中国科学 : 信息科学

SCIENTIA SINICA Informationis

论文

南京大学《Java高级程序设计》项目报告

191220080 马英硕

南京大学, 南京

E-mail: 191220080@smail.nju.edu.cn

摘要 这是我2021年秋季南京大学《Java高级程序设计》课程大作业项目报告文档。

关键词 Java, 项目报告, 文件存储, 网络通信

1 Introduction and Developing Goals

项目地址链接: https://github.com/jwork-2021/jw08-final-PandaAwAke

单机版演示视频: https://www.bilibili.com/video/BV1T34y1o7Ea

我制作的游戏是葫芦娃与蛇精的炸弹泡泡堂，支持联机操作、日志记录与重放、关卡设定等特性。

1.1 Gamerule

- 该游戏最多可支持4个人类玩家，并自带4个电脑玩家；
- 每个人类玩家拥有3条命，表现形式为“葫芦娃”，可以安放并提前引爆炸弹；
- 每个电脑玩家拥有2条命，表现形式为“蛇精”，蛇精安放的炸弹不会伤害蛇精；
- 每个炸弹拥有3x3的爆炸范围（可穿墙），自动引爆时间为3秒；
- 人类全部消灭电脑玩家或电脑全部消灭人类玩家，则游戏结束。

1.2 Workflow

本游戏的操作流程如下：

(1) 首先打开一个服务端(要求com.pandaawake.gourdgame.Config中设定ServerMode=true)，该服务端同时也搭载了一个客户端；

(2) 接着打开不超过3个其他客户端(要求com.pandaawake.gourdgame.Config中设定ServerMode=false)；

(3) 在服务端窗口按F1开始游戏；

(4) 每个窗口中可以按WASD向四个方向移动，按J键安放炸弹，按空格键提前引爆；

(5) 游戏结束时，服务端将向所有客户端通知，并停止游戏。

1.3 Feature

游戏特征：

- 本游戏的地图详细内容由服务端决定，客户端不从本地加载地图而是接受服务端信息；
- 游戏信息在所有客户端完全同步；
- 每名玩家可以通过移动来移动相机，从而获得更好的地图视角，即地图并不只限于窗口大小。

1.4 Developing Goals

本游戏的灵感来源是童年时玩的泡泡堂游戏。我受其启发从而模拟制作而成，想实现其中关于道具、炸墙等方面的内容。但由于开发时间、精力及水平十分有限，所以道具方面没有成果。

- 其实我开发的游戏并非完全努力向原游戏的功能靠拢，也有超出原游戏的部分，比如更大的地图、相机移动、提前引爆等。

2 Code Design

这一节中，我将简要说明我的代码设计思想，包括整体项目代码架构，Workflow等等。很可惜在临近期末时学业事情变多，我的时间不太足够在这个项目上继续实现部分设想中的内容；我在这个项目代码上的后期工作其实也有点力不从心，因为最初设计游戏时未考虑联机问题，导致游戏部分逻辑的实现在接入网络对战后显得异常棘手……在网络通信上的一些设计失败导致我写代码时总是看不到尽头，变得有些绝望，不知道这样写行不行得通，然后花几个小时踩雷再重新换一种思路实现……还好最后以一种能跑的通而且效果还不错的方式实现了，但是代码像屎山一样没处理emmm。。临近期末确实事情有些多，可能来不及处理这些代码了。不过这也是我第一次实现网络对战类游戏，踩过的那些雷总会成为经验的。

2.1 Mandas Tiled2D Game Engine

我曾有在C++上开发小型游戏引擎的经历，结合我们jw05中给出的很Raw的AsciiPanel，便诞生了在JPanel基础上做一个小型2D Tiled游戏引擎这个念头。本引擎的包名为com.mandas.tiled2d。

2.1.1 Application and Window

包名：com.mandas.tiled2d.core

游戏引擎之所以称为“引擎”，是因为它可以带动整个游戏转动，并为整个游戏铺设基础设施：即不断更新游戏，以及为游戏提供绘制窗口这两个功能。

[设计来源与初衷]

我从jw04中WorldScreen、Main等代码受到启发，觉得jw04中给出的示例代码耦合度太高，非常迷惑人。Main类实际上是一个JFrame，它还集成了一个AsciiPanel；WorldScreen本身不属于JPanel又不属于Component，其设计功能应是把World和AsciiPanel 衔接起来，提供中间服务；但在它这里还要处理按键事件，而且Screen这个名字也有点费解……我想降低这种耦合度，使代码更加清晰，结构更明朗。对于一个游戏而言，我们要尽可能将游戏的逻辑与其他部分分离。这就要分开窗口的事件、绘制等各种功能。

[设计思路]

