

第三年:2021/2022年

计算机科学项目

最终报告

“Bomberman狂欢



字数:9886个

学生名:徐宣伟

学号:2089260

导师姓名:Mirco Giacobbe

课程名称:计算机科学

内容

1 介绍	3
1.1 炸弹人系列游戏	3
1.2 动机和项目目标	4
2 背景	6
2.1 以一系列游戏开发框架	6
2.2 2D动画技巧.	7
2.3 碰撞检测问题	7
3 软件设计	8
3.1 关键特性	8
3.2 功能需求.	10
3.3 非功能性需求	13
3.4 用户界面	14
4 问题分析和解决方案设计	20
4.1 整体软件架构	20
4.2 核心子模块	23
4.2.1 图形	23
4.2.2 音频	24

4.2.3 输入.	26
4.3 核心子系统	27
4.3.1 艾尔系统.	27
4.3.1.1 查询算法	27
4.3.1.2 整体工作流程.	29
4.3.2 网络	31
4.3.2.1 网络体系结构	31
4.3.2.2 网络通信框架.	31
4.3.2.3 网络技术	33
5 验证评估	35
5.1 软件测试.	35
5.1.1 黑盒测试	35
5.1.2 可用性测试.	36
5.2 评价	40
6 总结	40
参考书目	41
附录A	42
附录B	46

1 介绍

说到游戏，很多家长觉得游戏对孩子没有意义，玩完游戏没有收获。虽然专门为娱乐而设计的游戏，但它们也具有促进计算机领域学习的教育价值。很多人因为游戏的趣味性而受到游戏的启发，去理解和探索游戏开发领域，这无疑对计算机科学的教育意义产生了很大的影响。作为最受欢迎的游戏系列之一，炸弹人已经成为许多人童年的美好回忆，给他们带来了快乐。在这篇论文中我将使用libGDX框架重新打造一款玩法更有创意的《炸弹人》游戏，其中包含了多种可选的难度和模式。

1.1 炸弹人系列游戏

炸弹人，也被简称为Dynablast，是一款基于迷宫的战略电子游戏，最初由Hudson Soft[1]于1985年开发。下面图1是这个版本的部分截图。这款游戏自发布以来吸引了很多粉丝和追随者。在那之后，越来越多的炸弹人系列的新衍生作品在不同的游戏平台上发布，都取得了巨大的商业成功，销量超过1000万套。[2]此外，Bomberman有自己的在线游戏社区“炸弹人wiki”，许多忠实的玩家和业余爱好者自学的程序员在这里表达他们的意见或建议，分享他们的游戏经验，这表明炸弹人系列的受欢迎程度。

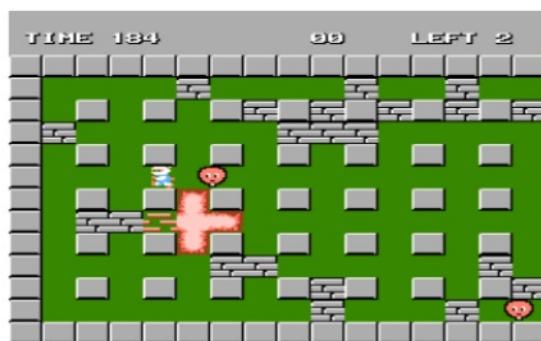


图1:NES版《炸弹人》截图，摘自老游戏网站[3]

通常《炸弹人》系列游戏的玩法意图是，玩家可以在以迷宫为基础的网格世界中，通过战略性地植入炸弹，杀死所有敌人(AI或其他玩家)，从而获胜。炸弹放置后会爆炸，并以“十字”的形状向四个方向扩散，除不可摧毁的障碍物外，放置炸弹的爆炸半径内的任何障碍物或人物都会被摧毁。在游戏场景中，地图上的所有路径都会被许多不同类型的障碍物挡住，形成一个巨大的迷宫，有些类型的障碍物可以通过在它们附近放置炸弹来摧毁，这样玩家就可以通过了。

《炸弹人》系列之一的《Bomber Man World》被Play Meter列为当时最受欢迎的街机游戏中的第46名:[4]它在基本的炸弹人玩法中引入了4种不同的升级道具，在炸毁可破坏的障碍物后可以获得这些道具，它们是：

1. 速度:增加移动速度。
2. 火焰长度:增加放置炸弹的爆炸范围。
3. 炸弹:放置不止一枚炸弹。
4. 无敌:玩家不能被敌人或爆炸杀死，[5]

除此之外，本系列还有两种游戏模式：

1. “普通游戏” :单人模式，要求玩家击败所有人工智能敌人。
2. “超级游戏” :多人模式，即玩家之间的战斗，目的是击败所有其他玩家。[6]

《轰炸机侠世界》的升级道具的引入和各种游戏模式的设计都增加了游戏的可玩性，增强了玩家的游戏体验。这个系列的游戏玩法和戒律对很多人产生了巨大的吸引力，给他们带来了欢乐，包括我。

1.2 动机和项目目标

然而，“Bomber Man World” 系列游戏的一个重要限制是，这款街机游戏不具有跨平台兼容性，这意味着玩家无法在其他平台(如个人电脑)上体验到这款游戏的乐趣

或者是移动设备。跨平台的《炸弹人》系列越来越罕见，因为游戏公司想要增加自己的游戏机销量，如果自己开发的游戏可以在任何平台上玩，没有人会购买游戏公司的独家游戏机。多年来，很多玩家都梦想着能够和别人跨平台玩这款游戏，这样就不用特意和朋友去电玩厅，也不用花一大笔钱从游戏公司买游戏机了。所以这激发了我去开发一款跨平台的游戏，这样在完成桌面平台的游戏开发之后，就可以在不同的平台上发行，这保证了我开发的游戏可以接触到更多的人，让他们体验到这款游戏的乐趣。

开发一款跨平台游戏是一件很复杂的事情，而且技术上有很大的挑战，因为不同的设备之间的工作方式不同，包括硬件、数据格式，所以很难保证代码可以在不同的平台上交叉编译。我的解决方案是考虑和使用一个合适的跨平台游戏框架和编程语言，它可以处理不同设备之间的这些差异，帮助代码适应多个系统。

此外，“轰炸机侠世界”系列仍有一些改进的空间。首先，游戏道具的数量和种类都比较少，可以设计引入更多有新意的武器道具，丰富玩法，提高玩的乐趣，比如火箭筒、手榴弹、地雷等等。其次，不允许玩家选择游戏难度，这意味着玩家必须从最简单的关卡开始，逐渐玩到越来越难的关卡。第三，这款游戏界面不美观，风格单调因为是街机游戏，可以通过添加华丽的画面来提升。所以我想改进这个游戏，从这些方面重新打造一个全新的《炸弹人》游戏，让它变得更好，更有趣。

最后，我的目标是利用跨平台游戏开发框架libGDX，在《轰炸机侠世界》的游戏玩法基础上，用java语言重新制作一款带有更多创新玩法的改进版bomberman游戏。这款游戏不仅包含了很多游戏道具，还为玩家提供了多种可选的难度和模式。在离线模式中，玩家可以体验不同的难度，他们需要独自对抗AI敌人。AI的智能在很大程度上决定了游戏的乐趣，所以我让人工智能随着难度的增加而变得更加智能，他们可以捡起能量来增强自己，使用武器攻击玩家。在线模式下，两名玩家需要互相杀死对方才能获胜。这就是我计划通过这款游戏实现的，用更多创新和有趣的玩法让游戏变得更有趣，让更多的人可以在不同的平台上接触到我开发的游戏，体验游戏的乐趣。

2 背景

在本章中，我将概述一些游戏开发入门所需的基本概念和原则。

2.1 以一系列游戏开发框架：

首先，我将介绍游戏框架的概念并谈谈它的用途。游戏框架是为创建游戏软件而优化的工作环境，它由一组库组成，这些库应该能促进游戏的开发。底层的库，在tun中，是一组程序员在工作中需要的数据(程序、子程序、对象、函数)，[7]一个游戏框架定义了许多必要的api来处理图形、声音、用户输入、数据文件等[8]因此，开发者不需要花费太多时间定义这些功能来开发一款游戏，它可以帮助开发者完成许多常见的开发任务，如渲染、处理音效、创建用户界面、绘制文本、线性代数等，这极大地推动了游戏开发的进步，提高了开发效率。因此，使用合适的游戏开发框架来开发游戏是一个明智的选择。

LibGDX是一个使用java编程语言编写的开源跨平台游戏开发框架[9]。因此，它便于在使用java创建游戏的同时调试代码。Java是一种与其他语言相比具有平台独立性的编程语言，由于其可移植性，它可以很容易地在多个平台或设备上编译和运行。除了包含许多一般游戏开发框架所拥有的库之外，LibGDX最大的特点是其良好的兼容性，它为许多平台提供了单一和统一的API，因此我可以访问多个主机平台，在这些平台上编写或调试代码。我也可以在很多不同的平台上发布代码，而不用在完成桌面游戏的开发后，为每个特定的平台编写代码。在游戏开发中使用libGDX和编程语言java，只满足了开发这款游戏的跨平台特性要求，也显著提高了我的开发效率。

2.2 2D动画:

2D动画是游戏开发中不可或缺的一项技术，它用于使用静态图像创造运动的幻觉。动画由多个帧组成，这些帧以固定的间隔序列显示。[10]游戏场景中游戏角色的移动和跳跃本质上是一个动画，其中多个静态图像在非常短的时间内以循环的方式进行切换和顺序显示。下图中的角色是我在我开发的游戏中使用的，这个图像包含了角色分别在四个方向(左、右、上、下)行走的每一帧动画，通过在一段时间内连续显示这一系列帧来创建行走的动作。



帧率指的是每秒显示的帧数。例如，一个角色完成向左移动的所有动作，这将被视为一个周期，在上图中每个周期有3帧动画，这意味着如果角色必须在一秒内完成一个周期，这3帧应该每秒显示，此时的帧率是3 FPS，所以显示每帧的时间大约是0.3。

2.3 碰撞检测问题:

碰撞问题是游戏开发中不可避免要考虑的问题之一。碰撞检测是检测两个或多个物体的交集的计算问题，应用于多个计算领域

碰撞在游戏中是非常常见的，例如，可能会有碰撞

角色与障碍，角色与子弹的碰撞，子弹与障碍的碰撞等等。游戏中这些元素之间的这些碰撞问题需要通过编写单独的代码来实时检测和处理。比如子弹与障碍物发生碰撞就会发生爆炸，就需要编写逻辑判断的代码来实时检查发射的子弹是否与障碍物发生碰撞，如果发生，就会播放爆炸的动画。

Hit-box是一种隐形的边界框，在实时[12]中用于检测游戏中元素之间的碰撞，它通常是一个附着在2D游戏中物体或元素周围的矩形，如人物、子弹、障碍物。如果连接在两个物体周围的两个轴对齐的矩形在碰撞前没有旋转或缩放，则可以通过检查这两个矩形是否重叠来实现最简单的碰撞检测。

在回顾了关于我的游戏开发的背景材料后，我会描述更多关于游戏设计和需求设置的细节。

3 软件设计

在本章中，我将根据项目目标介绍游戏设计的具体内容，包括游戏功能、功能需求和非功能需求、界面设计。

3.1 关键特性

首先，我将从游戏可玩性和创意方面提一下这款游戏的特点。

游戏的第一个特色是玩法和规则的改进，它在保持基本的炸弹人戒律不变的情况下，引入了新的可摧毁障碍“宝箱”和更多具有新意的游戏道具，包括强大的武器和增强角色属性的一般升级道具。每个游戏道具只能通过“以等概率炸毁宝箱”来获得。游戏道具一共有9件：

