编程教育游戏

概要设计说明书

周颢班第 6 组

目录

1. 文档介绍

1.1. 编写目的

本文档描述软件产品概要设计说明书的目的是：

1) 明确软件各功能的实现方式，确定游戏全部需求和组成模块；

2) 用于开发人员进行项目设计，供程序设计人员阅读；

3) 确定各模块的功能和用户接口，以此作为详细设计的依据和基础。

1.2. 文档范围

编程教育游戏概要设计说明书主要包含了该项目的详细设计，确认《需求规格说明书》后，根据其描述优化设计和分配，得出系统的体系结构和所有模块以及接口。

主要模块分为：管理员系统，用户登录系统，地图编辑器系统，社区和付费系统，游戏和评分系统。

1.3. 读者对象

需求分析人员、设计及开发者和相关的测试人员。

2. 设计选型

1) 前端框架：Vue

选择理由：Vue 采用自底向上增量开发的设计，适用于本项目类型的小型应用，简洁、轻便、快速。它的核心库只关注视图层，并且非常容易学习，易于其他库或已有项目整合。

2) Web 端样式库：AntDesign

选择理由：Ant Design 对 Vue 有专门的实现，即 Ant Design of Vue。且 Ant Design 提供完善的设计指引、最佳实践、设计资源和设计工具，来帮助设计者快速产出高质量产品。另外 Ant Design 采用 React 封装了一套 Ant Design 的组件库，即 Ant Design of React。

3) 数据库：MySQL

选择理由：MySQL 使用成本低，而同类数据库 SQL Server 和 Oracle 成本较为昂贵。另外 MySQL 跨平台支持，可运行在多个平台，对 Linux 系统友好。

4) Python 版本：Python3

选择理由：因为早期的 Python 版本在基础方面设计存在着一些不足之处。2008 年的时候 Guido van Rossum 重新开发了 Python 3.0(被称为 Python 3000,或简称 Py3k)，Python3 在设计的时候很好地解决了这些遗留问题，并且在性能上也有了一定的提升。

5) 编辑器：Visual Studio Code

选择理由：

- 轻量级：VScode 是一款轻量级的编辑器，安装包非常小，而且启动速度非常快。虽然这对实际的项目没什 么大的帮助,但是可以在无形间提高我们的开发体验。
- 丰富的插件系统：VScode 有着非常丰富的插件系统，无论你是编辑 HTML、CSS、JS、TS、Vue、React 等前端代 码，还是 JAVA、Python 等后端代码。我们都可以找到相对应的插件，让我们如虎添翼，更快速地 完成项目。
- 代码跟踪功能： 我们是一个团队，项目中往往都是协作开发，绝大多数情况下要使用 Git 来管理我们的代码，这个 时候 VScode 会跟踪我们的代码，非常明显地为我们标注：你更改了那些文件，文件中你修改可第 几行的代码，让我们对自己编写的代码一目了然。

6) 服务端框架：Django

选择理由：

- 其中 Django 的权限管理部分，与我们要实现的用户群体管理极为相似。
- Django 和 Flask，均支持较多扩展，使用简单，但 Flask 需要手动安装，较为繁琐。Django 自带一 些模块，能够避免繁琐的安装过程。
- 在之前的学习和 Django 实验中，我们成员都对 Django 有了初步的了解，可以节省学习成本。

7) 在线编程语言：Coffeescript

选择理由：

- 相对于 Javascript 硬绑了 C/Java 语法而言，Coffeescript 采用了类似 Ruby/Python 的语法，因此可以 用更少代码量去实现传统 JavaScript 的操作，并且是可使用又易懂的。通过官网(http://coffee-scri pt.org/)中给的样例代码来看，Coffeescript 的代码量显然更小；
- 选型时调研了几款编程教育游戏，有 CodeMonkey、CodeCombat 等游戏都是支持 Coffeescript 的，原因也是 Coffeescript 语法更加友好；其中 CodeCombat 还支持 Javascript 编程，对比过后感 觉 Coffeescript 代码量小，利于入门者理解编程的精髓。

- CoffeeScript 代码一一对应地编译到 Javascript，不会在编译过程中进行解释。而且已有的 JavaScript 类库可以无缝地和 CoffeeScript 搭配使用，反之亦然。编译后的代码是可读的，且经过 美化，能在所有 JavaScript 环境中运行，并且应该和对应手写的 JavaScript 一样快或者更快。