• 对于游戏上层逻辑，我们可以抽象出Application()。Application就是一个一直运转的机器，对整个游戏进行更新；

• 对于更基础的设施而言，我们抽象出Window。Window就是一个JFrame，它自然集成着类似于AsciiPanel的JPanel，并在窗口这里注册各类监听事件，比如监听键盘、鼠标、窗口事件等。它提供了我们绘制的基本环境；

• 窗口的实际阻塞运行不由我们控制，所以只能用一个线程去更新Application了。带动整个游戏转的原理在于Application的OnUpdate函数。更新线程不断调用Application的该函数，并告知距离上一次调用过了多长时间(timestep)。

• 我定义了用户接口GameApplication，引擎的用户必须实现它。这是用户编写的游戏的Application。在引擎的Application中，会不断更新GameApplication；而所有有关于Window的绘制、组件管理、事件监听已经完全被封装在引擎之中！也就是说，此时只需要用户在GameApplication的实现类中进行自己游戏的编写，而完全无需管理JFrame，JPanel的任何内容！

2.1.2 Renderer and Texture

包名：com.mandas.tiled2d.renderer

虽然我们封装了Application和Window，但用户总要自己定义绘制方式的吧？AsciiPanel对于做2D游戏而言其实并不好用，它的文件比较复杂，还指定了前后景色——说白了就是控制台绘制方式不完美适合2D游戏绘制。我参考AsciiPanel写出了Renderer，这是我们引擎的渲染器。渲染器是游戏引擎最重要的部分之一，诚然我们这里不使用OpenGL等Graphics API，我们只是继续封装JPanel的功能。Renderer提供了大量接口，以三种绘制方式帮助用户完成简单的引擎预设的2D游戏场景绘制架设。

此外为了方便用户绘制图片，我提供了Texture类来封装图片，它提供了一些简易的图片的缩放功能和空白颜色功能，Renderer绘制图片时也是以Texture为准的。与之对应，在com.mandas.tiled2d.utils.TileFileParser处提供了对Tile材质图片的解析功能。

2.1.3 Event System

包名：com.mandas.tiled2d.event

对我个人而言很想实现事件系统！虽然在jw05中最初没有实现，但现在已经完全实现了。这里定义了各种事件类型，以及每种事件对应的详细信息。用户只需要使用EventDispatcher.register来注册回调函数即可，使用非常方便。比如EventDispatcher.register(KeyEvents.Pressed.class, 回调lambda函数);即可实现用户对按键事件的响应。而且同一个事件还可以注册多个回调函数。

2.1.4 Scene and Entity Component System

包名：com.mandas.tiled2d.scene

这是游戏引擎的ECS，Entity Component System(我不知道怎么翻译)。每个Scene由一堆Entity组成，每个Entity由一堆Component组成。Component完成其最基本的功能，目前本引擎可以使用的Component有：

- TransformComponent: 可以完成对Entity的绘制位置设定；
- CameraComponent: 可以让Entity拥有游戏绘制的相机；
- TileTextureRenderComponent: 可以指定Entity的绘制材质。

2.1.5 Log Instrumentation and Log Management

本引擎集成了log4j2，拥有一些基础的配置设定，包括logging格式，自带Log.mandas()、Log.app()、Log.file()三个logger。还在utils.LogParser处给出了log文件解析功能。

2.2 GourdGame

引擎和游戏一般都是同步开发的，这样效果最好。

2.2.1 Game Logic

[地图设计] 地图设计参考了网络某地图。我希望地图的元素有可破坏性，于是设计了树和土墙；希望地图有阻挡属性，于是设计了石墙。于是地图便在实时变化，丰富了游戏内容。代码上自然有Tile类、Thing类，具体使用的类为Thing的子类Tree、Wall、TwiceBreakableWall。

[精灵设计] 精灵在代码上用Sprite类表示。蛇精、葫芦娃、炸弹均为精灵。在此基础上定义了MovableSprite作为可移动的精灵基类，在MovableSprite基础上又定义了玩家操作的PlayableSprite。Calabash和Snake代表葫芦娃和蛇精，它们都是PlayableSprite，可以进行安放炸弹等游戏行为。Bomb是炸弹，它将不断检查自己是否到达爆炸时间或者自己是否应该立即爆炸，并在爆炸时对Scene进行一些修改，比如移除树、移除精灵等。

[玩家设计] Player类定义了“玩家”的概念，它实际上在操控一个PlayableSprite，为是游戏操作的主要提供者。游戏的所有操作通过Player描述，玩家操作的也是自己的Player。

[关卡设计] Level类定义了关卡内容，包括地图大小、地图每一个Tile具体是什么，人类玩家和电脑玩家的出生点是哪些等。关卡可以通过服务端的SceneInitializer来对Scene完成初始化。

