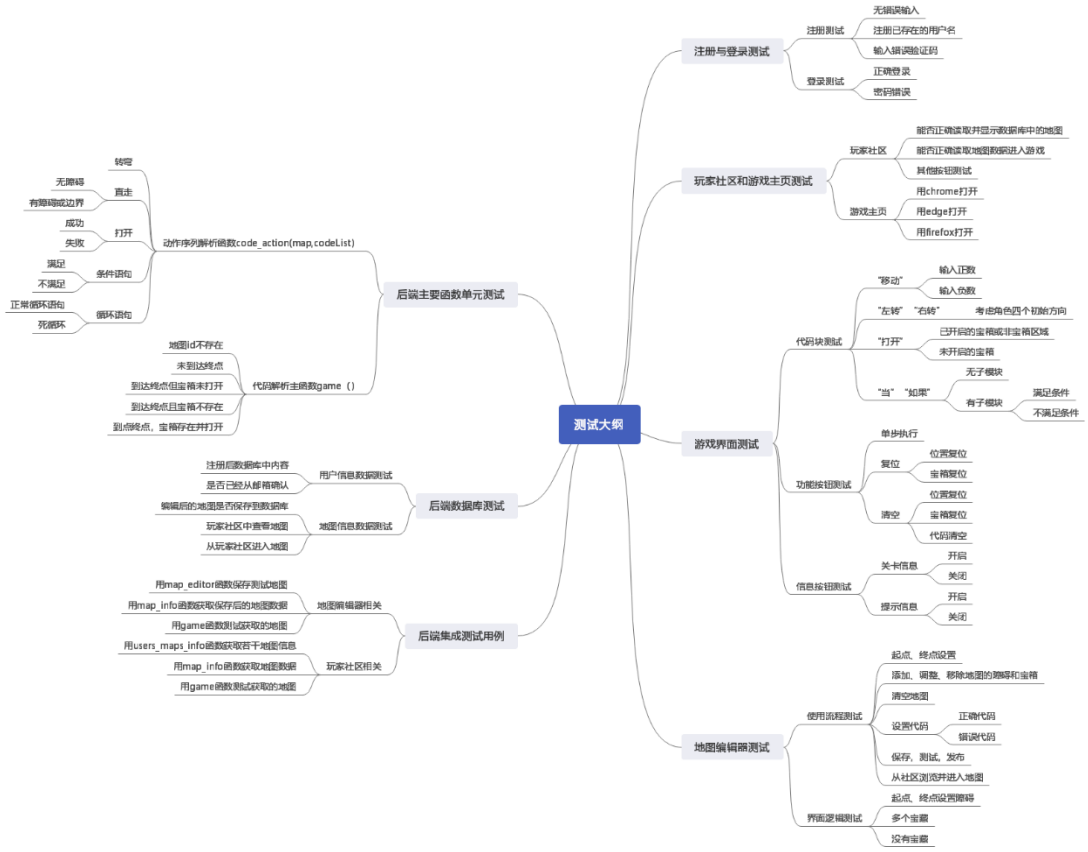
编程教育游戏

测试大纲

周颢班第 6 组

目录

思维导图:


测试大纲
注册与登录测试
注册测试
无错误输入
注册已存在的用户名
输入错误验证码
登录测试
正确登录
密码错误
玩家社区和游戏主页测试
玩家社区
能否正确读取并显示数据库中的地图
能否正确读取地图数据进入游戏
其他按钮测试
游戏主页
用chrome打开
用edge打开
用firefox打开
游戏界面测试
代码块测试
"移动"
输入正数
输入负数
"左转" "右转"
考虑角色四个初始方向
"打开"
已开启的宝箱或非宝箱区域
未开启的宝箱
"当" "如果"
无子模块
有子模块
满足条件
不满足条件
功能按钮测试
单步执行
复位
位置复位
宝箱复位
清空
位置复位
宝箱复位
代码清空
信息按钮测试
关卡信息
开启
关闭
提示信息
开启
关闭
地图编辑器测试
使用流程测试
起点、终点设置
添加、调整、移除地图的障碍和宝箱
清空地图
设置代码
正确代码
错误代码
保存、测试、发布
从社区浏览并进入地图
界面逻辑测试
起点、终点设置障碍
多个宝藏
没有宝藏
后端主要函数单元测试
动作序列解析函数code_action(map,codeList)
转弯
直走
无障碍
有障碍或边界
打开
成功
失败
条件语句
满足
不满足
循环语句
正常循环语句
死循环
代码解析主函数game（）
地图id不存在
未到达终点
到达终点但宝箱未打开
到达终点且宝箱不存在
到点终点，宝箱存在并打开
后端数据库测试
用户信息数据测试
注册后数据库中内容
是否已经从邮箱确认
地图信息数据测试
编辑后的地图是否保存到数据库
玩家社区中查看地图
从玩家社区进入地图
后端集成测试用例
地图编辑器相关
用map_editor函数保存测试地图
用map_info函数获取保存后的地图数据
用game函数测试获取的地图
玩家社区相关
用users_maps_info函数获取若干地图信息
用map_info函数获取地图数据
用game函数测试获取的地图

1. 游戏界面测试

1.1. 代码块测试

1) “走”代码块
   a) 拖动代码块到右侧代码区，应该可以正常显示
   b) 在输入框中输入正数，点击开始按钮后，小人应该行走对应的步数
   c) 在输入框中输入负数，点击开始按钮后，小人应该拒绝对非法输入做出响应
2) “向左转”代码块
   a) 拖动代码块到右侧代码区，应该可以正常显示
   b) 点击开始按钮后，根据小人的方向
      i. 初始为上，则变为左
      ii. 初始为左，则变为下
      iii. 初始为下，则变为右
      iv. 初始为右，则变为上
3) “向右转”代码块
   a) 拖动代码块到右侧代码区，应该可以正常显示
   b) 点击开始按钮后，根据小人的方向
      i. 初始为上，则变为右
      ii. 初始为右，则变为下
      iii. 初始为下，则变为左
      iv. 初始为左，则变为上
4) “打开”代码块
   a) 拖动代码块到右侧代码区，应该可以正常显示
   b) 当处于未开的宝箱上时，宝箱状态由未开启切换为开启
   c) 当处于已开启的宝箱或者非宝箱的地图块上时，无反馈
5) “当”代码块
   a) 拖动代码块到右侧代码区，应该可以正常显示
   b) 代码块没有子模块时，无法正常运行
   c) 代码块有子模块时，如果满足条件，那么子模块语句重复执行直到不再满足条