8) 编程游戏操作框架：Scratch

选择理由：此次实验我们选择最新的 Scratch3.0 版本作为基础框架。相对于 2.0 版本，3.0 版本不再使用即将停止维护的 flash，而是选择 HTML5 来编写，它支持跨平台使用，运行也不需要额外的插件。同时相比于 2.0 版 本还增加了诸如字符串包含判断、移至最下层积木等功能

3. 游戏内容设计

3.1. 主题设定

1) 编程教育游戏的主题设置为太空寻宝。贴合 7-12 岁青少年的年龄特征，也能和编程背景较好地结合起来。
2) 可控选择的角色：宇航员，遥控机器人，外星人。关卡给出目标，用户通过编程控制角色执行相关动作实现关卡目标。

3.2. 关卡设定：

3.2.1. 学习使用和运行编程语句：

1) 任务目标：沿直线通过目标区域，到达另一侧的出口。
2) 实现方式：对目标调用 goStraight()方法，点击运行实现目标向前方直线行走的任务。
3) 说明：第 1 关的主要目的在于帮助用户熟悉界面和 UI 交互机制，并能够正确添加最简单的语句块。
4) 背景故事：北极，神秘人的基地。马大叔已经全副武装，在飞来此处的路上，男孩已经把事情的经过告诉了他。男孩名为陆飞，他们家族拥有着世界上最大的飞行物设计公司。此次出发，正是为了寻找祖上流传下来的传说——名为”one piece"的宇宙大秘宝。他希望马大叔能和他一起，向着无垠的宇宙进发。“准备开始，起飞！”，随着机械女声响起，飞碟的发动机开始轰鸣。让我们一鼓作气，冲破大气层吧

5) 背景图：

3.2.2. 了解顺序结构：

1) 任务目标：通过曲折路径，到达另一侧的出口。在这一关中不能仅使用单条语句就完成任务，需要依次排布相应的语句才能完成整个移动过程。

2) 实现方式：对目标调用 goStraight()+turnLeft()+turnRight()方法，实现目标通过曲折路径的任务。

3) 说明：顺序结构是程序的最基本结构，在这一结构下指令逐条执行。第 2 关通过简单的语句排列帮助用户学习编程中基本的顺序结构，便于程序运行模式的理解与后续部分的学习。

4) 背景故事：在马大叔和陆飞的操作下，飞船终于冲出了大气层，可问题还没有结束。一块卫星残骸正位于他们面前的轨道上，请你操纵人物，避开障碍，继续探险吧！

5) 背景图：

3.2.3. 设置参数

1) 任务目标：在通过区域的过程中，需要在指定位置停下，并采集地块信息。这时系统提示发现一个宝藏。采集完成后，继续移动通过路径。

2) 实现方式：在调用 goStraight()方法的过程中为 goStraight()添加参数，如

```
Astronaut.goStraight(5)  # 人物前进5格
Astronaut.inspect()      # 对当前地块进行检查，获取相关信息
Astronaut.goStraight(2)  # 人物前进2格，通过目标区域
```

3) 说明：参数是程序执行的重要部分。通过参数可以对函数实现重用，或使函数的执行得到精确控制。通过学习为函数设置参数，让用户可以更加精确操控人物，实现后续任务。

4) 背景故事：马大叔和陆飞来到了美丽的太阳系，远处是广阔而深邃的宇宙，身后是蔚蓝的地球。曾经的梦想如今化为现实，年少的轻狂最后成为了眼前实景，请你操纵人物，在这美丽的星海中漫游吧

5) 背景图：

3.2.4. 选择条件语句

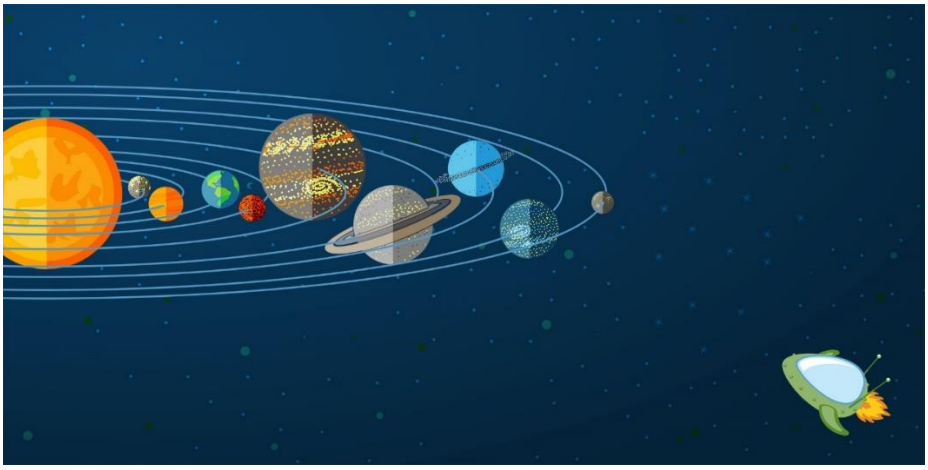
1) 任务目标：解锁宝藏的安全问题 v 任务描述：在上一关中我们获得了一个宝藏，这时我们发现需要回答对宝藏上的三个问题才能解锁宝藏的安全限制。已知当问到“A”时我们要回答“a”；当问到“B”时我们要回答“b”；当问到“C”时我们要回答“c”。设计程序使得我们的机器人可以智能应答。

