期刊论文管理系统

期刊论文管理系统旨在高效管理期刊论文的作者、评审专家及论文信息，提升期刊管理工作的整体效率和质量。本系统通过设计并在DBMS上实现合理的数据库结构，涵盖作者、评审专家、论文等核心数据表，并采用面向对象技术实现了作者管理、评审专家管理、论文管理等功能模块。系统功能包括添加、修改、删除和查询作者及评审专家信息，管理论文的基本信息和评审状态。用户界面友好，提供直观的操作方式，支持多条件组合查询，实现精准的数据检索。系统还具备数据访问控制和备份恢复功能，确保数据安全，并具有良好的扩展性和可维护性。

**关键词：**期刊论文管理；数据库；面向对象；数据管理；评审流程

目 录

[期刊论文管理系统 I](#_Toc166216013)

[目 录 II](#_Toc166216014)

[第一章 绪 论 1](#_Toc166216015)

[1.1 期刊论文管理系统介绍 1](#_Toc166216016)

[1.1.1背景与意义 1](#_Toc166216017)

[1.1.2功能模块 1](#_Toc166216018)

[1.2 Qt开发框架介绍 1](#_Toc166216019)

[第二章 系统分析 2](#_Toc166216020)

[2.1 功能分析 2](#_Toc166216021)

[2.1.1基本功能 2](#_Toc166216022)

[2.1.2拓展功能 2](#_Toc166216023)

[2.2 类关系分析 2](#_Toc166216024)

[第三章 详细设计及实现 4](#_Toc166216025)

[3.1场景与视窗 4](#_Toc166216026)

[3.2 背景类 4](#_Toc166216027)

[3.3 小球类 4](#_Toc166216028)

[3.4 弹珠类 4](#_Toc166216029)

[3.5 平板类 4](#_Toc166216030)

[3.6 生命值及关卡数显示 5](#_Toc166216031)

[3.6游戏控制类 5](#_Toc166216032)

[3.6.1各元素的协调 5](#_Toc166216033)

[3.6.2键盘事件 5](#_Toc166216034)

[3.6.3碰撞检测 5](#_Toc166216035)

[.......... 5](#_Toc166216036)

[3.6.4游戏状态的切换 5](#_Toc166216037)

[...... 5](#_Toc166216038)

[第四章 测试 5](#_Toc166216039)

[4.1 游戏应用环境的构建 5](#_Toc166216040)

[4.1.1游戏需要的硬件环境 5](#_Toc166216041)

[... 5](#_Toc166216042)

[4.1.2 游戏需要的软件环境 5](#_Toc166216043)

[........ 5](#_Toc166216044)

[4.2 游戏界面显示 5](#_Toc166216045)

[4.3 平板移动测试 6](#_Toc166216046)

[........ 6](#_Toc166216047)

[4.4 小球碰撞测试 6](#_Toc166216048)

[........ 6](#_Toc166216049)

[4.5 游戏失败测试 6](#_Toc166216050)

[......... 6](#_Toc166216051)

[4.6 游戏通关测试 6](#_Toc166216052)

[........ 6](#_Toc166216053)

[4.7 小结 6](#_Toc166216054)

[参考文献 7](#_Toc166216055)

第一章 绪 论

1.1 期刊论文管理系统介绍

1.1.1背景与意义

期刊论文管理系统是为了提升学术期刊管理的效率和质量而设计的。传统的期刊论文管理主要依赖手工操作，存在着信息散乱、效率低下、错误频发等问题。随着信息技术的发展，采用信息化手段进行期刊论文管理成为必然趋势。通过建立系统化、信息化的管理平台，可以有效解决传统管理中的各种问题，提高期刊论文的管理效率和准确性，促进学术交流和科研成果的传播。

1.1.2功能模块

期刊论文管理系统主要包括以下几个功能模块：

1. 作者管理：包括作者基本信息的添加、修改、删除和查询等功能。
2. 评审专家管理：包括评审专家基本信息的添加、修改、删除和查询等功能。
3. 论文管理：包括论文基本信息的录入、修改、删除和查询，以及评审状态的管理。

1.2开发框架介绍

**1.2.1 Spring Boot框架**

**Spring Boot是由Pivotal团队开发的一个基于Spring框架的快速开发框架。它通过简化配置和内置常用功能，极大地提升了开发效率。Spring Boot可以自动配置Spring应用程序，并提供了内置的Tomcat服务器，使开发者可以专注于业务逻辑的实现，而不需要过多关注配置细节。**

