超级玛丽游戏项目开发报告

1. 项目概述

本项目是一款基于 Java 开发的超级玛丽游戏，旨在为玩家提供一款经典的游戏体验。游戏界面采用 Java Swing 库进行界面开发，实现了基本的游戏玩法，包括玩家控制、敌人移动、子弹发射等。游戏地图通过文本文件配置，支持多个关卡。游戏中的角色和道具包括马里奥、金币、砖块、水管、乌龟等。游戏还包含背景音乐和游戏结束提示。

创新点&技术难点：本项目在传统的马里奥游戏的基础上将单人游戏设置为双人，两个角色的得分和血量分别单独统计，同时还通过连接数据库实现了游戏记录的保存和游戏进度的更新。

2. 业务需求分析和基本模块功能

2.1 游戏玩法&需求分析

玩家可以通过键盘控制马里奥移动和跳跃。

玩家可以发射子弹攻击乌龟。

马里奥可以收集金币，蘑菇增加得分。

马里奥可以与乌龟和障碍物碰撞，减少血量。

马里奥可以到达城堡通关。

2.2 功能模块

2.2.1 登录和注册界面：连接数据库实现用户的登录和注册，登录成功后打印用户的最新游戏进度

2.2.2 关卡选择界面及关卡设计：在关卡选择界面选择想要游玩的关卡后，通过读取对应关卡的文本文件加载对应关卡开始游戏。

2.2.3 游戏窗口：

(1) 加载背景音乐和地图资源：根据地图文本文件创建并绘制各种道具及障碍物，创建 Mario 角色

(2) 人物的移动碰撞逻辑：在 Mario 类中实现人物的移动，同时在 KeyListener 类中处理对应的按键事件

(3) 乌龟和蘑菇的移动逻辑：分别在 Turtle 类和 Mogu 类中处理转向和图片加载等逻辑，然后在游戏界面的主线程中调用对应的 move 函数。

(4) 角色发射子弹以及清理死亡的乌龟：为每个 mario 创建一个子弹列表，子弹命中乌龟后将被命中乌龟列入待清除序列，同时增加该角色得分。

(5) 角色碰撞金币和蘑菇增加得分：每个 coin 对象在首次撞击时掉落金币并增加得分，蘑菇拾取后即消失并增加得分。

(6) 角色血量管理：角色碰到 wave 对象即扣一滴血，碰到乌龟立刻死亡，当两个角色中有一个死亡时游戏即结束。

(7) 游戏的暂停和结束：通过键盘输入 P 暂停游戏，Q 结束游戏。

(8) 数据保存更新：游戏结束时连接数据库保存本次游戏记录到 game_results，同时若当前关卡数大于已有记录中关卡或当前两个角色的总得分高于记录得分，则更新用户记录

到 game_records.同时返回关卡选择模块以开始下一轮游戏。

3. 系统设计与实现

3.1 计算机体系设计

开发平台：Windows 11

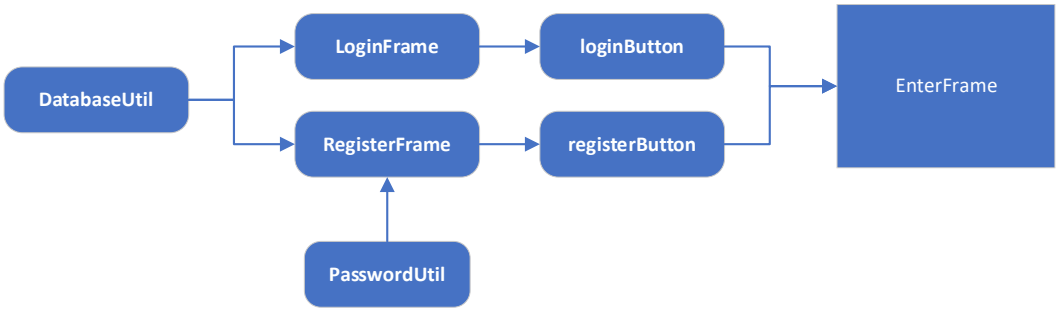
开发工具：IntelliJ IDEA，Mysql 数据库

开发语言：Java

开发库：Java Swing

3.2 功能详细设计与数据库字典

3.2.1 登录和注册界面：为了便于分工和管理，我们通过设计 LoginFrame 类和 RegisterFrame 类分别绘制登录和注册窗口以及分别处理登录和绘制逻辑，其中我们将数据库的连接和资源的关闭分离出来，创建一个 DatabaseUtil 来单独管理，在登录和注册过程中直接通过 DatabaseUtil 获得数据库连接，同时为了确保数据安全，我们还对用户输入的密码通过 SHA-256 算法进行加密处理，创建了一个 PasswordUtil 来实现加密密码的创建和校验工作。在注册逻辑中，我们对用户输入的密码进行 SHA-256 加密后插入到 user 数据表中，同时我们还会进行重名检测从而避免重复注册。在登录逻辑中，我们针对用户的输入的密码调用 PasswordUtil 中的 checkPassword 方法来进行密码校验，校验成功则进入关卡选择界面。在设计好 LoginFrame 类和 RegisterFrame 类之后，我们又重新设计了一个初始界面 EnterFrame 类，并通过 loginButton 和 registerButton 来控制创建 LoginFrame 和 RegisterFrame 的实例，从而将登录和注册功能集成到一起。

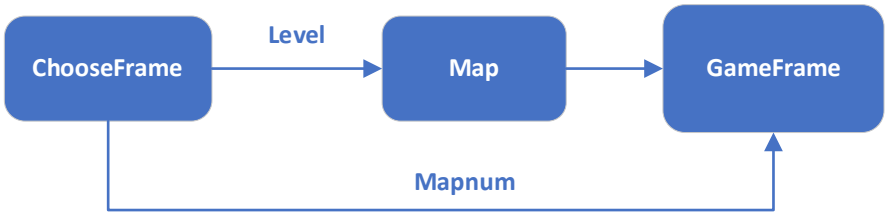


3.2.2 关卡选择界面及关卡设计：

首先绘制关卡选择界面 ChooseFrame 类的背景图片，然后再为每一个关卡创建一个 button，为每个按钮添加动作监听，当点击某一关卡时 actionPerformed 方法将按钮的动作命令转换为整数以表示关卡，获取关卡参数后即可创建对应的游戏窗口 GameFrame，同时为其添加键盘监听器。