件

d) 代码块有子模块时，如果不满足条件，那么跳过子模块的运行

6) “如果”代码块

a) 拖动代码块到右侧代码区，应该可以正常显示

b) 代码块没有子模块时，无法正常运行

c) 代码块有子模块时，如果满足条件，那么执行子模块语句

d) 代码块有子模块时，如果不满足条件，那么跳过子模块的运行

1.2. 功能按钮测试

1) 单步执行

a) 点击按钮后，应该执行代码区中的一步

2) 复位

a) 点击按钮后，小人的位置回到最开始

b) 点击按钮后，如果有打开的宝箱，宝箱也需要回到最开始的状态

3) 清空

a) 点击按钮后，小人的位置回到最开始

b) 点击按钮后，如果有打开的宝箱，宝箱也需要回到最开始的状态

c) 点击按钮后，代码区所有代码清空

1.3. 信息按钮测试

1) 关卡信息

a) 点击按钮，应该弹出关卡信息

b) 弹出关卡信息后点击关闭按钮，则关闭关卡信息面板

2) 提示信息

a) 点击按钮，应该弹出提示信息

b) 弹出提示信息后点击关闭按钮，则关闭提示信息面板

2. 地图编辑器测试

2.1. 地图编辑器使用流程测试

1) 测试类型：集成测试
2) 测试环境：Chrome 78.0.3904.97
3) 测试方法：通过浏览器模拟用户使用地图编辑器设计地图流程
4) 测试目的：验证地图编辑器可以正确完成地图编辑、测试和发布需求
5) 测试用例：
    a) 设置起点 (2,4)
    b) 设置终点 (5,1)
    c) 向地图添加障碍物块 (1,3)(1,4)(1,5)(2,3)(3,4)
    d) 向地图添加宝藏 (3,3)
    e) 移除地图上的障碍物块 (1,5)
    f) 通过方向按钮调整选中的物块
    g) 清空地图
    h) 重新设置起点 (3,3)
    i) 设置终点 (5,6)
    j) 向地图添加障碍物块 (4,3)(4,4)(5,5)
    k) 向地图添加宝藏 (6,2)
    l) 控制人物转向
    m) 点击保存
    n) 点击测试
    o) 点击发布
    p) 设置错误的执行代码
    q) 设置正确的执行代码
    r) 点击发布
    s) 进入社区，打开新发布的地图

6) 测试结果：

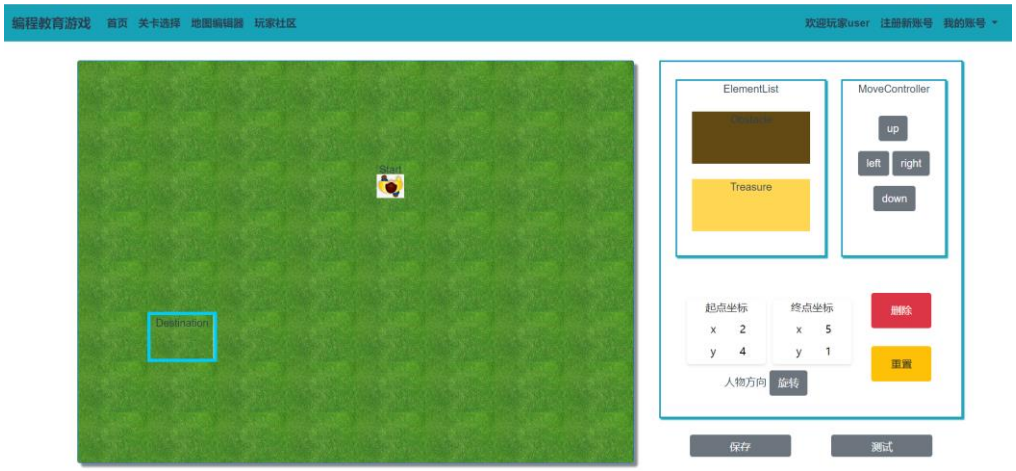
测试编号	测试内容	预期结果	实际结果
1	设置起点 (2,4)	起点变为(2,4)	成功设置起点
2	设置终点 (5,1)	终点变为(5,1)	成功设置终点
3	向地图添加障碍物块(1,3)(1,4)(1,5)(2,3)(3,4)	地图中增加障碍物块	成功设置障碍物
4	向地图添加宝藏 (3,3)	地图中增加宝藏	成功设置宝藏
5	移除地图上的障碍物块(1,5)	障碍物块被删除	成功删除障碍物
6	通过方向按钮调整选中的物块	光标根据方向按钮操控移动	成功移动光标
7	清空地图	地图中障碍物块和宝藏被清除	地图中不存在障碍物和宝藏
8	重新设置起点 (3,3)	起点变为(3,3)	
9	设置终点 (5,6)	终点变为(5,6)	
10	向地图添加障碍物块(4,3)(4,4)(5,5)	地图中增加障碍物块	成功设置障碍物
11	向地图添加宝藏 (6,2)	地图中增加宝藏	成功设置宝藏
12	控制人物转向	人物面向方向改变	人物朝向变为向左
13	点击保存	地图被上传到后端数据库	提示成功保存
14	点击测试	进入测试页面	进入测试页面
15	点击发布	报错	提示需要先通过测试

测试编号	测试内容	预期结果	实际结果
16	设置错误的执行代码	提示未能成功通关	显示通关失败信息
17	设置正确的执行代码	提示成功通关	显示通关成功信息
18	点击发布	地图被上传到后端数据库且可在社区中访问	社区中可以看到新发布的地图
19	进入社区，打开新发布的地图	能够游玩新地图	点击新地图后进入游戏界面

7） 测试截图：

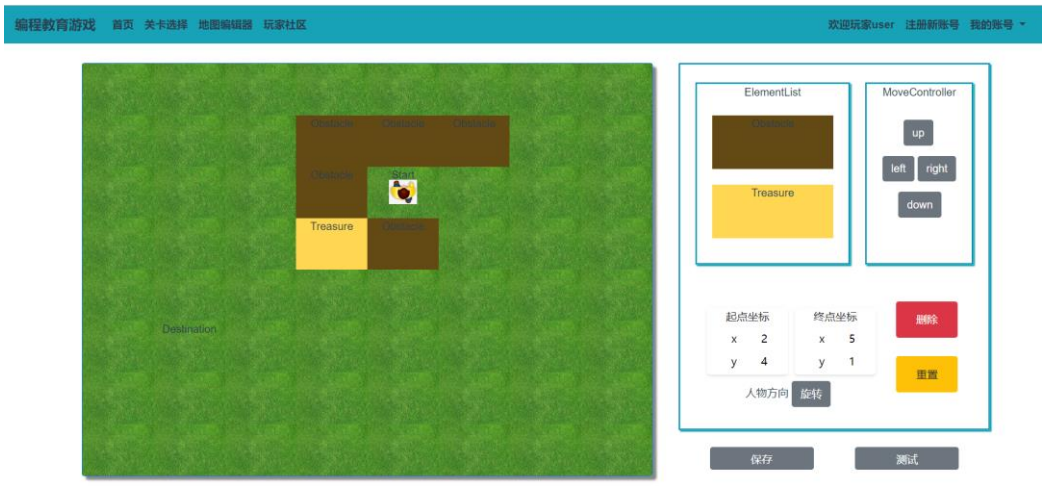
a） 设置起点 (2,4)

b） 设置终点 (5,1)