2) 实现方式：通过 if-else 语句判断给出的问题并进行回答。

3) 说明：选择条件语句可以帮助我们对程序执行的状况进行判断，并指导我们选择正确的分支。通过学习条件语句为构建一个更加“聪明”的程序，可以智能的根据不同情况做出不同应对，而无需我们手动控制。

4) 背景故事：背景故事：根据宇宙地图的指示，他们要寻找的秘宝，正位于一个名叫拉夫德鲁的星系中。为了完成友人陆飞的梦想，让我们继续前进吧，笔直地冲出太阳系。不要停下来啊！

5) 背景图:



3.2.5. 循环语句

1) 任务目标: 打开宝藏

2) 任务描述：在上一关中我们解开了宝藏的安全问题，现在我们需要尝试打开这个宝藏。我们发现宝藏被装在一个 E 合金制作成的盒子里。根据地球上科学家的研究，已知任何一种其他物体都无法切开 E 合金，但其内部结构与液体类似，不断旋转会导致其中不同成分分层，导致其强度下降。下面我们要让其不断旋转，直到其强度低于临界强度后进行切割。

3) 实现方式：使用循环语句不断调用 turnAround()方法，循环终止条件为 Box.hardness()<CriticalPoint，终止后使用 cut()方法打开宝藏。

4) 背景故事：经历了漫长的航行，如今大秘宝已近在眼前。克制住自己的冲动，谨慎地避开每一个障碍，到达那路途的尽头，梦想之地吧！

5) 背景图：

4. 前端设计

4.1. 主页

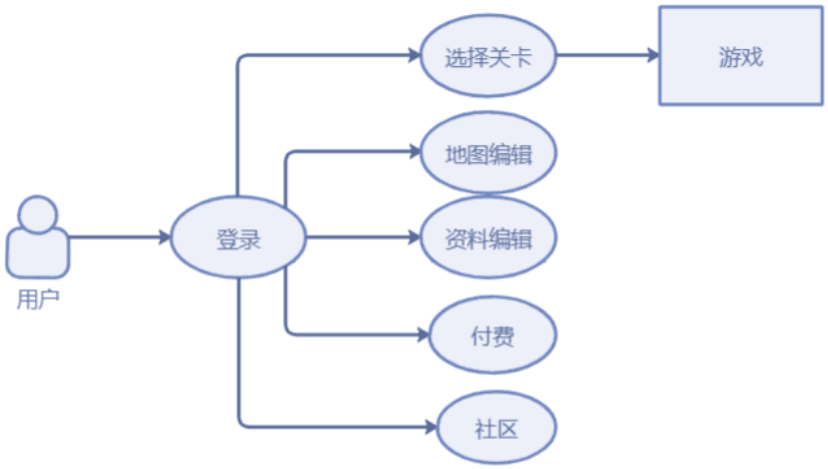
1) 美工设计思想

一个网站的主页是整个网站的门户，所需要担负的职责是给出各个功能的入口以及大致的介绍。因此主页由四个部分组成：开始游戏，关卡选择，游戏帮助和游戏设置，美化主要围绕这四个部分来进行