3.2.3 游戏窗口

(1)加载背景音乐和地图资源：背景音乐的添加通过 MusicUtil 工具类实现。然后创建 Mario 和 Mario2 对象，初始化血量和得分，同时开始加载地图资源。为了方便对关卡进行扩充，地图资源的加载通过文本文件的形式加载，其具体实现路线是先创建好各种道具类(Brickl 类，coin 类，pipe 类，Mogu 类，Turtle 类等)，然后由 Map 类读取文本文件，将其转化为整形二维数组，每一个道具类对应唯一一个编号（例如 1 对应 brick，2 对应 coin），然后在 GameFrame 里遍历读取之后的 map 数组来批量创建道具对象。最后根据创建的对象通过 paint 方法进行地图资源绘制。



（2）人物的移动碰撞逻辑：人物的移动碰撞逻辑主要依靠 KeyListener 类和 Mario 类实现，通过 KeyListener 监听键盘修改 mario 中的信号值来控制人物的移动方向同时记录角色最后一次的朝向来控制角色发射子弹的方向。通过 hit 方法来处理与金币蘑菇和障碍物的碰撞，从而对人物的速度和血量得分作出调整。

（3）乌龟和蘑菇的移动逻辑：我们在 Turtle 和 Mogu 类中分别设计 checkCollision 方法来控制乌龟和蘑菇的运动方向，当乌龟和蘑菇碰到障碍物或者到达窗口边缘时反向，然后在 GameFrame 中调用各自的 move 方法即可实现移动逻辑。

（4）角色发射子弹以及清理死亡的乌龟：在 Turtle 类中设计 isHitByBullet 方法来检测乌龟是否被子弹击中若被击中则更新 isDead，同时在 GameFrame 中设计 checkBoom 方法来清理超出窗口的子弹，同时将被击中的乌龟加入清除列表进行批量清楚。同时增加得分。

（5）角色碰撞金币和蘑菇增加得分：为 coin 类新增两个 bool 变量 isCollected 和 showSun 来分别金币是否已经被收集以及是否显示金币图片。然后在 Mario 的 hit 方法中检测碰撞的同时更新 isCollected 和 showSun 来控制金币图片的显示，同时增加得分。

（6）角色血量管理：血量的管理统一在 GameFrame 的 checkmario 方法中进行，turtle 导致的血量变动直接通过在 checkmario 中比较 mario 和 turtle 的位置实现，而 wave 导致的血量变动直接根据 hitflag1 来判断即可，hitflag1 的更新在 Mario 中进行。

（7）游戏的暂停和结束：暂停的检测在 KeyListener 中进行，键入 P 后更新 isPaused，然后在 GameFrame 的主线程中每次检测暂停状态来判断是否暂停。Q 键结束关卡的实现较为复杂，因此在 GameFrame 的 KeyListener 实例中再添加对 Q 的检测，若键入 Q 键则直接调用 dispose 方法结束游戏。

（8）数据保存更新：通过 GameFrame 中的 saveGameResult 来实现数据的保存更新，其有两种重载模式，private void saveGameResult(String result, int userid, String username)用于将本次游戏记录存储到 game_results 数据表，public void saveGameResult(String username, int level) 用于更新该用户的游戏进度到 game_records 数据表。在每次结束游戏前依次调用这两个函数即可实现数据的保存更新。

3.3 编码实现

3.3.1 登录和注册界面：

```java
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

/**
 * DatabaseUtil 类用于管理数据库连接和资源的关闭。
 */
public class DatabaseUtil {  11 个用法
    // 数据库连接的URL，其中包括了数据库的地址、端口和数据库名称
    private static final String URL = "jdbc:mysql://localhost:3306/mario?useSSL=false&serverTimezone=UTC";
    // 数据库用户名
    private static final String USER = "hang";  1 个用法
    // 数据库密码
    private static final String PASSWORD = "8892297";  1 个用法

    /**
     * 获取数据库连接的方法。
     *
     * @return 返回一个 Connection 对象，用于与数据库进行交互。
     * @throws Exception 如果加载驱动或建立连接失败，则抛出异常。
     */
    public static Connection getConnection() throws Exception {  6 个用法
        // 加载数据库驱动
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        // 建立并返回数据库连接
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    /**
     * 关闭数据库资源的方法，包括连接、预编译语句和结果集。
     *
     * @param conn   要关闭的数据库连接
     * @param pstmt  要关闭的预编译语句
     * @param rs     要关闭的结果集
     */
    public static void close(Connection conn, PreparedStatement pstmt, ResultSet rs) {  1 个用法
        try {
            // 关闭结果集
            if (rs != null) rs.close();
            // 关闭预编译语句
            if (pstmt != null) pstmt.close();
            // 关闭数据库连接
            if (conn != null) conn.close();
        } catch (Exception e) {
            e.printStackTrace(); // 打印异常堆栈跟踪信息
        }
    }

    /**
     * 关闭数据库资源的方法，仅包括连接和预编译语句。
     *
     * @param conn   要关闭的数据库连接
     * @param pstmt  要关闭的预编译语句
     */
    public static void close(Connection conn, PreparedStatement pstmt) {  1 个用法
        // 调用重载的 close 方法，传递 null 作为结果集参数
        close(conn, pstmt,  rs: null);
    }
}
```

图 1 建立数据库连接

```java
    // 登录逻辑
    try (Connection conn = DatabaseUtil.getConnection()) {// 获取数据库连接
        String sql = "SELECT id, password_hash, salt FROM users WHERE username = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString( parameterIndex: 1, username);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            int userid = rs.getInt( columnLabel: "id");
            String storedHash = rs.getString( columnLabel: "password_hash");
            String salt = rs.getString( columnLabel: "salt"); // 从数据库中获取盐值

            // 验证密码
            if (PasswordUtil.checkPassword(password, storedHash, salt)) {
                // 加载用户的游戏记录
                String gameRecord = loadGameRecord(conn, username);
                JOptionPane.showMessageDialog( parentComponent: null, message: "登录成功", title: "信息", JOptionPane.INFORMATION_MESSAGE);

                // 更新EnterFrame的状态
                enterFrame.updateLoginState( isLoggedIn: true, username, userid, gameRecord);

                dispose(); // 关闭登录窗口
            } else {
                JOptionPane.showMessageDialog( parentComponent: null, message: "密码错误", title: "错误", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: null, message: "用户名不存在", title: "错误", JOptionPane.ERROR_MESSAGE);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog( parentComponent: null, message: "登录失败", title: "错误", JOptionPane.ERROR_MESSAGE);
    }
}
```