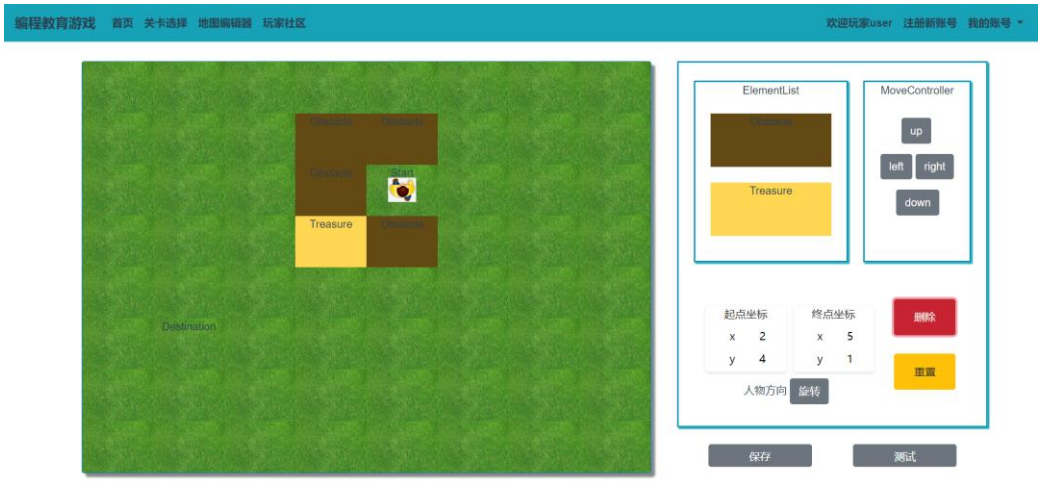


c） 向地图添加障碍物块 (1,3) (1,4) (1,5) (2,3) (3,4)

d） 向地图添加宝藏 (3,3)

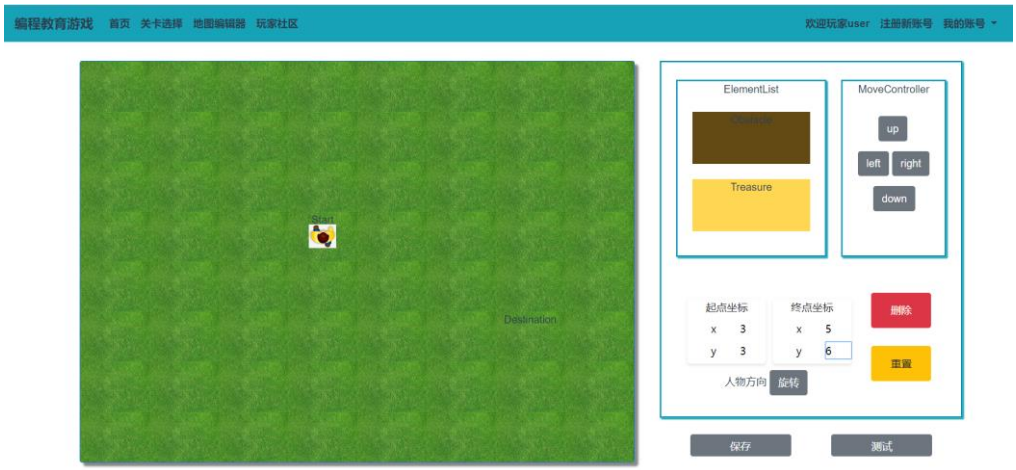


e） 移除地图上的障碍物块（1, 5）



f） 通过方向按钮调整选中的物块

g） 清空地图

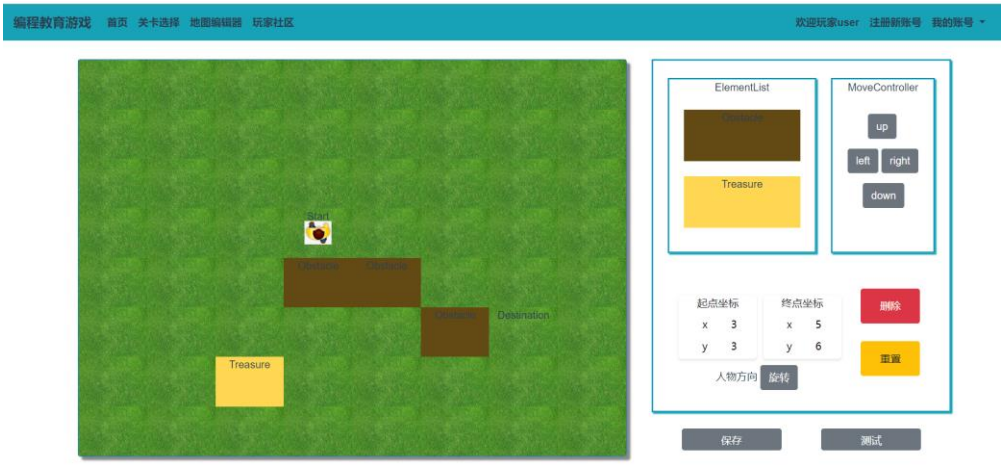


h) 重新设置起点（3,3）

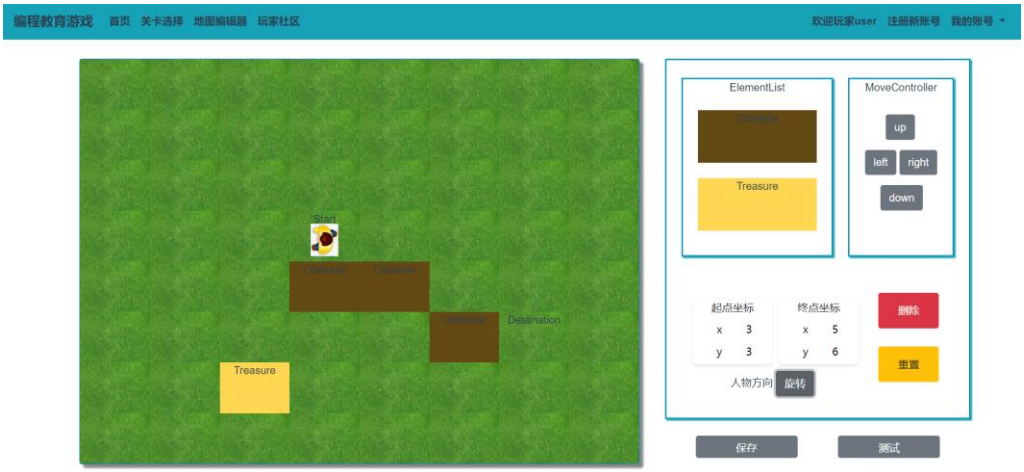
i) 设置终点（5,6）

j) 向地图添加障碍物块（4,3）(4,4)(5,5)

