中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

Developing a Java Game from Scratch

母舰

南京大学, 南京 210023
E-mail: 201250137@smail.nju.edu.cn

收稿日期: 2021-xx-xx; 接受日期: 2021-xx-xx; 网络出版日期: 2022-xx-xx

摘要 本文是南京大学秋季学期课程高级 Java 程度设计的大作业的报告，经过一学期的学习，借助重写的 AsciiPanel 作为 UI，并结合 Java 泛型，输入输出，lamda 表达式，并发编程，网络编程技术以及 Maven 自动构建基本完成一个类似于泡泡堂的小游戏。报告中基本陈述了游戏的灵感来源，设计理念，以及游戏中用到的技术细节

关键词 游戏，存档，联机，并发，Java，NIO

1 游戏介绍

该游戏的灵感源于经典的 4399 游戏泡泡堂，玩家操纵葫芦娃在地图上上下左右移动，按下 J 键可以放置炸弹，炸弹将会在一定时间后爆炸，并对周围可以摧毁的物体和生物造成一点伤害，当物体和生物生命值清零时即摧毁成功，炸弹同样会误伤自己。

引用格式: 作者 1, 作者 2, 作者 3, 等. 中国科学 LaTeX 模版. 中国科学: 信息科学, 在审文章
啊啊啊, Xing M M, Xing M, et al. Title for citation (in Chinese). Sci Sin Inform, for review

2 设计理念

```
|  ├─main
|  |  └─java
|  |      |  Main.java
|  |      |
|  |      ├─asciiPanel
|  |      |      AsciiCharacterData.java
|  |      |      AsciiFont.java
|  |      |      AsciiPanel.java
|  |      |      TileTransformer.java
|  |      |
|  |      ├─file  # 存档文件
|  |      |      data.txt
|  |      |
|  |      ├─img   # 游戏贴图
|  |      |      bloom.png
|  |      |      fire.png
|  |      |      floor.png
|  |      |      flower.png
|  |      |      gourd_only.png
|  |      |      scorpion.png
|  |      |      start_background.png
|  |      |      start_background_(4,
3).png
|  |      |      stone.png
|  |      |      tree.png
|  |      |
|  |      ├─map
|  |      |      Map.java
|  |      |      MazeGenerator.java
|  |      |      Node.java
|  |      |
|  |      ├─network
|  |      |  |  GameClient.java
|  |      |  |  GameSever.java
|  |      |  |
|  |      |  └─doubleGame
|  |      |          Packge.java
|  |      |
|  |      ├─progress    # 线程控制，怪物管理
|  |      |      CreateMonster.java
|  |      |      DealExplore.java
|  |      |      Game.java
|  |      |      MonsterThread.java
|  |      |      State.java
|  |      |
|  |      ├─screen   # 屏幕
|  |      |      ClientScreen.java
|  |      |      ClientStartScreen.java
|  |      |      EndScreen.java
|  |      |      Screen.java
|  |      |      StartScreen.java
|  |      |      WorldScreen.java
|  |      |
|  |      └─thing    # 物件，生物和物体
|  |              Bloom.java
|  |              Creature.java
|  |              Fire.java
|  |              Floor.java
|  |              Flower.java
|  |              Goods.java
|  |              Monster.java
|  |              Player.java
|  |              Stone.java
|  |              Thing.java
|  |              Tree.java
|  |              World.java
|  |
|  └─test
|      └─java
|          ├─progress
|          |      CreateMonsterTest.java
|          |
|          └─thing
|                  ThingTest.java
|                  WorldTest.java
|
```

游戏主要依靠一个二维数组来实现基本逻辑，并通过对 AsciiPanel 的修改来使得游戏贴图变为自定义贴图来实现基本的游戏界面。游戏中所有的物体都继承自 Thing 类，根据物体的不同特点例如可破坏和不可破坏，可移动和不可移动在继续划分继承关系。游戏的主体是 World，Word 持有一个二维的 Thing 类数组。其中葫芦娃由玩家键盘输入控制，怪兽由怪物制造类产生，每个怪兽由一个线程控制模拟，每个炸弹同样由一个线程控制。