图 2 登录逻辑

```java
// 注册操作的具体实现
private class RegisterAction implements ActionListener { 1 个用法
    @Override
    public void actionPerformed(ActionEvent e) {
        // 获取用户输入的用户名和密码
        String username = usernameField.getText();
        String password = new String(passwordField.getPassword());

        // 检查用户名或密码是否为空
        if (username.isEmpty() || password.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "用户名或密码不能为空", title: "错误", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // 生成盐值并加密密码
        String salt = PasswordUtil.generateSalt(); // 生成盐值
        String hashedPassword = PasswordUtil.hashPassword(password, salt); // 使用盐加密密码

        try (Connection conn = DatabaseUtil.getConnection()) {
            // 插入新用户信息到数据库
            String sql = "INSERT INTO users (username, password_hash, salt) VALUES (?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString( parameterIndex: 1, username); // 设置用户名
            pstmt.setString( parameterIndex: 2, hashedPassword); // 设置加密后的密码
            pstmt.setString( parameterIndex: 3, salt); // 设置盐值

            int rows = pstmt.executeUpdate(); // 执行SQL语句并返回影响的行数
            if (rows > 0) {
                // 注册成功提示
                JOptionPane.showMessageDialog( parentComponent: null, message: "注册成功", title: "信息", JOptionPane.INFORMATION_MESSAGE);
                dispose(); // 关闭注册窗口
            }
        } catch (Exception ex) {
            // 注册失败提示
            JOptionPane.showMessageDialog( parentComponent: null, message: "注册失败", title: "错误", JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace(); // 打印异常信息
        }
    }
}
```

图 3 注册逻辑

```java
package util;

import ...

public class PasswordUtil {// 密码工具类  5个用法
    // 生成盐
    public static String generateSalt() {  1个用法
        SecureRandom random = new SecureRandom();
        byte[] salt = new byte[16];
        random.nextBytes(salt);
        return Base64.getEncoder().encodeToString(salt);
    }

    // 使用 SHA-256 加密密码，带盐
    public static String hashPassword(String password, String salt) {  2个用法
        try {
            MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256");
            md.update(salt.getBytes()); // 添加盐
            byte[] hashedPassword = md.digest(password.getBytes());
            return Base64.getEncoder().encodeToString(hashedPassword); // 返回 Base64 编码的哈希值
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("加密算法不存在", e);
        }
    }

    // 校验密码
    public static boolean checkPassword(String plainPassword, String hashedPassword, String salt) {
        String newHash = hashPassword(plainPassword, salt);
        return newHash.equals(hashedPassword); // 比较加密后的密码
    }
}
```

图 4 加密算法

3.3.2 关卡选择界面及关卡设计：

```java
// 初始化按钮数组
buttons = new JButton[5];
Font buttonFont = new Font( name: "方正舒体", Font.PLAIN,  size: 30); // 按钮字体

// 使用循环创建并设置多个按钮
for (int i = 0; i < buttons.length; i++) {
    buttons[i] = new JButton(String.valueOf( i: i + 1)); // 按钮文本为关卡编号
    buttons[i].setFont(buttonFont); // 设置按钮字体
    buttons[i].setForeground(Color.yellow); // 设置按钮前景色
    buttons[i].setBackground(mycolor); // 设置按钮背景色
    buttons[i].setBounds( x: 10 + 101 * i,  y: 95,  width: 65,  height: 65); // 设置按钮位置和大小
    buttons[i].setHorizontalAlignment(JButton.CENTER); // 设置按钮文本居中
    buttons[i].addActionListener( l: this); // 为按钮添加动作监听
    this.getContentPane().add(buttons[i]); // 将按钮添加到内容面板
}
```

图 5 创建关卡按钮

```java
@Override
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();
    int level = Integer.parseInt(command); // 将按钮的动作命令转换为整数以表示关卡

    dispose(); // 关闭当前窗口
    try {
        // 创建一个新的游戏窗口，并为其添加键盘监听器
        GameFrame g = new GameFrame(level, this.userid, this.username);
        KeyListener kl = new KeyListener(g);
        g.addKeyListener(kl);
    } catch (Exception e1) {
        e1.printStackTrace(); // 打印异常信息
    }
}
```

图 6 选择关卡并创建游戏窗口

3.3.3 游戏窗口：

（1）加载背景音乐和地图资源：

```java
package util;

import ...

public class MusicUtil {  2个用法
    private static Clip clip;  4个用法
    static {
        File bgMusicFile = new File( pathname: "music/bg1.wav");
        try {
            AudioInputStream audioInputStream =
                    AudioSystem.getAudioInputStream(bgMusicFile);
            clip = AudioSystem.getClip();
            clip.open(audioInputStream);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void playBackground(){  1个用法
        //音乐播放
        clip.setFramePosition(0);
        clip.loop(Clip.LOOP_CONTINUOUSLY);
    }
}
```

图 7 音乐工具类

```
// 地图配置 创建各种道具对象
for (int i = 0; i < map.length; i++) {
    for (int j = 0; j < map[0].length; j++) {
        switch (map[i][j]) {
            case 0:
                break;
            case 1:
                brick = new Brick( x: j * 30,  y: i * 30,  width: 30,  height: 30, new ImageIcon( filename: "image/brick.png").getImage());
                eneryList.add(brick);
                break;
            case 2:
                coin = new Coin( x: j * 30,  y: i * 30,  width: 30,  height: 30, new ImageIcon( filename: "image/coin_brick.png").getImage());
                eneryList.add(coin);
                break;
            case 3:
                pipe = new Pipe( x: j * 30,  y: i * 30,  width: 60,  height: 120, new ImageIcon( filename: "image/pipe.png").getImage());
                eneryList.add(pipe);
                break;
            case 4:
                wave = new Wave( x: j * 30,  y: i * 30,  width: 30,  height: 30, new ImageIcon( filename: "image/wave.png").getImage());
                eneryList.add(wave);
                break;
            case 5:
                mogu = new Mogu( x: j * 30,  y: i * 30,  width: 30,  height: 30, new ImageIcon( filename: "image/mogu.png").getImage());
                eneryList.add(mogu);
                break;
            case 6:
                castle = new Castle( x: j * 30,  y: i * 30,  width: 90,  height: 90, new ImageIcon( filename: "image/tower.png").getImage());
                eneryList.add(castle);
                break;
            case 7:
                Turtle turtle = new Turtle( x: j * 30,  y: i * 30,  width: 30,  height: 30, new ImageIcon( filename: "image/Ltortoise2.png").getImage());
                eneryList.add(turtle);
                break;
        }
    }
}
```