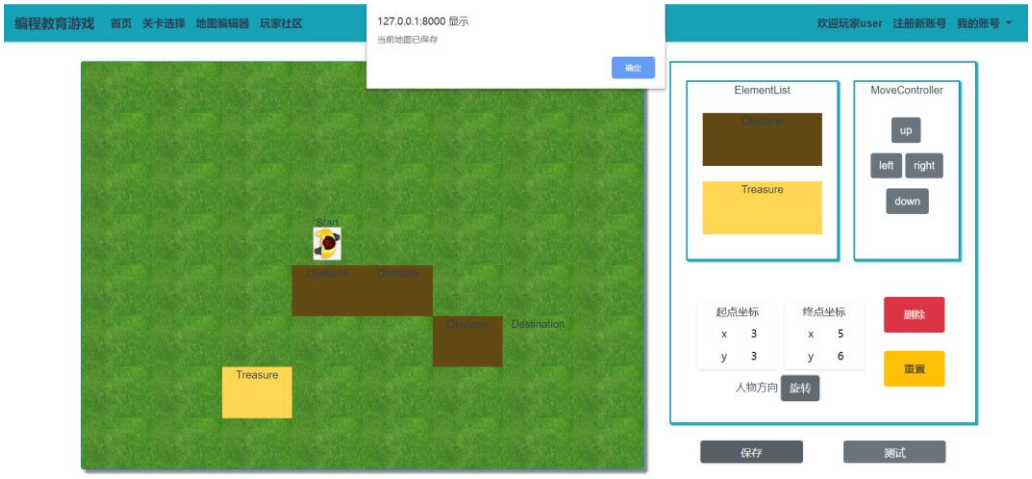
k) 向地图添加宝藏（6,2）



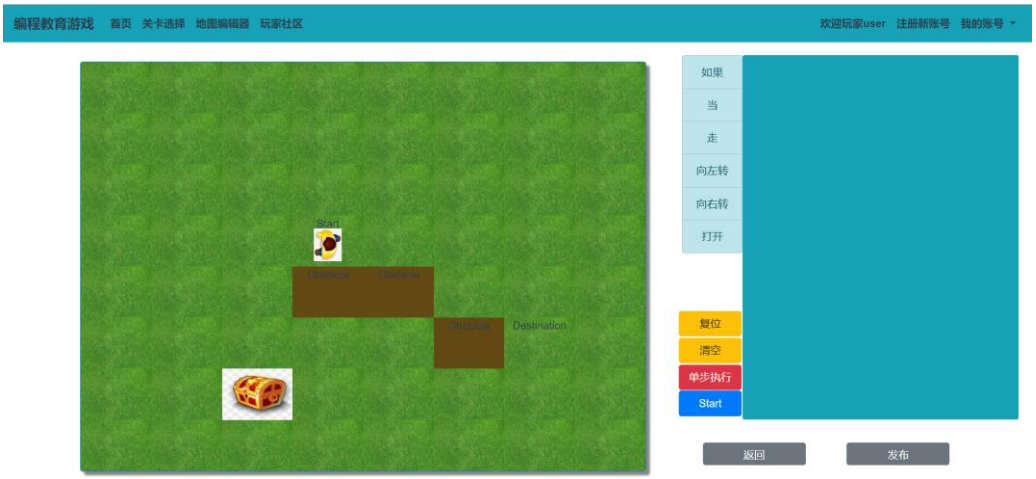
l) 控制人物转向



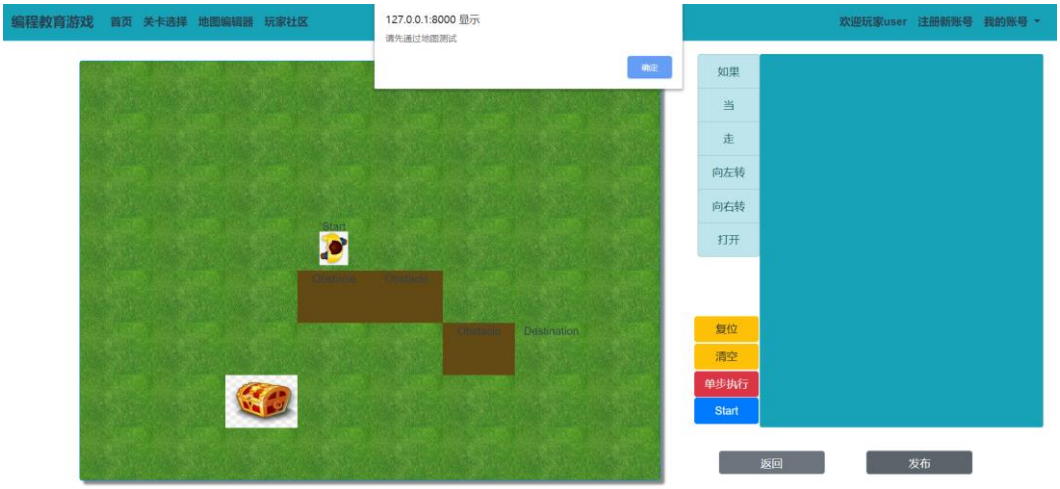
m) 点击保存



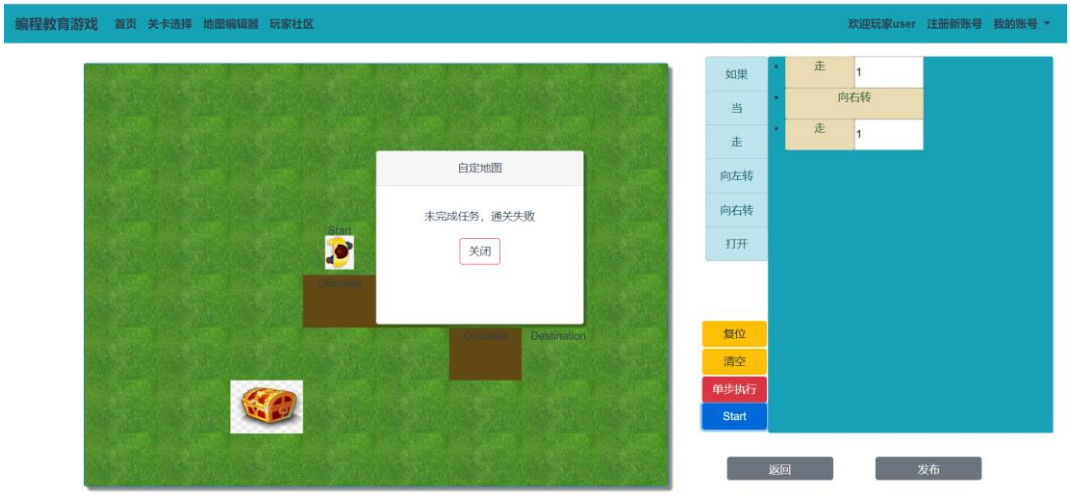
n） 点击测试



o） 点击发布



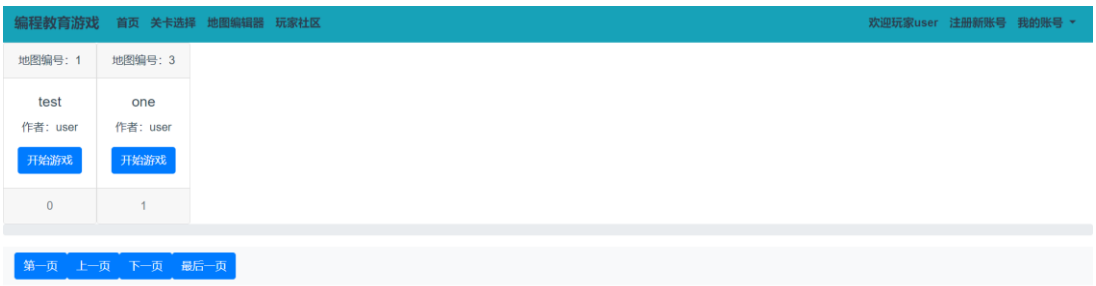
p） 设置错误的执行代码

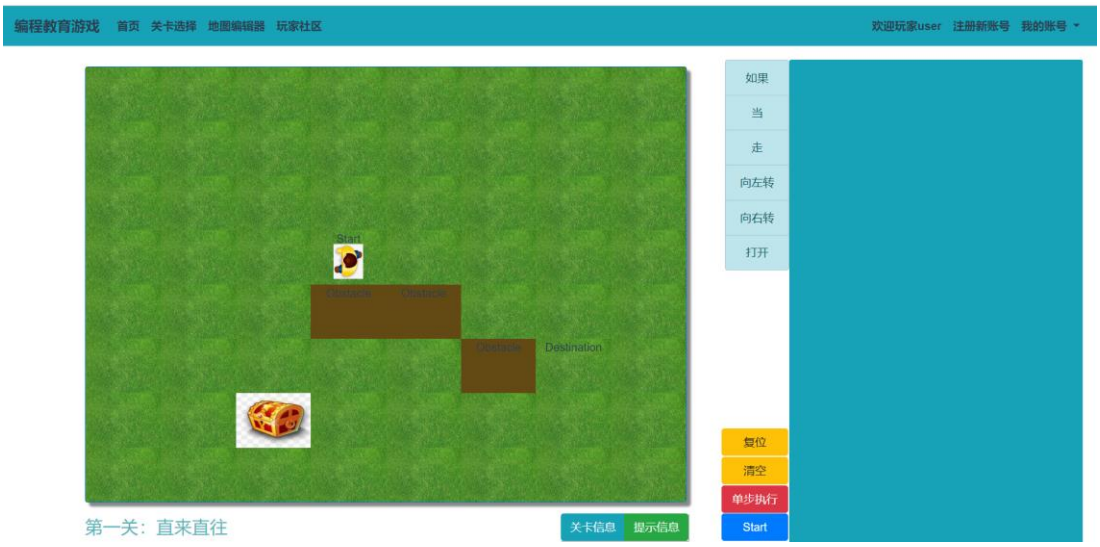


q） 设置正确的执行代码

r） 点击发布

s） 进入社区，打开新发布的地图





2.2. 地图编辑器编辑界面逻辑测试

1) 测试类型：单元测试
2) 测试环境：Chrome 78.0.3904.97
3) 测试方法：通过浏览器模拟用户通过鼠标操作对地图进行编辑
4) 测试目的：验证地图编辑器编辑逻辑的正确性
5) 测试用例：
    a) 设置起点 (2,4)
    b) 设置终点 (5,1)
    c) 向地图添加障碍物块 (1,3)(1,4)(1,5)
    d) 向地图添加宝藏 (3,3)
    e) 在地图起点设置障碍物块
    f) 在地图终点设置障碍物块
    g) 向地图增加宝藏 (4,4)
    h) 删除宝藏 (3,3)
    i) 点击测试
    j) 向地图增加宝藏 (4,4)
    k) 点击测试

6) 测试结果：

测试编号	测试内容	预期结果	实际结果
1	设置起点 (2,4)	起点变为(2,4)	成功设置起点
2	设置终点 (5,1)	终点变为(5,1)	成功设置终点
3	向地图添加障碍物块(1,3)(1,4)(1,5)	地图中增加障碍物块	成功设置障碍物
4	向地图添加宝藏(3,3)	地图中增加宝藏	成功设置宝藏
