少儿启蒙编程游戏
软件分析设计文档

计蒜客粉丝队[1]

[1]于纪平、朱佳豪、路橙、林锦坤、于志竟成

Contents

第 1 章. 引言

1.1 标识

本文档适用于计蒜客粉丝队开发的“少儿编程启蒙游戏”1.0 版本。

1.2 系统概述

“少儿启蒙编程游戏”是一款基于 Web 的、以帮助处于小学阶段的儿童进行编程学习为目标的游戏。该系统由计蒜客粉丝队于 2017 年 10 月至 2018 年 1 月间开发和维护。计蒜客是该项目的需求方，清华大学计算机系软件工程课程为该项目提供支持。

1.3 文档概述

本文档是“少儿启蒙编程游戏”系统的软件分析设计文档。本文档包含了以下内容：

- 软件开发过程中根据需求方说明进行的需求分析的结果
- 根据需求得到的系统结构设计方案
- 系统各个模块间接口的设计
- 系统各个模块的具体设计
- 数据库的具体设计
- 对系统进行的软件测试所使用的方法及测试的结果

第 2 章. 需求分析

2.1 功能需求

2.1.1 需求列表

用户系统

- 用户通过账号和密码进行注册和登录
- 若用户忘记密码，可通过邮箱验证后找回密码
- 已登录的用用户可以修改密码

关卡系统

用户登录后可以看到关卡选择页面

- 用户可以重复进入已经通过的关卡，已通过的关卡会保留用户通过时的代码
- 用户可以进入最近一个未通过的关卡进行游戏，其他未通过的关卡无法进入（但前五关无论如何均可进入）
- 每个关卡应当存在标程
- 每个关卡有通过后的评级，满分为 3 星，通过将用户使用的 Blockly 控件块数与标程的块数相比较得出。用户使用的块数小于等于标程块数，通过后获得 3 星；大于标程块数但小于标程块数的两倍，通过后获得 2 星；大于标程块数的两倍，通过后获得 1 星；未通过为 0 星

- 每个关卡通过后，可以将解法分享给其他人，其他人（无论登录还是未登录），都可以进入该关卡并执行该用户分享的代码，但不允许修改
- 在游戏中，执行代码后会对结果进行判断，若用户未通过关卡，应允许用户继续编写代码或重新开始这一关卡

页面布局

用户选择关卡后，进入游戏

- 页面左侧为游戏界面，包含地图、角色、物品和地图元素的显示
- 页面右侧为操作区，用户使用 Blockly 控件组合定义好的语句控制角色的动作
- 每个关卡有一个既定的完成目标
  - 结合程序设计的思想进行关卡设计
  - 关卡的目标应是固定的，类似「兔妮妮」，通关条件是兔子避过各种障碍吃掉所有胡萝卜

地图编辑器

用户可以使用地图编辑器自行设计关卡。

- 地图编辑器左侧为地图界面，右侧列出所有可用的角色、物品和地图元素
- 用户可以在地图编辑器中使用所有的角色、物品和地图元素
- 地图编辑器不关心用户设计的地图是否有解
- 用户设计的地图在保存后可以分享给其他人，其他人可以完成该关卡
- 用户设计的关卡只存在通过和不通过两种状态，无须评级
- 用户设计的关卡通过后，可以将解法分享给其他人，其他人（无论登录还是未登录），都可以进入该关卡并执行该用户分享的代码，但不允许修改

2.1.2 用例列表

所有的用例如下：

用户部分

- 注册
- 登录
- 退出登录
- 找回密码
- 修改个人信息（含修改密码）

游戏部分

- 玩一个关卡
- 分享解法
- 查看（被分享的）解法

地图部分

- 关卡列表（主线关卡或分享的关卡）
- 创建地图
- 编辑地图（含设置地图属性、分享等）

2.1.3 用例图

UML 用例图如图 2.1。

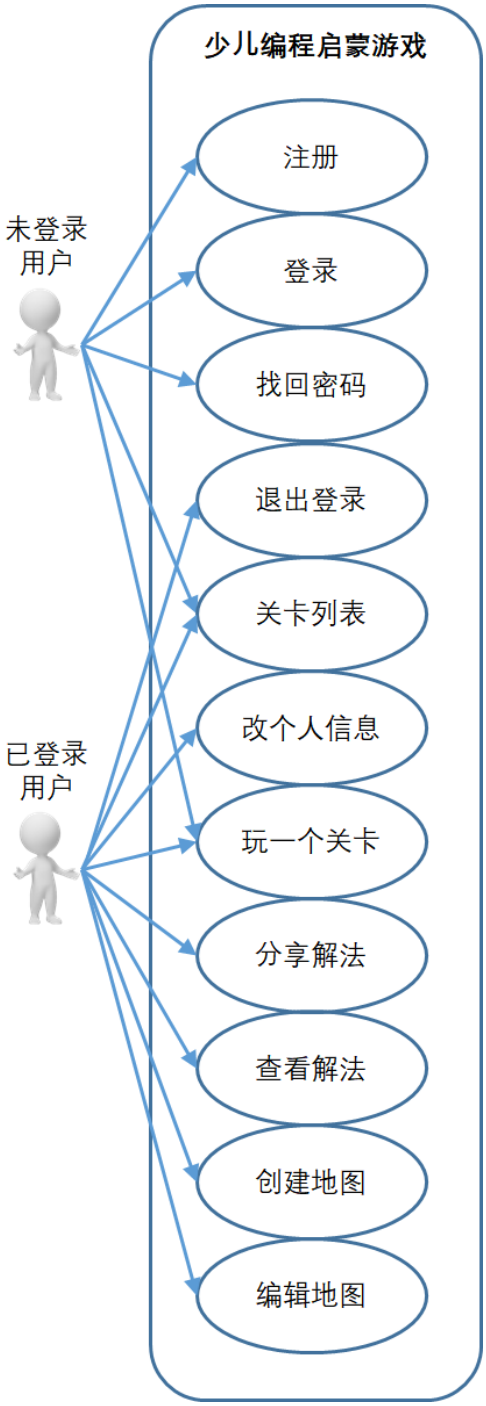

Figure 2.1: UML 用例图

2.1.4 用例描述

注册

说明	新用户输入邮箱、昵称等个人信息，并设置密码，成为已注册的用户。
前提条件	用户当前未登录
触发条件	用户希望注册，并点击了页面“注册”按钮
流程	正常流程 • 系统提示用户输入邮箱、昵称、密码、重复密码 • 用户输入上述信息，提交给系统 • 系统检查用户提供的信息，如果出现以下情况之一，则转异常流程处理 – 邮箱格式不合法 – 已有其他用户使用了相同的邮箱 – 已有其他用户使用了相同的昵称 – 密码与重复密码不一致 • 系统将提供的信息存入用户数据库 • 系统提示用户注册成功 异常流程 • 系统根据出错原因，向用户提示对应的错误信息 • 系统提示用户重新输入注册所需的信息
后置条件	用户可以登录系统

登录

说明	用户输入邮箱、密码的信息进行登录
前提条件	用户当前已注册但未登录
触发条件	用户希望登录，并点击了页面“登录”按钮
流程	正常流程 • 系统提示用户输入邮箱、密码 • 用户输入上述信息，提交给系统 • 系统检查用户提供的信息，根据这些信息查询用户数据库，如果出现以下情况之一，则转异常流程处理 – 数据库中未查到该邮箱对应的用户 – 提供的密码与数据库中的密码不同 • 系统提示用户登录成功 异常流程 • 系统根据出错原因，向用户提示对应的错误信息 • 系统提示用户重新输入登录所需的信息
后置条件	用户成功登录系统，可以进行后续上传得分、分享解法等需要登录才可以进行的操作

退出登录

说明	用户退出登录
前提条件	用户已登录
触发条件	用户希望退出登录，并点击了页面“退出登录”按钮
流程	正常流程 • 系统记录用户退出了登录
后置条件	用户退出登录，相关特权同时取消

找回密码

说明	用户忘记了自己的密码，通过发送邮件的方式重设自己的密码
前提条件	用户当前已注册但未登录
触发条件	用户希望找回密码，并点击了页面“找回密码”按钮
流程	正常流程 • 系统提示用户输入邮箱 • 用户输入上述信息，提交给系统 • 系统检查用户提供的信息，根据这些信息查询用户数据库，如果数据库中未查到该邮箱对应的用户，则转异常流程处理 • 系统生成一随机字符串，作为该用户的新密码 • 系统将上述新密码保存到用户数据库 • 系统向该邮箱发送新密码 • 系统向用户提示找回密码邮件已发送 • 异常流程 • 系统提示不存在对应的用户 • 系统提示用户重新输入找回密码所需的信息
后置条件	用户可以通过查询邮箱的方式获得自己的新密码，并用此密码登录系统（用户可以之后通过其他流程将密码修改为自定义的密码）