2) 前端界面设计思路

a) 实际效果图展示：

b) 用户用例图：



c) 布局

根据用户用例图，其中因为只有四个需要展示的部分，因此将整个页面纵向地等分成四列。每列的结构相同，由说明文字，跳转按钮，背景图组成

d) 背景图选取

因为我们的编程教育游戏项目的主题是宇航员的探险，因此背景图选用四张宇航员的卡通图片，旨在体现游戏主题

3) 前端逻辑实现

因为主页的功能是作为整个网站的门户，因此只提供了简单的跳转，无复杂逻辑。

4) 使用的组件

主页使用了 Bootstrap 中的 card 组件，四个主要组成部分都是通过 card 组件搭建的。

4.2. 游戏界面

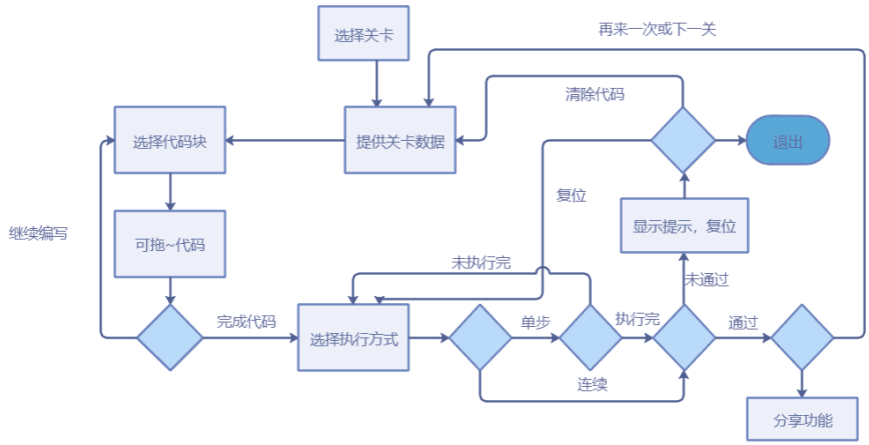
1) 美工设计思想

游戏界面是这次的编程教育游戏项目的核心部分，而美工设计主要围绕着如何使得游戏界面更加的通俗易懂，直观而进行的。为了实现这一点，需要使用尽可能简单以及直观的说明，各类按钮的位置也以满足人眼舒适为目的

2) 前端界面设计思路

a) 实际效果图展示：

b) UML 图：



c） 布局

游戏界面由三个部分组成，如上所示：最右边是核心的拖拽代码块编程区域，左边的大部分是游戏地图区域，左边区域的底部则是地图名字，关卡信息和提示信息等说明性文档所在

3） 前端逻辑实现

a） 拖拽代码块部分

因为代码块的数量是不受限制的，因此为了美观，引入了 vue-custom-scrollbar 组件，即一个内嵌的滚动区域。拖拽黏合功能的实现是通过引入 vuedraggable 组件实现的，通过该组件，玩家可以自由地组合各个代码块，来通过各种关卡。而代码的分析则是通过一个遍历函数实现的，该函数通过遍历代码块区域的每一个拖拽块，来生成一个约定好的 json 文件，并发送给后端，然后再接收相应的动作序列。

b） 游戏地图区域

该区域的核心功能是根据接收的地图数据以块为单位，显示出正常的地图，具体的实现是通过 v-for 完成的，接收了后端数据需要逐项分析每一个地图块的属性并加以显示。另一个核心功能是根据后端发送的动作序列表现出正确的行为，同样是通过遍历后端发送的动作序列完成的，动作序列中的每一个动作都会改变地图区域的数据，然后通过 props 管道将数据变化体现在地图上。

c） 文档区域

文档区域的逻辑则是点击相应按钮后显示对应的说明文字，这些说明文字来自从后端数据库中获取到的数据

4） 使用的组件

游戏界面使用了 Bootstrap 中的 btn-group 组件和 card 组件、vue-custom-scrollbar 以及 vuedraggable 组件