1. 医疗包:恢复1点生命值。
2. 跑鞋:增加移动速度，可以累积。



3. 硝化甘油溶液:增加放置炸弹的爆炸范围，

可以积累的。

4. 地雷:放置一个所有角色都看不见的地雷。
5. 火箭炮:获得火箭炮可以发射两枚火箭。



6. 手榴弹:在障碍物后投掷手榴弹。



7. 炸弹包:放置多个单枚炸弹，放置的炸弹数量可以累积。



- ☒。防暴盾:一次性承受炸弹、地雷、火箭筒、手榴弹的爆炸伤害。



9. 手套:在爆炸前3秒将炸弹推出。



对于游戏规则来说，玩家需要通过策略性放置炸弹或使用武器来杀死所有敌人才能获胜，如果玩家在时间限制内没有杀死所有敌人或被敌人杀死，则游戏失败。游戏性的提高丰富了游戏内容，提高了趣味性。

第二个特色是为单人模式提供多种可选难度，让玩家可以选择任何难度不同的关卡。随着难度的增加，AI会变得越来越聪明，可以合理地使用游戏道具，他们会不断地寻找“宝箱”并放置炸弹来摧毁它们，以收集更多的游戏道具，他们也会使用武器攻击玩家，这使得AI对玩家来说更难被杀死。可选的难度总共有3个：

1. 简单级别:人工智能不能捡起和使用任何掉落的道具。
2. 中级:AI只能使用可以增强其属性的非武器道具(包括医疗包，跑鞋，硝化甘油溶液)。
3. 魔梦级:AI可以使用任何掉落的道具来增强其属性，并使用武器攻击玩家。

更智能的人工智能增加了游戏的难度，使游戏更具挑战性和趣味性。

第三个功能是增加了在线模式下两名玩家的聊天室功能，它允许其中一名玩家通过创建或加入聊天室来与另一名玩家进行一对一的游戏，在一个本地的网络区域建立连接，玩家可以发送消息。

在游戏过程中可以互相交流，这增加了玩家之间的互动性和竞争性

根据这些游戏特性，需要进行需求分析，确定游戏需要实现哪些具体功能，所以我将对功能需求和非功能需求进行描述。

3.2 功能需求

明确和定义一个产品的功能需求是很重要的，这可以帮助我检查这个游戏产品是否可以实现这些目标或者识别缺失的需求。

这个软件系统可以分为7个子组件，我列出了每个子组件需要执行的具体功能，以便检查它们捕获的行为是否符合目标中提到的要求。

3.2.1 障碍

- 铁块不能被炸弹或武器摧毁。砖块可以被炸弹或武器摧毁。宝箱可以被炸弹或武器摧毁。摧毁宝箱后掉落的游戏道具。
-

3.2.2 炸弹

- 放置的炸弹会在3秒后爆炸。
- 炸弹可以杀死玩家和阿尔。
- 玩家和AI无法通过炸弹。
- 炸弹的爆炸范围可以增加和积累。
- 可以增加并积累三秒内放置的炸弹数量。炸弹的散布范围要被障碍物阻挡
- 炸弹可以由AI或玩家使用非武器道具手套来推。被推的炸弹如果碰到障碍物就会爆炸。
- 炸弹不能推到地图边界之外。

3.2.3 球员

- 玩家可以通过方向键操纵炸弹人在四个方向(上、下、左、右)上移动。
玩家可以通过按空格键放置炸弹。
- 玩家可以捡起掉落的任何游戏道具。
- 玩家可以通过按数字键1、2、3、4来使用4种不同的武器。玩家可以被炸弹或武器杀死。
- 玩家不能通过障碍物和炸弹。
- 玩家不能走出地图的边界。
- 玩家可以在游戏中随时点击相应的按钮暂停或恢复游戏。
- 玩家可以在游戏中随时点击相应按钮查看游戏规则或帮助。
- 玩家在游戏中可以随时退出并返回游戏主菜单。玩家可以在决斗模式中随时点击“发送”按钮互相发送信息。
- 玩家可以在决斗模式中随时接收信息。
如果玩家受到炸弹或武器的伤害，将会失去生命值。
- 在单人模式中，如果在限定时间内消灭所有AI敌人，玩家将获得胜利。
- 在单人模式中，如果所有的AI敌人都没有在时间限制内被消灭，玩家就会输。
- 如果玩家被AI敌人消灭，他们就会失败。
如果玩家输了，应该会弹出一个弹出窗口来显示游戏失败。如果玩家赢了，应该会弹出一个显示游戏成功的弹出窗口。玩家可以在弹出式游戏失败窗口中选择重启游戏或返回主屏幕。
- 玩家只能在弹出式游戏胜利窗口中选择返回主屏幕。
在决斗模式中，如果两名玩家仍然在时间限制内存活，那么游戏应该是平局。
- 玩家可以通过点击游戏主菜单中的相应按钮来选择游戏难度、游戏模式。
- 玩家可以通过点击游戏主菜单中的对应按钮来查看游戏规则或帮助。
- 玩家可以在设置界面左右滑动，调节背景音乐或游戏音效的音量。
- 玩家可以在设置画面中开启和关闭游戏音效或背景音乐。
- 玩家可以在决斗模式的屏幕上点击相应的按钮创建一个房间或加入一个房间。
-
-
-

3.2.4 游戏道具

3.2.4.1 核武道具

- 获得医疗包可以恢复1点生命值。
- 买一双跑鞋可以增加移动速度。
- 获得一杯硝酸甘油溶液，可以在四个方向(上、下、左、右)分别增加放置炸弹的爆炸范围一格。获得防暴盾牌可以抵御炸弹、火箭筒、手榴弹、地雷造成的一次爆炸伤害。

3.2.4.2 武器

- 获得火箭炮可以让角色发射两枚火箭。
- 按数字键1可以使用火箭筒发射火箭。发射的火箭不能飞出地图的边界。
- 按数字键3可以埋设一枚地雷，三秒后就看不到了
地雷只能放置在空的格子上。
- 手榴弹只能在角色前方的格子上有障碍物时投掷。
按4号数字键可以在障碍物后投掷手榴弹，并在前方第三格上造成 3×3 格的爆炸。
- 获得炸弹包可以在3秒内增加一个额外放置的炸弹。使用手套按数字2键，
三秒内将炸弹推出。
-

3.2.5 游戏面板

- 游戏面板应实时统计并显示玩家和AI的生命值。
游戏面板需要实时统计并显示玩家获得的游戏道具。
- 游戏面板应实时记录并显示游戏倒计时
游戏面板可显示决斗模式下两名玩家之间的聊天记录。
-

3.2.6 AI

- 阿尔必须绕过障碍物才能移动。
- AI可以放置炸弹。
- AI应该避开放置炸弹的爆炸半径。
- AI不应该通过障碍。
- ai可以找到通往宝箱的路径。
- AI可以寻找掉落的游戏道具。
- AI在简单难度下无法拾取任何游戏道具。
- 在中等难度下，AI可以拾取非武器道具。
- AI可以在噩梦难度下使用所有游戏道具。
- AI应该在噩梦难度中使用武器而不伤害自己。AI可以被炸弹或者武器杀死。
- 如果受到炸弹或武器的伤害，AI会失去生命值。
- ai可以赢，也可以输。

3.2.7 网络

- 通过创建一个房间来构建服务器。
- 客户端可以通过在局域网中输入IP地址来搜索服务器。服务器应该接受来自客户端的请求。
- 通过加入房间建立服务器和客户端之间的连接。游戏数据必须在服务器和客户端之间正常传输。
- 客户端和服务器之间的游戏进度必须同步并实时更新。
- 客户端和服务器之间发送和接收的消息必须始终保持同步。
- 这应该会弹出一个弹出窗口，显示离线状态，如果服务器或客户端离开游戏。
-

3.3 非功能性需求

我们已经从不同的子组件介绍了软件的功能需求，现在我将介绍描述这个游戏如何行为或运行的规范的非功能需求，而不是定义具体的功能，这与功能需求是完全不同的。

- 兼容性:本游戏必须兼容并运行在具有不同环境的多个平台上，包括Windows、macOs、Android、Linux、ios、WebGL。使用LibGDX开发这个游戏，LibGDX是一个跨平台的游戏框架，具有兼容性强的特点，它为许多平台提供了一个单一统一的API，这意味着我只需要为这些不同的平台进行一些部署配置和定义启动器，这样我就可以在桌面上编译后在这些平台上发布代码，而无需为特定的平台编写代码。

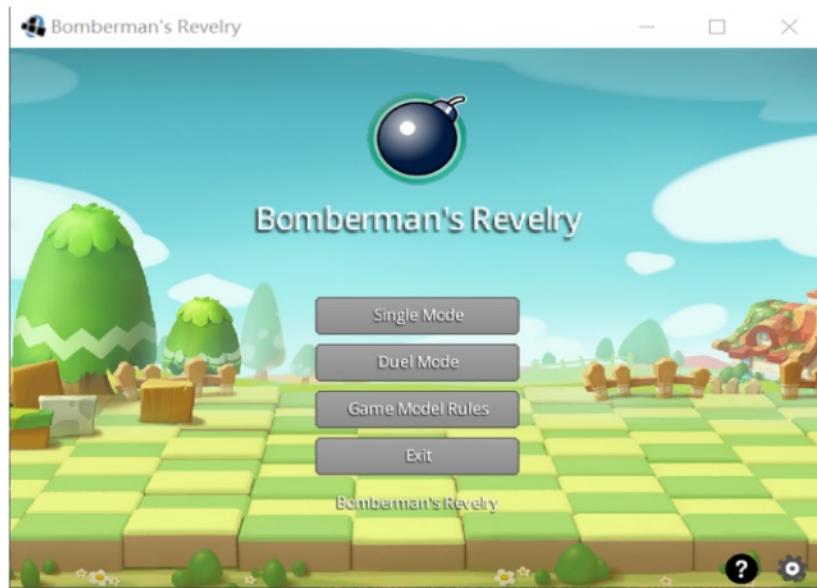
- 可用性:
 1. 用户界面要直观，便于操作。用户可以根据自己的想法，用更少的时间点击游戏界面中的相应按钮，成功完成这一系列操作:查看游戏帮助、调整游戏音量、选择游戏模式和难度、暂停和继续游戏、创建或加入房间、发送消息。用户界面可以简单易学，允许用户在浏览用户界面的同时执行某些特定操作，而无需经过培训或教程等复杂的学习过程，确保用户对游戏满意，
 2. 游戏界面应该快速加载。一旦玩家开始玩游戏，游戏资源就应该快速加载，包括图像、游戏元素、游戏面板。在等待游戏开始的过程中，玩家不会失去耐心和兴趣。

- 性能:在线模式下网络正常时，服务器与客户端的连接时间总计不应超过4秒。创建服务器的时间不应超过1秒，客户端在局域网中搜索服务器的响应时间不应超过1.5秒，服务器接受客户端请求的时间不应超过1.5秒。等待连接的时间不能太长，以免影响在线模式下双方玩家的体验。

3.4 用户界面设计

玩家与计算机之间的交互是决定用户能否正确使用游戏系统的重要因素。
下面我来演示一下游戏的用户界面设计。发射后

游戏中，游戏的主画面会先显示出来，如下图所示：



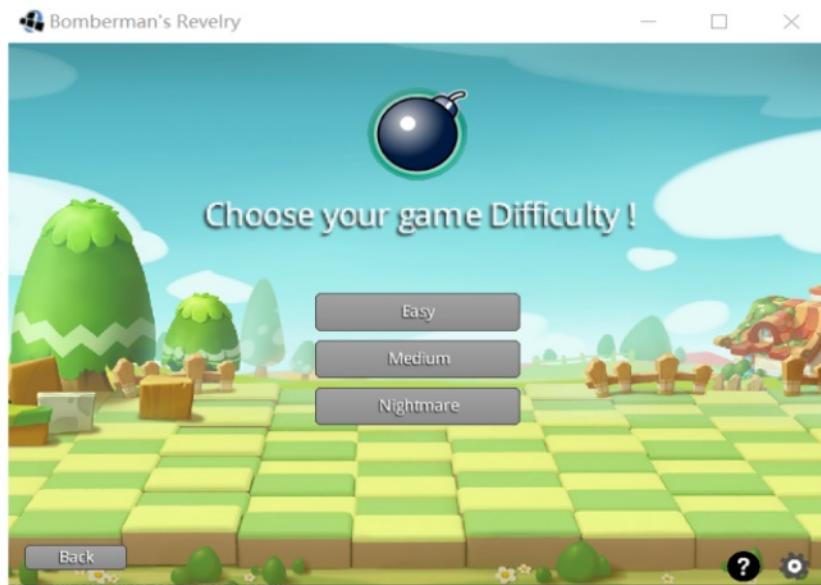
截图1:游戏主菜单

游戏主菜单包含3种基本操作：

1. 选择不同的游戏模式开始玩游戏
 - 单一的模式
 - 决斗模式
2. 查看游戏帮助和规则
 - 游戏控制
 - 游戏道具说明
 - 游戏规则
3. 音频管理
 - 开启或关闭背景音乐和游戏音效调整背景音乐和游戏音效的音量

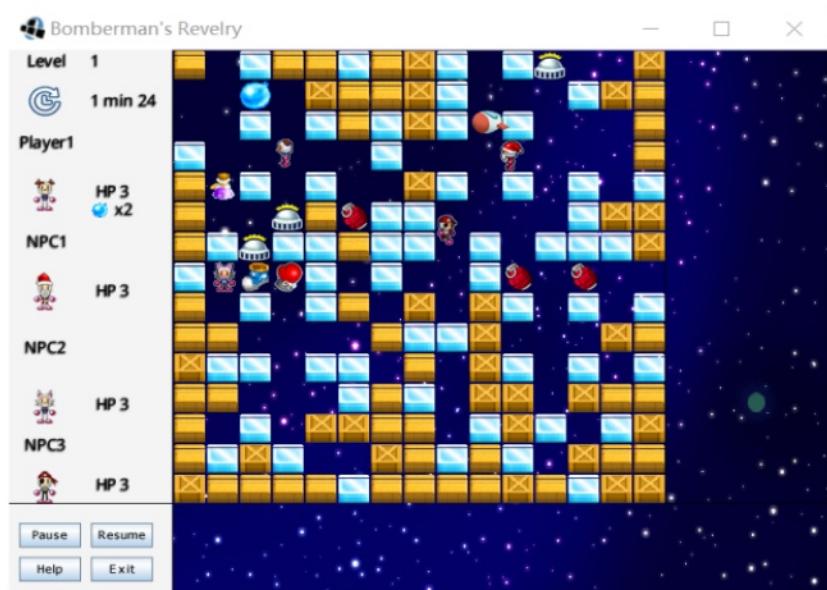
我将依次向大家介绍和描述这三种操作的游戏界面，从第一个操作即游戏模式的选择开始，玩家可以选择单人模式和决斗模式(双人模式)：

选择单人模式：单人模式有三种可选的游戏难度供玩家选择，它们是简单、中等和噩梦。难度选择画面如下图：



屏幕截图2:难度选择画面

选择其中一个游戏难度后，游戏画面会加载并显示，如下图截图3所示：



截图3:单人模式下的游戏画面

从截图3来看，游戏左边面板的顶部统计游戏结束的剩余时间，其次是玩家和其他三个敌人AI的剩余生命值。此外，游戏面板还增加了一个额外的功能，记录玩家已经拿了哪些游戏道具及其剩余数量，以便玩家可以知道哪些道具还没有使用，并允许他们战略性地使用这些道具。面板底部允许玩家在玩游戏时进行一些常规操作，玩家可以暂停或恢复游戏，查看游戏帮助并随时退出当前游戏。

在游戏结局的画面上，当玩家达到胜利或失败的条件时，会弹出一个胜利或失败的窗口来提示玩家这场游戏的胜利或失败，我们可以看到下面两张截图：



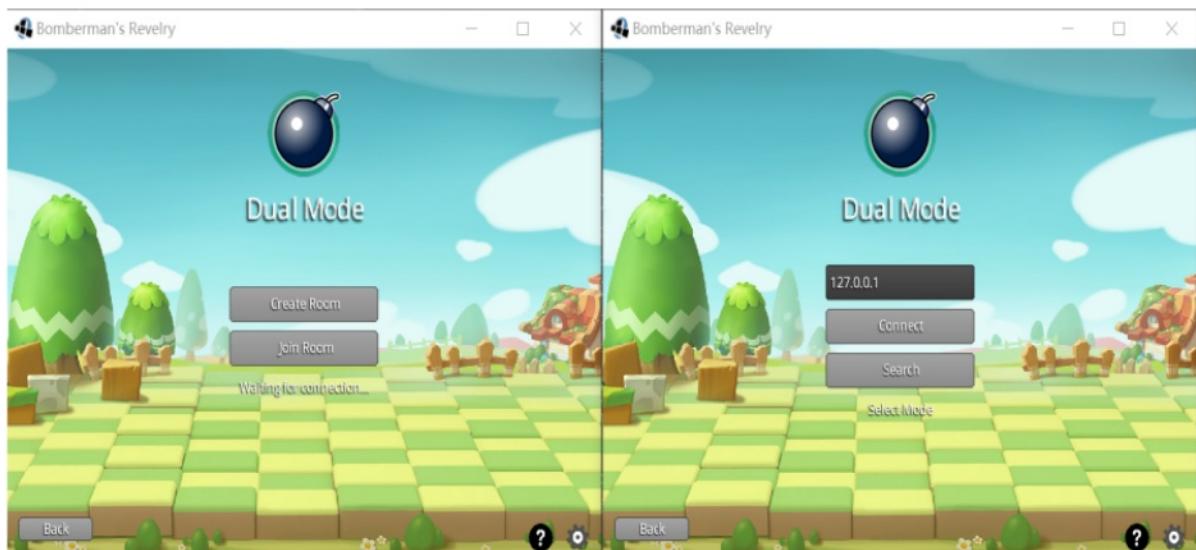
截图4:胜利窗口



截图5:失败窗口

游戏结束，玩家在本局获胜时会被引导回到主界面，玩家输了本局可以选择直接点击对应按钮重新开始游戏，这样玩家就不需要回到主菜单，然后选择上一轮相同模式的相同难度，这样可以为玩家节省时间和精力。

- 选择决斗模式:两名玩家中有一人可以选择创建或加入一个房间



截图6:决斗模式下的连线画面

从截图6中我们可以看到，创建房间的玩家将代表一个服务器，需要等待另一个代表客户端加入房间的玩家在决斗模式下建立连接。在这个过程中，代表客户端的玩家需要输入主机服务器的IP地址，默认情况下通常是127.0.0.1。它还提供了搜索同一局域网中相邻服务器的功能。

当两名玩家成功连接后，会出现网游界面：



截图7:决斗模式的游戏画面

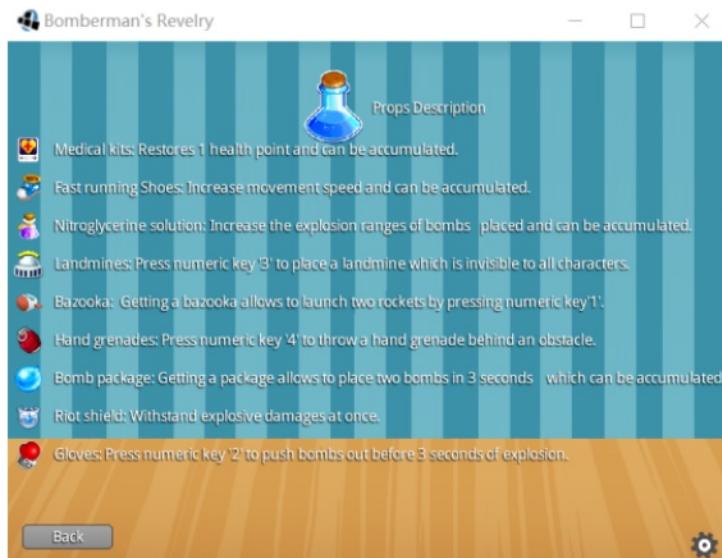
右边的游戏面板记录了游戏中的剩余时间、对手的生命值、道具使用信息。还有一个额外的功能，允许两个玩家通过发送消息进行交流，左边面板记录并显示两个玩家之间的聊天记录，这大大增加了互动性。

第二个操作允许玩家查看和了解游戏操作、道具描述和游戏规则，确保他们知道如何玩这个游戏。



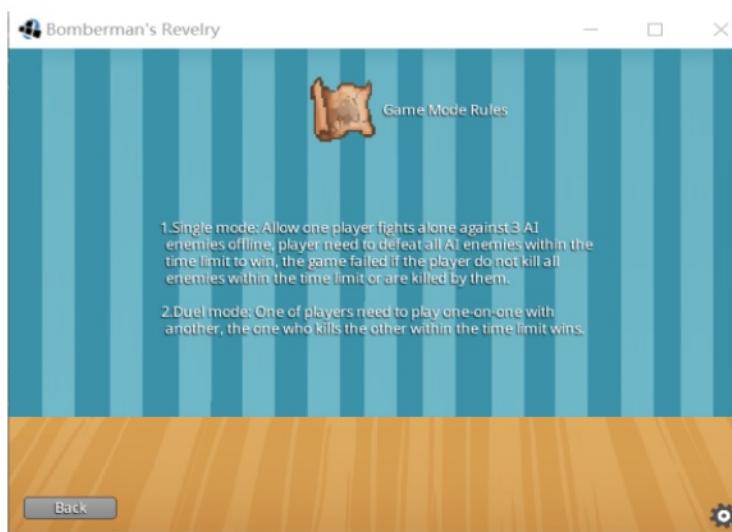
截图8:游戏控制说明的画面

从截图8中，我们可以看到操纵炸弹人在四个方向(上、下、左、右)移动和放置炸弹的操作说明。



截图8:游戏控制说明的画面

截图9详细描述了游戏道具介绍和使用的信息，每个道具都有自己的特点，可以给玩家带来特效，让玩家可以有策略地使用道具。

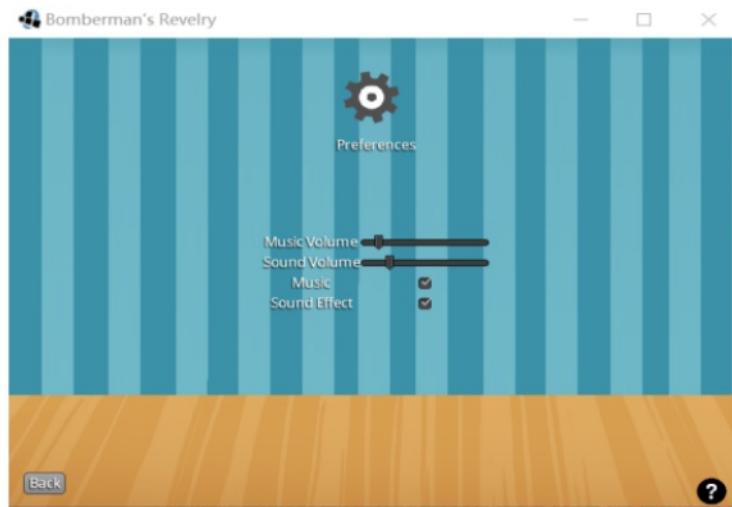


截图9:道具介绍画面

截图10描述了两种不同模式的游戏规则，玩家需要

了解这些规则，才能赢得这款游戏。

最后一个操作是音频管理，我们可以从下面的截图11中看到。玩家可以通过在偏好画面中左右滑动来调节背景音乐或游戏音效的音量，也可以在偏好画面中开启和关闭游戏音效或背景音乐。