关卡列表

说明	用户浏览关卡列表（包括主线关卡列表、分享的所有自定义地图列表、用户自己创建的地图列表）
前提条件	无特殊条件
触发条件	用户希望浏览关卡列表，并点击了页面“主线关卡”按钮、“所有地图”按钮或“我的地图”按钮
流程	正常流程 • 系统检查用户希望获取的关卡列表（可能还有页码），查询地图数据库获取用户当前有权限浏览的所有关卡，如果找不到合法的关卡，则转异常流程处理 – 所有的主线关卡都可以被浏览 – 用户可以浏览自己创建的所有地图 – 只有被分享的自定义地图才可以被其他人浏览 • 系统向用户显示当前页（默认为第 1 页）的关卡，并根据当前页码与总页数等信息决定是否提示用户可以浏览上一页或下一页关卡 • 用户看到关卡列表，如果用户希望浏览上一页或下一页地图，则更新当前页码，并重复执行该流程 异常流程 • 系统提示找不到可见的地图 • 系统向用户显示一个空白列表
后置条件	用户可以选择显示的某个关卡，进入其中进行游戏

修改个人信息

说明	用户修改自己的昵称、密码等个人信息
前提条件	用户已经登录
触发条件	用户希望修改自己的个人信息，并点击了页面“用户中心”按钮
流程	正常流程 • 系统提示用户输入原密码、新昵称、新密码、重复新密码等信息 • 用户输入上述信息，提交给系统 • 系统检查用户提供的信息，根据这些信息查询用户数据库，如果出现以下情况之一，则转异常流程处理 – 提供的原密码与数据库中的密码不同 – 提供的新昵称已被其他用户使用 – 提供的新密码与重复新密码不一致 • 系统根据提供的信息修改用户数据库 • 系统提示用户修改成功 异常流程 • 系统根据出错原因，向用户提示对应的错误信息 • 系统提示用户重新输入修改所需的信息
后置条件	无特殊条件

玩一个关卡

说明	用户玩一个关卡
前提条件	无特殊条件
触发条件	用户从关卡列表中选择了某个关卡，或通过历史记录等方式希望玩一个关卡
流程	正常流程 • 系统检查用户希望玩的关卡，查询地图数据库，如果出现以下情况之一，则转异常流程处理 – 用户试图玩其他玩家创作但未被分享的关卡 – 用户未登录，但试图玩非前五关的主线关卡 – 用户未登录，但试图玩其他玩家分享的关卡 – 用户已登录，试图玩非前五关的主线关卡，但尚未成功通关其上一关 • 系统向用户显示游戏页面，包括 Blockly 编程界面与游戏图形画面 • (*) 用户与 Blockly 交互，完成编程操作 • 用户将代码提交系统运行 • 系统运行用户编写的代码，同时在游戏图形画面实时展示运行状态 • 如果代码执行过程中，任意时刻达成了游戏的目标，则游戏成功 • 如果出现以下情况之一，则转“游戏失败”可选流程处理 – 代码执行过程中，按照游戏规则判定为失败（例如主角死亡） – 代码执行的步数过长，超过 10000 – 代码执行完毕，但未达成游戏的目标

流程	• 系统提示用户游戏成功 • 系统根据关卡的标准 Blockly 块数与玩家的块数计算玩家获得的星级，并显示 • 如果用户已登录，则将该次游戏的程序、星级信息存入解法数据库，并提示用户可以分享该解法 • 用户仍可以修改程序并运行，即跳转到（*）处重复执行 “游戏失败”可选流程 • 系统提示用户游戏失败，并根据出错原因，向用户提示对应的错误信息 • 用户可以修改程序并运行，即跳转到（*）处重复执行 异常流程 • 系统根据出错原因，向用户提示对应的错误信息 • 系统返回上一个页面（这通常是来源的地图列表页面）
后置条件	如果玩家已登录并通关，则可以分享该次解法，且用户下次再进入该地图时，解法会自动显示在 Blockly 编程界面中；用户也可能因为此次通关而解锁进入新关卡的权限

分享解法

说明	用户分享自己某个关卡的通关解法
前提条件	用户已经登录
触发条件	用户刚刚通过了某个关卡，希望分享该解法，并点击了“分享”按钮
流程	正常流程 • 系统修改解法数据库，将该解法标记为分享的 • 系统根据该解法在数据库的编号等信息生成一个分享链接 • 系统提示用户分享成功，并显示生成的分享链接
后置条件	任何人均可以通过该分享链接查看该次解法

查看解法

说明	用户查看分享的解法
前提条件	无特殊条件
触发条件	用户访问了一个被分享的解法链接
流程	正常流程 • 系统根据链接中的编号等信息查询解法数据库，如果提供的编号不合法，则转异常流程处理 • 系统向用户显示查到的 Blockly 程序，并显示一游戏图形界面 • 用户可以运行该程序，系统将会在游戏图形界面实时展示运行状态（用户不能进行编程操作） 异常流程 • 系统提示解法查看出错 • 系统向用户显示一空白页面
后置条件	无特殊条件

创建地图

说明	用户创建一个新的地图
前提条件	用户已经登录
触发条件	用户希望创建新的地图，并点击“地图编辑器”按钮
流程	正常流程 • 系统显示地图编辑器页面，包括地图编辑器图形界面和交互表单 • 用户通过与表单交互在地图上进行创建、修改、删除元素等编辑操作，并设置是否分享 • 用户声明编辑完毕 • 系统将该地图插入到地图数据库中 • 系统提示地图创建成功
后置条件	该用户之后可以在“我的地图”列表中看到该地图，并对此地图进行编辑、分享等操作；如果已分享，则其他用户可以在“所有地图”列表中看到该地图

编辑地图

说明	用户编辑已创建的地图（含设置分享操作）
前提条件	用户已经登录
触发条件	用户通过选择“我的地图”列表中的地图，希望编辑该地图
流程	正常流程 • 系统查询地图数据库，显示地图编辑器页面，包括地图编辑器图形界面和交互表单，显示地图原有的状态 • 用户通过与表单交互在地图上进行创建、修改、删除元素等编辑操作，并设置是否分享 • 用户声明编辑完毕 • 系统将该地图在地图数据库中修改 • 系统提示地图修改成功
后置条件	如果用户分享了该地图，则其他用户可以在“所有地图”列表中看到该地图

2.2 非功能需求

我们设计了 10 个主线关卡，以如下方式体现以下程序设计的思想：

- 第 1、2、3 关：游戏基本设定，基本“前进”Blockly 块介绍
- 第 4 关：顺序结构程序设计
- 第 5、6 关：循环结构程序设计
- 第 7 关：循环与条件结构程序设计
- 第 8 关：函数与递归
- 第 9、10 关：上述各部分综合训练与应用

每个主线关卡也设定了其允许使用的 Blockly 块的集合。

第 3 章. 体系结构设计

3.1 模块划分

我们将系统划分为以下模块:

- 游戏相关的模块
  - 游戏页面：玩游戏的主页面
  - 游戏界面：游戏的 3D 图形显示界面
  - 游戏逻辑：处理游戏的逻辑部分并指示游戏界面运行
  - Blockly：与 Blockly 交互并指示游戏逻辑运行
- 地图编辑器相关的模块
  - 地图编辑器页面：地图编辑器的主页面
  - 地图编辑器显示界面：地图编辑器的主页面 3D 图形显示界面
- 其他页面
  - 关卡列表页面（主线关卡列表、分享的地图列表、我的地图列表）
  - 修改用户信息页面
- 后端