**1.2.2 Vue.js框架**

**Vue.js是一个由尤雨溪开发的前端JavaScript框架，旨在通过简洁易用的API和灵活的组件系统来构建用户界面。Vue.js采用渐进式架构，既可以作为一个轻量级的库使用，也可以通过其生态系统构建复杂的单页面应用程序。Vue.js的核心特性包括双向数据绑定、虚拟DOM和组件化开发，使开发者能够高效地构建和维护前端界面。**

**1.2.3 MySQL数据库**

**MySQL是一个开源的关系型数据库管理系统，由瑞典MySQL AB公司开发。MySQL以其快速、可靠、可扩展性强的特点，成为了许多Web应用开发中的首选数据库。它支持标准的SQL查询语言，提供了多种存储引擎，具备事务处理、外键约束和视图等高级功能。MySQL数据库在期刊论文管理系统中，用于存储和管理系统中的各种数据，如用户信息、作者信息、评审专家信息和论文信息等。**

第二章 系统分析

2.1 功能分析

2.1.1基本功能

1.用户登录

* 用户输入自己的ID和密码登录系统。系统验证用户身份，确保用户信息的安全性和唯一性。

2.个人主页

* 显示用户的个人信息，包括用户名、性别、联系方式等。
* 提供编辑个人信息功能，用户可以更新自己的个人资料。
* 提供退出功能，用户可以安全退出系统。。
* 提供注销功能，用户可以注销自己的账户。

3.管理员管理

* 管理员基本信息的录入、修改、删除和查询功能。
* 管理员可以管理系统中的其他用户和信息，确保系统的正常运行。

4.用户管理

* 用户基本信息的录入、修改、删除和查询功能。
* 管理员可以管理普通用户的账户信息，确保用户数据的准确性和完整性。

5.作者管理

* 作者基本信息的录入、修改、删除和查询功能。
* 关联作者与其发表的论文，便于管理和查询。

6.评审专家管理

* 评审专家基本信息的录入、修改、删除和查询功能。
* 关联评审专家与其评审的论文，便于管理和查询。

7.论文管理

* 论文基本信息的录入、修改、删除和查询功能。
* 标注论文的作者和评审专家，跟踪论文的评审状态。
* 评审状态管理，支持多状态管理（如未评审、评审中、评审完成等）。