5	在地图起点设置障碍物块	报错	显示无法在起点设置障碍
6	在地图终点设置障碍物块	报错	显示无法在终点设置障碍
7	向地图增加宝藏(4,4)	报错	显示地图中最多只能有一个宝藏
8	删除宝藏 (3,3)	地图中宝藏被删除	成功删除宝藏
9	点击测试	报错	显示地图中至少要有一个宝藏
10	向地图增加宝藏(4,4)	地图中增加宝藏	成功设置宝藏
11	点击测试	进入测试状态	成功进入测试状态

3. 注册与登录测试

3.1. 注册测试

1） 测试思路

注册是玩家进行游戏的第一步，也是后续操作的起点，在整个项目中处于十分重要的位置。注册模块与其他模块也有着十分紧密的联系，所以对注册模块的测试既有单元测试也有集成测试，既需要功能性测试，也需要非功能性测试。基于以上分析，我们设计了下文中的测试用例对注册模块进行了多方面的测试，主要是测试在各种非法情况下注册系统能否正常响应并返回对应的结果。

2） 测试过程

后端数据库已有用户 user

用户名：user

密码：user

a） 注册已存在用户名

注册时用户名填入已存在的 user，注册失败，并显示提示信息“用户名已经存在”。

b） 输入错误验证码

注册时输入错误的验证码，显示提示信息“认证码错误”

c） 无错误输入

用户名：testu

密码：1234

注册成功，收到邮件确认信息

点击链接完成邮箱确认。

3.2. 登录测试

1） 测试思路

玩家在完成注册后还需要通过登录来认证身份，从而开始游戏。登录过程作为身份认证的过程，也有着十分重要的地位。由于登录模块的运行也需要前后端的结合，因此也以集成测试为主，同时进行单元测试。主要的测试目的是检测登录前端页面能否正常

显示、数据能否发送到后端、后端能否正确响应、用户登录信息错误时能否快速返回对应提示等。

2） 测试过程

a） 登录新的帐号 testu

登录成功

说明刚刚注册的 testu 已在后端保存。

b） 输入错误密码

页面没有跳转，登陆失败，显示提示信息“密码错误”

