通信阶段，遇到了一些困难，最后也没有很好的解决。首先是因为我没有选择一服务端，多客户端的模式，为了简便我的设计目标直接是两个玩家端进行信息传输。但由于思路不够清晰，出现了很多问题。通信方面我才用了UDP协议利用DatagramSocket发送DatagramPacket进行通信。而网络IO的内容，我选择了利用序列化直接传输对象。

而在网络编程时，要合理打印信息，思考接发信息的顺序来调试代码，这是我在实践中体会到的。

4　设计方法

在实现游戏的基本框架时，运用的是JAVA设计模式中的—-工厂方法（Factory Method），工厂方法模式允许将产品类的实例化推迟到具体的创建者子类，由创建者子类决定实例化哪一个产品类。

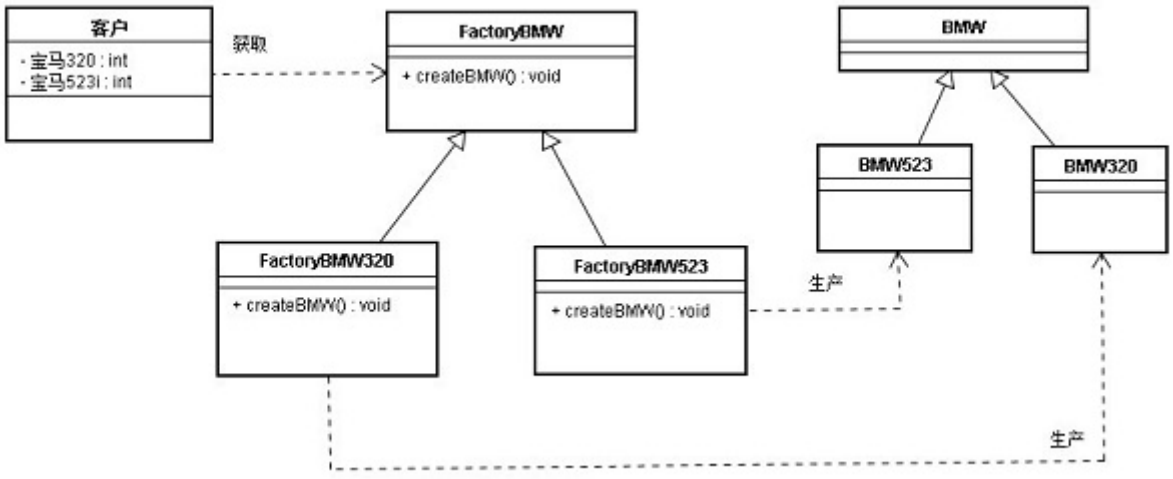

图 2　factory

Figure 2　factory

使用工厂方法后，调用端的耦合度大大降低了。工厂方法模式对对象的创建进行了包装，使得客户程序中仅仅处理抽象产品角色提供的接口。在我的程序中，工厂就是CreatureFactory，绑定一个世界负责所有的生物创建。而多种多样的生物们继承自CreatureAI基类，World不需要知道怎么创建生物，只需要交给工厂就可以了。结构如图2的CarFactory模式。

5　总结展望

这是本人第一次使用JAVA语言来编写有一定复杂性的程序。这次的开发涉及到JAVA基本语言逻辑的使用，面向对象思想的实践，JAVA异常机制的使用，JAVA 并发编程的实现，单元测试的编写，文件与网络IO，以及用于思考整个程序框架的JAVA设计模式。我认为涵盖了学习内容的大部分方面，是一次非常有意义的实践，尤其是对于我这样一个初学者。但实现成果中依然有太多不足，在之后，将会继续完善该项目。而JAVA的学习之路，算是有了一个很满意的起步，未来还会有长的路会走!

参考文献

1　Trystan. Roguelike tutorial. Journal, Thursday, January 21, 2016

1

1. *Affiliation, City* 000000*, Country*
E-mail: