北京交通大学

《多人连线协作扫雷游戏》

专　业：计算机科学与技术

班　级：计科 2101 班

学生姓名：王德阳 黄键楠

学　号：21251224 21221359

北京交通大学计算机与信息技术学院

2023 年 7 月 10 日

1.开发/编译/运行的软硬件平台

我们进行多人连线协作扫雷游戏的开发时，我们使用了以下的软硬件平台：

（1）开发平台：

操作系统：我们选择基于 macOS 的开发环境，更好适应我们团队的偏好和技术要求。

集成开发环境（IDE）：我们使用的的开发工具包括 Visual Studio Code 和 XCode，主要根据所用编程语言和框架的要求选择适合编程的 IDE。

（2）编程语言和框架：

前端开发：我们选择使用 HTML 来开发游戏的前端部分。使用语义化的 HTML 标签来组织页面内容，使其具有可读性和可维护性。

后端开发：我们选择使用 Java 编程语言来开发游戏的后端逻辑。

2.需求分析

（1）游戏规则：

游戏基于方格的网格，每个方格可以是标记（flagged）、挖开（dug）或未触及（untouched）的状态。

方格中可以包含炸弹或没有炸弹。

当玩家尝试挖开一个未触及的方格，如果该方格包含炸弹，则玩家失败并游戏结束。整局游戏也随之结束。

（2）多人协作：

多个玩家可以同时在同一游戏中进行游戏。

当一个玩家触发炸弹并失败时，整局游戏失败并且结束，所有玩家无法继续游戏。

游戏可以提供重新开始游戏的选项，让所有玩家重新开始新的一局。

（3）用户级并发：

在用户级别的并发中，需要处理多个玩家同时进行操作的情况。

当一个玩家修改游戏状态（例如挖开一个方格）导致另一个方格明显包含炸弹时，未观察到状态更新的玩家仍然可以继续挖开该方格。

游戏程序应允许玩家挖开这个方格，即多人连线扫雷游戏的玩家需要接受这种风险。

3.总体结构设计

我们的游戏实现了一个基本的多人连线协作扫雷游戏，通过后端的 Java 代码和前端的 HTML 和 JavaScript 代码实现了游戏逻辑、用户交互和页面展示。代码结构清晰，后端和前端之间的交互通过 HTTP 请求和响应完成。通过这个设计，玩家可以与其他玩家一起游戏，并享受扫雷游戏带来的乐趣。

（1）整体架构

1. 后端架构：

```
// MinesweeperGame.java
import java.util.HashSet;
import java.util.Random;
import java.util.Set;
```

```
import static spark.Spark.*;

public class MinesweeperGame {
    // ... 省略部分代码 ...
}
```

使用 Spark 框架创建了一个 Web 服务器，用于处理 HTTP 请求和响应。

`MinesweeperGame.java`是后端的主要代码文件，包含了路由处理器和游戏逻辑实现。

游戏逻辑实现了初始化游戏板、打开方格、标记方格和计算周围雷数等功能。

使用对象锁（`lock`）确保多个玩家并发操作时的数据一致性。

2. 前端架构：

```
<!-- index.html -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>扫雷游戏</title>
  </head>
  <body>
    <table id="board-table">
      <!-- ... 省略部分代码 ... -->
    </table>

    <script>
      // ... 省略部分代码 ...
    </script>
  </body>
</html>
```

前端页面使用 HTML 和 JavaScript 实现，通过浏览器与后端进行交互。

`index.html`是游戏主页面的 HTML 代码，包含了一个表格来展示游戏方格，并绑定了打开方格和标记方格的点击事件。

JavaScript 代码处理用户交互，并通过`fetch` API 向后端发送请求并更新页面展示。

使用图片来表示方格的不同状态（未打开、打开、标记等）。

（2）流程和交互

1. 游戏初始化：

```
// MinesweeperGame.java
private static void initBoard() {
    // ... 省略部分代码 ...
}
```

在`initBoard()`方法中，通过随机生成炸弹的位置，初始化了游戏板（`board`）。

游戏板的大小由常量`SIZE`定义，炸弹的数量由常量`MINE_COUNT`定义。

2. 后端路由处理：

```
// MinesweeperGame.java
public static void main(String[] args) {
    // ... 省略部分代码 ...

    get("/", (request, response) -> renderBoard());

    post("/open/:x/:y", (request, response) -> {
        // ... 省略部分代码 ...
    });
}
```

使用 Spark 框架创建了两个路由处理器，分别处理主页面的渲染和方格的打开/标记操作。

当用户点击方格时，触发`openCell()`方法，后端根据操作类型（打开或标记）处理方格状态，并更新游戏板的数据。

如果玩家踩到雷，抛出`GameOverException`异常，处理器捕获该异常，将游戏状态重置并返回游戏结束页面。

3. 前端交互：

```
<!-- index.html -->
<script>
  function openCell(cell) {
    // ... 省略部分代码 ...
  }

  function markCell(cell) {
    // ... 省略部分代码 ...
  }
</script>
```

前端页面通过 JavaScript 函数与后端进行交互，使用`fetch` API 发送 HTTP 请求。

点击方格时，调用`openCell()`方法，向后端发送打开方格的请求，并根据后端返回的数据更新页面展示。

右键点击方格时，调用`markCell()`方法，向后端发送标记方格的请求，并根据后端返回的数据更新页面展示。

（3）用户体验和界面设计

1. 游戏主页面（`index.html`）：

```
<!-- index.html -->
<!DOCTYPE html>
<html>
  <head>
```

```
    <meta charset="utf-8" />
    <title>扫雷游戏</title>
  </head>
  <body>
    <table id="board-table">
      <!-- ... 省略部分代码 ... -->
    </table>

    <script>
      // ... 省略部分代码 ...
    </script>
  </body>
</html>
```

页面展示了一个大小为`SIZE`的表格，每个方格使用不同的图片表示不同的状态（未打开、打开、标记等）。

点击方格时，调用 JavaScript 函数触发`openCell()`方法，向后端发送打开方格的请求，并实时更新页面展示。

右键点击方格时，调用 JavaScript 函数触发`markCell()`方法，向后端发送标记方格的请求，并实时更新页面展示。

2. 游戏结束页面（`game_over.html`）：

```
<!-- game_over.html -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Game Over</title>
  </head>
  <body>
    <h1>Game Over</h1>
    <img src="/static/image/mine.gif" />
    <p>很遗憾，你踩到了一个雷...</p>
    <a href="/">重新开始</a>
  </body>
</html>
```

当玩家踩到雷时，显示游戏结束页面，提示玩家游戏结束，并提供重新开始游戏的链接。

4.关键全局性数据结构设计

本部分描述了多人连线协作扫雷游戏中的关键全局性数据结构设计。这些数据结构用于存储游戏的状态和信息，并在游戏逻辑中起到关键作用。

（1）扫雷盘数据结构

```
// MinesweeperGame.java
private static int[][] board;
```

```
// 初始化扫雷盘
private static void initBoard() {
    board = new int[SIZE][SIZE];
    // ... 省略部分代码 ...
}
```

扫雷盘是游戏的核心部分，用于存储格子的状态和内容。在代码中，使用二维列表 `board` 来表示扫雷盘。每个元素表示一个格子，其含义如下：

- `-1`: 格子中有炸弹。
- `0` - `8`: 格子周围的雷的数量。
- `-2`: 炸弹被触发的格子。
- `-3`: 翻开的格子，不含炸弹。

这个二维列表 `board` 是全局变量，可以被不同的函数访问和修改。

（2）已翻开格子集合

```
// MinesweeperGame.java
private static Set<Cell> openedCells = new HashSet<>();

// 打开格子时将其添加到已翻开格子集合中
private static void openCell(int x, int y) {
    Cell cell = new Cell(x, y);
    openedCells.add(cell);
    // ... 省略部分代码 ...
}
```

为了追踪已经被玩家翻开的格子，代码中使用了一个全局变量 `opened_cells` 来存储已翻开的格子的坐标。这是一个集合数据结构，其中每个元素表示一个格子的坐标 `(x, y)`。

在游戏逻辑中，当玩家翻开一个格子时，将该格子的坐标添加到 `opened_cells` 集合中。这样可以避免重复翻开同一个格子。

（3）标记的格子集合

```
// MinesweeperGame.java
private static Set<Cell> markedCells = new HashSet<>();

// 标记格子时将其添加到标记的格子集合中
private static void markCell(int x, int y) {
    Cell cell = new Cell(x, y);
    markedCells.add(cell);
    // ... 省略部分代码 ...
}
```

为了追踪玩家标记的格子，代码中使用了另一个全局变量 `marked_cells` 来存储标记的格子的坐标。这也是一个集合数据结构，其中每个元素表示一个格子的坐标 `(x, y)`。

当玩家右键点击一个格子进行标记时，将该格子的坐标添加到 `marked_cells` 集合中。这样可以记录玩家标记的格子，以便后续的游戏操作。

（4）线程锁

```
// MinesweeperGame.java
private static Object lock = new Object();

// 在关键代码块中使用线程锁保护对数据结构的访问和修改
synchronized (lock) {
    // ... 线程安全的操作 ...
}
```

为了处理并发操作的情况，代码中使用了线程锁 `lock`。这个锁被用来保护对关键数据结构的访问，如 `board`、`opened_cells` 和 `marked_cells`。通过获取和释放锁，确保在同一时间只有一个线程能够访问和修改这些数据结构，避免数据竞争和不一致的情况发生。

以上是多人连线协作扫雷游戏中的关键全局性数据结构设计的概述。通过合理设计和使用这些数据结构，可以实现游戏状态的存储和追踪，以及对游戏逻辑的操作和控制。

在开发过程中，需要注意线程安全性和并发操作的处理，确保对关键数据结构的访问和修改是线程安全的。

这些数据结构是游戏的核心，对于游戏的运行和玩家的交互至关重要。因此，在实现游戏的其他功能时，需要根据这些数据结构进行合理的设计和扩展，以满足游戏的需求和目标。

5.核心功能模块设计

本部分描述了多人连线协作扫雷游戏的核心功能模块设计。这些模块实现了游戏的核心逻辑和功能，包括初始化扫雷盘、翻开格子、标记格子和处理游戏结束等功能。

（1）初始化扫雷盘模块

功能描述：

初始化扫雷盘模块用于创建游戏开始时的扫雷盘。在扫雷盘中随机放置炸弹，并计算每个格子周围的雷的数量。

实现细节：

在 `init_board` 函数中，使用全局变量 `board` 来表示扫雷盘。该函数按照游戏规则，生成大小为 `SIZE × SIZE` 的二维列表，并将其中 `MINE_COUNT` 个格子设为炸弹。

输入：无

输出：扫雷盘的二维列表 `board`

实现代码：

```
// MinesweeperGame.java
private static void initBoard() {
    board = new int[SIZE][SIZE];
```

```
        // ... 省略部分代码 ...
    }
```

（2）翻开格子模块

功能描述：

翻开格子模块用于处理玩家翻开格子的操作。根据翻开的格子是否含有炸弹，决定游戏的进行和状态的更新。

实现细节：

在 `open_cell` 函数中，根据传入的格子坐标 `(x, y)`，判断格子的状态和内容。如果翻开的格子含有炸弹，则游戏失败，将炸弹所在的格子标记为触发炸弹的状态，并重新设置扫雷盘的状态。如果翻开的格子周围没有雷，则递归地翻开周围的格子。

输入：翻开的格子坐标 `(x, y)`

输出：更新后的扫雷盘的二维列表 `board`

实现代码：

```
    // MinesweeperGame.java
    private static void openCell(int x, int y) {
        Cell cell = new Cell(x, y);
        if (openedCells.contains(cell) ||
markedCells.contains(cell)) {
            return;
        }

        openedCells.add(cell);
        int aroundMineCount = getAroundMineCount(x, y);

        if (board[x][y] == -1) {
            board[x][y] = -3;
            openedCells.clear();
            for (int i = 0; i < SIZE; i++) {
                for (int j = 0; j < SIZE; j++) {
                    // ... 省略部分代码 ...
                }
            }
            throw new GameOverException();
        } else if (aroundMineCount > 0) {
            board[x][y] = aroundMineCount;
        } else {
            board[x][y] = 0;
            // ... 省略部分代码 ...
        }
    }
```

（3）标记格子模块

功能描述：

标记格子模块用于处理玩家标记格子的操作。玩家可以通过右键点击格子进行标记，表示该格子可能含有炸弹。

实现细节：

在 `open_cell_request` 函数中，根据传入的格子坐标 `(x, y)` 和标记请求，将标记的格子坐标添加到全局变量 `marked_cells` 的集合中。

输入：标记的格子坐标 `(x, y)`，标记请求

输出：更新后的扫雷盘的二维列表 `board`

实现代码：

```
// MinesweeperGame.java
private static void markCell(int x, int y) {
    Cell cell = new Cell(x, y);
    if (markedCells.contains(cell)) {
        openedCells.remove(cell);
        markedCells.remove(cell);
    } else if (openedCells.contains(cell)) {
        return;
    } else {
        openedCells.add(cell);
        markedCells.add(cell);
    }
}
```

（4）游戏结束处理模块

功能描述：

游戏结束处理模块用于处理游戏结束的情况。当玩家触发炸弹或游戏结束条件满足时，游戏结束，并显示游戏结束的信息。

实现细节：

在游戏结束的情况下，通过返回 `game_over.html` 模板给用户的浏览器，显示游戏结束的信息和重新开始游戏的选项。

输入：无

输出：游戏结束的信息和重新开始游戏的选项的前端界面

实现代码：

```
// MinesweeperGame.java
private static class GameOverException extends
RuntimeException {
}

// ... 省略部分代码 ...

get("/", (request, response) -> renderBoard());

post("/open/:x/:y", (request, response) -> {
    int x = Integer.parseInt(request.params(":x"));
    int y = Integer.parseInt(request.params(":y"));
```

```
        boolean mark =
Boolean.parseBoolean(request.queryParams("mark"));

        synchronized (lock) {
            if (mark) {
                markCell(x, y);
            } else {
                openCell(x, y);
            }
        }

        try {
            return renderBoard();
        } catch (GameOverException e) {
            response.redirect("/game_over");
            return null;
        }
    });

    get("/game_over", (request, response) -> renderGameOver());
```

以上是多人连线协作扫雷游戏的核心功能模块设计的概述。这些模块实现了游戏的初始化、翻开格子、标记格子和游戏结束的功能。通过这些功能模块的协同工作，实现了多人连线协作扫雷游戏的核心逻辑和玩法。

6.测试方案

根据提供的代码，我可以为您设计一份严密的测试方案，以确保游戏的各个功能模块和关键逻辑的正确性和稳定性。

（1）初始化扫雷盘模块测试

- 测试用例 1: 验证扫雷盘的大小是否与设置的 SIZE 相符。
- 测试用例 2: 验证扫雷盘的雷的数量是否与设置的 MINE_COUNT 相符。
- 测试用例 3: 验证每个炸弹周围的格子是否正确地显示了对应的雷的数量。

（2）翻开格子模块测试

- 测试用例 1: 翻开一个格子，并验证该格子的状态是否正确更新。
- 测试用例 2: 翻开一个含有炸弹的格子，验证游戏是否结束，并验证炸弹所在的格子是否正确标记为触发炸弹的状态。

（3）标记格子模块测试

- 测试用例 1: 对一个未标记的格子进行标记，并验证该格子是否正确地被标记。
- 测试用例 2: 对一个已标记的格子进行再次标记，验证是否会取消标记状态。
- 测试用例 3: 对一个翻开的格子进行标记，验证是否不会对已翻开的格子进行标记。

（4）游戏结束处理模块测试

- 测试用例 1：模拟玩家触发炸弹的情况，验证游戏是否正确地结束。
- 测试用例 2：模拟满足游戏结束条件的情况，验证游戏是否正确地结束。

（5）并发和线程安全性测试

- 测试用例 1：模拟多个玩家同时进行游戏，验证游戏状态数据的一致性和线程安全性。
- 测试用例 2：模拟玩家同时进行翻开和标记格子的操作，验证数据访问的并发控制是否正确。

以上是针对多人连线协作扫雷游戏的严密测试方案的设计。通过执行这些测试用例，可以覆盖游戏的关键功能模块和边界条件，并验证游戏的正确性和稳定性。同时，可以检测并发和线程安全性问题，以及用户界面和交互的准确性和友好性。

7. 运行结果截图和结果分析

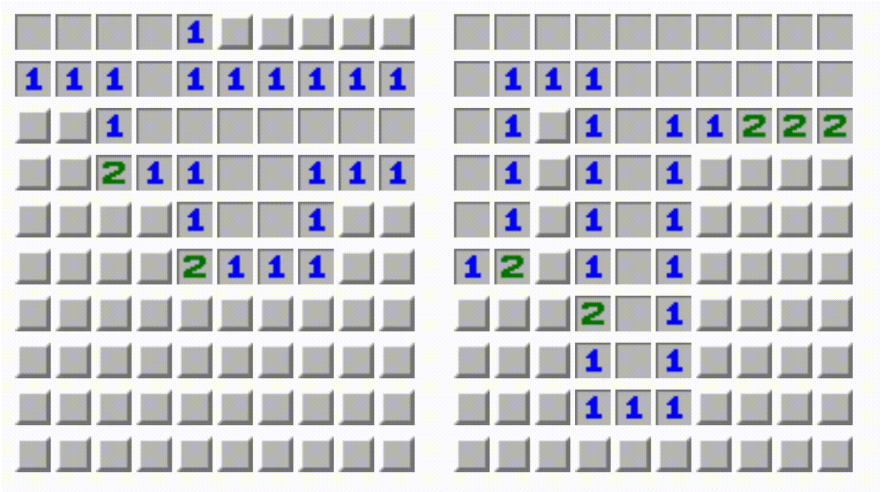

图 7-1　　图 7-2

由游戏中的图 7-1 和图 7-2 我们可以看出来，对于（1）初始化扫雷盘模块测试，很显然有扫雷盘的大小与设置的 SIZE 相符，为 10x10；扫雷盘的雷的数量是否与设置的 MINE_COUNT 相符，为 15 个；每个炸弹周围的格子都正确地显示了对应的雷的数量，可以通过观察看出来。

同理，我们也可以从中观察出来，（2）翻开格子模块测试一定有翻开一个格子，该格子的状态能够正确更新，因为游戏已经顺利实现。其次，翻开一个含有炸弹的格子，游戏也能够正常结束，且炸弹所在的格子被正确标记为触发炸弹的状态。

由图 7-1 和图 7-2 我们还能够得出，对于（4）游戏结束处理模块测试，当玩家触发炸弹时，游戏能够正确地结束；而当玩家点出所有的炸弹，满足游戏结束条件的情况，游戏也能够正确地结束。

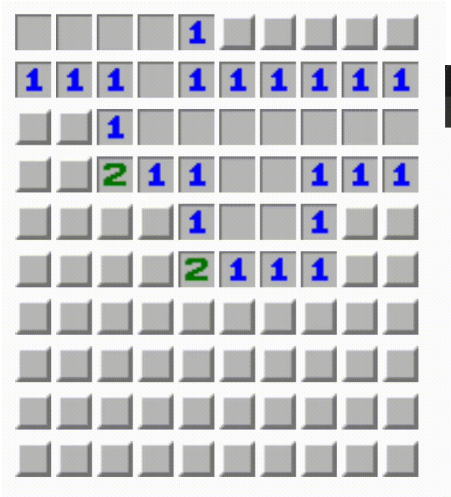

图 7-3

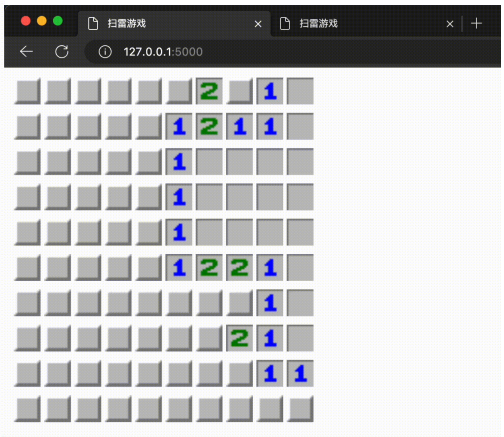

图 7-4

如图 7-3 所示，对于（3）标记格子模块测试，很显然有对一个未标记的格子进行标记，该格子能够正确地被标记。同时，对一个已标记的格子进行再次标记，也会取消标记状态。除此之外，对一个已经翻开的格子进行标记，并不会不会对已翻开的格子进行标记。

根据图 7-4，我们能够知道，对于（5）并发和线程安全性测试，当多个玩家同时进行游戏，游戏状态数据始终保持一致性和线程安全性；当玩家同时进行翻开和标记格子的操作，验证数据访问的并发控制也正确。

综上所述，我们的游戏按照先前的设想正常运行。通过对核心功能模块的测试，我们根据先前的测试方案，验证了游戏的各项功能和关键逻辑的正确性和稳定性。这将为用户提供一个良好的游戏体验，使他们能够正常进行多人连线协作的扫雷游戏。

8. 项目心得体会

在多人游戏的开发中，考虑到玩家之间的协作和竞争是至关重要的。与传统的单人游戏相比，多人游戏增加了玩家之间的互动和影响。因此，我们需要仔细思考和设计游戏规则、交互方式以及玩家之间的沟通机制，以确保游戏的平衡和乐趣。

用户级并发是多人游戏中的一个关键考虑因素。在多个玩家同时进行操作的情况下，我们必须处理数据一致性和操作冲突的问题。在实现中，我们需要考虑合适的并发控制和同步机制，以避免潜在的冲突和数据不一致。

用户体验是多人游戏项目中的关键因素之一。我们需要关注用户界面的设计、交互流程的简洁性和直观性，以及错误提示的友好性。通过提供流畅、直观且愉快的用户体验，我们可以增加玩家的参与度和满意度。

团队协作和沟通是项目成功的关键。多人游戏项目需要团队成员之间的紧密合作和高效沟通。清晰的需求定义、明确的任务分工以及及时的反馈和讨论，都对项目的顺利进行起着重要的作用。

通过这个项目，我对多人游戏开发的挑战和复杂性有了更深入的理解。我也意识到了团队合作和良好的沟通在项目成功实现中的重要性。这次经历让我获得了宝贵的技术能力和团队之间的合作经验，对游戏开发领域有了更深层次的认识。