图 8 根据 map 数组创建道具对象

（2）人物的移动碰撞逻辑:

```java
/**
 * 马里奥线程的主循环，处理移动和游戏逻辑。
 */
public void run() {
    while (true) {
        if (left) {
            lastDirection = Direction.LEFT; // 更新最后方向
            if (!speedUp) {
                this.xspeed = 5; // 正常速度
            }
            if (hit(Dir_Left)) {
                this.xspeed = 0; // 如果碰撞则停止
            }
            if (this.x >= 0) {
                this.x -= this.xspeed; // 向左移动
                this.img = new ImageIcon( filename: "image/mari_left.gif").getImage(); // 更新图像
            }
        }

        if (right) {
            lastDirection = Direction.RIGHT; // 更新最后方向
            if (hit(Dir_Right)) {
                this.xspeed = 0; // 如果碰撞则停止
            }
            if (this.x < 600) {
                this.x += this.xspeed; // 向右移动
                this.img = new ImageIcon( filename: "image/mari_right.gif").getImage(); // 更新图像
            }
            if (this.x >= 600) {
                gf.bg.x -= this.xspeed; // 移动背景
                if (gf.bg.x == -6500) {
                    gf.bg.endflag = true; // 到达关卡末尾
                }
                for (int i = 0; i < gf.eneryList.size(); i++) {
                    Enemy enery = gf.eneryList.get(i);
                    enery.x -= this.xspeed; // 移动所有敌人
                }
                gf.mario2.x2 -= this.xspeed; // 移动第一个马里奥
                this.img = new ImageIcon( filename: "image/mari_right.gif").getImage(); // 更新图像
            }
            this.xspeed = 5; // 重置速度
        }

        if (up) {
            if (jumpFlag && !isGravity) {
                jumpFlag = false;
                new Thread(() -> {
                    jump(); // 执行跳跃动作
                    jumpFlag = true;
                }).start();
            }
        }

        try {
            this.sleep( millis: 20); // 控制循环速度
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

图 9Mario 类移动逻辑

```java
/**
 * 键盘按下事件监听，用于处理不同按键的按下操作。
 *
 * @param e 键盘事件
 */
@Override
public void keyPressed(KeyEvent e) {
    int code = e.getKeyCode();
    switch (code) {
        // 向右走
        case 39: // Key code for the right arrow key
            gf.mario.right = true; // 设置信号位为真，表示向右移动
            break;
        // 向左走
        case 37: // Key code for the left arrow key
            gf.mario.left = true; // 设置信号位为真，表示向左移动
            break;
        // 发射子弹
        case 32: // Key code for the space bar
            addBoom(); // 调用发射子弹方法
            break;
        // 向上跳跃
        case 38: // Key code for the up arrow key
            gf.mario.up = true; // 设置信号位为真，表示跳跃
            break;
        // 第二个角色（mario2）向右走
        case 68: // Key code for the 'D' key
            gf.mario2.right2 = true; // 设置信号位为真，表示向右移动
            break;
        // 第二个角色（mario2）向左走
        case 65: // Key code for the 'A' key
            gf.mario2.left2 = true; // 设置信号位为真，表示向左移动
            break;
        // 暂停或继续游戏
        case 80: // Key code for the 'P' key
            gf.setPaused(!gf.isPaused()); // 切换游戏暂停状态
            if (gf.isPaused()) {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "游戏已暂停，按 P 键继续",  title: "暂停", JOptionPane.INFORMATION_MESSAGE);
            }
            break;
        // 第二个角色（mario2）发射子弹
        case 83: // Key code for the 'S' key
            addBoom2(); // 调用发射子弹方法
            break;
        // 第二个角色（mario2）向上跳跃
        case 87: // Key code for the 'W' key
            gf.mario2.up2 = true; // 设置信号位为真，表示跳跃
            break;
    }
}
```

图 10 键盘事件监听

（3）乌龟和蘑菇的移动逻辑：

```java
// 检测是否碰到障碍物或边界，反向移动
public void checkCollision(GameFrame gf) {  1 个用法
    for (Enemy e : gf.eneryList) {
        if (e != this && this.getBounds().intersects(e.getBounds())) {
            moveRight = !moveRight; // 碰撞后反向移动
            updateImage(); // 更新图片方向
            break;
        }
    }

    // 边界检测
    if (this.x <= 0 || this.x + this.width >= gf.getWidth()) {
        moveRight = !moveRight;
        updateImage(); // 更新图片方向
    }
}
```

图 11 乌龟和蘑菇的移动逻辑

（4）角色发射子弹以及清理死亡的乌龟：

```java
// 检查子弹碰撞
public void checkBoom() {  1 个用法
    for (int i = 0; i < boomList.size(); i++) {
        Boom b = boomList.get(i);
        if (b.x < 0 || b.x > 800) {
            boomList.remove(i);
        }
    }

    for (int i = 0; i < boomList2.size(); i++) {
        Boom b2 = boomList2.get(i);
        if (b2.x < 0 || b2.x > 800) {
            boomList2.remove(i);
        }
    }

    ArrayList<Enemy> toRemove = new ArrayList<>();
    for (int i = 0; i < eneryList.size(); i++) {
        Enemy e = eneryList.get(i);
        if (e instanceof Turtle) {
            Turtle turtle = (Turtle) e;
            if (turtle.isHitByBullet(boomList2) || turtle.isHitByBullet((boomList))) {
                if(turtle.isHitByBullet(boomList))
                {mario.score += 5;
                score1 =mario.score;}
                if(turtle.isHitByBullet(boomList2))
                {mario2.score2 += 5;
                score2 =mario2.score2;}
                turtle.isDead = true;
                toRemove.add(turtle);
            }
        }
    }
    // 移除死亡的敌人
    for (Enemy enemy : toRemove) {
        eneryList.remove(enemy);
    }
}
```

图 12 子弹碰撞检测

（5）角色碰撞金币和蘑菇增加得分：

```java
/**
 * 检查马里奥在指定方向上是否发生碰撞。
 * 处理与金币、敌人和障碍物的碰撞。
 */
public boolean hit(String dir) {  5个用法
    Rectangle myrect = new Rectangle(this.x, this.y, this.width, this.height); // 马里奥的边界框
    Rectangle rect = null;

    for (int i = 0; i < gf.eneryList.size(); i++) {
        Enemy enery = gf.eneryList.get(i);

        // 根据方向定义碰撞检测的边界框
        if (dir.equals("Left")) {
            rect = new Rectangle( x: enery.x + 5, enery.y, enery.width, enery.height);
        } else if (dir.equals("Right")) {
            rect = new Rectangle( x: enery.x - 5, enery.y, enery.width, enery.height);
        } else if (dir.equals("Up")) {
            rect = new Rectangle(enery.x,  y: enery.y + 2, enery.width, enery.height);
        } else if (dir.equals("Down")) {
            rect = new Rectangle(enery.x,  y: enery.y - 2, enery.width, enery.height);
        }