截图11:游戏音效设置的画面

最后，在描述完游戏的特点、功能和游戏的界面设计之后，我将在下一章中对游戏系统进行建模，使其形象化并制定其结构。

4 问题分析与解决方案设计

4.1 软件架构

软件架构为开发一个软件提供了一个框架来处理它的复杂性和巨大的规模，它可以为我开发的游戏定义一个结构化的解决方案，以满足所有功能，非功能，技术和操作的要求。[13]

代码的组织将是混乱和混乱的，真的很难管理
在我的游戏中不使用任何架构的代码。因此，选择一个好的和
合适的建筑模式对我来说非常重要，它代表了我的设计

决定这个游戏的整体结构，这将显著决定它是否是一个成功的游戏。因此，在本章中，我将阐述我的游戏系统中使用的架构模式，并描述每个子组件的逻辑组织，以及它们在整个系统内如何相互工作。

一个架构模式是一个通用的，可重用的解决方案，在给定的上下文中软件架构中一个常见的问题我在我的整体游戏架构中应用了MVC，它是架构模式之一，它将游戏系统分为三个部分，包括模型、视图和控制器。这些模块分别执行不同的任务，我将通过下面这两张图展示结构并描述它们是如何一起工作的：

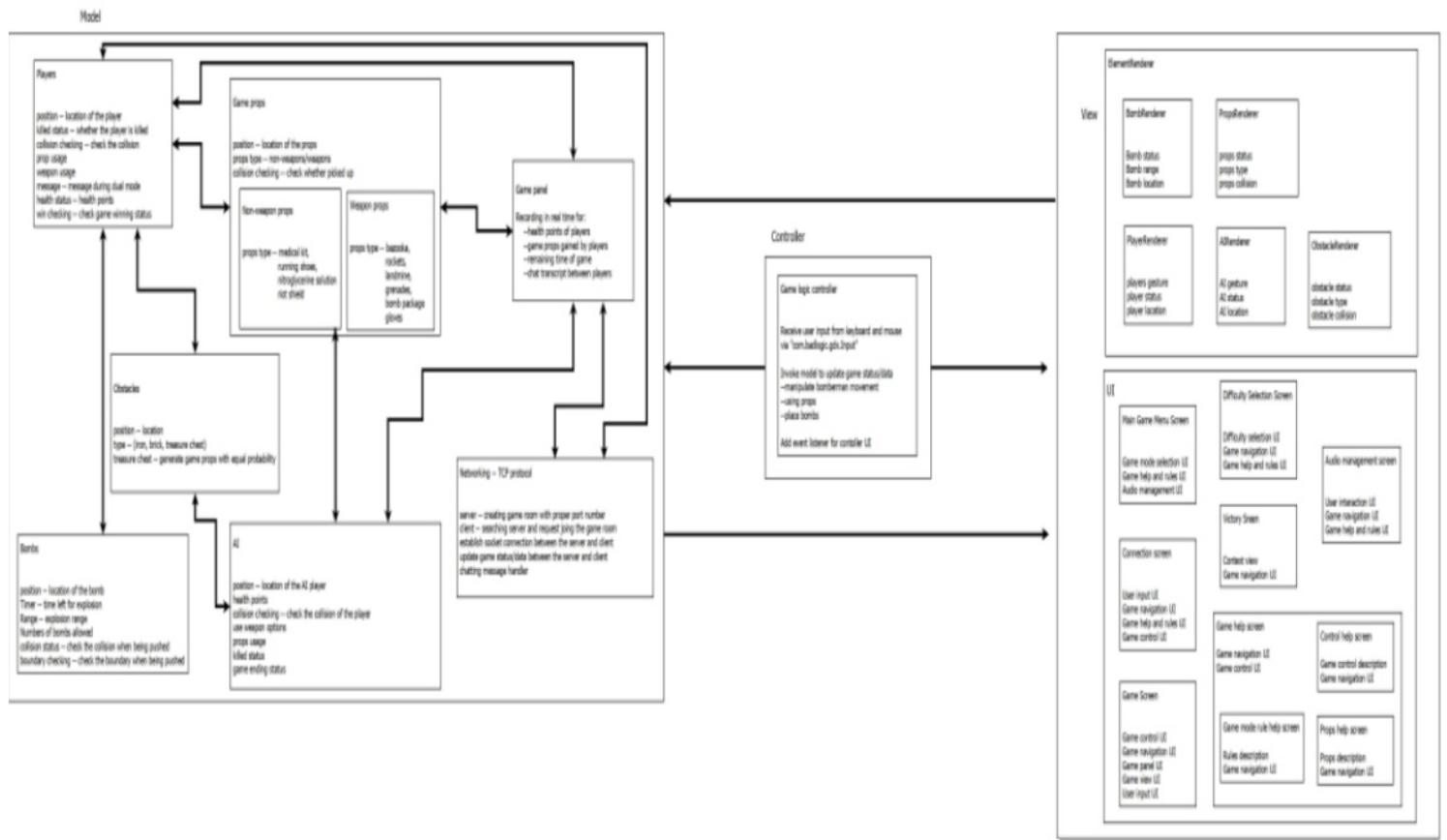


图1: 我的游戏中的MVC架构模式

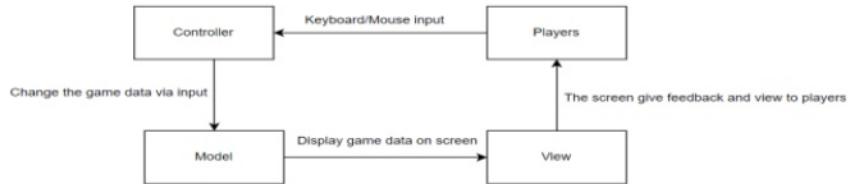


图2:MVC的工作流程

控制器用于通过添加事件监听器实时处理所有来自键盘和鼠标的用户输入，然后在模型中翻译和更改游戏数据，然后通过视图模块显示在屏幕上，例如玩家使用鼠标点击用户界面上的按钮，通过按相应的键操纵角色在屏幕上移动、使用道具或放置炸弹。

Model是用来管理我的游戏逻辑和数据的，当游戏数据发生变化时，它会将游戏数据传输到view模块并显示在屏幕上。它不仅定义了所有游戏元素及其属性，包括道具、角色、炸弹、障碍物，还说明了它们之间是如何关联或影响的。例如，砖块被摧毁，玩家因炸弹爆炸而失去生命值。另外，它定义了游戏中涵盖的所有功能，我举几个例子：1. 游戏面板用于记录关于生命值、游戏道具和剩余时间的信息。2. 用于实现两名玩家在线对战和聊天室的网络模块。

view可以用来将游戏显示在屏幕上，本质上指的是用户界面。它是用户交互的部分，当model的数据发生变化时，它将更新的内容呈现在屏幕上。例如，如果玩家在玩游戏时点击暂停按钮，游戏屏幕会迅速冻结。

MVC为我开发这个游戏提供了很多好处。首先，它将我的代码分成三个模块，它将是结构化和模块化的，这意味着它将非常容易管理和修改。我可以很容易地找到特定模块中对应的代码，以便在有bug的时候进行修改。比如，如果玩家移动太快，那么一定是模型模块中定义的玩家属性有问题。其次，我的代码会有很好的扩展性，方便添加新功能。每个模块只需要执行特定的任务，互不干扰，比如视图只需要处理游戏的显示，这样我只要在这个模块中创建一个游戏帮助屏幕如果我想添加一个新的用户界面来引入游戏规则，就不会影响整体结构。第三，它使代码变得可读、可理解，从而可以方便游戏开发，显著提高效率。

Libgdx框架集成了几个通用模块，为构建游戏架构的每一步提供服务。下面我将介绍核心模块和类

libraries that I used in my game development provided by libGDX framework.

4.2 核心Sub-module

4.2.1 图形

libGDX中的图形模块为图形开发提供了一套库，OpenGL API允许在屏幕上绘制图像。纹理类可以用来解码具有PNG格式的图像，并通过导入库“com.badlogic.gdx.graphics”将其加载到GPU内存中。Texture”，可以看作是一个实例化的纹理对象来表示一个图像。使用纹理后，需要使用texture.dispose()方法释放内存，否则会导致内存的泄漏。AssetManager类负责加载多种类型的资源，包括纹理、音乐、声音效果、字体等。我把所有的图像资源，包括背景，人物，障碍，游戏道具，爆炸到资产文件夹中，这样我就可以使用assetManager方法。由AssetManager类提供的load(fileName, type)来加载我所有的资源，给定资源的类型和文件名。使用AssetManager API来统一管理和加载我所有类型的资源是非常方便的，这为我节省了很多时间和精力。

```
private static <T> void LoadResource(字符串名称, 字符串fileName, 类<T>类型)assetManager。加载 {  
    (fileName, 类型);  
    assetNames。put (name, fileName);  
}
```

```
LoadResource (name: "door", fileName:"bricks/door.png", Texture.class);  
LoadResource(name: "brick", 文件名:"bricks/brick.png", Texture.class);  
LoadResource(name: "broken_brick_animation", 文件名 :"bricks/broken_brick_animation.png", Texture.class);LoadResource  
(name: "chest", 文件名:"bricks/chest.png", Texture.class);  
LoadResource(name: "iron", 文件名:"bricks/iron.png", Texture.class);
```

例如，通过调用assetManager.load("player/bomb_idle.png", Texture.class)方法加载带有PNG格式的图片“brick”作为纹理资源。

这些背景图片实例可以在加载这些图片资源后通过调用assetManager创建纹理对象来获得。为了方便游戏界面的开发，libGDX框架提供了许多常见的小部件，可以通过导入库“com.badlogic.gdx.scenes.scene2d”来使用。ui”，他们扩展了actor类，拥有了所有属性和功能，这样我就不用花时间去定义Myactor了。Image类提供了获取纹理对象后的图像缩放操作，例如，

我使用setSize()的方法来调整背景图片的大小，以便我可以将其添加到舞台上显示在游戏屏幕上。

```
public static <T> T get (string 资产名称, Class<T> type){返回assetManager.get(assetNames.get(资产名称), 类型);}
```

```
}
```

```
私有void createUIC {
    背景 = ResourceManager.get(资产名称 :"tutorial_bg1", Texture.class);背景 = 新纹理(Gdx.files.internal
    ("background/white.png"));

    Image 背景图像 = new Image(背景);
    背景图像.setSize(常数.SCREEN_WIDTH Constants, SCREEN_HEIGHT);stage.addActor(背景图像);
```

libGDX还提供了动画类，以显示人物的运动，武器道具的释放，爆炸效果。在背景中我已经描述了动画的基本概念以及它的工作原理，所以我将谈谈如何通过使用这个类来创建一个动画在屏幕上显示。动画是通过定义Animation <T>来创建的，其中T表示动画中帧的一般类型，我使用TextureRegion作为动画帧的对象，它需要提供两个参数来创建动画实例：具有特定类型的图像列表，也称为关键帧2。每一帧播放之间的时间间隔。下面是组成2D动画的操作，炸弹人向右移动。