3 技术细节

3.1 解决多线程冲突问题

将一系列涉及到对游戏主体的操作都改为用 world 中提供的接口进行处理，并为这些方法加上关键字 synchronized 即可实现同一时间只有一个对象在地图上进行判断和移动，可以避免两个对象同时观察到可以前进而同时站上同一个 tile 的问题发生

```
1     public synchronized void dealExplore(int x, int y, int range){
2
3         }
4
5         public synchronized void canGoUp(int x, int y){
6
7         }
8
9         public synchronized boolean setMonster(Monster monster, int x,
              int y){
10
11        }
```

3.2 多线程设计

游戏线程分为四类，一类是怪物控制线程，一类是炸弹控制线程，一类是玩家操作主线程，一类是游戏刷新线程。

为了模拟每个一个怪物为单独的个体，为怪物的产生的生命周期内的活动写了一个线程，CreatMonster 根据玩家需要创建出一定数量的怪物线程类，每个线程负责创建一个怪物，后续怪物的行为逻辑都将由独立的线程控制。

```
1     public class CreateMonster {
2     private final int numberOfMonster;
3     private World world;
4
5     public CreateMonster(int n, World world){
```

```
6           numberOfMonster = n;
7           this.world = world;
8           ExecutorService exec = Executors.newFixedThreadPool(
                numberOfMonster);
9           for (int i = 0; i < numberOfMonster; i++){
10              exec.execute(new MonsterThread(world));
11          }
12          exec.shutdown();
13          System.out.println("Create Successfully! ");
14      }
```

炸弹需要倒计时 2 秒，放在主线程中会造成阻塞，采用单独的线程控制每一个炸弹，这里不采用 lamda 表达式，因为要将每个活跃的线程记录下来为了后续的存档做准备

```
1       public class DealExplore implements Runnable, Serializable {
2       Bloom b;
3       World world;
4
5       public DealExplore(Bloom b, World world){
6           this.b = b;
7           this.world = world;
8           world.addExplore(this);
9
10      }
11      @Override
12      public void run() {
13          try {
14              b.explore();
15          } catch (Exception e) {
16              e.printStackTrace();
17          }
18          world.removeExplore(this);
19      }
20  }
```

单的游戏刷新主线程

```
1       while (true){
2                   try {
3                       Thread.sleep(60);
```

```
4                       game.repaint();
5                   } catch (InterruptedException e) {
6                       e.printStackTrace();
7                   }
8               }
```

3.3 游戏暂停的实现

通过一个状态类来表明游戏的状态，每个线程的循环监控静态变量，修改静态成员变量即可实现对游戏进程的控制

```
1       /**
2  * 表示游戏状态
3  * run（运行）
4  * stop（暂停）
5  * end（结束）
6  */
7  public class State implements Serializable {
8
9
10     public static String state = "run";
11
12 }
```

3.4 游戏存档的实现

游戏存档的实现是简单的，通过 Java 的序列化机制可以将类序列化写进文本文件中保存，想要读取存档时再序列化取出即可，本游戏存档实现的关键在于在 World 类中实时存入未结束的线程，这样才能实现对其它线程游戏状态的保存，在下一次读取存档开始游戏时将之前保存过的线程循环启动即可基本实现游戏存档的恢复。

```
1      public void reSetExplore(){
2          if (explores.size() != 0){
3              ExecutorService exec = Executors.newFixedThreadPool(
                   explores.size());
4
5              for (Runnable r : explores){
6                  exec.execute(r);
7              }
```

```
8               exec.shutdown();
9           }
10
11      }
```

3.5 游戏联机

游戏主体放在服务器上，玩家在客户端进行操作，客户端将操作编码后发送给服务器进行解析，服务器解析命令后控制指定的玩家进行相应的操作。服务器会将游戏内容编码后发送给客户端，客户端解码后将画面显示给用户。客户端初次连接时会得到一个服务器分配的 id 用于分配角色并识别玩家的身份。