4.3. 关卡选择、社区和地图编辑器

1) 美工设计思想

对于一个网站而言，网页美工的设计将直接影响到用户的体验。所以网站的美工设计必须按照正确的设计原则来进行，本项目主要遵循的原则有：

a) 整体风格要一致：各个页面的风格要保持一致，各个页面的素材、布局与整体结构也保持一致，网页元素大小排版保持工整有序、不突兀。

b) 主题元素切合网站内容：由于故事主题是太空与宇航员，网页素材也基本是太空与宇航员相关，整体颜色基调也偏向蓝色以更好表现出太空给人带来的辽阔感。

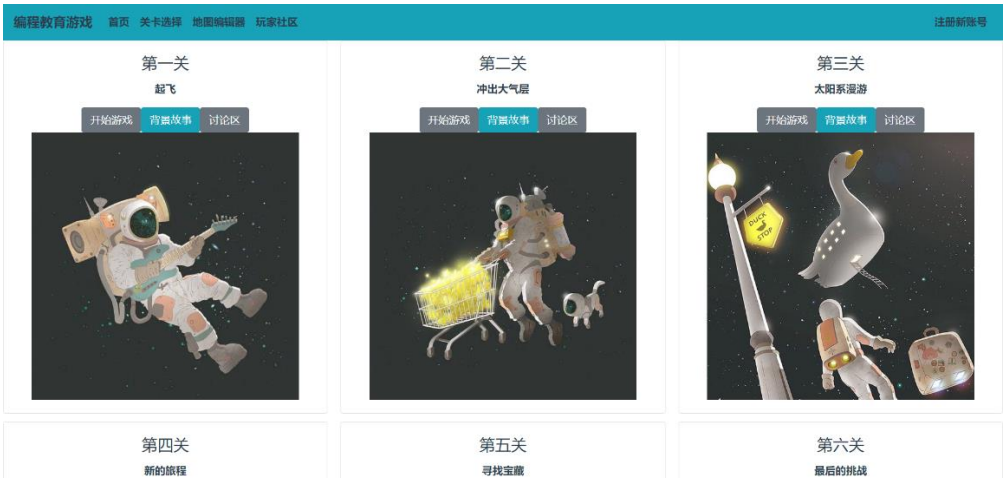
c) 设计简洁不杂乱：一个界面不能有太多的内容和功能，只显示最突出的元素，避免分散用户注意，方便用户找到想要的内容。

d) 内容安排有逻辑：同一个页面的元素要相关，不能胡乱堆砌，不同的功能分类安排在不同的页面中，以让用户更好的使用。

2) 前端界面设计思路

a) 关卡选择界面

关卡选择界面的目的是为了使玩家能方便地选择关卡，同时向玩家展示关卡的相关信息。为了实现这一功能，前端部分采用的是 bootstrap 框架的卡片（card）组件。每一关都将被展示在一个卡片之中，每个卡片显示对应关卡的标题并提供开始对应关卡游戏的按钮，并以相关的图片作为背景。



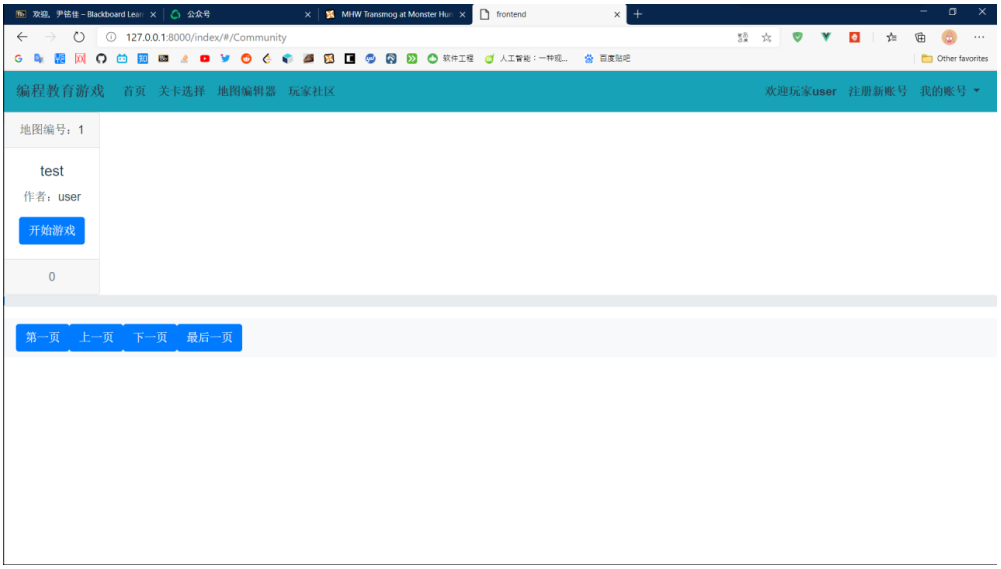
关卡的卡片还要展示这一关的背景故事。由于背景故事有一定长度，直接显示在卡片中则不够简洁所以在每个关卡的卡片中设置了背景故事按钮，点击后再显示出对应的背景故事。

b)　玩家社区界面

玩家社区的主要作用是给玩家提供一个交流的平台，玩家在地图编辑器中自行设计的地图可以在这里展示给其他的玩家，玩家也可以通过玩家社区游玩玩家自制的地图，能充分发掘游戏的潜力。为了实现这一点，需要前后端进行结合，后端部分负责存储玩家自制的地图数据，前端部分则负责以合适的方式将地图的基本信息展示在用户面前。

为了实现展示地图的目的，基本框架仍然使用 bootstrap 的 card，每一个 card 显示一个自制关卡，每个卡片展示对应关卡的基本信息，并提供开始游戏的按钮。用户选择关卡后点击即可跳转到游戏界面开始游戏。