3.2 系统行为描述

主要通过用户、地图、游戏等各画一个 UML 序列图来描述（分别如图 3.1、3.2、3.3所示）。

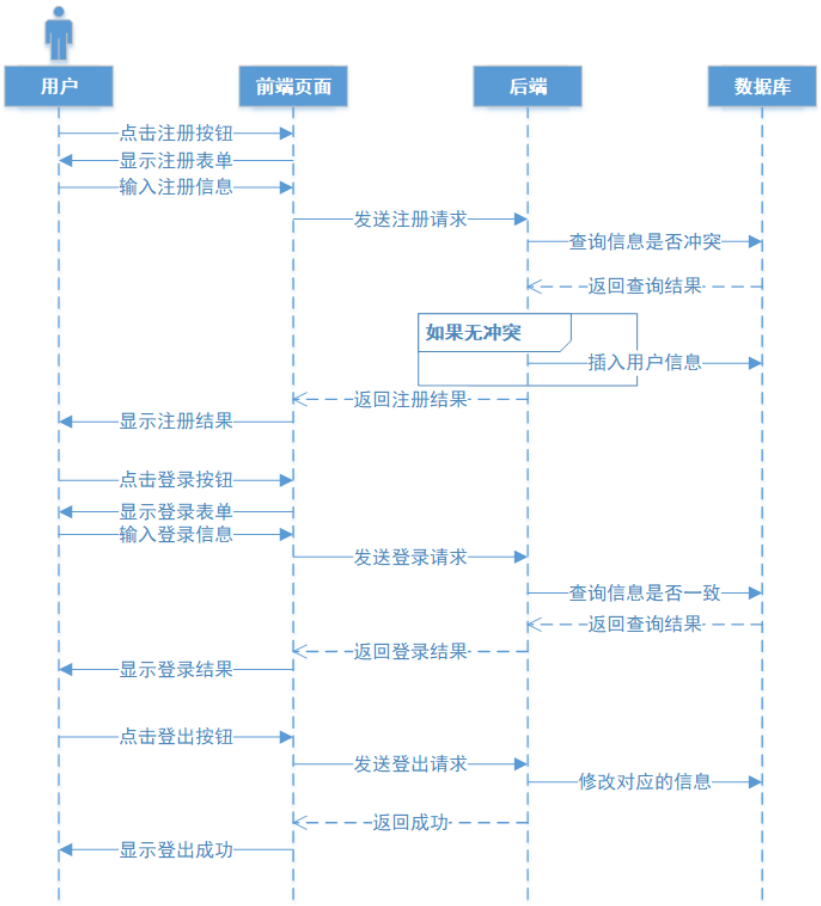

Figure 3.1: 用户操作 UML 序列图

3.3 系统模块图

游戏系统的模块图描述如图 3.4所示。

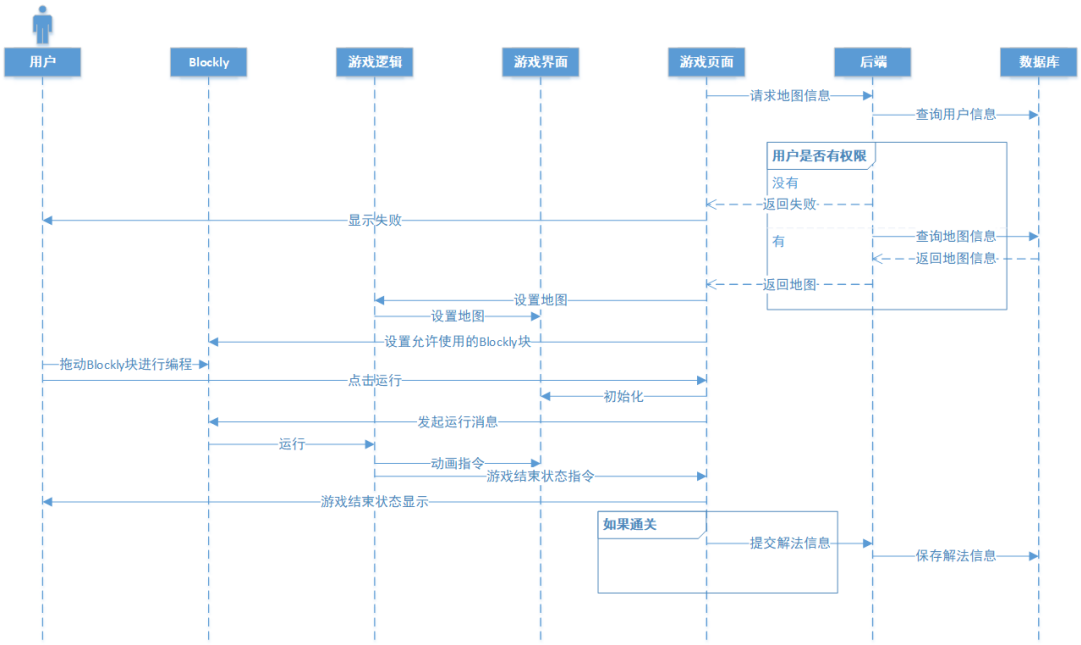

Figure 3.2: 地图操作 UML 序列图

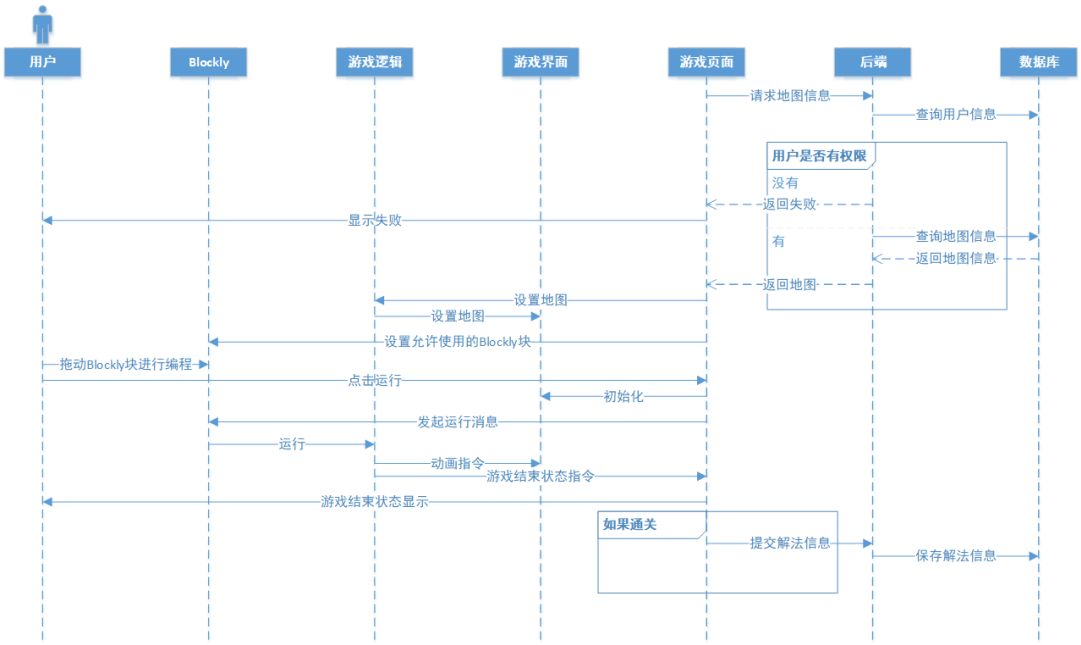

Figure 3.3: 游戏操作 UML 序列图

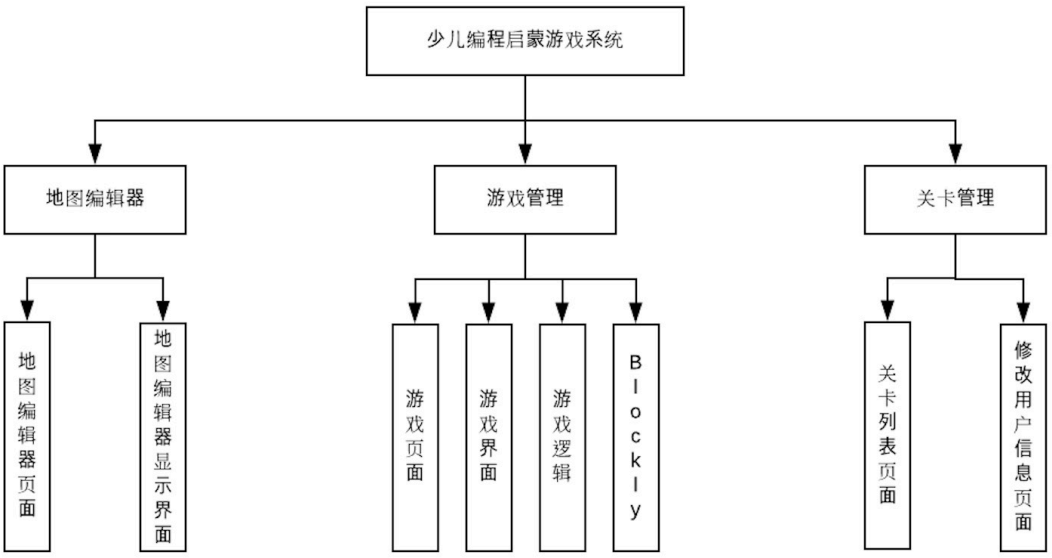

Figure 3.4: 系统模块划分

第 4 章. 模块间主要接口设计

4.1 地图编辑器显示界面——地图编辑器页面

4.2 前端各页面——后端

API 根 URL：`/api/0.1/`。API 均通过 JSON 格式交换数据。传入及返回数据均为 JSON 对象。

API 一览表如下：

URL	GET	POST	PUT
`/user/`	获取用户列表	创建用户	无
`/user/<user_id>/`	获取用户信息	无	修改用户信息
`/forget/`	无	重置用户的密码，并将密码发至用户邮箱	无
`/modify/`	获取当前用户信息	修改用户信息	无
`/token/`	无	登录获取访问令牌	无
`/map/`	获取地图列表	创建新地图	无
`/map/<map_id>/`	获取地图信息	无	修改地图信息
`/stage/`	获取关卡信息	无	无
`/solution/`	获取解法列表	创建新的解法	无
`/solution/<sol_id>/`	获取解法信息	无	修改解法信息