        // 处理与金币的碰撞
        if (myrect.intersects(rect) && dir.equals("Up")) {
            if (enery instanceof Coin) {
                Coin coin = (Coin) enery;
                if (!coin.isCollected()) {
                    coin.setCollected(true); // 设置为已收集状态
                    coin.img = new ImageIcon( filename: "image/coin_brick_null.png").getImage(); // 修改图片
                    score=gf.score1;
                    score++; // 增加得分
                    gf.score1 = score;
                    System.out.println("得分增加，当前分数：" + score);
                    // 显示太阳图片
                    coin.setShowSun(true);
                    Timer timer = new Timer( delay: 200, new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            coin.setShowSun(false); // 隐藏太阳图片
                            gf.repaint(); // 刷新界面以消失太阳图像
                        }
                    });
                    timer.setRepeats(false);
                    timer.start();
                }
                return true;
            }
        } else if (myrect.intersects(rect) && dir.equals("Down")) {
            if (enery.getClass() == Wave.class) {
                gf.hitflag1 = true; // 标记被波浪击中
            }
            return true;
        }

        // 处理与蘑菇的碰撞
        if (myrect.intersects(rect)) {
            if (enery.getClass() == Mogu.class) {
                score += 10; // 增加蘑菇得分
                gf.score1 = score;
                gf.eneryList.remove(i); // 移除蘑菇
                System.out.println("吃到蘑菇，加10分！");
            }
            return true;
        }
    }
    return false; // 没有碰撞
}
```

图 13Mario 与金币蘑菇的碰撞逻辑

（6）角色血量管理：

```java
public void checkmario() throws Exception {  1 个用法
    for (Enemy e : eneryList) {
        if (e instanceof Turtle) {
            Turtle turtle = (Turtle) e;
            if (mario.x + mario.width > turtle.x && mario.x < turtle.x + turtle.width &&
                    mario.y + mario.height > turtle.y && mario.y < turtle.y + turtle.height) {
                mario.blood--;
                if (mario.blood == 0 && !endflag) {
                    saveGameResult( result: "Mario1 is dead", userid, username);
                    saveGameResult(username, mapnum);
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Mario1 is dead",  title: "提示消息", JOptionPane.WARNING_MESSAGE);
                    endflag = true;
                    dispose();
                    ChooseFrame choose = new ChooseFrame(username, userid);
                }
            }
        }
    }

    if (hitflag1 && !changeflag) {
        mario.blood--;
        changeflag = true;
        if (mario.right)
            mario.x -= 80;
        else
            mario.x += 80;
        hitflag1 = false;
    }

    int bufferZone = 10;// 碰撞检测的缓冲区

    if (mario.x + mario.width > castle.x - bufferZone && mario.x < castle.x + castle.width + bufferZone &&
            mario.y + mario.height > castle.y - bufferZone && mario.y < castle.y + castle.height + bufferZone &&
            mario2.x2 + mario2.width2 >                castle.x - bufferZone && mario2.x2 < castle.x + castle.width + bufferZone &&
            mario2.y2 + mario2.height2 > castle.y - bufferZone && mario2.y2 < castle.y + castle.height + bufferZone) {
        if (!endflag) {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "通关！",  title: "游戏提示", JOptionPane.INFORMATION_MESSAGE);
            endflag = true;
            saveGameResult( result: "Success", userid, username);
            dispose();
            ChooseFrame choose = new ChooseFrame(username, userid);
        }
    }
    if (mario.blood == 0 && !endflag) {
        saveGameResult( result: "Mario1 is dead", userid, username);
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Mario1 is dead",  title: "提示消息", JOptionPane.WARNING_MESSAGE);
        endflag = true;
        dispose();
        ChooseFrame choose = new ChooseFrame(username, userid);
    }
}
```

图 14 血量管理和死亡判断

（7）游戏的暂停和结束：

```java
// 添加键盘监听器
this.addKeyListener(new KeyListener() {
    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_Q) {
            int confirm = JOptionPane.showConfirmDialog(
                    parentComponent: null,
                    message: "确定要退出游戏并返回选择关卡界面吗？",
                    title: "退出游戏",
                    JOptionPane.YES_NO_OPTION
            );
            if (confirm == JOptionPane.YES_OPTION) {
                endflag = true;
                dispose();
                try {
                    ChooseFrame choose = new ChooseFrame(username,userid);
                } catch (Exception ex) {
                    throw new RuntimeException(ex);
                }
            }
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {}
});
```

图 15 退出游戏监听，暂停游戏监听见图 10

（8）数据保存更新：

```java
// 保存游戏结果到数据库
private void saveGameResult(String result, int userid, String username) {  7 个用法
    String sql = "INSERT INTO game_results (userid, username, player1_score, player1_blood, player2_score, player2_blood, result, timestamp) " +
            "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = DatabaseUtil.getConnection();
        pstmt = conn.prepareStatement(sql);

        pstmt.setInt( parameterIndex: 1, userid); // 用户ID
        pstmt.setString( parameterIndex: 2, username); // 用户名
        pstmt.setInt( parameterIndex: 3, score1); // 角色1得分
        pstmt.setInt( parameterIndex: 4, mario.blood); // 角色1血量
        pstmt.setInt( parameterIndex: 5, score2); // 角色2得分
        pstmt.setInt( parameterIndex: 6, mario2.blood2); // 角色2血量
        pstmt.setString( parameterIndex: 7, result); // 游戏结果 (Success / Dead)
        pstmt.setTimestamp( parameterIndex: 8, new Timestamp(System.currentTimeMillis())); // 当前时间

        pstmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        DatabaseUtil.close(conn, pstmt);
    }
}
```

图 16 数据保存函数重载方法 1

```java
public void saveGameResult(String username, int level) {  3 个用法
    Timestamp currentTimestamp = new Timestamp(System.currentTimeMillis());
    GameRecordDAO gameRecordDAO = new GameRecordDAO();

    // 获取用户的现有记录
    List<GameRecord> existingRecords = gameRecordDAO.getGameRecordsByUsername(username);

    int totalScore = score1 + score2; // 计算总得分
    int maxBlood = Math.max(mario.blood, mario2.blood2); // 计算最高血量

    if (existingRecords.isEmpty()) {
        // 如果没有现有记录，创建新记录
        GameRecord newRecord = new GameRecord(username, totalScore, maxBlood, currentTimestamp, level);
        gameRecordDAO.saveGameRecord(newRecord);
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Game result saved as new record!");
    } else {
        GameRecord existingRecord = existingRecords.get(0);
        int existingLevel = existingRecord.getLevel();