技术部分采用 NIO 机制，服务器采用 Selector 监视多个通道实现非阻塞机制。

部分代码

```
1       public void listen() throws IOException {
2           System.out.println("服务器启动成功");
3           while(true) {
4
5               int readyChannels = selector.selectNow();
6
7               if(readyChannels == 0) {
8                   continue;
9               }
10
11              Set<SelectionKey> selectedKeys = selector.selectedKeys();
12
13              Iterator<SelectionKey> keyIterator = selectedKeys.iterator
                    ();
14
15              while(keyIterator.hasNext()) {
16
17                  SelectionKey key = keyIterator.next();
18
19                  if(key.isAcceptable()) {
20                      // a connection was accepted by a
                            ServerSocketChannel.
21                      SocketChannel socketChannel = serverSocketChannel.
                            accept();
```

```
22                      socketChannel.configureBlocking(false);
23                      socketChannel.register(selector, SelectionKey.
                            OP_READ|SelectionKey.OP_WRITE);
24
25                      socketChannel.write(CHARSET.encode("ID"+(number++))
                            );
26
27
28
29                  } else if (key.isConnectable()) {
30                      // a connection was established with a remote
                            server.
31                  }
32                  ..............
33      }
```

（以下为百度的 NIO 介绍）

NIO 和传统 IO（一下简称 IO）之间第一个最大的区别是，IO 是面向流的，NIO 是面向缓冲区的。Java IO 面向流意味着每次从流中读一个或多个字节，直至读取所有字节，它们没有被缓存在任何地方。此外，它不能前后移动流中的数据。如果需要前后移动从流中读取的数据，需要先将它缓存到一个缓冲区。NIO 的缓冲导向方法略有不同。数据读取到一个它稍后处理的缓冲区，需要时可在缓冲区中前后移动。这就增加了处理过程中的灵活性。但是，还需要检查是否该缓冲区中包含所有您需要处理的数据。而且，需确保当更多的数据读入缓冲区时，不要覆盖缓冲区里尚未处理的数据。

IO 的各种流是阻塞的。这意味着，当一个线程调用 read() 或 write() 时，该线程被阻塞，直到有一些数据被读取，或数据完全写入。该线程在此期间不能再干任何事情了。NIO 的非阻塞模式，使一个线程从某通道发送请求读取数据，但是它仅能得到目前可用的数据，如果目前没有数据可用时，就什么都不会获取。而不是保持线程阻塞，所以直至数据变得可以读取之前，该线程可以继续做其他的事情。非阻塞写也是如此。一个线程请求写入一些数据到某通道，但不需要等待它完全写入，这个线程同时可以去做别的事情。线程通常将非阻塞 IO 的空闲时间用于在其它通道上执行 IO 操作，所以一个单独的线程现在可以管理多个输入和输出通道（channel）。

4 课程总结

通过这门课我对与 Java 的各种机制有了更多的了解和掌握，深入学习到了 Java 中的诸如泛型，并发，输入输出，构建，网络编程方面的知识。虽然我最后完成了一个具备存档，联机功能的具有图形化界面的游戏，但是其中仍然有很多问题，bug 之类的存在，与此同时，在大作业之中我也很少考虑到优化方面的问题，大部分都在以完成，可以运行为目的，但也较上课之前有了很大的进步，

我也会在今后继续学习，努力打更好的代码。

参考文献

1 chun cao. ppt of Advanced Java programming

Developing a Java Game from Scratch

Jian Mu

NJU, NanJing 210023*, China*
E-mail: 201250137@smail.nju.edu.cn

Abstract This article is a report on the major assignments designed for the advanced Java level of the Nanjing University fall semester course. After one semester of study, with the help of the rewritten AsciiPanel as the UI, combined with Java generics, input and output, lamda expressions, concurrent programming, network programming technology

Keywords game, load, network, concurrent, Java, NIO