说明

权限 所有权限要求非公开的接口在调用时还需通过 HTTP Request Headers 额外传入：

```
Authorization: Token <token>
```

名称	类型	缺省值	描述
token	字符串	不可缺省	访问令牌

所有接口返回的 res_code 均以-1 表示缺少访问权限。

用户资源

用户数据

- id* 整型 [0,] 用户 id
- name*^ 字符串用户名（昵称）
- email*^ 字符串用户绑定的电子邮箱
- gender*^ 整型 [0, 3] 用户性别（0= 未知，1= 男，2= 女，3= 其他）
- privilege* 整型 [0, 2] 用户权限类型（0= 普通用户，1=VIP，2= 管理员）
- expiration 日期 VIP 过期的日期
- join_date 日期用户注册日期

  注：

  - 用户信息分为完整信息、简略信息和变更信息三种。上列所有字段均包含于完整信息中，带有 * 的字段包含于简略信息中，带有 ^ 的字段包含于变更信息中。
  - id 为 0 的用户称为 root 用户，其权限类型为 2（管理员）。

/user/

GET 获取用户列表。

权限要求：管理员

参数：

名称	类型	缺省值	描述
pageNo	整型 [1,]	1	分页页码
pageSize	整型 [1, 40]	20	分页每页中包含项目数量

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 1]	0	请求成功 =1，失败 =0
list	用户简略信息列表	空列表	包含分页中用户的简略信息
has_prev	整型 [0, 1]	0	是否存在上一页（0= 否，1= 是）
has_next	整型 [0, 1]	0	是否存在下一页（0= 否，1= 是）

POST 创建用户。（注意：暂时没有考虑反图灵测试）

权限要求：公开

传入数据：

名称	类型	缺省值	描述
new_user_info	用户完整信息	不可缺省	新建用户的信息
password	字符串	不可缺省	新用户的密码

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 3]	0	请求成功 =1，未知错误 =0，email 已存在 =2，数据不合法 =3

名称	类型	缺省值	描述
user_id	整型 [0,]	0	新建用户的 id

`/user/<user_id>/` 参数:

名称	类型	缺省值	描述
user_id	整型 [0,]	（不可缺省）	请求的用户的 id

PUT 更改一指定用户的信息（包括更换密码）。

权限要求: `((普通用户 or VIP) and current_user.id == user_id) or 管理员`

传入数据:

名称	类型	缺省值	描述
change_password	整型 [0, 1]	0	是否变更密码（0= 不变更，1= 变更)
new_user_info	用户变更信息	空对象	新的用户信息
old_password	字符串	空	旧密码
new_password	新密码	空	新密码

说明:

- new_user_info 中的字段可以缺省，缺省的字段将不发生更改。

返回:

名称	类型	缺省值	描述
res_code	整型 [-1, 3]	0	请求成功 =1，未知错误 =0，数据不合法 =2, 不存在该用户 =3

GET 获取某一用户的信息。

权限要求：公开

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，不存在该用户 =2
user	用户完整信息	空对象	指定用户的完整信息

/modify/

POST 更改一指定用户的信息（包括更换密码）。

权限要求：`((普通用户 or VIP) and current_user.id == user_id) or 管理员`

传入数据：

名称	类型	缺省值	描述
old_password	字符串	空	旧密码
new_password	新密码	空	新密码
gender	用户变更信息	空	新的用户信息
username	用户变更信息	空	新的用户信息

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 1]	0	请求成功 =1，未知错误 =0

GET 获取当前用户的信息。

权限要求：当前用户

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 1]	0	请求成功 =1，未知错误 =0
gender	用户性别信息	空对象	用户性别信息
username	用户名信息	空对象	用户名信息

/token/

POST 获取会话令牌。(暂时未考虑反图灵测试)

权限要求：公开

传入数据：

名称	类型	缺省值	描述
email	字符串	（不可缺省）	请求的用户的电子邮件
password	字符串	（不可缺省）	用户的密码

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 3]	0	请求成功 =1，未知错误 =0，密码不匹配 =2, 不存在该用户 =3
token	字符串	""	生成的会话令牌
user	用户简略信息	空对象	会话令牌所属用户

/forget/

POST 重置指定邮箱用户的密码。

权限要求：公开。

传入数据：

名称	类型	缺省值	描述
email	字符串	空	邮箱

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 1]	0	请求成功 =1，未知错误 =0

地图资源

地图（关卡）的具体 json 表示:

- `title`，非空字符串，地图的名称。
- `author`，用户简略信息，地图的作者。
- `n_blockly`，整数，标程用的 Blockly 块数，如果是用户自定义关卡则为 0。
- `height`，正整数，地图的高度。
- `width`，正整数，地图的宽度。

代码中二维数组，第一维是 `height` 方向，第二维是 `width` 方向，下标从 0 开始，按照从上到下和从左到右的方向增大下标（但是注意 3D 视角下有旋转）。

- `init_pos`，长度为 2 的数组，[0] 为 `height` 方向坐标，[1] 为 `width` 方向。
- `init_dir`，整数，为 16 到 19，表示初始方向。
- `init_hp`，正整数，初始生命值。
- `init_attack`，非负整数，初始攻击力。
- `init_AI_infos`，数组，每项拥有 `id`、`pos`、`dir`、`hp`、`attack` 属性，`id` 代表一个 ai 的编号（是个字符串）`pos` 与 `dir` 同上；此外，如果是后端向前端返回地图时，每项还拥有 `code` 属性（字符串），表示这个 ai 的代码；`hp` 为正整数、`attack` 为非负整数，表示生命值和攻击力。

- `instr_set`，一个长度为 100 的数组，第 i 位表示代码为 i 的指令是否可用（`false` 或 `true`），如果一个指令代码还未定义则建议先填 `false`（但是用户自定义关卡 100 位全为 true）。指令编号表见表 4.18。**目前暂时不考虑用户自定义关卡可用的指令集，即认为所有指令都可用**。
- `final_pos`，格式同 `init_pos` 或 null，表示通关状态要求主角的位置，或没有要求。

Table 4.18: 指令编号表

编号	Blockly 中标识符名	含义
0	Text(Debug)	
1	Library	
11	`gameMove`	向前移动一格
12	`gameTurn`	转向
14	`GameCW`	顺时针（向右转）
15	`GameCCW`	逆时针（向左转）
16	`GameUp`	向上
17	`GameLeft`	向左
18	`GameDown`	向下
19	`GameRight`	向右
21	`gameAttack`	攻击面前的人
31	`gameLookAheadName`	向前看，获取其名字
32	`gameGetPosX`	给定名字获取 x 坐标
33	`gameGetPosY`	
34	`gameGetDir`	
35	`gameGetAttack`	
36	`gameGetHp`	
40	Math	Blockly Builtin
50	Logic	
60	Functions	
70	Loops	

Table 4.18: 指令编号表

编号	Blockly 中标识符名	含义
80	Lists	
90	Variables	

`/map/<map_id>/` 参数：

名称	类型	缺省值	描述
map_id	整型 [0,]	不可缺省	请求的地图 id

GET 获取地图信息。

权限要求：公开

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，不存在该地图 =2
map	地图（关卡）	空	地图信息

PUT 修改地图。

权限要求：登录用户

传入数据：

名称	类型	缺省值	描述
map	地图（关卡）	不可缺省	新建地图的信息

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，信息不合法 =2

DELETE 删除地图。(暂未实现)

/map/

GET 获取地图列表。

权限要求：公开

参数：

名称	类型	缺省值	描述
pageNo	整型 [1,]	1	分页页码
pageSize	整型 [1, 40]	20	分页每页中包含项目数量
authorId	整型 [0,]	无	地图作者用户 Id（若缺省则不对作者进行筛选）

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0
list	地图列表	空列表	请求的地图列表信息
has_prev	整型 [0, 1]	0	是否存在上一页（0= 否，1= 是）
has_next	整型 [0, 1]	0	是否存在下一页（0= 否，1= 是）

POST 创建地图。发出请求的用户即新建的地图的作者。

权限要求：登录用户

传入数据：

名称	类型	缺省值	描述
map	地图（关卡）（除了 author 字段）	不可缺省	新建地图的信息

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，信息不合法 =2
map_id	整型 [0,]	0	新建的地图的 id（如果成功）