首先，加载一个具有纹理类的sprite sheet来显示动画中的每一帧。其次，将这个精灵片分割成具有特定宽度和高度的小单元格，然后将它们存储在2D数组中，该数组具有TextureRegion类型。第三，将存储在2D数组中的所有单元格放入1D数组中，作为创建动画时要播放的关键帧。最后，通过创建构造函数，使用这些关键帧和播放每帧的时间0.15秒来创建动画实例，以便此动画将显示在屏幕上。释放武器或爆炸效果的动画使用相同的技术来创建。

```
纹理 spritesheet = ResourceManager.get(spritesheetPrefix, Texture.class) TextureRegion[][]
frames = TextureRegion.split(spriteSheet, spriteSheet.getWidth() / FRAMES, spriteSheet.getHeight()
() / TextureRegion[] rightFrames = new TextureRegion[FRAMES];
for (int i = 8; i < FRAMES; i++)
    rightFrames[i] = frames[e][i];
```

```
}
```

```
右化: new Animation(TextureRegion>(frameDuration 8.15f, rightFrames);
```

4.2.2 音频

libGDX中的音频模块提供了音频资源的管理和处理，包括音乐和音效，支持mp3, ogg, wav三种不同的格式。音乐类负责处理较长片段的背景音乐，声音类通常处理较短片段的各种音效(爆炸，放置炸弹，使用道具)在游戏中，包括玩，

暂停和重放。加载音频资源的过程与加载图片资源的过程相同，首先我将所有的音频文件放入assets文件夹中，assetManager类也可以通过创建相应的实例来读取和加载这些音频资源：

```
LoadResource(name: "pLaceBomb", 文件名:"sounds/pLaceBomb.wav", Sound.cLass);LoadResource(name: "expLore",
文件名:"sounds/explorer.wav", sound.cLass);LoadResource(name: "pLaceMine")。文件名:"sounds/pLaceMine.mp3",
Sound.cLass);

LoadResource(name: "bg_main_001" 文件名: "music /bg_main_001.wav", Music.cLass);LoadResource(name: "bg_ad_001").
文件名:"music /bg_ad_001.wav", Music.cLass);LoadResource(name: "victory", 文件名:"music /victory.wav", Music.cLass);
LoadResource(name: "Lose", 文件名:"music /Lose.wav", Music.cLass);
```

这些实例可以通过调用assetManager加载音频资源后创建音乐和声音对象来获得。Get(资产名称，类型)。

```
public static <T> T get (String assetName, Class<T> type) {
    return assetManager.get(assetNames.get(assetName), type);
}

mainMenuTheme = ResourceManager.get(assetName: "bg_main_001", Music.class);
gameTheme = ResourceManager.get(assetName: "bg_ad_001", Music.class);

victory = ResourceManager.get(assetName: "victory", Music.class);
lose = ResourceManager.get(assetName: "lose", Music.class);

placeBomb = ResourceManager.get(assetName: "placeBomb", Sound.class);
placeMine = ResourceManager.get(assetName: "placeMine", Sound.class);
explore = ResourceManager.get(assetName: "explore", Sound.class);
```

assetManager管理的所有资源都需要暂停资源的加载，并通过assetManager.update()方法保存加载状态：

```
公共静态布尔更新(){返回assetManager.update();}
```

在加载所有音频资源后，music.play()和music.stop()方法用于播放特定的背景音乐，并在手动调用时停止播放，它允许在音乐结束或暂停时重新启动或恢复音乐。音乐。setVolume(浮动音量)允许设置背景音乐、音乐的范围。setLooping(布尔值)用来循环播放背景音乐。

```

public void playMusic(String name, boolean looping) {
    if (musicEnabled) {
        Music music = musics.get(name);

        if (music != null && !music.isPlaying()) {
            music.setVolume(musicVolume);
            music.setLooping(looping);
            music.play();
        }
    }
}

public void stopMusic(String name) {
    Music music = musics.get(name);

    if (music != null) {
        music.stop();
    }
}

```

4.2.3 输入

在我的桌面游戏开发中，libGDX提供了输入模块来处理键盘和鼠标的输入。GDX提供了几种方法。Input根据用户的输入给出响应，判断用户是用鼠标点击还是按下键盘上的某个键。例如，方法isKeyPressed(keys.SPACE)返回一个布尔值来表示玩家是否按下空格键，如果该值为真，由玩家操作的字符将放置炸弹。

如果股票指型基金(简称eft)。输入。isKeyPressed(键。4 . spritmanager . player . placebomb(炸弹人。

NetworkType.LOCAL);spriteManager.computePowerupRate ();



在本例中，我使用addListener(通过导入“com.badlogic.gdx.scenes.scene2d”向设置按钮添加事件侦听器)。Actor”，它在监听器接收到特定事件后，将输入转换为另一个不同的事件。ClickListenerO和touchUPO方法在玩家点击鼠标左键点击设置按钮时被调用，然后游戏画面会切换到偏好画面，这样用户就可以在不退出当前游戏的情况下调整背景音乐和音效的音量。

设置按钮。setPosition(x常量。屏幕宽度- 50.0f, y: 0.0f);
设置按钮。setSize(宽度:40.0f, 高度:40.0f);

设置按钮。addListener(new ClickListener() {
 public void touchUp(InputEvent event, float x, float y, int pointer, int button)
 {
 handle();
 game.setScreen(new PreferencesScreen());
 }
});

4.3 核心子系统

4.3.1 AI系统

AI是我的游戏中最关键的部分之一，我们可以通过应用数据结构算法创造一个聪明的敌人，这让玩家觉得它是被另一个玩家操纵的。智能AI代理将在增强玩家的游戏体验方面发挥重要作用，他们会不断寻找宝盒以收集更多的游戏道具来增强自己的属性，并使用武器攻击玩家。在这个过程中，他们还可以避免放置炸弹爆炸的伤害，理性地使用武器道具，玩家需要与他们对战，击杀所有敌人才能在离线模式下赢得这场游戏，这使得游戏变得更加具有挑战性和趣味性。

4.3.1.1 寻径:广度优先搜索算法

AI通常可以通过游戏中的各种搜索算法执行特定的命令。在我的游戏中，AI敌人通过广度优先搜索算法执行寻找游戏道具或宝箱的任务，它从一个初始节点开始，然后搜索相同深度的所有相邻节点，然后扩展到下一个深度，该算法将继续搜索节点，直到找到目标节点，并返回AI向目标节点移动的路径解，如果没有其他节点可以扩展，则结束。这个寻路问题可以表述为4个部分：

初始状态:AI的当前坐标。

搜索空间:空网格的所有坐标。

目标节点:可破坏的障碍物或游戏道具的坐标。路径解:坐标数组。

AI首先定位当前坐标作为初始节点，然后按右、下、左、上的顺序探索初始节点附近的所有邻近坐标节点。之后，扩展所有这些邻域节点，重复这个步骤，直到探索到一个不可摧毁的障碍物，然后停止扩展当前节点，开始向其他三个方向扩展邻域节点，以此类推，直到找到一个可摧毁的障碍物或游戏道具。最后，返回一个可以到达目标节点的所有坐标的数组，作为AI的路径解，以便它们可以向目标位置移动，放置炸弹或拾取游戏道具。在我设计的游戏中

地图，在这个迷宫中每个障碍的放置是非常合理的，玩家或AI不会被四个方向的四个坚不可摧的障碍包围，并被困在特定的区域。因此，永远不会出现没有更多节点可以扩展，算法找不到路径解的情况，除非不再有可摧毁的障碍物或游戏道具。我之所以使用BFS算法，是因为该算法考虑了前一个节点附近的所有相邻节点，适合于寻找从原点到目标节点的最短路径，保证了AI可以用更少的时间或成本从当前位置找到到可破坏障碍物的最短路径，并快速收集游戏道具以增强属性或准备使用武器攻击玩家。当AI收集到足够的游戏道具时，他们将很难打败他们。

我们可以看到下面的伪代码，BFS应用队列的数据结构来存储和探索坐标信息，新的坐标节点将从队列的末尾插入，已经扩展的节点将从队列的开始处移除，有一个数组记录当前坐标节点之前是否被访问过，如果节点被访问过，则不会插入队列中进行探索。如果发现了可破坏的障碍物或游戏道具，这个算法将结束并返回一个路径解决方案，这是这个寻路问题的路径解决方案。

BFS算法的伪代码

1. input: Given the map spriteManager.map, current coordinate(x0, y0), target coordinate(x1, y1)
2. output: path <- Array<Vector2>
3. Procedure: BFS(x0, y0, x1, y1):
4. Define queue <- Node array
5. Add the current node into the tail of the queue:
queue[tail] := (x0, y0, parent, step)
6. visited[x0][y0] = True
7. tail := tail + 1
8. let flag := 0 label for stop criteria
9. While queue is not empty: tail > head
10. Loop: search four directions(up, right, down, left)
11. if out of the map then continue; end if
12. if the next step can be reached and is not visited then
13. Set visited[tx][ty] := 1 where (tx, ty) denotes the step coordinate
14. Add the step to the tail of the queue:
queue[tail] := (tx, ty, parent, step + 1)
15. Update the tail: tail = tail + 1
End if
16. if find the target (x1, y1) then set flag := 1; break; end if
17. End Loop.
18. if flag is equal to 1 then break; end if
19. Dequeue: set head := head + 1
20. End

4.3.1.2 整体工作流程

AI还需要在找到通往可破坏障碍物或游戏道具的路径后执行其他任务，他们需要在可破坏障碍物附近放置炸弹以摧毁它们，以便从宝箱中获得掉落的游戏道具。在这个过程中，AI应该避免放置炸弹的爆炸，这意味着他们必须找到不会爆炸的安全区域，直到炸弹爆炸，然后不断寻找新的路径来获得游戏道具。在这个过程中，绝对不会出现AI在放置炸弹后，因为炸弹爆炸并呈“十字”形向四个方向扩散而找不到安全区域的情况，也就是说，无论放置炸弹的爆炸范围有多大，人工智能只需要移动到放置炸弹的对角线上就可以绕过爆炸区域。尽管AI只能在同一方向上放置炸弹，但我将炸弹在一个方向上传播的最大爆炸半径设置为4格，以便AI在放置炸弹后有足够的时间向前移动超过4格以远离爆炸区域。综上所述，AI的状态会从寻找路径到放置炸弹，再到寻找安全区域，然后是等待状态。我们可以看到下面的状态机图，它说明了人工智能系统的所有状态：

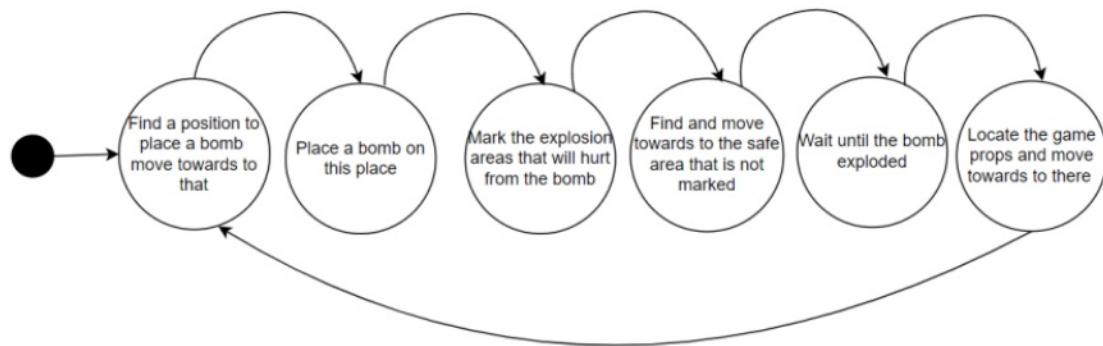


图1:状态机图

活动图是状态机的一种特殊情况，它强调AI系统行为的顺序性和并发性。人工智能在按顺序执行这些任务时，如果受到炸弹或武器道具的伤害，就会失去生命值或死亡，这个过程是同时进行的。我们可以看到下面的活动图：