        // 如果新关卡高于现有记录，则更新记录
        if (level > existingLevel) {
            existingRecord.setScore(totalScore);
            existingRecord.setBlood(maxBlood);
            existingRecord.setCreatedAt(currentTimestamp);
            existingRecord.setLevel(level);
            gameRecordDAO.updateGameRecord(existingRecord); // 更新记录
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Level updated to " + level + "!");
        } else if (level == existingLevel) {
            // 如果关卡相同，检查得分是否更高
            if (totalScore > existingRecord.getScore()) {
                existingRecord.setScore(totalScore);
                existingRecord.setBlood(maxBlood);
                existingRecord.setCreatedAt(currentTimestamp);
                gameRecordDAO.updateGameRecord(existingRecord); // 更新记录
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Score updated to " + totalScore + "!");
            } else {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "No update needed. Existing score is higher or equal.");
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "No update needed. Existing level is higher.");
        }
    }
}
```

图 17 数据保存重载方法 2

4. 系统测试

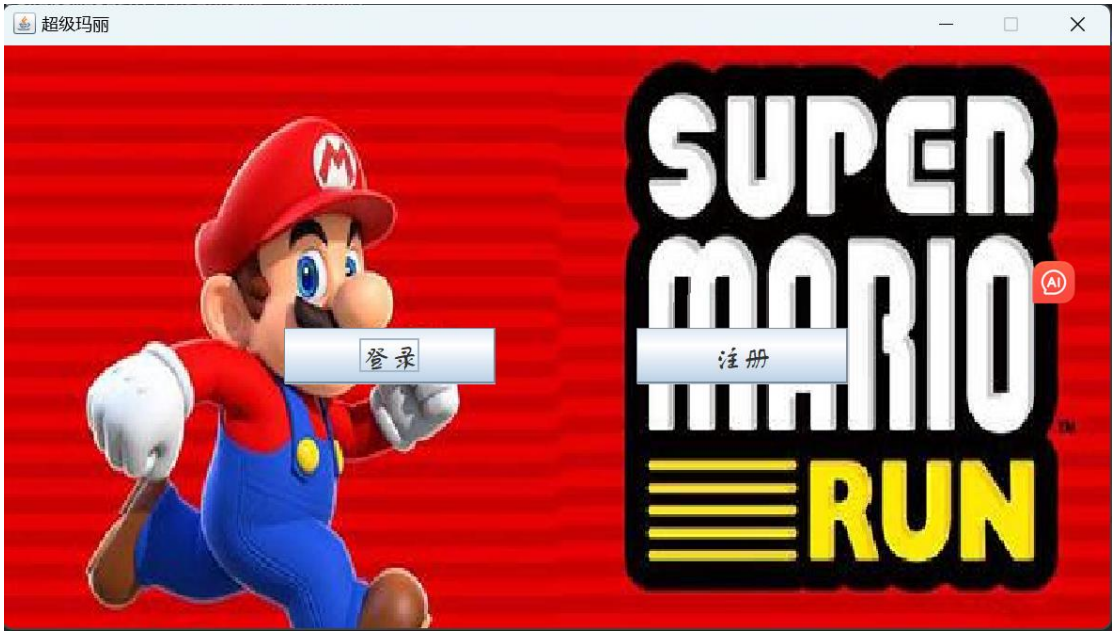


图 18 开始界面测试

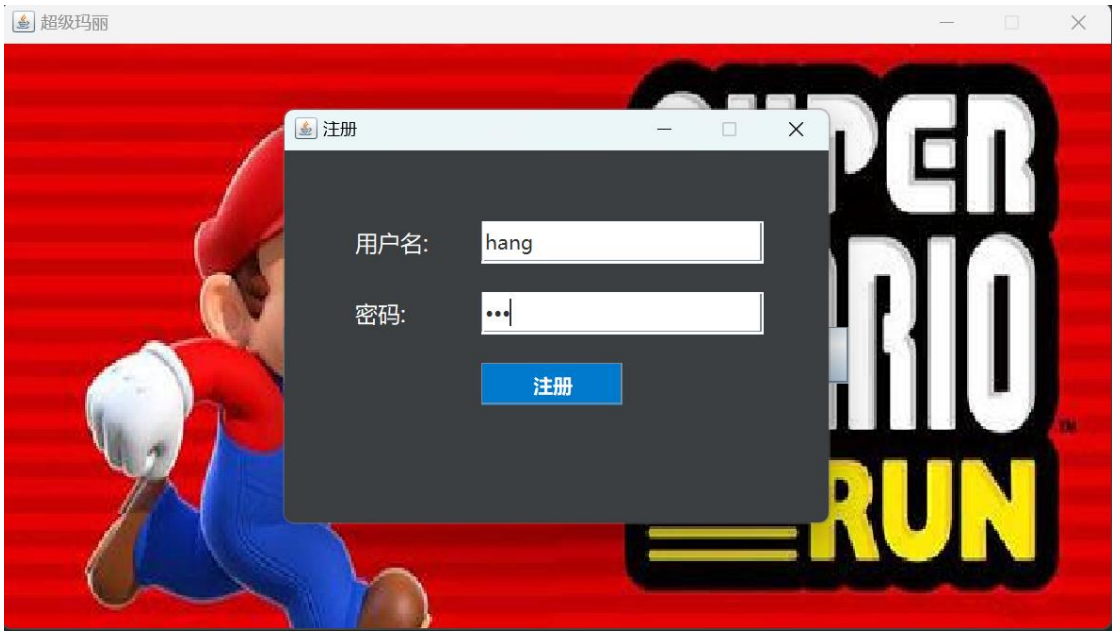


图 19 注册界面测试

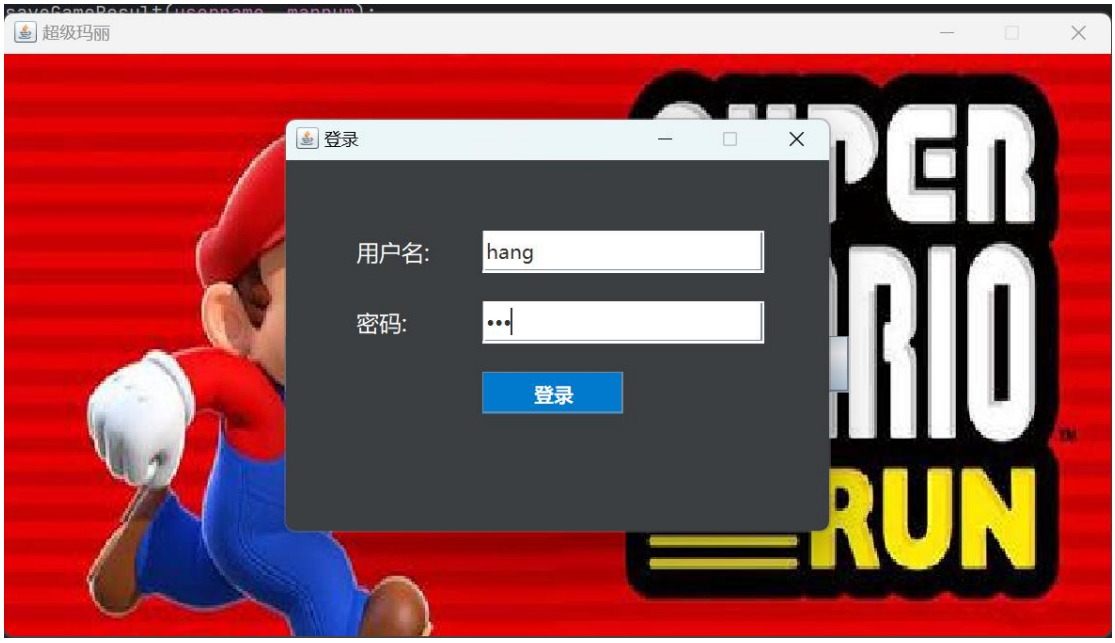


图 20 登录界面测试

图 21 开始游戏界面测试

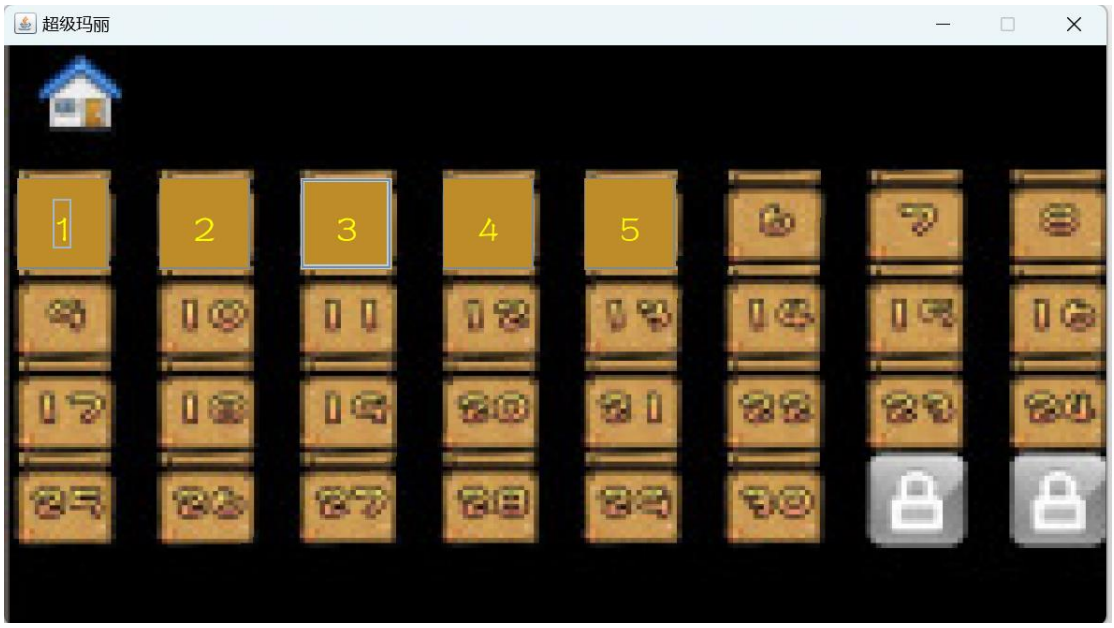


图 22 关卡选择界面测试

图 23 游戏界面测试

图 24 暂停功能测试

图 25 结束功能测试

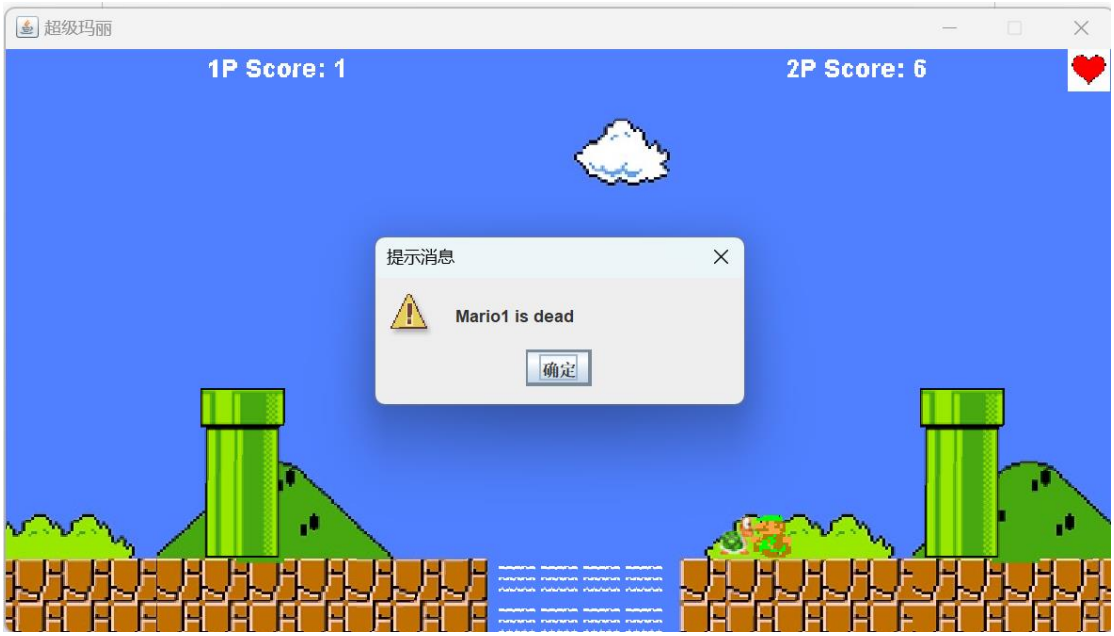


图 26 死亡测试

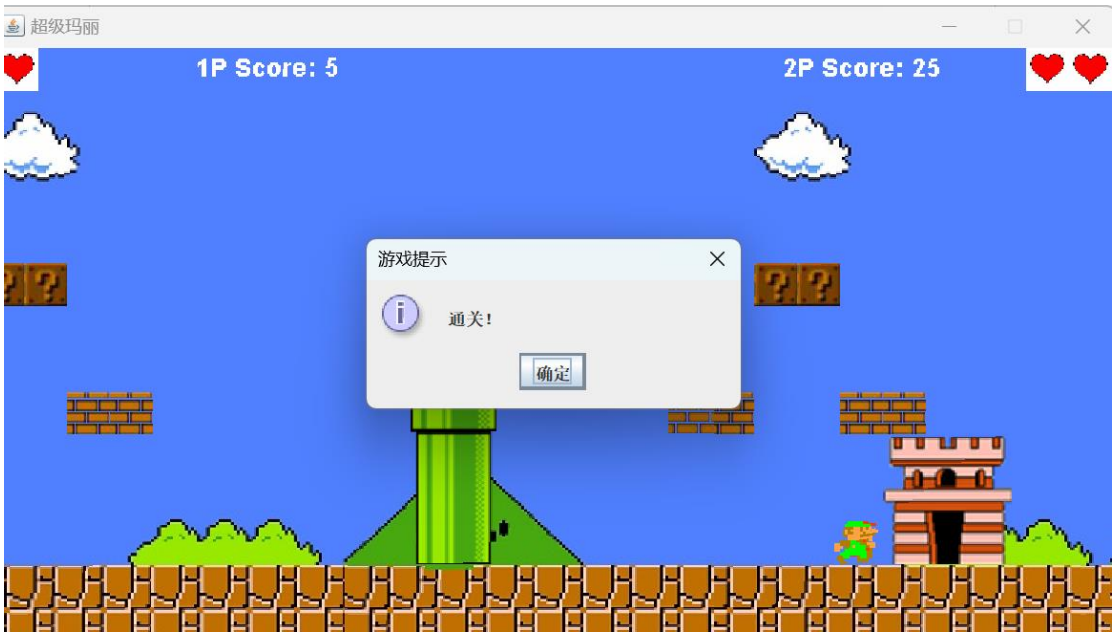


图 27 通关测试