`/stage/<stage_id>/` 参数：

名称	类型	缺省值	描述
stage_id	整型 [1,]	不可缺省	请求的关卡编号

GET 获取关卡信息。

权限要求：对于游客只开放 `stage_id<=5` 的关卡，登录用户开放 `stage_id<=` 通关的最后关卡 `id+1`，超级用户开放所有关卡。

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，不存在该关卡 =2
map	地图	空	关卡的地图信息

解法资源

- user 整型 [0,]，用户 id
- map 整型 [0,]，地图 id
- code 字符串，blockly 代码

- shared 布尔，是否共享
- stars 整型 [0,]，星级

/solution/

GET 获取解法列表。

权限要求：登录用户

传入数据：

名称	类型	缺省值	描述
pageNo	整型 [1,]	1	分页页码
pageSize	整型 [1, 40]	20	分页每页中包含项目数量
user	整型 [0,]	0	用户的 id
map	整型 [0,]	0	地图的 id

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 1]	0	请求成功 =1，失败 =0
list	地图信息列表	空列表	地图信息列表
has_prev	整型 [0, 1]	0	是否存在上一页（0= 否，1= 是）
has_next	整型 [0, 1]	0	是否存在下一页（0= 否，1= 是）

POST 创建解法。

权限要求：公开

传入数据：

名称	类型	缺省值	描述

名称	类型	缺省值	描述
code	字符串	""	Blockly 代码
shared	布尔类型	false	是否共享
stars	整型 [0,]	1	星级评价
map	整型 [0,]	空	地图的 id

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，数据不合法 =2
sol_id	整型 [0,]	0	新建解法的 id

/solution/<sol_id> 参数：

名称	类型	缺省值	描述
sol_id	整型 [0,]	（不可缺省）	请求的解法的 id

PUT 更改一指定解法的信息。

权限要求：登录用户（暂时未考虑权限问题）

传入数据：

名称	类型	缺省值	描述
code	字符串	""	Blockly 代码
shared	布尔类型	false	是否共享
stars	整型 [0,]	1	星级评价

名称	类型	缺省值	描述
map	整型 [0,]	空	地图的 id

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，不存在该解法 =2

GET 获取某一解法的信息。

权限要求：登录用户

返回：

名称	类型	缺省值	描述
res_code	整型 [-1, 2]	0	请求成功 =1，未知错误 =0，不存在该解法 =2
solution	解法完整信息	空对象	指定解法的完整信息

4.3 游戏页面——游戏逻辑

4.3.1 游戏逻辑提供的接口

方法名	参数	功能
`gameSetMap`	`map`	设置游戏地图，供游戏页面在收到游戏地图时与点击“重置”时调用
`gameInit`	无	在用户代码运行之前需要调用

4.3.2 游戏页面提供的接口

方法名	参数	功能
gameSetState	gameState: string	设置游戏状态，供逻辑调用的包括通关、不通关

4.4 游戏页面——Blockly

4.4.1 Blockly 提供的接口

方法名	参数	功能
getCode	无	获取用户编写的 Blockly 程序对应的 JavaScript 代码
getToolboxXml	map	根据地图计算允许使用的 Blockly 块对应的 XML 以显示

此外，Blockly 还会检查 window.blocklyCallback 与 window.blocklyShouldRun 的状态来改变自己的行为。游戏页面可以修改这两个变量来控制 Blockly 中的解释器是否运行。

4.5 Blockly——游戏逻辑

4.5.1 游戏逻辑提供的接口

游戏逻辑提供了如下有关游戏功能的函数（即每个 Blockly 块的功能实现），供 Blockly 解释器调用：

- gameMove
- gameAttack
- gameTurn
- gameLookAheadName

- gameGetPosX
- gameGetPosY
- gameGetDir
- gameGetAttack
- gameGetHp

4.5.2 Blockly 提供的接口

游戏逻辑可以通过修改 window.blocklyCallback 与 window.blocklyShouldRun 来控制 Blockly 中的解释器是否运行。

4.6 游戏逻辑——游戏界面

4.6.1 游戏界面提供的接口

游戏界面提供了如下有关显示的函数，供游戏逻辑调用：

方法名	参数	功能
setTargetPos	x, z	设置目标点位置
addMonster	id, x, z, maxHp	新增一个怪物
setMonsterHp	id, hp	设置怪物生命值
monsterMoveForward	id	怪物前进一步
monsterTurnCCW	id	怪物逆时针旋转
monsterTurnCW	id	怪物顺时针旋转
monsterAttack	id	怪物攻击
createMap	height, width	新增一个怪物
createPlayer	x, z, maxHp	设置玩家状态
setPlayerHp	hp	设置玩家生命值
setPlayerDirection	x, z	设置玩家朝向

方法名	参数	功能
setMonsterDirection	id, x, z	设置怪物朝向
playerTurnCW		玩家顺时针旋转
playerTurnCCW		玩家逆时针旋转
playerMoveForward		玩家前进一步
playerAttack		玩家攻击

4.7 游戏界面——Blockly

游戏界面可以通过修改 window.blocklyCallback 与 window.blocklyShouldRun 来控制 Blockly 中的解释器是否运行。

第 5 章. 各模块的主要设计

5.1 游戏总体页面

游戏总体页面类 App 是一个 React 组件，建立前端游戏页面中的所有顶层组件并将其组合起来，通过为子组件传递回调函数而响应子组件的操作，并与后端进行交互。App 组件为整个前端的最顶层框架。

5.1.1 Methods

方法名	参数	功能
`reloadUser`	无	根据 token 更新用户信息
`onLoginChange`	`user: object`	当用户登录信息被更改时调用，修改当前用户信息并针对该用户权限修改相应的权限控制

5.1.2 子组件

组件类名	功能	Props 传递
`Nav`	导航栏，显示用于登录信息	`user=this.state.user, onLoginChange=this.onLoginChange`
`Route`	地图编号的路由管理	`path=/game/:map_id/, component=this.GameDashBoard`

组件类名	功能	Props 传递
PrivateRoute	用户个人地图编辑的路由管理，带权限控制	path="/editor/:map_id/", component=MapEditor
PrivateRoute	地图编辑器的路由管理，带权限控制	path="/editor/", component=MapEditor
Route	查看已分享解法的路由控制	path="/solution/:sol_id/", component=SolutionViewer
Route	登录的路由控制	path="/login", component=PleaseLogin
Route	地图查找的路由控制	path="/searchmaps/:author_id/", component=StageGallery
Route	显示所有地图 (包括关卡与他人分享的地图) 的路由控制	path="/allmaps/", component=StageGallery
Route	显示所有关卡的路由控制	path="/allmaps/", component=StageGallery
PrivateRoute	修改用户个人信息的路由控制	path="/info/", component=InfoModify
Route	显示用户个人地图库的路由控制	path="/mymaps/", component=MapLibrary

5.2 导航栏

导航栏 Nav 是游戏的顶端导航栏，用于显示用户信息和各个操作按钮。

5.2.1 Props

属性名	功能
user	用户登录信息，当用户未登录时，无法显示游戏界面
onLoginChange	回调函数，当用户登录信息被修改时调用

属性名	功能
history	用户历史页面栈，用来在用户无权限访问某页面时回退到用户上一次访问的页面

5.2.2 Method

方法名	参数	功能
handleClick	name: string, value: boolean	响应点击事件并弹窗

5.2.3 子组件

组件类名	功能	Props 传递
AppBar	material-ui 的库组件类，显示静态导航栏	position="static"
Toolbar	material-ui 的库组件类，显示工具栏	path=/game/:map_id/, component=this.GameDashBoard
IconButton	显示菜单栏的图标	className=this.classes.menuButton, color="contrast", aria-label="Menu"
Typography	logo，显示“JiSuanKeFans”	type="title", color="inherit", className=this.classes.flex
Button	主线关卡按钮	color="contrast", onClick= 转到主线关卡页面的回调函数
Button	所有关卡按钮	color="contrast", onClick= 转到所有关卡页面的回调函数
Button	地图编辑器按钮	color="contrast", onClick= 转到地图编辑器页面的回调函数

组件类名	功能	Props 传递
Button	我的地图按钮	color="contrast", onClick= 转到我的地图页面的回调函数
Button	用户信息按钮	color="contrast", onClick= 显示用户信息的回调函数
Button	登出按钮	color="contrast", onClick= 退出登录的回调函数
Button	登录按钮	color="contrast", onClick= 显示登录界面的回调函数
Button	注册按钮	color="contrast", onClick= 显示注册界面的回调函数
LoginFormDialog	登录页面, 在 open 为 true 时显示	open=this.state.loginOpen, onRequestClose=this.handleClick('loginOpen', false), onLogin=this.props.onLoginChange
RegisterFormDialog	注册页面, 在 open 为 true 时显示	open=this.state.registerOpen, onRequestClose=this.handleClick('registerOpe false)