由于关卡数目可能较多，前端一次性无法全部显示，后端也无法承受每次查询所有地图的查询代价，所以需要采用分页的方式来进行展示。每次前端都向后端提供一个地图编号区间，后端根据区间向前端展示这一页地图的基本信息以供玩家进行选择。与之配套的还有翻页按钮和进度条等组件。

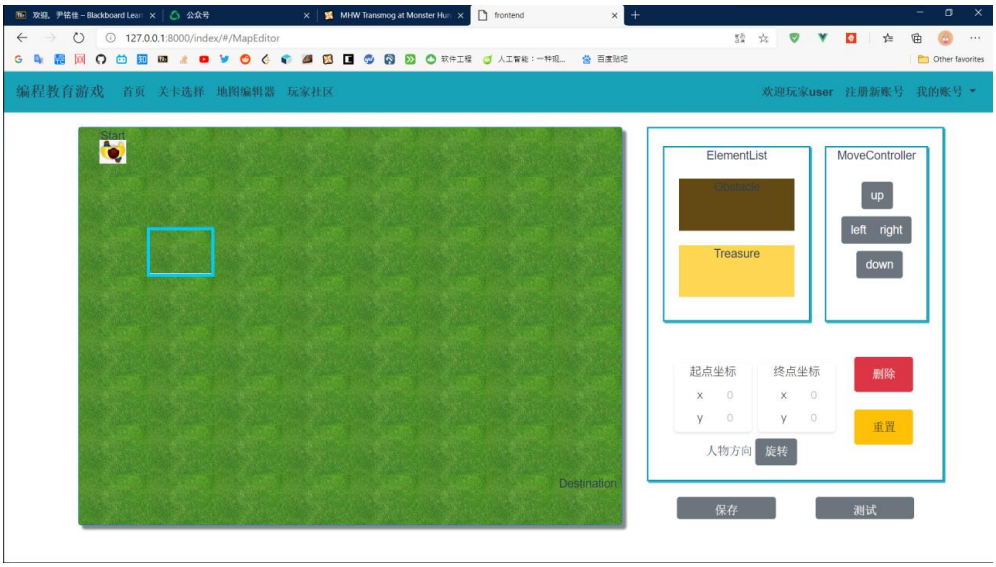


c)　地图编辑器

地图编辑器是提供给玩家的自制地图的基本工具，目的是为了方便玩家进行自

制地图。为了使大部分玩家都能简单容易地实现地图制作，避免直接使用代码，地图编辑器需要实现全部可视化操作。

最终实现的地图编辑器界面如下所示，左侧是可编辑的地图区域，右侧则是工具和测试面板。编辑地图时只要先点击右侧需要放置的区块，然后点击左侧地图区域要放置的格子即可。在实现之后还可以自己拖动测试代码块到测试区对地图进行测试，小人将按照给定的代码运动。在最终测试无误后玩家可以提交代码到后端，由后端进行存储并展示在玩家社区中。



5. 数据库模式

5.1. 地图数据

```
1.  class Map(models.Model):
2.
3.      id = models.IntegerField(primary_key=True)
4.      name = models.CharField(max_length=128)
5.      length = models.IntegerField()
6.      width = models.IntegerField()
7.      state = models.CharField(max_length=200)
8.
9.      #关卡起点、终点和宝藏坐标
10.     startx = models.IntegerField()
```

```
    starty = models.IntegerField()
    endx = models.IntegerField()
    endy = models.IntegerField()
    treasurex = models.IntegerField()
    treasurey = models.IntegerField()

    #人物形象和初始时朝向信息
    characterType = models.IntegerField()
    characterState = models.CharField(max_length=2)

    class Meta:
        ordering = ['id']
```

用于保存地图信息的数据库模型：

- id为主键，代表地图编号
- name为地图名字，最大长度128
- length为地图的纵向格数
- width为地图的横向格数
- state为地图的状态信息，使用一个最大长度为200的字符串，字符串的每一位代表一个格子，不同的数字代表不同的地形（空地、障碍物、宝箱等）。
- startx起始位置的第一个坐标
- starty起始位置的第二个坐标
- endx终点位置的第一个坐标
- endy终点位置的第二个坐标
- treasurex宝藏位置的第一个坐标
- treasurey宝藏位置的第二个坐标
- characterType人物形象
- characterState人物初始的朝向