4. 玩家社区和游戏主页测试

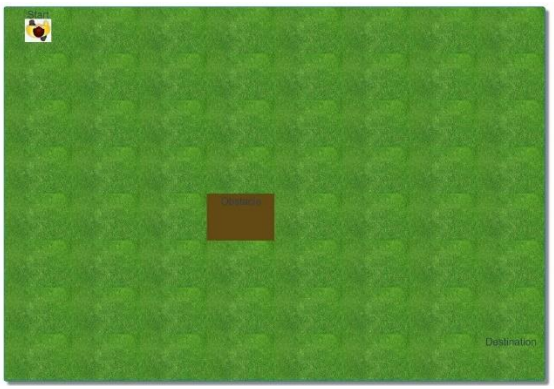
4.1. 玩家社区测试

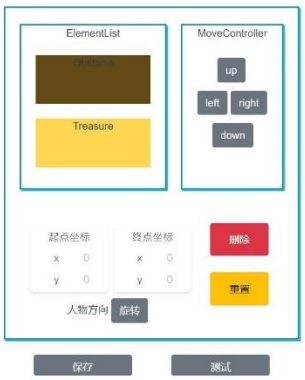
1) 测试思路

玩家社区读取后端数据库的地图，并提供选择地图的按钮，供玩家选择，因此测试主要分为两个方面，一方面测试能否正确的读取数据库，与地图编辑器保存的新地图产生互动，一方面测试选择对应的地图后，能否正确读取地图内容，跳转到游戏界面。对玩家社区的测试主要是各种按钮的测试，以及与后端数据库的互动情况，因此属于单元测试和集合测试。

2) 测试用例与过程

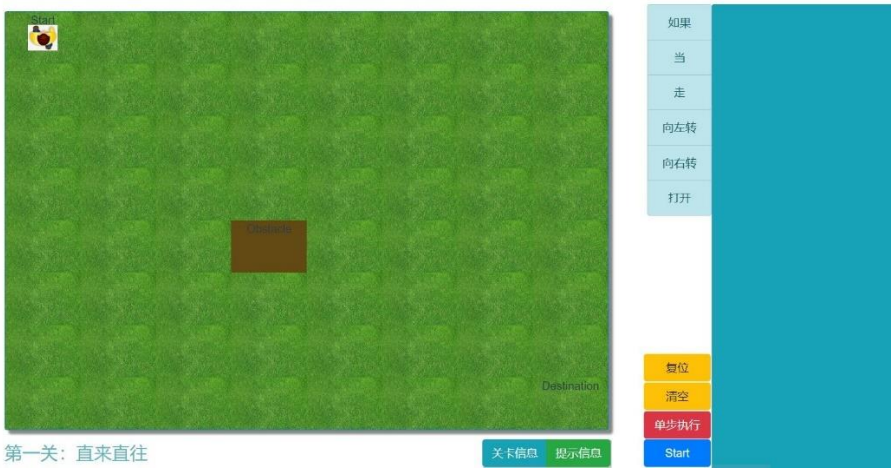
a) 在地图编辑器中发布一个新地图，并查看玩家社区是否能成功显示





进入玩家社区 ，能成功看到 testu 发布的新地图

b) 点击新地图开始游戏，查看能否正确读取后端地图数据



可以看到与之前发布的新地图一致，说明读取地图成功，各接口运行情况良好。

另外测试了一些与后端无关的按钮，都能成功的运行。

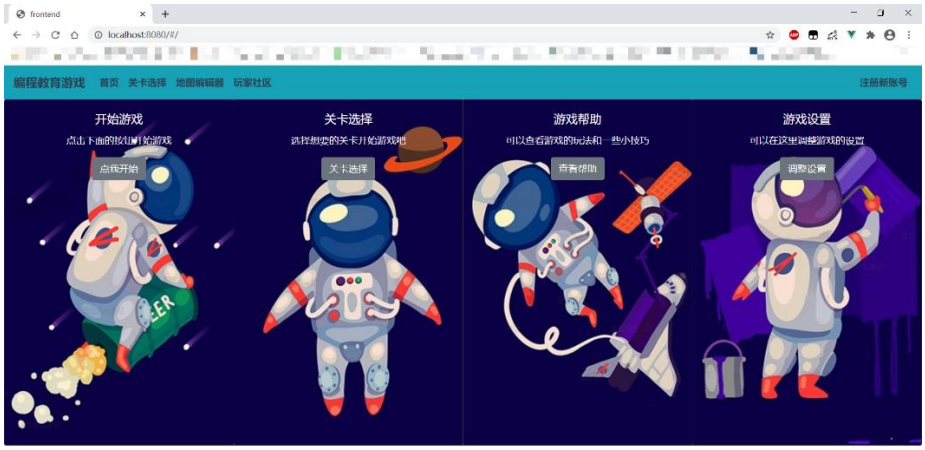
4.2. 游戏主页测试

1） 测试思路

游戏主页面的显示本身并不依赖于后端数据库的数据，主页本身也不实现游戏的功能，仅作为展示和提供游戏功能接口的用途，所以对主页的测试基本上是单元测试与非功能性测试。对主页的主要测试目的也是检测主页在各个浏览器以及系统环境下能否正常打开、正常显示。