5.3 游戏操作界面

游戏操作界面类 DashBoard 是一个 React 组件，用于组合游戏界面与 Blockly 操作界面，实现游戏显示与游戏逻辑的交互。

5.3.1 Props

属性名	功能
user	用户登录信息。当用户未登录时，无法显示游戏界面
onUpdate	一个回调方法，当用户通关或分享解法时被调用。调用更上层的 App:reloadUser() 重新加载用户信息
match	保存用户当前地图的编号

属性名	功能
history	用户历史页面栈，用来在用户无权限访问某页面时回退到用户上一次访问的页面

5.3.2 Method

方法名	参数	功能
handleClick	name: string, value: boolean	响应点击事件并弹窗或关闭窗口
gameSetState	gameState: string	当用户通关时，计算用户的解法评级
handleClick	name: string, value: boolean	响应点击事件并弹窗或关闭窗口
initMap	无	在游戏初始化时加载地图、blockly 代码解释器、初始化 blockly 和游戏界面动画的状态
run	无	读取用户代码并调用 blockly 代码解释器运行代码

5.3.3 子组件

组件类名	功能	Props 传递
GameContainer	游戏画面显示界面	gameSetState=this.gameSetState, gameState=this.state.gameState, reportHeight= 回调函数，用于统一游戏界面和 blockly 界面的浏览器大小自适应

组件类名	功能	Props 传递
MessageDialog	进入游戏时的提示窗口	title="提示", open=this.state.welcomeOpen, closeText="关闭", onRequestClose= 关闭提示窗口后的回调函数
MessageDialog	分享解法时的提示窗口	title="提示", open=this.state.sharedOpen, closeText="好的", onRequestClose= 关闭分享窗口后的回调函数
MessageDialog	通关时的提示窗口	title="提示", open=this.state.passedOpen, closeText="关闭", onRequestClose= 关闭通关窗口后的回调函数，同时处理解法的自动保存，显示分享解法的按钮
MessageDialog	游戏失败时的提示窗口	title="提示", open=this.state.failedOpen, confirmText="重试", onRequestConfirm= 点击重试按钮后的回调函数，closeText="关闭"，onRequestClose= 关闭通关窗口后的回调函数，同时处理解法的自动保存，显示分享解法的按钮
BlocklyContainer	blockly 操作界面	onError= 失败时触发的回调函数，refCallback= 用于 DashBoard 组件更新 blocklyContainer 的回调函数，onLoaded= 加载时的回调函数，加载用户保存的历史解法，对用户权限判断

5.4 Blockly 操作界面

Blockly 操作界面类 BlocklyContainer 是一个 React 组件，用于封装原始的 Blockly 为一个 React 部件，对外提供接口，并自适应窗口大小。

5.4.1 Props

属性名	功能
`onLoaded`	一个回调方法。当 Blockly 库的 JavaScript 加载完后调用
`readOnly`	Blockly 工作区是否不允许用户修改的
`onError`	当 Blockly 库的 JavaScript 加载失败后调用
`defaultBlocks`	初始的 Blockly 工作区中的 Block
`toolboxXml`	Blockly toolbox 中可用的 Block

5.4.2 Methods

方法名	参数	功能
`clear`	无	清空工作区
`getCode`	无	获取工作区的用户代码
`getNBlocks`	无	获取工作区的用户代码的 Block 块数
`getXmlText`	无	获取工作区的用户代码的 XML 表示
`highlightBlock`	`id: integer`	高亮标号为 `id` 的 Block
`loadXmlText`	`xmlText: string`	往工作区载入 XML 表示的 Blockly 代码
`setReadOnly`	`readOnly: boolean`	设置用户是否能够编辑工作区
`resize`	`h: float`	把 blockly 缩放至高度为 `h`
`init`	无	当 Blockly JavaScript 库载入后进行的一些初始化操作。最后会回调 `props.onLoaded`

方法名	参数	功能
update	newProps: object, prevProps: object	当收到一个新的 Props 后做的更新操作

5.5 游戏画面显示界面

游戏画面显示界面类 GameContianer 是一个 React 组件，用于加载并绘制游戏的 3D 模型动画，并通过 blockly 的代码运行调用游戏逻辑进行动画的更新。

5.5.1 Props

属性名	功能
gameSetState	一个回调方法。Blockly 运行结束时调用，判断游戏是否通关。
onLoaded	一个回调方法。当游戏画面的资源全部加载成功后调用。
reportHeight	一个回调方法。向 blockly 汇报当前自适应算法算出的界面高度。

5.5.2 Method

方法名	参数	功能
setWeight	action: Object, weight: int	动画切换时的权值更新
prepareCrossFade	startAction: Object, endAction: Object, duration: float	将动画从 startAction 平滑切换到 endAction，切换所用时间为 duration
setTargetPos	x: float, z: float	将终点位置设置为 (x,0,z)
addMonster	id: int, x: float, z: float, maxHp: int	在地图上 (x,0,z) 的位置新增加一个血量为 maxHp 的怪物，其编号为 id
setMonsterHp	id: int, hp: int	设置编号为 id 的怪物的血量为 hp

方法名	参数	功能
monsterMoveForward	id: int	编号为 id 的怪物向前移动一步
monsterTurnCCW	id: int	编号为 id 的怪物逆时针旋转 90 度
monsterTurnCW	id: int	编号为 id 的怪物顺时针旋转 90 度
monsterAttack	id: int	编号为 id 的怪物攻击面前的敌人一次
createMap	height: int, width: int	创建大小为 height * width 的地图
createPlayer	x: float, z: float, maxHp: int	在 (x,0,z) 的位置放置血量为 maxHp 的玩家角色
setPlayerHp	hp: int	设置玩家角色的血量为 hp
setPlayerDirection	x: float, z: float	将玩家角色的朝向设置为 (x,y,z)，其中 y 由 x 和 z 共同决定
setMonsterDirection	id: int, x: float, z: float	将编号为 id 的怪物的朝向设置为 (x,y,z)，其中 y 由 x 和 z 共同决定
setCameraPosition	x: float, y: float, z: float	将相机的位置设置为 (x,y,z)
setLookAt	x: float, y: float, z: float	将玩家打怪时向前看的朝向设置为 (x,y,z)
playerMoveForward	id: int	玩家向前移动一步
playerTurnCCW	id: int	玩家逆时针旋转 90 度
playerTurnCW	id: int	玩家顺时针旋转 90 度
playerAttack	id: int	玩家攻击面前的敌人一次
gameLoop	time: int	游戏的执行函数，循环更新人物、怪物和 blockly 的状态
onMouseDown	event: Object	鼠标点击时调用，捕捉鼠标移动事件来进行视角移动
onMouseMove	event: Object	鼠标移动时调用，取消监听鼠标松开事件，持续进行视角移动
onMouseUp	event: Object	鼠标松开时调用，取消捕捉鼠标移动事件

5.5.3 子组件

组件类名	功能	Props 传递
Game	游戏 3D 画面显示界面	width=width, height=height, cameraPosition=cameraPosition, lookAt=lookAt, playerPosition=playerPosition, playerRotation=playerRotation, mapBlocks=mapBlocks, knightMesh=knightMesh, monsters=monsters, playerHp=playerHp, playerMaxHp=playerMaxHp, targetPosition=targetPosition, mapMesh=mapMesh, lssMesh=lssMesh

5.6 地图编辑器页面

地图编辑器 MapEditor 是一个 React 类，用来根据用户输入更新地图编辑器界面以及一些用户登录信息，并显示当前地图编辑器的地图。包含了地图编辑器显示界面和地图编辑器的操作界面。

5.6.1 Methods

方法名	参数	功能
getInitialState	无	初始化工作区
handleTextChange	无	处理文本框内容变化
updateMap	event: object	更新地图编辑器界面地图
setMap	无	设置地图编辑器界面地图
submitMap	map: map	提交当前地图编辑器界面内地图
choosePlayer	无	选择游戏内玩家起始位置

方法名	参数	功能
chooseMonster	无	选择游戏内怪兽位置
chooseTarget	无	选择游戏内目标位置

5.6.2 子组件

组件类名	功能	Props 传递
EditorGameContainer	地图编辑器画面显示界面	aiName=this.state.aiName, onLoaded= 加载已编辑地图的回调函数
TextField	地图名字的输入框	onChange= 更新地图名字的回调函数, value=this.state.mapName
Button	设置玩家位置的按钮	onClick= 回调函数调用 choosePlayer()
Button	设置目标位置的按钮	onClick= 回调函数调用 chooseTarget()
Button	设置怪物位置的按钮	onClick= 回调函数调用 chooseMonster()
FormControlLabel	分享地图的表单	无
Button	提交保存地图的按钮	onClick= 回调函数调用 submitMap()

5.7 地图编辑器显示界面

地图编辑器显示界面类 EditorGameContainer 是一个 React 组件，用于显示地图编辑器中人物、怪物和终点位置，并利用光线追踪支持鼠标点击放置目标。