5.2. 玩家个人信息

```
class User(models.Model):

    gender = (
        ('male', "男"),
        ('female', "女"),
    )

    name = models.CharField(max_length=128, unique=True)
    password = models.CharField(max_length=256)
    email = models.EmailField(unique=True)
    sex = models.CharField(max_length=32, choices=gender, default="男")
    mobile = models.CharField(max_length=20)
```

```
    c_time = models.DateTimeField(auto_now_add=True)
    has_confirmed = models.BooleanField(default=False)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ["-c_time"]
        verbose_name = "用户"
        verbose_name_plural = "用户"




{
    "user_id": "用户 id",
    "user_name": "用户名",
    "user_level": "用户级别",   //1：游客，2：登录用户
}
```

用于保存玩家个人信息的数据库模型：

- name 为用户名，最大长度为 128，不可与已有名字重复
- password 为登录密码，最大长度为 256
- email 为玩家邮箱，不可重复
- sex 为玩家性别，可选 male（男）或者 female（女）
- mobile 为玩家手机号
- c_time 为玩家注册时间，注册时自动登记为当前时间
- has_confirmed 代表玩家是否经过邮箱认证，默认为否

5.3. 玩家已过关卡数的信息

```
class PlayerInfo(models.Model):

    name = models.CharField(max_length=128, unique=True) #用户名
    passMapNumber = models.IntegerField() #已过关卡数
```

用于保存玩家已过关卡数的数据库模型：

- name为用户名，不可重复
- passMapNumber为已经通过的关卡数

5.4. 玩家已过关卡历史信息

```
class HistoryInfo(models.Model):

    name = models.CharField(max_length=128, unique=True) #用户名
    mapNum = models.IntegerField() #第几关
    score = models.IntegerField() #得分
    code = models.TextField() #过关代码
```

用于保存玩家已过关卡的历史信息的数据库模型：

- name为用户名，不可重复
- mapNum为关卡数
- score为得分信息
- code保存用户过关时使用的代码

6. 数据交换格式

6.1. 登录界面的数据交换格式

用户登录时，由后端传给前端

```
{
    "user_id": "用户 id",
    "user_name": "用户名",
    "user_level": "用户级别",   //1：游客，2：登录用户
}
```

在登录界面用户正常登录或使用“游客登录”后，后端验证用户名以及密码，验证通过后，向前端发送用户信息，包含用户 id、用户名以及用户级别（游客或者登录用户）。

6.2. 地图信息的数据交换格式

游戏界面初始化时，由后端传给前端。

```
{
    "id": "关卡编号",
```

```
    "name": "关卡名字", // 勇闯 xxxx 之类的
    "length": "地图长度",
    "width": "地图长度",
    "start": {
        "x": "x 坐标",
        "y": "y 坐标"
    },
    "end": {
        "x": "x 坐标",
        "y": "y 坐标"
    },
    "treasure": { // 由于先只实现前三关，因此只需要两个参数，后续可以继续添加
        "x": "x 坐标",
        "y": "y 坐标"
    },
    "character": {
        "type": "人物形象", //人物形象，1 为宇航员，2 为遥控机器人，3 为外星人
        "position": {
            "x": "x 坐标",
            "y": "y 坐标"
        },
        "state": "人物朝向" // u,d,l,r
    }
}
```

在进入游戏界面的时候，前端向后端请求该关卡的地图信息，后端从数据库中取出对应关卡的地图信息，发送给前端。其内容包括：关卡 id、关卡名字、地图规模（长宽）、起始位置、终点位置、宝藏位置、人物形象、人物位置以及人物朝向。

6.3. 代码运行部分的数据交换格式

6.3.1. 代码解析

在用户运行代码时，由前端发给后端。

```
{
    "id": "关卡编号", //只有该项为必须项，其他项都为可选可重复的
    "type": "程序执行方式", // 可选值为 continue 和 onestep，onestep 的话只返回一条动作
```

```
    "code": {
        "goStraight": "具体步数",
        "turnLeft": "NULL",
        "turnRight": "NULL",
        "inspect": "NULL", // 探索面前一格的信息，空格返回 1，障碍 2，宝藏 3，边沿 4
        "if": {
            "condition": { // 必须包含一个 inspect 函数
                "expression": "具体的关系符号", // 支持==,!=
                "val": "条件对应的限制值" // 取值范围与 inspect 函数返回值一致
            },
            "code": { // 可以嵌套其他代码
                ...
            },
            "else": { //可以为空，嵌套其他代码

            }
        },
        "while": {
            "condition": { // 必须嵌套一个 inspect 函数
                "expression": "具体的关系符号", // 支持==, !=
                "val": "条件对应的限制值" // 取值范围与 inspect 函数返回值一致
            },
            "code": { // 嵌套其他代码

            }
        },
        "open": "NULL" //打开面前一格的宝箱,成功返回 1,失败返回 0
    }
}
```