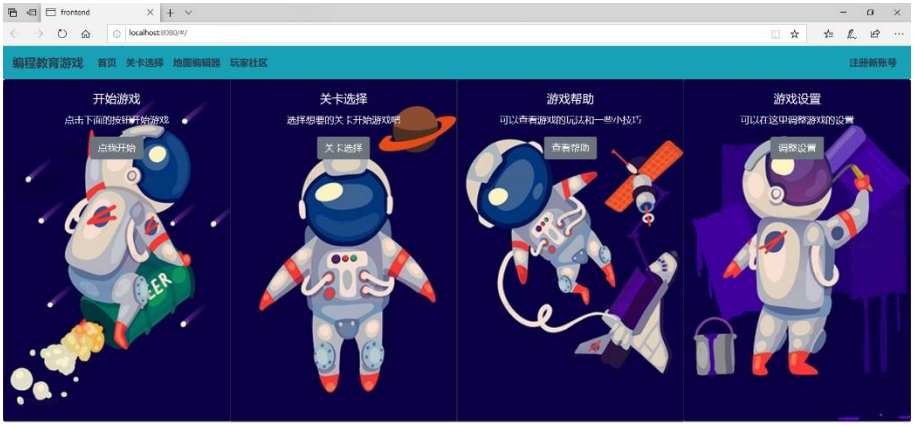
2） 测试用例与过程

a） 在 Windows 环境下 chrome 浏览器打开主页



可以看出，主页能够正常打开，主页的各个元素均能正常显示，无明显的加载时间，能较快响应。

b） 在 Windows 环境下 Edge 浏览器打开主页



可以看出，主页在 Edge 浏览器下也能正常打开，能达到预期的显示效果，无明显的加载时间。

在 Ubuntu 虚拟机环境下的 Firefox 等浏览器中也能正常打开主页，无明显变形、卡顿等异常情况。

5. 后端数据库测试

5.1. 用户信息数据测试

测试用例：在注册页面使用以下信息进行注册

点击注册按钮后，查看数据库中用户信息相关的表格：

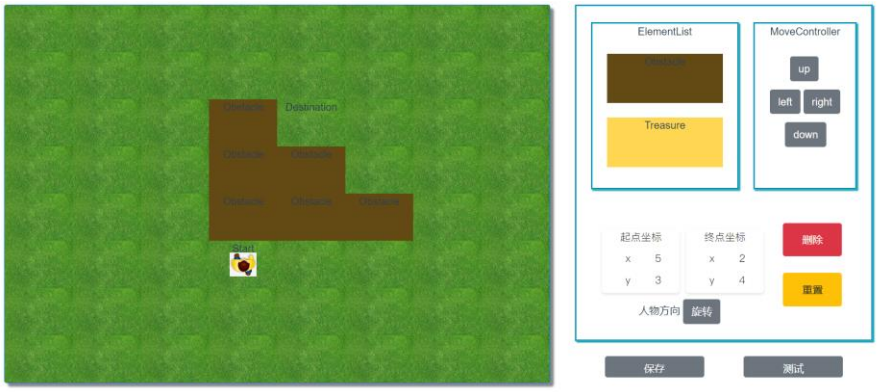
可以看到该测试用户的相关信息已经存到数据库中了，但此时 has_confirmed 一项值为 0，意味着用户尚未从邮箱中确认。

前往邮箱确认完成后，再次查看数据库中的用户信息：

此时 has_confirmed 从 0 变为 1，说明用户已经完成邮箱确认。

5.2. 地图信息数据测试

测试用例：在地图编辑器页面设计如下地图



点击保存后，在数据库中查看存放地图信息的表格：

id	name	length	width	state	startx	starty	endx	endy
1	test	10	10	1111111111111111111111111111111111111111	1	1	6	6
5	one	8	8	1111111111111111112111111122111111222111111	5	3	2	4

可以看到，由 testuser 创建的地图已经成功存入数据库中，其中包含了用户创建的地图的各项信息以及创建者的用户名。

前往玩家社区页面，查看创建的地图：

可以看到刚刚创建的地图已经显示在玩家社区里面了，点击开始游戏

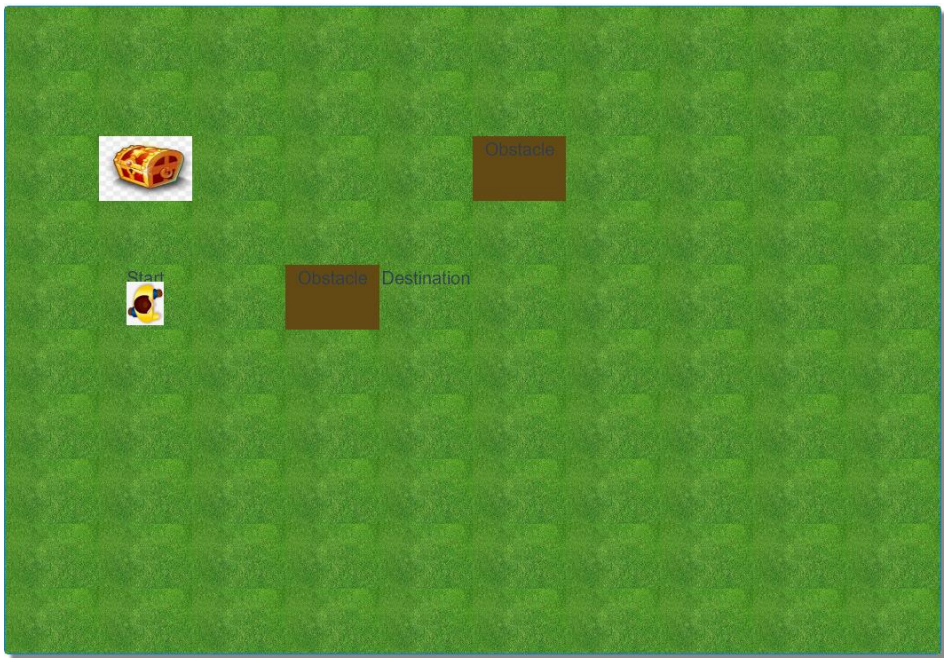
与地图编辑器页面设计的地图一致

6. 后端主要函数单元测试

6.1. 动作序列解析函数 code_acion(map, codeList)

函数	code_acion()
描述	该函数用于解析游戏中用户代码序列，得到实际游戏中人物动作序列
输入值	地图数据及代码序列
返回值	动作序列

以下测试均针对地图 id=10005，该地图如下所示



测试编号	测试内容	输入 codelist	输出 actionlist
1	转弯	{'turnRight' : '1'}, {'turnLeft' : '1'}	turnRitht, turnLeft
2	各方向无障碍物、无碰壁直走	{'goStraight' : '1'}	goRight
3	直走碰到障碍物	{'goStraight' : '5'}	goRight

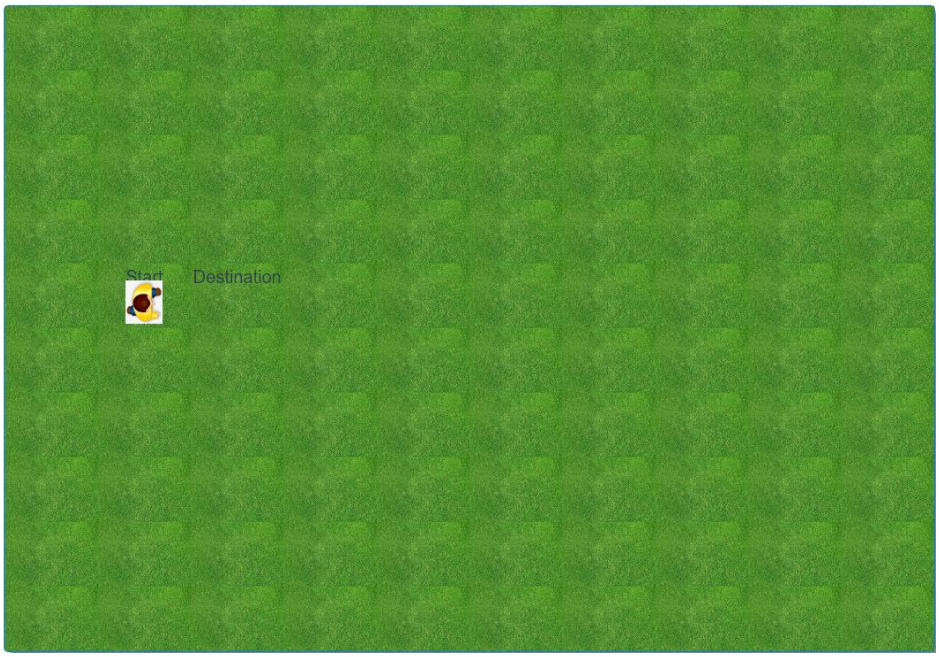
测试编号	测试内容	输入 codelist	输出 actionlist
4	直走碰到边界	{’turnRight‘ : ’1‘}, {’goStraight‘ : ’10‘}	turnRight, goDown, goDown, goDown, goDown, goDown,
5	打开宝箱失败	{’open‘ : ’1‘}	collectFail
6	打开宝箱成功	{’turnLeft‘ : ’1‘}, {’goStraight‘ : ’2‘}, {’open‘ : ‘1’}	turnLeft, goUp, goUp, collectSuccess
7	满足条件时条件语句	{'condition': {'expression': 1, 'val': 1, 'code': [{'turnRight': '1'}]}}	turnRight, turnLeft
8	不满足条件是条件语句	{'condition': {'expression': 2, 'val': 1, 'code': [{'turnRight': '1'}]}}	turnLeft
9	正常循环语句	{'circulate': {'expression': 1, 'val': 1, 'code': [{'goStraight': '1'}]}}	goRight
10	死循环	{'circulate': {'expression': 1, 'val': 1, 'code': [{'turnRight': '1'}]}}	重复 100 次 turnRight