5.7.1 Props

属性名	功能
onLoaded	一个回调方法。当游戏画面的资源全部加载成功后调用
aiName	怪物的名字

5.7.2 Method

方法名	参数	功能
setTargetPos	x: float, z: float	将终点位置设置为 (x,0,z)
addMonster	id: int, x: float, z: float, maxHp: int	在地图上 (x,0,z) 的位置新增加一个血量为 maxHp 的怪物，其编号为 id
createMap	height: int, width: int	创建大小为 height * width 的地图
createPlayer	x: float, z: float, maxHp: int	在 (x,0,z) 的位置放置血量为 maxHp 的玩家角色
setPlayerDirection	x: float, z: float	将玩家角色的朝向设置为 (x,y,z)，其中 y 由 x 和 z 共同决定
onTrackballChange	无	鼠标滚轮变化时调用，改变相机位置
onMouseDown	event: Object	鼠标点击时调用，捕捉鼠标移动事件来进行视角移动
onMouseMove	event: Object	鼠标移动时调用，取消监听鼠标松开事件，持续进行视角移动
onMouseUp	event: Object	鼠标松开时调用，取消捕捉鼠标移动事件，通过光线追踪算法实现对鼠标点击处位置的精确计算并放置相应的地图元素

5.7.3 子组件

组件类名	功能	Props 传递
EditorGame	游戏 3D 画面显示界面	width=width, height=height, camera=camera, playerPosition=playerPosition, playerRotation=playerRotation, mapBlocks=mapBlocks, knightMesh=knightMesh, monsters=monsters, targetPosition=targetPosition, mapMesh=mapMesh, lssMesh=lssMesh

5.8 地图元素显示组件

地图元素显示组件 Game 类是一个 React 组件，用于和 three.js 结合，显示地图中的各个 3D 元素。

5.8.1 Props

属性名	功能
width	地图宽度
height	地图高度
cameraPosition	相机位置
lookAt	相机方向
mapBlocks	地图块
playerPosition	玩家位置
playerRotation	玩家朝向
monsters	怪物列表，包含怪物网格模型和怪物的一系列信息
playerHp	玩家血量
playerMaxHp	玩家最大血量
targetPosition	终点位置

属性名	功能
knightMesh	玩家角色 3D 网格模型
mapMesh	地图 3D 网格模型

5.8.2 子组件

组件类名	功能	Props 传递
Monster	怪物类	position=monsters[i].position, rotation=monsters[i].rotation, monsterMesh=monsters[i].mesh
Player	角色类	position=playerPosition, rotation=playerRotation, playerMesh=knightMesh
Map	地图类	mapMesh=mapMesh
perspectiveCamera	相机类	position=cameraPosition, lookAt=lookAt
ambientLight	全局光照类	color=0xffffff
Bar	血条类	position=playerPosition, curValue=playerHp, maxValue=playerMaxHp

5.9 游戏逻辑

游戏逻辑的表现形式为一个 React class，但所有的函数均为 static 函数，方便游戏页面和（尤其是）Blockly 解释器调用。

该类中维护一个 static 变量 `Game.map` 作为其“状态”的保存处。

根据前述的接口定义，当游戏逻辑被游戏页面调用时，会初始化自身的相关信息。

而当游戏逻辑被 Blockly 解释器调用时，会根据游戏规则修改 `Game.map` 的状态，并通知游戏界面对画面做对应的修改（动画）。在必要（通关、失败）时，通知游戏页面进行相关的提示操作。

5.10 地图列表

地图列表包含了游戏关卡列表和用户地图列表，前者在点击后进入游戏界面，后者点击后进入地图编辑界面。

5.10.1 关卡列表页面

关卡列表类 StageGallery 是一个 React 类，显示用户权限范围内的所有关卡并响应点击，在点击后转换到相应的关卡页面。

Props

属性名	功能
`user`	用户信息
`author_id`	地图作者账号
`history`	用户历史页面栈，用来在用户无权限访问某页面时回退到用户上一次访问的页面

Method

方法名	参数	功能
`lock`	`map: Object`	判断 map 相对于用户而言是否锁定
`getInitialState`	无	获取初始的关卡列表

子组件

组件类名	功能	Props 传递
MapChooser	选择地图列表界面	mapFetcher=this.mapFetcher, lock=this.lock, onClick= 点击地图时的回调函数，进入相应的关卡

5.10.2 用户地图列表页面

用户地图列表类 MapLibrary 是一个 React 类，显示用户自己编辑并保存过的所有地图的列表。

Props

属性名	功能
author	用户信息，包括用户账号和用户名

Method

方法名	参数	功能
mapFetcher	author_id: string	从后端获取用户地图列表

子组件

组件类名	功能	Props 传递
MapChooser	选择地图列表界面	mapFetcher=this.mapFetcher, lock=false, onClick= 点击地图时的回调函数，进入相应的用户地图进行编辑

5.10.3 地图选择界面

地图选择界面类 MapChooser 是一个 React 类，用于显示当前地图列表中的所有地图的缩略图和相应的按钮。

Props

属性名	功能
`mapFetcher`	回调函数，用于从后端获取地图的所有信息
`onClick`	回调函数，用于响应用户鼠标点击，获得地图编号后进入对应的地图

Method

方法名	参数	功能
`getInitialState`	无	初始化地图列表
`loadMapList`	无	调用 mapFetcher，从后端载入地图列表
`goToPrev`	无	翻到上一页
`goToNext`	无	翻到下一页

子组件

组件类名	功能	Props 传递
`Grid`	地图缩略图的一个块	无

`Button`	地图缩略图的点击按钮	`className=this.props.classes.nav_button,`
		`onClick=this.goToPrev,`
		`disabled=!this.state.hasPrev`

5.11 登录及注册对话框

5.11.1 Props

属性名	功能
onLogin	一个回调方法，登录成功后回调
onRequestClose	试图关闭对话框时的回调方法

其他属性会被直接传递至内部的 Dialog 中，例如 open 属性等。

5.11.2 Method

方法名	参数	功能
handleInputChange	name: string	返回一个当用户输入内容时，将当前 state 中 name 键的值修改为用户输入的函数。
handleRequestForget	e: object	点击忘记密码时执行的函数，会往服务器 post 重置密码请求
removeMessage	msg	删除一条消息
addMessage	msg	增加一条消息

5.11.3 子组件

组件类名	功能	Props 传递
Dialog	所封装的对话框	除了 props.classes 都会传至 Dialog
SnackbarContent	消息提示条	key= 消息编号， className=css 样式， message= 消息内容， action= 关闭按钮

组件类名	功能	Props 传递
TextField	邮箱输入框	required={true}, margin="dense", id="email" type="email", label="邮箱", value={this.state.email}, fullWidth, onChange={this.handleInputChange('email')}
TextField	密码输入框	required={true}, margin="dense", id="password" type="password", label="密码", value={this.state.password}, fullWidth, onChange={this.handleInputChange('password')}
DialogActions	重置密码、登录、取消 3 种操作	无，但有 3 个 Button 作为 props.children

对于注册对话框,仅仅是少了 onLogin 的 Props,并稍微改动了 handleRequestSubmit 函数，修改了 post 数据和网址，以及一些 TextField 输入框，其余都无差。

5.12 后端

5.12.1 技术选型

我们的后端搭建于 Python 3 环境，并依赖于如表 5.33所示的包。

Table 5.33: 后端依赖的包

包名	版本	说明
Django	1.11.5	Web 开发框架
djangorestframework	3.7.0	用于 RESTful API 的实现
PyMySQL	0.7.11	用于访问 MySQL 数据库
pytz	2017.2	用于进行与时区相关的处理

5.12.2 结构

后端的逻辑结构如图 5.1所示。

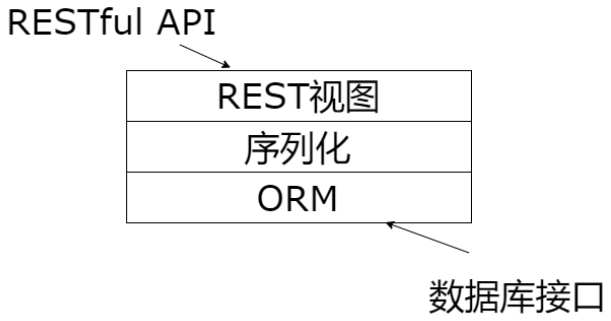

Figure 5.1: 后端逻辑结构

REST 视图负责提供 RESTful API，对 RESTful API 的调用进行响应。后端中所有的 REST 视图及对应的 RESTful API 部分如表 5.34所示。