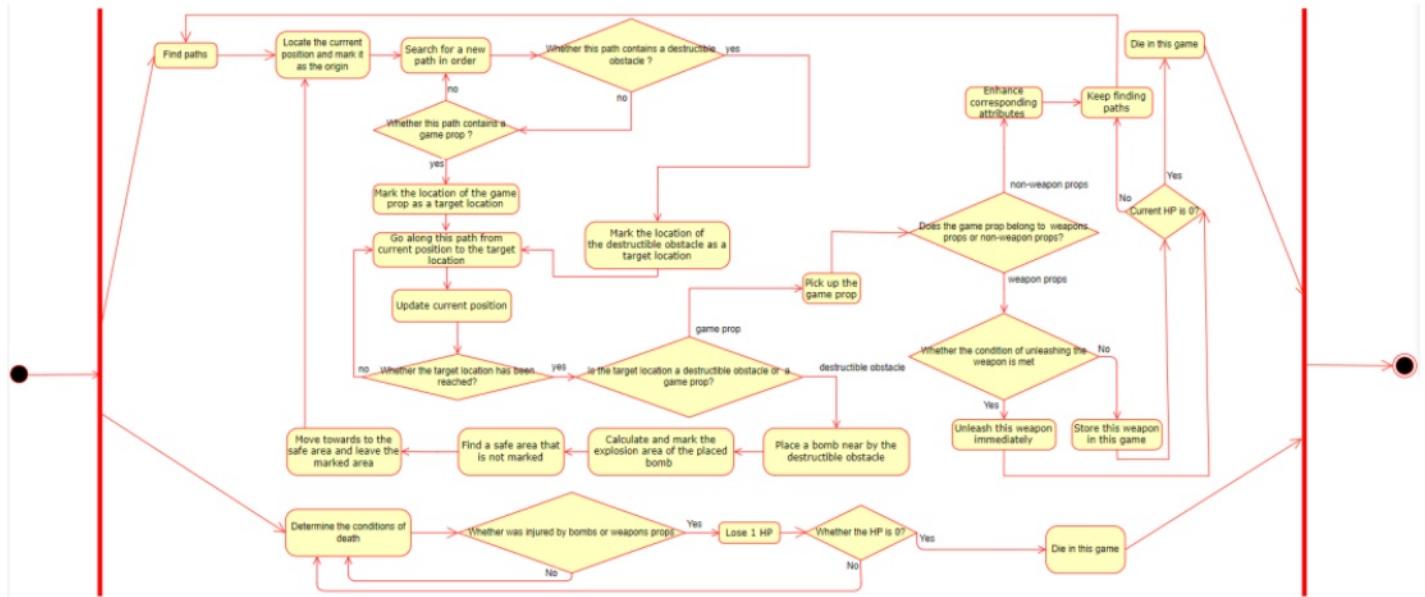


图2:活动图

该图展示了AI系统的整体工作流程，描述了他们的决策过程。考虑到AI在使用武器道具时可能会伤害到自己的问题，我为人工智能添加了每个道具的释放条件，使他们能够合理地使用所有武器。

1. 火箭筒:火箭筒发射火箭弹的爆炸范围为1格，AI根据发射火箭弹的移动速度和AI的移动速度，只有在前方前两格没有障碍物的情况下才能发射火箭弹。
2. 地雷:人工智能可以记录下地雷放置后的坐标，并绕过这个区域，虽然看不见，但如果只有一条路径，它们会向放置地雷的相反方向移动。
3. 炸弹包:AI只能在可破坏障碍物附近放置多个炸弹，在一个方向放置多个炸弹后，它们会移动到炸弹的对角线上，以防止爆炸伤害。
4. 手套:AI只能在前4格没有障碍物的情况下推动炸弹，因为炸弹的最大爆炸半径可以在一个方向上扩展为4格。
5. 手榴弹:手榴弹的爆炸范围以3 * 3的着陆点为中心是9格，所以人工智能必须在第二个网格上投掷手榴弹，这样他们才不会伤到自己。而且，他们只有在第二格上没有障碍物的情况下才能扔出手榴弹。

AI只会在满足特定条件时释放这些武器，这使得AI更难对付。这个游戏将会变得更具挑战性和趣味性。

4.3.2 网络系统

4.3.2.1 网络架构:对等架构

我在我的游戏的网络部分中应用了对等架构，每个对等点或节点都有完全相同的责任，并且都可以作为提供服务的服务器或发送请求的客户端。创建房间的玩家将代表一个服务器来监听和接受传入的请求，因此另一个玩家将作为客户端向服务器发送加入房间的请求，当请求被接受时游戏将开始。这个机制非常灵活，玩家不必在每次游戏开始前都扮演特定的角色。

我使用这个架构的主要原因是它的可靠性高，在线模式下服务器崩溃时客户端不会受到影响，仍然可以正常工作，这保证了玩家在克服网络不稳定和断网的情况下有很好的用户体验。我们可以看到架构图：

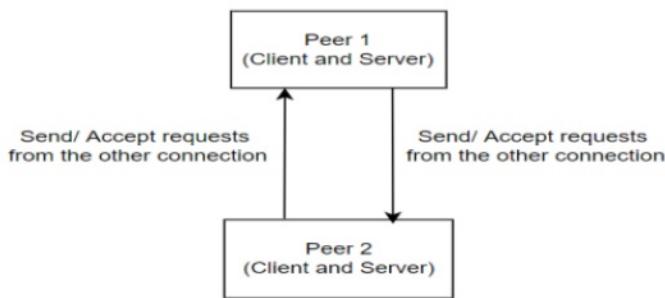


图4:对等架构图

4.3.2.2 网络通信框架:二人之战

球员

我的游戏中的双人模式支持两名玩家在同一个局域网上相互战斗。我使用了java库提供的网络通信框架k \otimes yoNet，它基于TCP和UDP连接协议，非常适合应用client/server和peer-to-peer架构的游戏网络部分，因为它允许序列化对象并转换为字符串格式以存储在内存中，然后在服务器和客户端之间快速传输游戏数据，从而保证了两个玩家之间的游戏状态或状态可以同步。让我谈谈如何在我的游戏中实现这种在线模式。

首先，如果其中一个玩家想在双人模式下创建一个房间，它将建立一个服务器，给定TCP端口号为8888, UDP端口号为7777

通过导入“com.esotericsoftware.kryonet.Server”并创建服务器对象，如果它没有被创建，将会出现I/O异常。

```
game.server = new Server();

try {
    game.server.bind(tcpPort: 8888, udpPort: 7777);
} catch (IOException e) {
    e.printStackTrace();
}

Kryo kryo = game.server.getKryo();
kryo.register(MessageProtocol.class);
kryo.register(EnterProtocol.class);
kryo.register(ConnectProtocol.class);
kryo.register(MoveProtocol.class);
kryo.register(PlaceBombProtocol.class);
kryo.register(LaunchRocketProtocol.class);
kryo.register(ThrowGrenadeProtocol.class);
kryo.register(PlaceMineProtocol.class);
kryo.register(PauseGameProtocol.class);
kryo.register(ExitGameProtocol.class);
kryo.register(Vector2.class);

game.server.start();
```

其次，另一个玩家应该通过导入“com.esotericsoftware.kryonet.Client”创建客户端来加入这个房间，并创建一个客户端对象。

有必要分别获得服务器和客户端的Kryo实例，然后在启动服务器和客户端之前注册这些用于相互传输数据的协议类。需要注意的是，客户端和服务器中的协议类必须以相同的顺序注册。

```
game.client = new Client();

Kryo kryo = game.client.getKryo();
kryo.register(MessageProtocol.class);
kryo.register(EnterProtocol.class);
kryo.register(ConnectProtocol.class);
kryo.register(MoveProtocol.class);
kryo.register(PlaceBombProtocol.class);
kryo.register(LaunchRocketProtocol.class);
kryo.register(ThrowGrenadeProtocol.class);
kryo.register(PlaceMineProtocol.class);
kryo.register(PauseGameProtocol.class);
kryo.register(ExitGameProtocol.class);
kryo.register(Vector2.class);

game.client.start();
```

第三，通过给它添加一个事件监听器，使服务器可以接收和处理来自客户端发送的请求。

```
game.server.addListener(new Listener() {
    public void received(Connection connection, Object object) {
        if (object instanceof ConnectProtocol) {
            gameReady = true;
            game.clientConnectionId = connection.getID();
            System.out.println(connection.getEndPoint());
            game.server.removeListener(this);
        }
    }
});
```

第四，输入主机服务器的IP地址后，通过服务器绑定的端口号建立客户端与服务器之间的连接。我将连接的最大超时时间为1000毫秒，如果它们之间的连接超时，将会有个I/O异常提示连接失败。

```
connect.addListener((ChangeListener) (event, actor) -> {
    try {
        game.client.connect(timeout: 1000, ipAddress.getText(), tcpPort: 8888, udpPort: 7777);
        game.networkType = Bomberman.NetworkType.REMOTE;
        gameReady = true;
    } catch (IOException e) {
        e.printStackTrace();
    }
});
```

最后，当客户端准备好使用sendTCP方法启动游戏时，客户端将通过TCP将连接协议对象发送到服务器。当服务器接受请求时，游戏屏幕将显示。

```
Timer.schedule(() -> {
    if (gameReady) {
        ConnectProtocol connectProtocol = new ConnectProtocol();
        game.client.sendTCP(connectProtocol);
        Timer.schedule(() -> {
            game.changeScreen(Bomberman.GAME);
            cancel();
        }, delaySeconds: 1.0f);
        cancel();
    }
}, delaySeconds: 0.0f, intervalSeconds: 1.0f);
```

4.3.2.3 网络技术:聊天室的功能

在双人模式下聊天室的实现方面，1使用libgdx提供的网络模块和多线程技术来处理消息。这些是实现这一功能的网络操作：

1. 创建一个服务器套接字，用于通过TCP接收和接受给定端口号的客户端发送的所有请求。
2. 在给定主机服务器的IP地址和通过TCP的特定端口号的情况下，创建用于与服务器建立连接的客户端套接字。
3. 创建输入流和输出流，通过套接字读取数据和写入数据，这是发送或接收消息的过程。

有必要创建两个线程来处理服务器和客户端之间的实时消息接收和发送，因为每个线程都可以表示一个顺序的工作流程，并同时执行它们的特定任务，这样就会提高从另一个连接接收消息的响应时间和效率，这就确保了当它们相互发送消息时，消息可以同步和快速地显示在它们的屏幕上。此外，每个线程将独立运行，互不干扰，玩家可以随时互相聊天而不影响游戏状态的更新和同步，这保证了他们可以顺畅地相互交互。这也是我使用多线程技术来实现这个过程的原因。