2.1.2拓展功能

1.权限控制

* 不同角色（如管理员、普通用户、评审专家等）具有不同的操作权限，确保系统的安全性和数据的保密性。

2.多字段查询

* 支持多条件组合查询，如按作者、评审专家、论文标题等进行筛选，提高数据检索的精准性和效率。

3.Vue界面刷新保存

* 实现Vue.js前端界面的自动刷新和数据保存，提升用户体验和操作便捷性。

2.2 类关系分析

整个期刊论文管理系统大致分为前端、后端和数据库三部分。各部分之间通过RESTful API进行通信，以确保数据的一致性和操作的高效性。

前端使用Vue.js框架进行开发，主要负责用户界面的显示和交互逻辑。前端主要类和组件包括：

App.vue：主应用组件，作为整个应用的入口，包含全局配置和路由。

Main.vue：主要界面组件，包含导航栏和主要内容区域。

Login.vue：登录组件，处理用户的登录操作。

Home.vue：首页组件，展示用户的基本信息和系统概况。

UserComponents.vue：用户管理相关组件，包括用户信息的增删改查功能。

AuthorComponents.vue：作者管理相关组件，包括作者信息的增删改查功能。

ReviewerComponents.vue：评审专家管理相关组件，包括评审专家信息的增删改查功能。

PaperComponents.vue：论文管理相关组件，包括论文信息的增删改查和评审状态管理功能。

后端使用Spring Boot框架进行开发，主要负责业务逻辑的实现和数据的处理。后端主要类和服务包括：

UserController：处理用户相关的请求，如注册、登录、获取用户信息等。

AuthorController：处理作者相关的请求，如作者信息的增删改查。

ReviewerController：处理评审专家相关的请求，如评审专家信息的增删改查。

PaperController：处理论文相关的请求，如论文信息的增删改查和评审状态管理。

UserService：包含用户相关的业务逻辑，实现用户信息的管理和验证。

AuthorService：包含作者相关的业务逻辑，实现作者信息的管理。

ReviewerService：包含评审专家相关的业务逻辑，实现评审专家信息的管理。

PaperService：包含论文相关的业务逻辑，实现论文信息的管理和评审流程的跟踪。

数据库使用MySQL进行数据存储和管理。数据库主要表结构包括：

User：存储用户信息，包括用户ID、用户名、密码、联系方式等。

Author：存储作者信息，包括作者ID、姓名。

Reviewer：存储评审专家信息，包括评审专家ID、姓名。

Paper：存储论文信息，包括论文ID作者ID、审稿人、审稿人、评审状态等。

三大类的协作工作

1. 前端与后端

前端通过Axios或Fetch API等HTTP客户端库向后端发送请求，例如用户登录请求、获取用户信息请求、提交论文信息请求等。后端接收到请求后，通过相应的控制器（如UserController、AuthorController等）处理请求，调用业务服务层（如UserService、AuthorService等）完成相应的业务逻辑处理。处理完成后，后端将结果以JSON格式返回给前端，前端再将数据渲染到用户界面上。例如：

用户登录时，前端将用户输入的ID和密码发送到后端的UserController。后端验证用户信息后，将验证结果返回给前端，前端根据验证结果决定是否跳转到主页或显示错误信息。

用户在个人主页修改个人信息时，前端将修改后的信息发送到后端的UserController。后端更新数据库中的用户信息后，将更新结果返回给前端，前端根据结果更新界面显示。

2. 后端与数据库

后端通过JPA（Java Persistence API）或其他ORM（对象关系映射）框架与MySQL数据库交互。每个控制器在处理请求时，调用相应的服务层，服务层通过Repository接口与数据库进行交互，执行数据的增删改查操作。例如：

用户登录时，UserService会从数据库中查询用户信息，验证用户ID和密码是否匹配。

用户修改个人信息时，UserService会更新数据库中的用户信息记录。

3. 前端与数据库

前端不直接与数据库交互，而是通过后端间接与数据库进行交互。前端的每个操作（如用户登录、信息修改、数据查询等）都会触发一个HTTP请求，后端接收到请求后，通过服务层与数据库交互，将结果返回给前端。通过这种方式，确保了前端操作的安全性和数据的一致性。

第三章 详细设计及实现

3.1场景与视窗

在MainWindow类里，创建场景QGraphicsScene类，视窗QGraphicsView类，游戏控制gamecontroller类的实例。

.......

3.2 背景类

本程序的背景是一个图片插入进去的。因为视窗坐标的局限性，背景作为画刷覆盖scene场景相对更困难，需要重写类中的函数，所以直接使用QGraphicsItem来作为背景图片的载体，

.....

3.3 小球类

小球类继承于QGraphicsItem，由于功能需要，需要传入当前的游戏控制类实例作为参数。

........

3.4 弹珠类

弹珠类是继承于QGraphicsItem，由于弹珠是静止不动的，所以只需要重写三个虚函数。

.......

3.5 平板类

平板类继承于QGraphicsItem，需要重写三个虚函数。平板的绘制类似于小球，直接插入图片到画刷画出图形。因为要进行与小球的碰撞检测，所以shape函数要返回精确轮廓。

.......

3.6 生命值及关卡数显示

.......

3.6游戏控制类

.......

3.6.1各元素的协调

......

3.6.2键盘事件

......

3.6.3碰撞检测

..........

3.6.4游戏状态的切换

......

第四章 测试

4.1 游戏应用环境的构建

4.1.1游戏需要的硬件环境

...

4.1.2 游戏需要的软件环境

........

4.2 游戏界面显示

......

4.3 平板移动测试

........

4.4 小球碰撞测试

........

4.5 游戏失败测试

.........

4.6 游戏通关测试

........

4.7 小结

.....

参考文献

1. 百度百科.打砖块游戏. https://baike.baidu.com/item/打砖块游戏/9093237?fr=aladdin.
2. 百度百科.Qt. https://baike.baidu.com/item/qt/451743?fr=aladdin.
3. 杨连刚.Qt开发的国内外知名软件[EB/OL].2018-04-19.
4. wiki百科.Qt for Python Signals and Slots. https://wiki.qt.io/Qt\_for\_Python\_Signals\_and\_Slots.
5. 邱升.求点关于直线坐标的一种简捷方法[J].中学数学, 2000.