对象 users @mario (hang的mysql) - 表

表配置文件 开始事务 单元格编辑器 筛选 & 排序 列 数据分析 工具

id # int(11)	username varchar(50)	password_hash varchar(255)	created_at timestamp	salt varchar(255)
1	zhangsan	V+hrLaAjA+Blsg1CoPq/	2024-11-30 04:53:53	P8W9rFj6i19tV8pDhSA=
2	lisi	fYa03ObumsxUGCLohS)	2024-11-30 05:37:13	TKB8Z3E98JyKYgr3Aqw
3	11	NB2HjdX0FzMlv6gk3m/	2024-12-04 10:01:18	3zttD153mjET2s2id/PK\
4	1	lLtc4lkkxV7Ar8xoi2TuQ/	2024-12-09 04:52:19	m1lP6kRvvtwBZ8Q0cyB
5	2	FHrAPRfVJRGHjUR6ml7	2024-12-09 05:46:18	TxwZCtjcincHn4yMlLlkh
6	hang	Q0wazKdn8O1CheGNvi	2024-12-12 17:19:35	5YS4fh/rCjlR8LVxLVSpv
7	hang2	0x6R+kgAvTHUxYsU0Q	2024-12-14 21:32:15	/+EHzB2B90tWnUWgpl
8	hang3	CDI8lTmMTxvBwrBug6p	2024-12-14 22:58:44	gGtxT9dfobZqQ6nV4dk

图 28 用户信息数据表

对象 users @mario (hang的mysql) - 表 game_results @mario (hang的mysql...

表配置文件 开始事务 单元格编辑器 筛选 & 排序 列 数据分析 工具

id # int(11)	userid # int(11)	username varchar(100)	player1_score # int(11)	player1_blood # int(11)	player2_score # int(11)	player2_blood # int(11)	result varchar(50)	timestamp datetime
56	6	Player1	2	3	10	0	Mario2 is dead	2024-12-14 22:40:44
57	6	Player1	21	2	15	3	Success	2024-12-14 22:41:51
58	6	Player1	6	0	10	3	Mario1 is dead	2024-12-14 23:29:27
59	6	Player1	1	0	5	3	Mario1 is dead	2024-12-14 23:31:43
60	6	hang	1	0	5	3	Mario1 is dead	2024-12-14 16:39:15
61	6	hang	11	3	15	3	Success	2024-12-14 16:42:43
62	7	hang2	1	2	5	0	Mario2 is dead	2024-12-14 16:53:42
63	7	hang2	2	0	15	3	Mario1 is dead	2024-12-14 17:01:22
64	7	hang2	1	3	5	0	Mario2 is dead	2024-12-14 17:04:01
65	8	hang3	0	3	0	0	Mario2 is dead	2024-12-15 02:17:11