我画了一个顺序图来清晰地描述双玩家模式下网络连接的整个过程，强调了物体在网络上交互的顺序，如下图所示：

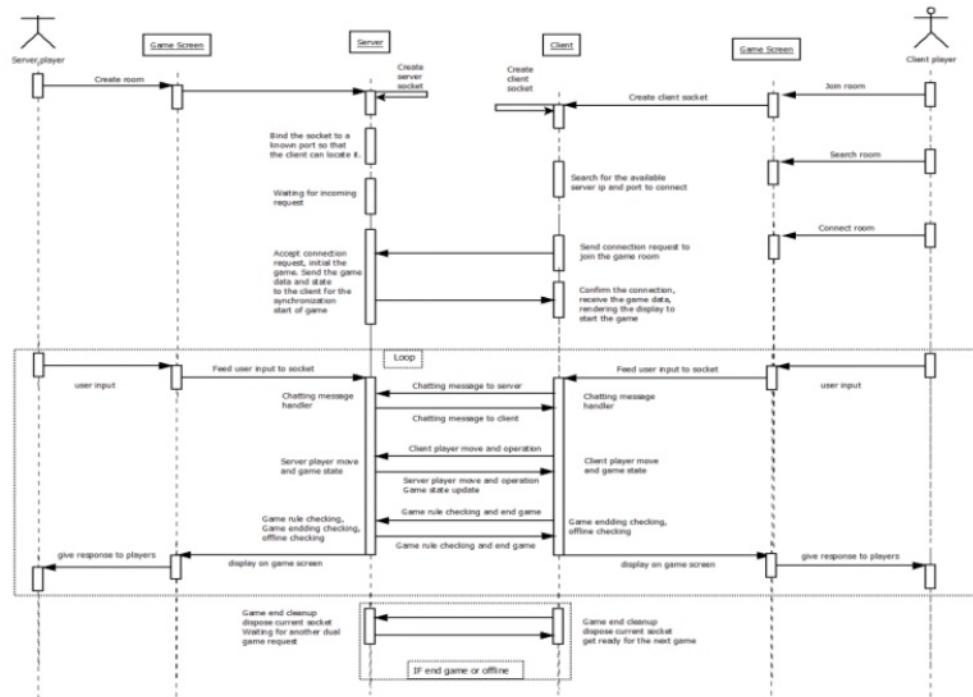


图5:顺序图

我之所以在网络中使用TCP协议，是因为它是一种适合实现小型网络游戏的协议，尤其是我的游戏中的双人对战和聊天室功能。它可以将大数据分割成合适的数据块或数据包，以便在网络上传输，在在线模式下数据包丢失并导致网络中断的概率很小，这保证了游戏数据和消息能够在服务器和客户端之间稳定、完整地传输。因此，它非常可靠，为玩家提供了良好的游戏体验。

5 验证与评价

测试可以验证我的游戏是否能够按照预期工作并满足我的要求，无论所有的功能是否都实现了，它都可以帮助我识别并发现游戏中潜在的问题或漏洞，我需要在游戏中进行修改，这保证了游戏的质量和可靠性。评估可以帮助我回顾游戏开发过程中哪些地方是好的，哪些地方需要改进，一些偏离最初计划的地方，为以后的职业生涯积累了工作经验。我将在下面描述游戏中使用的测试策略以及它的工作原理，然后是对我的项目的评估。

5.1 软件测试

5.1.1 黑盒测试

游戏系统被视为一个在测试执行时无法打开的黑盒，我应用了黑盒测试来检查游戏的用户界面和功能，这个测试可以帮助我检查这个游戏产品是否可以实现项目建议书中定义的这些目标，并识别缺失或错误的功能。测试是从玩家的角度进行的，检查用户从键盘和鼠标的输入是否与游戏系统输出相匹配，这样测试人员就不需要考虑游戏内部系统是如何工作的，黑盒输出的这些测试结果非常直观，通过输入和输出的差异，我很容易识别出我的游戏中存在的潜在问题。比如游戏画面显示玩家发射的火箭飞出了游戏地图的边界，在模型模块内的“火箭”类中一定缺乏对边界外的逻辑判断。

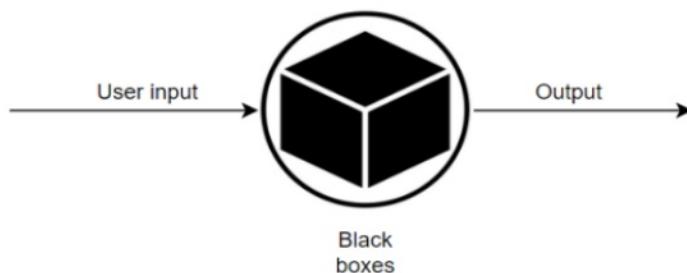


图6:黑盒测试

我在开始测试之前，根据设计部分中提到的功能需求定义测试用例，然后在测试过程中检查它们是否都通过了。幸运的是，经过无数次的测试运行和调试修改代码后，所有的测试用例都通过了，这确保了这款游戏和我预想的一样好。我会把每个测试用例的结果放在附录里。

5.1.2 可用性测试

可用性在游戏开发中起着不可或缺的作用，它在很大程度上决定了这款游戏的用户体验，游戏中的一个小故障或问题可能会带来糟糕的用户体验并导致玩家对这款游戏失去兴趣或耐心，他们最终可能会放弃玩这款游戏，不管它的玩法和戒律多么吸引人。评估游戏可用性的方法和其他应用是一样的，通过可用性测试来评估真实用户使用这款软件产品的容易程度，通过观察和记录用户在执行研究人员分配的特定任务时的行为，它帮助我通过分析这些来自用户的直观数据反馈来识别我的游戏中存在的问题或设计缺陷。它也是从玩家的角度来进行的，所以这种方法是最有效和最具代表性的。因此，这就是我将可用性测试应用于我的游戏的原因。我既是游戏开发者，也是这次测试的研究员，所以我将描述一下我所做的具体设计决策。

1. 找到具体的测试参与者

我希望我的游戏仍然能够为每个细分的受众带来良好的用户体验。所以我特地邀请了6位来自不同年龄段的当地朋友来参加这次测试玩原型游戏，他们会代表不同的目标群体，分别是10-17、18-25、26-40、41-54、55-64、65以上。

2. 为参与者设计并定义了收集问题的测试任务

我根据游戏的功能为他们设计了不同的任务，希望他们尝试独自完成这些任务，而不需要向我求助。在进行测试之前，我向他们解释了我的游戏规则以及如何玩，这让他们对这个游戏有了一个直观的印象。在进行测试时，我必须密切关注他们在屏幕上的动作，我会问为什么当他们开始执行与预期相反的任务时，会这样做，以便了解他们的想法，并确定可能需要改进的问题。我总结了他们在测试中总共面临的一个潜在问题，并将其放在附录b中。以下是我游戏中不同部分的任务定义：

为测试者设计的任务

● 音频:

- (1) 打开设置画面
- (2) 打开和关闭背景音乐
- (3) 调高或调低背景音乐
- (4) 调高和打开游戏中的音效
- (5) 调高游戏音效

● 游戏帮助和说明

- (1) 打开游戏帮助画面
- (2) 打开道具描述画面
- (3) 打开游戏规则画面
- (4) 打开障碍信息画面

● 游戏逻辑:

- (1) 单人模式中依次选择easy level、medium level、梦魇level.
- (2) 操纵炸弹人上、下、左、右移动
- (3) 操纵炸弹人在障碍物附近放置炸弹
- (4) 操纵炸弹人拾取道具
- (5) 操纵炸弹人使用火箭筒发射火箭
- (6) 操纵炸弹人埋设地雷
- (7) 利用手套操纵炸弹人推炸弹
- (8) 在得到一个炸弹包裹后，操纵炸弹人放置两个炸弹
- (9) 操纵投弹手向障碍物后投掷一枚手榴弹
- (10) 暂停并恢复游戏
- (11) 在游戏过程中查看游戏帮助
- (12) 游戏过程中退出当前游戏返回主菜单
- (13) 打开决斗模式画面
- (14) 在决斗模式中创建一个房间等待连接
- (15) 以双机模式查找邻居主机服务器的IP地址
- (16) 在文本框中输入需要搜索的IP地址
- (17) 加入一个房间开始决斗模式
- (18) 在文本框中输入消息，以决斗模式发送

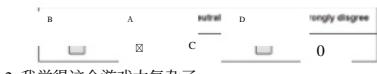
3. 可用性评估

测试完成后，使用的系统可用性量表从有效性、效率和用户满意度3个方面来衡量my game产品的可用性。这是一份SUS问卷，其中包含了10个问题让他们完成，以便获得直接和快速的反馈。^[15]本文提出的系统可用性量表的评价方法是基于Andrew(2020)的。他们有不超过2分钟的时间

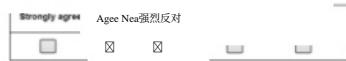
完成并根据他们的问卷结果通过李克特5分量表计算得分，每个问题的每个回答代表不同的分数。如下图所示，是我的问卷模板：

可用性量表问题

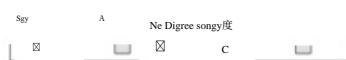
1. 我想我会经常玩这个游戏。



2. 我觉得这个游戏太复杂了。



3. 我认为游戏软件很好用。



4. 我想我需要一个技术人员的支持才能使用这个游戏产品。



5. 我发现这款游戏中的各种功能都整合得很好。



6. 我觉得这个游戏产品的不一致性太多了。



7. 我想大多数人会很快学会使用这款游戏产品。



8. 我发现这个游戏产品使用起来很麻烦。



9. 使用这款游戏产品，我觉得非常有信心。



10. 在开发这款游戏之前，我需要学习很多东西。



Response results	Usability Scores
Strongly agree	5 points
Agree	4 points
Neutral	3 points
Disagree	2 points
Strongly disagree	1 points

图7:可用性量表问题模板

我收集了所有6人的全部问卷结果，并按照以下标准计算出相应的SUS得分[15]:

1. 计算X:将所有奇数题的分数相加，total减去5，得到X。
2. 计算Y:把所有偶数题的分数加起来，用25减去这个总分。
3. SUS分数:计算X + Y并将总和乘以2.5得到SUS分数。 [15]

Testers/Scores	Q1/pt	Q2/pt	Q3/pt	Q4/pt	Q5/pt	Q6/pt	Q7/pt	Q8/pt	Q9/pt	Q10/pt	X/pt	Y/pt	SUS Scores	Average SUS Scores
10-17	5	1	5	1	4	1	5	1	5	1	19	20	97.5	
18-25	5	2	5	1	5	2	4	1	5	1	19	18	92.5	
26-40	4	1	5	2	5	1	5	1	4	2	18	18	90.0	(97.5+92.5+90.0+87.5+77.5+70.0)/6 ≈ 85.8
41-54	4	2	4	2	4	1	5	1	5	1	17	18	87.5	
55-64	3	2	5	1	4	2	3	1	4	2	14	17	77.5	
More than 65	3	1	4	2	5	2	4	2	3	4	14	14	70.0	

表8:可用性量表得分结果

从表中我们可以看到一个现象，年龄越大的受众，他们的SUS得分越低，这意味着由于不同年龄段的思维模式不同，他们不能很容易地使用这款游戏产品。代表这6个不同年龄段的6位参与者的平均SUS得分为85.8分，在SUS得分的评分图式中属于优秀类别[15]，这意味着大多数用户群体对我的游戏产品比较满意和认可，我的产品在功能和用户界面设计上没有较大的缺陷。这种方法是有效使用的，并通过直观的分数数据展示了my游戏产品的可用性。

5.2 评价

总的来说，我的项目基本符合我最初的预期，几乎所有计划中定义的功能都已经实现，在进行黑盒测试后没有出现严重的问题或故障，所以我对我现在的游戏产品非常满意。唯一的小缺陷是，AI无法实时检测并避免玩家放置的炸弹爆炸造成的伤害，这意味着AI并不像你想象的那样难以被杀死。不幸的是，我不能将这个功能添加到AI中，因为它会影响它的状态变化，因为它们必须不断地按顺序执行任务，如果AI可以躲避炸弹，它们将无法继续执行其他任务，所以它教会了我在编写程序时最好确保你的代码具有可扩展性和适应性。

但是开发过程并不顺利，特别是AI的设计，我花了很长时间调试和修改。除了在AI中应用数据结构算法寻找路径外，还需要计算和标记放置炸弹的爆炸区域，以避免爆炸伤害他们。在这个过程中，他们也可以使用所有武器道具而不伤害自己，所以你必须为AI添加每种武器的释放条件。此外，你需要为每个游戏道具编写逻辑fon，并将它们放在一起测试，以确保它们可以正常使用，道具也会相互影响，因为它们具有不同的功能，所以真的很难处理它们之间的逻辑。这个游戏中网络通信的部分对实时性要求很高，所以很难处理网络不稳定带来的延迟。