6.2. 代码解析主函数 game()

函数	game()
描述	该函数用于解析游戏中用户代码序列，得到实际游戏中人物动作序列、游戏内各物品状态及游戏任务完成状态。
输入值	包含地图任务信息及用户代码序列的 JSON 包。
返回值	包含动作序列及任务完成状态的 JSON 包。

测试使用数据库内有 id 为 1、1000、10001、10002、10003、10004、10005、10006 共八张地图数据，考虑到前面测试过动作序列，此处不再列出测试输出内容中的动作序列。测

试需要用到的地图 10005 见 code_action()测试，地图 10001 如下所示：



编号	测试内容	输入(编号 2 以后指输入的代码序列)	输出 state	输出 message
1	地图 id 不存在	id：0	error	map not exist
2	未到达终点（针对地图 id=10005）	{'goStraight': '1'}	end	destination not arrive
3	到达终点但是宝箱存在且未被打开（针对地图 id=10005）	{'turnLeft': '1'}, {'goStraight': '2'}, {'turnRight': '1'}, {'circulate': {'expression': 1, 'val': 1, 'code': [{'goStraight': '1'}]}}, {'turnRight': '1'}, {'goStraight': '2'}	end	treasure not collected
4	到达终点且宝箱不存在（针对地图 id=10001）	{'goStraight': '1'}	end	success
5	到达终点和宝箱存在且被打开（针对地图 id=10005）	{'turnLeft': '1'}, {'goStraight': '2'}, {'turnRight': '1'}, {'circulate':	end	success

编号	测试内容	输入(编号 2 以后指输入的代码序列)	输出 state	输出 message
		{'expression': 1, 'val': 1, 'code': [{'goStraight': '1'}]}}, {'turnRight': '1'}, {'goStraight': '2'}		

7. 后端集成测试用例

7.1. 后端接口描述

函数名	接口	输入	输出
game	game/	地图 id，代码序列	解释后的代码序列，运行结果
map_info	mapInfo/	地图 id	地图数据
map_editor	mapEditor/	地图数据	保存成功的信息
users_maps_info	usersMaps/	起始地图 id，所需地图数	用于展示的满足要求的地图数据

7.2. 测试用例

7.2.1. 地图编辑器相关

用例描述：用于测试与地图编辑器有关的后端接口
构件：game，map_info，map_editor
协作关系：用 map_editor 函数保存测试地图，用 map_info 获取保存后的地图，用 game 测试获取的地图。
测试步骤： 1、用 map_editor 函数保存测试地图 2、用 map_info 函数获取保存后的地图数据 3、输入正确的代码，用 game 函数测试获取的地图

输入：
mapData = {
"user_id": 1000,
"user_name": "test",
"map": {"name": "mapForTest", "length": 10, "width": 10, "state": [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], "start": {"x": 0,"y": 1 }, "end": {"x": 1, "y": 2},"character": { "state": "r","type": 1 }
}
}
gameData = {"id": 1000,"type": "step", "codeList": [{ "goStraight": 1 },{"turnRight": "1"},{"goStraight": 1}]}

7.2.2. 玩家社区相关

用例描述：用于测试与玩家社区有关的后端接口
构件：game，map__info，users__maps__info
协作关系：用 users__maps__info 获取一些地图 id 信息，用 map__info 利用 id 获取一幅地图数据，用 game 测试地图
测试步骤： 1、创建三幅地图 2、用 users_maps_info 函数获取三幅地图信息 3、用步骤 2 获取的第三幅地图 id 以及 map_info 函数获取第三幅地图数据 4、用 game 测试获取的地图
输入： models.Map.objects.create(id=3, name=’map1’, length=10, width=10, state=map0, startx=0, starty=1, endx=1,endy=2, characterType=1, characterState='r') models.Map.objects.create(id=2, name='map2', length=10, width=10, state=map1, startx=4, starty=1, endx=4,endy=2,characterType=1, characterState='r') models.Map.objects.create(id=1, name=’map3', length=10, width=10, state=map2, startx=4, starty=1, endx=5,endy=3,characterType=1, characterState='r') gameData = {"id": 1000,"type": "step", "codeList": [{ "goStraight": 1 },{"turnRight": "1"},{"goStraight": 1}]}

7.3. 测试过程

7.3.1. 地图编辑器相关

测试步骤： 1、用 map__editor 函数保存测试地图 2、用 map__info 函数获取保存后的地图数据 3、输入正确的代码，用 game 函数测试获取的地图
预期结果： 游戏运行成功，过关
实际结果：

```
(venv) E:\Pycharm\mygame\TheGame-master\TheGame-master\django_sys>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Store map successfully!
Get map successfully!
End mission successfully!
Test1 passed.
.
Get maps' information successfully!
Get map successfully!
End mission successfully!
Test2 passed!
.
----------------------------------------------------------------------
Ran 2 tests in 0.053s

OK
Destroying test database for alias 'default'...
```

测试用例一通过

7.3.2. 玩家社区相关

测试步骤：

1、创建三幅地图

2、用 users_maps_info 函数获取三幅地图信息

3、用步骤 2 获取的第三幅地图 id 以及 map_info 函数获取第三幅地图数据

4、用 game 测试获取的地图

预期结果：

游戏运行成功，过关

实际结果：

```
(venv) E:\Pycharm\mygame\TheGame-master\TheGame-master\django_sys>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Store map successfully!
Get map successfully!
End mission successfully!
Test1 passed.
.
Get maps' information successfully!
Get map successfully!
End mission successfully!
Test2 passed!
.
----------------------------------------------------------------------
Ran 2 tests in 0.053s

OK
Destroying test database for alias 'default'...
```

测试用例二通过