前端需要对用户写好的代码进行一些处理，将解析后需要执行的代码发送给后端，其内容包括：关卡id、程序执行方式、代码列表。代码列表中每一项对应一个或者多个操作，单个操作包括直行、转向、侦查、打开宝箱。另外还有条件语句和循环语句的解析。if条件语句包含三部分:判断条件、判断条件为真时执行的代码、判断条件为假时执行的代码。while循环语句包含两个部分：判断条件以及循环执行的代码。其中嵌套的代码块可以包含所有类型的代码，即可以实现条件判断以及循环的嵌套。

6.3.2. 动作序列

后端收到前端发来的代码后，运行代码，将小人需要执行的动作序列和结果发给前端。

```
{
```

```
    "turnLeft": "NULL",
    "turnRight": "NULL",
    "goUp": "走的步数",
    "goDown": "走的步数",
    "goLeft": "走的步数",
    "goRight": "走的步数",
    "collectSuccess": "NULL", // 前端根据宝箱开启的成功与否展示不同的特效
    "collectFail": "失败的原因",
    "endMissionSuccess": "得分", //当前端读取到这个，意味着指令序列结束
    "endMissionFail": "失败的原因" //当前端读取到这个，意味着指令序列结束
}
```

后端会向前端发送一个列表，列表的每一项代表小人应当顺序执行的一个操作，其中包含：左转、右转、向上走、向下走、向左走、向右走、打开宝箱成功、打开宝箱失败。该列表的最后一项为“endMissionSuccess”或者“endMissionFail”其中之一，代表指令序列结束且任务成功/任务失败。

6.4. 地图编辑器界面的数据交换格式

地图编辑器功能，由前端发送给后端

```
{
   "user_id": "用户 id",
    "map": {
        "name": "地图名称",
        "length": "地图长度",
        "width": "地图宽度",
        "state": [[2, 1, 1, 1, 1],
                  [1, 1, 3, 1, 1],
                  [1, 2, 2, 1, 1],
                  [1, 1, 1, 1, 1],
                  [1, 1, 1, 1, 1]
                 ],  // 示例 5*5 地图，1：空格，2：障碍，3：宝藏
        "start": {
            "x": "x 坐标",
            "y": "y 坐标"
        },
        "end": {
            "x": "x 坐标",
            "y": "y 坐标"
        },
        "treasure": {
            "x": "x 坐标",
            "y": "y 坐标"
```

```
        },
        "character": {
            "state": "任务朝向",
            "type": "人物形象", //人物形象，1 为宇航员，2 为遥控机器人，3 为外星人
        }
    }
}
```

在地图编辑器页面，当玩家编辑完地图后，前端会将玩家设计的地图发送给后端，其内容包括：用户 id（在储存该地图到数据库中时，会使用玩家的 id 作为主键）、地图信息（包含地图名称、地图长宽、地图每一格的内容、起点、终点、宝藏位置）、人物信息（包含人物初始朝向以及人物形象）。

7. 重要函数与算法

1) 代码分析函数 game()

函数功能：接收来自前端的包含游戏代码的 json 包，并返回解析的动作序列、得分等内容给前端。（数据交换格式见前面第 5 点）

函数实现：先从数据库中查找接收到的地图 id，得到该地图的相关数据，并使用 transfer()将地图数据格式化。而后通过 code_action()函数，根据现有人物数据及前端传来的代码序列进行解析，得到人物的动作序列，期间可能需要调用 inspect()函数来判断游戏中人物前方是否有障碍、宝箱等。最终，根据解析后的人物位置、终点位置及宝箱开启情况判断游戏任务是否成功完成。

2) 自动发送认证邮件 send_email()

函数功能:在用户注册之后实际登陆之前,发送一份认证邮件到用户的注册邮箱中。

函数实现：实际上 python 中已经内置了一个 smtp 邮件发送模块，并且 django.core.mail 子包封装了电子邮件的自动发送 SMT 协议。

3) 自制地图保存函数 map_editor()

函数功能：接收前端传来的自制地图 json 包，并将该地图数据保存在数据库中。

4) 地图信息读取函数 users_maps_info()及 map_info()

函数功能：前者是读取所有用户自制地图信息并返回相应 json 包，后者是通过地图 id 读取游戏自带地图并返回相应的 json 包。