6 总结

我的目标是重新打造一个具有更多创新玩法的改良版《炸弹人》游戏，看来我已经实现了这个目标。在游戏内容上，我的游戏既有独特的游戏规则，也有丰富的玩法，包括各种游戏道具，智能的AI敌人，多种游戏难度和模式，大大增加了我的游戏乐趣。绝大多数用户群体对我的游戏产品感到满意，在完成可用性测试后无需任何教学就可以轻松使用，他们都认为这是一款享受且有趣的游戏。

由于游戏框架具有跨平台的特点，如果我有更多的时间，我会继续为其他平台开发这款游戏的其他版本。

参考书目

- [1] McFerran, Damien (2009). Hudson Profile - Part 2 (RG). Issue 67. Retro Gamen Magazine. pp. 44-49. Retrieved January 19, 2011
- [2] Fandom (2011). Bomberman Wiki. [online] Available at: https://bomberman.fandom.com/wiki/Main_Page [Accessed 26 Dec. 2021].
- [3] Hudson Soft Company Ltd (2020). Bomberman download. [online] Available at: <https://www.oldgames.sk/en/game/bomberman/translations/> [Accessed 26 Dec. 2021].
- [4] Play Meter (1994). Equipment Poll - Video & Pinball Combined. Vol. 20, no. 3. Skybird Publishing. p. 8.
- [5][6] Gaming-History (1992). Bomber Man World. [online] Available at: <https://www.arcade-history.com/?n=bomber-man-world&page-detail&id=309> [Accessed 27 Dec. 2021].
- [7] Martin Eden (2020). The Essentials of a Common Game Development Framework. [online] Available at: <https://meliorgames.com/game-development/the-essentials-of-a-common-game-development-framework/> [Accessed 1 Jan. 2022].
- [8] Wikipedia (2022). Game framework. [online] Available at: https://wiki.freepascal.org/Game_framework#See_also [Accessed 1 Jan. 2022].
- [9] libGDX (2022). Goals and Features. [online] Available at: <https://ibgdx.com/> [Accessed 2 Jan 2022].
- [10] libGDX (2021). 2D Animation. [online] Available at: <https://www.bookstack.cn/read/libgdx-1.10-en/c8601e303e116d6d.md> [Accessed 5 Jan 2022].
- [11] Teschner, M.; Kimmerle, S.; Heidelberger, B.; Zachmann, G.; Raghupathi, L.; Fuhrmann, A.; Cani, M.-P.; Faure, F.; Magnenat-Thalmann, N.; Strasser, W.; Volino, P. (2005). Collision Detection for Deformable Objects. Computer Graphics Forum. 24: 61-81. doi:10.1111/j.1467-8659.2005.00829.x. S2CID 1359430.
- [12] Valve Developer Community (2012). Hitbox. [online] Available at: <https://developer.valvesoftware.com/wiki/Hitbox> [Accessed 10 Jan 2022].
- [13] Tutorialspoint (2013). Software Architecture & Design Introduction. [online] Available at: https://www.tutorialspoint.com/software_architecture_design/introduction.htm [Accessed 2 Feb 2022].

[14] N. Medvidovic and R. N. Taylor (2010). Software architecture: foundations, theory, and practice. 2010 ACM/IEEE 32nd International Conference on Software Engineering, pp. 471-472, doi: 10.1145/1810295.1810435.

[15] S. Andrew (2020). The System Usability Scale & How It's Used in UX. [online] Available at:

<https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/> [Accessed 20 Mar 2022].

附录A

测试用例的结果:

1. 障碍测试用例

- 铁块不能被炸弹或武器摧毁。通行证砖块可以被炸弹或武器摧毁。
- Pass宝箱可以被炸弹或武器摧毁。传递摧毁宝箱后掉落的游戏道具。
- 通过
-
-

2. 炸弹的测试案例

- 放置的炸弹会在3秒后引爆。通过
- 炸弹可以杀死玩家和AI。通过
- 玩家和AI不能通过炸弹。通过炸弹的爆炸范围可以增加和积累。通过
- 通过
- 三秒内放置炸弹的数量可以增加和积累。通过
- 炸弹的扩散范围应该被障碍物挡住。通过
- 炸弹可以由AI或玩家使用非武器道具手套来推。通过
- 被推的炸弹如果碰到障碍物就会爆炸。通过
- 炸弹不能推到地图边界之外。通过
-

3. 玩家测试用例

- 玩家可以使用方向键操纵轰炸机在四个方向(上、下、左、右)上移动。通过
- 玩家可以按空格键放置炸弹。通过
- 玩家可以捡起任何掉落的游戏道具。通过
- 玩家可以通过按数字键1、2、3和4使用4种不同的武器。通过
- 玩家可能会被炸弹或武器杀死。通过
- 玩家不应该通过障碍和炸弹。通过
- 玩家不能走出地图的边界。通过
- 玩家可以在游戏过程中随时点击相应的按钮暂停或继续游戏。通过
- 玩家可以在游戏过程中随时点击相应的按钮查看游戏规则或帮助。通过
- 玩家可以在游戏过程中随时退出并返回游戏主菜单。通过
- 在决斗模式中，玩家可以随时通过点击“发送”按钮向对方发送消息。通过
- 玩家可以在决斗模式中随时接收消息。通过
如果受到炸弹或武器的伤害，玩家就会失去生命值。在单人模式中，如果在
- 限定时间内消灭所有AI敌人，玩家将获胜。通过
- 在单人模式中，如果没有在限定时间内消灭所有AI敌人，玩家将会失败。通过
- 如果玩家被AI敌人消灭，他们就会失败。通过
这应该会弹出一个弹出窗口，显示游戏失败，如果玩家输了。如果玩家获胜，
- 应该会弹出一个弹出窗口来显示游戏成功。通过玩家可以在弹出的游戏失败
- 窗口中选择重新开始游戏或返回主界面。通过
- 玩家只能在弹出的游戏胜利窗口中选择返回主界面。通过
- 在决斗模式中，如果两名玩家仍在时间限制内存活，则游戏应为平局。通过
- 玩家可以通过点击游戏主菜单中的相应按钮选择游戏难度、游戏模式。通
- 过
- 玩家可以通过点击游戏主菜单中的相应按钮查看游戏规则或帮助。通过
- 玩家可以在设置界面上左右滑动来调整游戏音效的背景音乐音量。通过
- 玩家可以在设置界面中打开或关闭游戏音效或背景音乐。通过
-
-

- 玩家可以在决斗模式的屏幕上点击相应的按钮创建一个房间或加入一个房间。通过

4. 游戏道具测试用例

4.1 核武道具

- 获得一个医疗包可以恢复1点生命值。通过
- 获得一双跑鞋可以提高运动速度。获得一杯硝化甘油溶液可以使放置的炸弹在四个方向(上、下、左、右)分别增加一个格子的爆炸范围。获得一个防暴盾牌可以承受炸弹，火箭筒，手榴弹，地雷的一次爆炸伤害。通过
-

4.2 武器

- 获得火箭炮可以让角色发射两枚火箭。通过
- 按数字键1可以用火箭筒发射火箭。Pass发射的火箭不能飞出地图的边界。
- Pass按数字3键可以埋设地雷，三秒后就看不到了。通过
- 地雷只能放置在空的格子上。通过
- 只有当角色前面的格子上有障碍物时，才能投掷手榴弹。通过
- 按4号数字键可以扔一枚手榴弹到障碍物后面，在前方第三格上造成 3×3 格的爆炸。通过
- 得到一个炸弹包裹可以在三秒钟内增加一个额外放置的炸弹。通过
- 使用手套按数字2键，在三秒内推出炸弹。通过
-

5. 游戏面板测试用例

- 游戏面板应该实时统计和显示玩家和人工智能的生命值。通过
- 游戏面板应实时统计并显示玩家获得的游戏道具。通过
- 游戏面板应实时记录并显示游戏倒计时。Pass游戏面板可以在决斗模式下
- 显示两名玩家之间的聊天记录。通过
-

6. AI测试用例

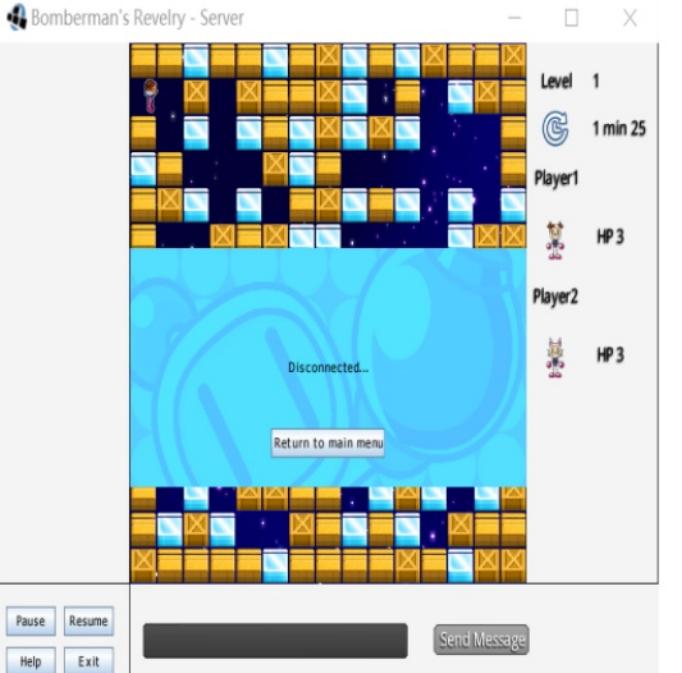
- 人工智能必须绕过障碍物才能移动。通过
- AI可以放置炸弹。通过
- AI应该避开放置炸弹的爆炸半径。通过
- AI不应该通过障碍。通过
- AI可以找到通往宝箱的路径。通过
- AI可以寻找掉落的游戏道具。通过
- 在简单难度下，艾尔无法捡起任何游戏道具。通过
- 在中等难度下，AI可以拾取非武器道具。通过
- AI可以在噩梦难度下使用所有游戏道具。通过
- 在噩梦难度下，AI应该使用武器而不伤害自己。AI可以被炸弹或武器杀死。通过
- 如果受到炸弹或武器的伤害，AI会失去生命值。通过AI可以赢得或输掉这个游戏。通过

7. 网络测试案例

- 通过创建一个房间来构建服务器。通过
- 客户端可以在局域网中输入IP地址来查找服务器。通过
 - 服务器应该接受来自客户端的请求。通过
- 通过加入房间建立服务器和客户端之间的连接。Pass Game数据必须在服务器和客户端之间正常传输。客户端和服务器之间的Pass Game进度必须同步和实时更新。通过
 - 客户端和服务器之间发送和接收的消息必须始终保持同步。通过
- 这应该会出现一个弹出窗口，显示离线状态，如果服务器或客户端离开游戏。通过
 -
 -

附录B

在可用性测试过程中，只存在一个问题：

Problem Users did not know if their opponent was still online in two-player mode	 A screenshot of the game interface titled "Bomberman's Revelry - Server". It shows a 10x10 grid of the game board. On the right side, there is a sidebar with player information: "Level 1", "1 min 25", "Player1" (with a blue icon), "HP 3", "Player2" (with a pink icon), and "HP 3". In the center of the screen, a message box displays "Disconnected..." and a button labeled "Return to main menu". At the bottom left, there are buttons for "Pause", "Resume", "Help", and "Exit". At the bottom right, there is a "Send Message" button.
Description If one player quits the game for network reasons or on his own, the other player doesn't know and waits blindly for a connection	
Solution Designed and Added a window to prompt disconnection, it will pop up and allow users to return to the main menu when one of the players disconnects for some reasons.	