Table 5.34: REST 视图及对应 RESTful API

REST 视图	RESTful API
`UserListView`	`/user/`
`UserView`	`/user/<user_id>/`
`MapView`	`/map/<map_id>/`
`MapListView`	`/map/`
`StageView`	`/stage/<stage_id>/`
`SolutionListView`	`/solution/`
`SolutionView`	`/solution/<solution_id>`
`ModifyView`	`/modify/`
`ForgetView`	`/forget/`
`ObtainExpiringAuthToken`	`/token/`

ORM 负责向上层提供所有数据模型的实例，并维护这些实例与数据库中数据的对应关系。序列化模块由一组序列化器组成，负责 ORM 提供的数据实例和 RESTful API 数据交换使用的 JSON 格式之间的双向转换。序列化器及 ORM 之间的对应关系如所示。

Table 5.35: 序列化器和 ORM 的对应关系

序列化器	ORM 模型类名	说明
`UserSerializer`	`User`	用户信息序列化器
`ModifySerializer`	`User`	更改用户信息请求的序列化器
`MapBriefSerializer`	`Map`	地图简略信息序列化器
`MapFullSerializer`	`Map`	地图完整信息序列化器
`StageSerializer`	`Stage`	关卡信息序列化器
`TokenPostSerializer`	`User`	获取访问令牌请求的序列化器
`SolutionSerializer`	`Solution`	解法信息序列化器

第 6 章. 数据库设计

6.1 技术选型

数据库选用的 MySQL 数据库，版本采用的是项目开始时的最新稳定版 v5.7.19.

6.2 模型描述

数据库存储的数据类型可以分为三类：

- 用户模型 (User)： 用于存储用户账户的个人信息，包括性别，密码，VIP 信息等
- 地图模型 (Map)： 用于存储游戏地图信息，包括系统内设地图和用户自定义地图
- 解法模型 (Solution)：用于存储用户解法

6.3 用户模型

域	类型	限制	用途
email	字符串	-	用户邮箱，用于用户登录，邮箱验证和密码找回
username	字符串	50	用户名，用于个人展示
password	字符串	100	用户密码，用于用户登录，存储时以加密后的形式存储，加密算法是 SHA512
gender	整数	-	性别，用于个人展示
is_admin	布尔	-	用户管理标志，用于权限判定
expiration	日期	-	用户 VIP 过期日期，用于判断是否具有 VIP 权限
join_date	日期	-	用户注册日期
latest_level	整数	-	用户最高关卡，用户进行游戏时自动加载到最新未完成关卡

6.4 地图模型

域	类型	限制	用途
title	字符串	30	地图标题，用于游戏内显示
author	用户	-	地图作者，用于标记地图的原作者，判断权限等
height	整数	-	地图高度
width	整数	-	地图宽度
shared	布尔	-	分享标志，用于判断地图是否被原作者分享公开
stage	整数	-	不明（!!!!）

n_blockly	整数	-	blockly 块数，用于记录最优解法的 blockly 块使用情况
instr_set	文本	-	指令集，用于记录地图可用的指令集合
init_AI_infos	文本	-	AI 信息，用于游戏运行时的 AI 动作模拟
init_pos_x	整数	-	起始位置 x 坐标
init_pos_y	整数	-	起始位置 y 坐标
init_dir	整数	-	起始方向
init_hp	整数	-	起始血量
init_attack	整数	-	用户起始攻击力
final_pos_x	整数	-	目标位置 x 坐标
final_pos_y	整数	-	目标位置 y 坐标
final_gold	整数	-	目标金币，到达目标点时最少的金币储备
welcome_msg	文本	-	欢迎信息，进入关卡时弹出
passed_msg	文本	-	通关成功信息，通关成功时弹出
failed_msg	文本	-	通关失败信息，通关失败时弹出
std_blockly_code	文本	-	标准解法的 blockly 形式

6.5 解法模型

域	类型	限制	用途
email	字符串	-	用户邮箱，用于用户登录，邮箱验证和密码找回
username	字符串	50	用户名，用于个人展示
password	字符串	100	用户密码，用于用户登录，存储时以加密后的形式存储，加密算法是 SHA512
gender	整数	-	性别，用于个人展示
is_admin	布尔	-	用户管理标志，用于权限判定
expiration	日期	-	用户 VIP 过期日期，用于判断是否具有 VIP 权限
join_date	日期	-	用户注册日期
latest_level	整数	-	用户最高关卡，用户进行游戏时自动加载到最新未完成关卡

6.6 地图模型

域	类型	限制	用途
user	用户	-	记录本解法的原作者
map	地图	-	记录本解法对应的地图
shared	布尔	-	记录本解法是否被分享公开，用于浏览权限判断
code	文本	-	解法的 blockly 形式，用于浏览时 blockly 显示
stars	整数	-	解法评级

第 7 章. 软件测试

7.1 概述

在开发过程中，我们对软件进行了细致的测试，以保障其具有充分的可靠性。

具体地，我们为后端实现的 RESTful API 构造了单元测试，并计算了单元测试的代码覆盖率以帮助我们尽可能地提高测试样例的强度。

7.2 测试环境

测试在项目 Dockerfile 描述的 Docker 镜像中进行。该 Docker 镜像的一些重要的配置参数如 7.1所示。

Table 7.1: 测试环境配置

参数名称	参数值
操作系统	Debian 8
MySQL 引擎版本	5.7
Python 版本	3.4.2

测试使用了 Django 框架提供的单元测试功能，代码覆盖率使用

7.3 测试样例

我们的单元测试中包含了以下测试样例：

- 请求创建合法的新用户（/user）
- 以正确的登录信息请求获取令牌（/token/）
- 在已有有效令牌的情况下再次以正确的登录信息请求获取令牌（/token/）
- 以不存在的用户名请求获取令牌（/token/）
- 以错误的密码请求获取令牌（/token/）
- 以获得的令牌请求用户信息（/token/）
- 请求创建一个和已有用户同名的新用户（/user/）
- 请求创建新用户，但不给定密码（/user/）
- 以正确的user_id 请求获取用户信息（/user/user_id/）
- 在有两个用户的情况下请求获取第 1 页用户列表（/user/）
- 在有两个用户的情况下请求获取第 0 页用户列表（/user/）
- 在有两个用户的情况下请求获取第 2 页用户列表（/user/）
- 在有 22 个用户的情况下请求获取第 1 页用户列表（/user/）
- 在有 22 个用户的情况下请求获取第 2 也用户列表（/user/）
- 在有 22 个用户的情况下请求获取第 1 页用户列表，指定每页显示 30 条（/user/）
- 以正确的令牌请求创建合法的地图（/map/）
- 以正确的map_id 请求获取已分享地图的信息（/map/map_id/）
- 在有一个已分享地图的情况下请求获取第 1 页地图列表（/map/）
- 以地图作者的令牌请求修改地图信息（/map/map_id/）
- 请求修改不存在的地图的信息（/map/map_id/）
- 以非地图作者的令牌请求修改地图信息（/map/map_id/）
- 以地图作者的令牌请求修改地图信息,但请求中包含的修改信息为空(/map/map_id)
- 在未给出令牌的情况下请求获取用户自己创建的地图列表（/map/）
- 以创建了一张地图的用户的令牌请求获取用户自己创建的地图列表（/map/）
- 以非地图作者的令牌请求删除地图（/map/map_id/）

- 以地图作者的令牌请求删除地图（/map/map_id/）
- 在没有分享过的地图的情况下请求获取第 1 页地图列表（/map/）
- 以合法的令牌请求创建合法的解法（/solution/）
- 在未提供令牌的情况下请求获取现有解法的信息（/solution/sol_id/）
- 以解法作者的令牌请求修改解法信息（/solution/sol_id/）
- 以合法的令牌请求创建解法，但请求中给出的解法信息为空（/solution/）
- 以合法的令牌请求获取不存在的解法信息（/solution/sol_id/）
- 以解法作者的令牌请求修改解法信息,但请求中给出的修改信息为空(/solution/sol_id/)
- 以有一个解法的令牌请求获取用户自己创建的解法列表的第 1 页（/solution/）
- 在用户 A 仅有一未分享解法的情况下请求获 A 的解法的列表的第 1 页（/solution/）
- 在用户 A 有一已分享解法的情况下请求获 A 的解法的列表的第 1 页(/solution/)
- 在未提供令牌的情况下请求获取当前用户信息（/modify/）
- 在提供了正确令牌的情况下请求获取当前用户信息（/modify/）
- 在未提供令牌和修改信息的情况下请求更改当前用户信息（/modify/）
- 在提供正确令牌但修改信息为空的情况下请求更改当前用户信息（/modify/）
- 在提供正确令牌和完整修改信息但所给旧密码错误的情况下请求更改当前用户信息（/modify/）
- 以正确令牌、完整修改数据和正确旧密码的情况下请求更改（/modify/）

7.4 测试结果

我们项目通过了上述所有测试样例。最终测试的代码覆盖率为 95%。