2.2.2 Network

网络对战太难了，诚实的说没预料到大作业要引入网络通信，我的游戏实现方式转变过来太痛苦太困难了，现在的版本代码也非常垃圾。最开始尝试的版本是希望在服务端和客户端之间传递操作，比如客户端传递一个向上移动的操作到服务端，服务端再转发给所有客户端这种方式。但实现完成后发现完全不同步，而且几乎没什么修补的方法。只能重新实现了一个场景状态同步，确保服务端和客户端的Scene是一致的而并不转发操作，即客户端只向服务端传递操作而服务端只向客户端传递Scene状态。最后终于可以了。其中还踩了不知道多少坑，网络通信写了很久很久很久很久，代码肯定是几千行级别的，但也算有了点经验吧。

- 我实现了一些游戏控制信号，比如游戏开始、游戏初始化、游戏结束等，这使得每个客户端都知道自己是几号玩家，游戏也能由服务端实现控制。

3 Technical Problems

开发过程中确实遇到了不少技术问题，主要是并发编程和网络通信。我这两方面处理的都不算很好，虽然和网络通信相关的部分我确实实现了类序列化，但没用到接口，主要是我自己人工的调用；而且并发编程同步处理也略有草率，不过最后是能跑。对这方面我还有很多可以研究的方向，也踩了不少坑积累了不少经验。

4 Engineering Problems

我开发时的总体工程方法就是采取游戏引擎+用户代码实现的方式。网络通信采取Selector NIO方法，应该算是用了Reactor模式。

4.1 Network Battle Implementation

我定义了下述这些结构，用以在网络处理模块和游戏之间进行交互：

[**Action**] 这代表一个网络数据表示的操作，可以细分为：

- Game Action (包含游戏控制信号，比如游戏开始等)
- Connection Action (包含网络连接信号，比如有一个新客户端接入)
- Player Action (包含玩家操作信号，比如移动、放置炸弹等)
- Scene Action (包含场景更新信号)

[**DataProcessor**] 这代表一个网络数据处理器。

- ServerDataProcessor: 这代表服务端的网络数据处理器，服务端接收数据后在这里进行解析，解析成对应的Action。这里也可以进行数据合法性判别：比如定义服务端接收到的哪些数据是合理的，可以被解析；它也知道接收到哪些数据是不合理的，比如服务端接收到了游戏控制信号。
- ClientDataProcessor: 客户端的网络数据处理器。和上面同理。

[**ActionPerformer**] 这代表一个Action的执行器。

- ServerActionPerformer: 这里将上面提到的各种各样的Action在服务端中进行处理。每个Action包含了它的详细信息。比如服务端这里可以接收到客户端的Player Action，从而对服务端的玩家进行一个真正的操作，改变Scene。
- ClientActionPerformer: 和上面同理。

[**SocketServer and SocketClient**] 这两个类处理网络连接。

- SocketServer: 采取NIO Selector模式，它有一个待发送的队列，也有一个接收到信息的队列。说白了就是在处理数据，在轮询到某个channel的时候尝试从它读取并放入接收队列，并看看有没有数据需要向它写入。
- SocketClient: 和上面同理。

[**GameServer and GameClient**] 这两个类在总体上处理网络连接的一切。

- GameServer: 不断地看SocketServer是否接收到数据，如果接收到数据就用ServerDataProcessor先进行处理从而获取Action，接着用ServerActionPerformer去执行这个Action。它还提供sendAction功能，用于实时指定向指定的客户端发送需要的数据。该数据会进入SocketServer的待发送队列。
- GameClient: 和上面同理。

4.2 Testing

我写了一些单元测试用例，用以测试部分工具类的功能。单元测试确实很好用，测试简单的功能确实很方便。

5 Course Conclusion

这个课很精彩，讲的内容比较深入，同学们的能力让我大开眼界。老曹太有魅力了。我这个项目获得一个键盘奖品，也很荣幸。在获得的那个时间个人认为可能还算是没发错人，但到后来我自己也有些力不从心的时候，看见大家的项目都很优秀，有点觉得键盘发给我亏了2333。无论如何，感谢大气的老曹，我平时喜欢翘课但这门课我一节也没想翘（最后一节确实是起不来了）。对于课程内容方面我觉得个人没有什么建议，很喜欢这种不面向考试的学习，收获尽在过程之中。

参考文献

1 NativeHook, https://www.github.com/kwhat/jnativehook/
2 og4j2, https://logging.apache.org/log4j/2.x/

南京大学《Java高级程序设计》项目报告

191220080 Yingshuo Ma

Nanjing University, Nanjing, China
E-mail: 191220080@smail.nju.edu.cn