66	6	hang	17	0	17	2	Mario1 is dead	2024-12-15 02:52:46
67	6	hang	17	0	16	2	Mario1 is dead	2024-12-15 02:59:43
68	6	hang	6	3	5	0	Mario2 is dead	2024-12-15 03:01:29
69	6	hang	0	3	0	0	Mario2 is dead	2024-12-15 03:03:30
70	6	hang	5	3	5	0	Mario2 is dead	2024-12-15 03:04:53
71	6	hang	6	0	7	3	Mario1 is dead	2024-12-15 03:24:21
72	6	hang	6	3	13	0	Mario2 is dead	2024-12-15 03:25:33
73	6	hang	12	0	7	2	Mario1 is dead	2024-12-15 03:26:44
74	6	hang	13	0	7	3	Mario1 is dead	2024-12-15 03:35:21
75	6	hang	0	3	5	0	Mario2 is dead	2024-12-15 03:36:36
76	6	hang	0	3	0	0	Mario2 is dead	2024-12-15 03:44:25
77	6	hang	10	0	0	3	Mario1 is dead	2024-12-15 03:45:30
78	6	hang	1	0	6	3	Mario1 is dead	2024-12-15 03:50:31
79	6	hang	6	0	6	2	Mario1 is dead	2024-12-15 03:50:50
80	6	hang	2	3	11	0	Mario2 is dead	2024-12-15 03:51:25
81	6	hang	15	0	5	3	Mario1 is dead	2024-12-15 04:13:29
82	6	hang	8	0	17	3	Mario1 is dead	2024-12-15 07:26:37
83	6	hang	1	0	0	3	Mario1 is dead	2024-12-15 07:29:47
84	6	hang	1	0	6	1	Mario1 is dead	2024-12-15 09:05:53
85	6	hang	5	1	25	2	Success	2024-12-15 09:07:10

图 29 游玩记录数据表

对象 users @mario (hang的mysql) - 表 game_results @mario (hang的mysql... game_records @mario (hang的mys...

表配置文件 开始事务 单元格编辑器 筛选 & 排序 列 数据分析 工具

id # int(11)	username varchar(50)	score # int(11)	blood # int(11)	created_at timestamp	level # int(11)
1	2	0	3	2024-12-08 21:48:54	1
2	hang	13	3	2024-12-14 01:38:38	1
3	hang3	0	3	2024-12-15 02:17:11	1

图 30 用户闯关进度数据表

5. 项目展望

增加关卡：增加更多的关卡，增加游戏的挑战性。

增加角色：增加更多的角色，增加游戏的趣味性。

增加道具：增加更多的道具，增加游戏的策略性。

增加音效：增加更多的音效，增加游戏的沉浸感。

增加网络功能：增加网络对战功能，增加游戏的互动性。