# 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
2） Web 端样式库：Ant Design
3） 数据库：MySQL
4） Python 版本：Python3
5） 编辑器：Visual Studio Code
6） 服务端框架：Django
7） 在线编程语言：Coffeescript
8） 编程游戏操作框架：Scratch

# 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：

- 任务目标：沿直线通过目标区域，到达另一侧的出口。
- 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。
- 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。

### 3.2.2. 了解顺序结构：

- 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。
- 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。
- 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。

### 3.2.3. 设置参数

- 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。
- 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
```

- 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。

### 3.2.4. 选择条件语句

- 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。
- 实现方式：通过 if-else 语句判断给出的问题并进行回答。
- 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。

### 3.2.5. 循环语句

- 任务目标: 打开宝藏
- 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。
- 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. 数据库模式

## 4.1. 地图数据

```
class Map(models.Model):

    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=128)
    length = models.IntegerField()
    width = models.IntegerField()
    state = models.CharField(max_length=200)
```

```
8.
9.      #关卡起点、终点和宝藏坐标
10.     startx = models.IntegerField()
11.     starty = models.IntegerField()
12.     endx = models.IntegerField()
13.     endy = models.IntegerField()
14.     treasurex = models.IntegerField()
15.     treasurey = models.IntegerField()
16.
17.     #人物形象和初始时朝向信息
18.     characterType = models.IntegerField()
19.     characterState = models.CharField(max_length=2)
20.
21.     class Meta:
22.         ordering = ['id']
```

## 4.2. 玩家已过关卡数信息

```
1.  class PlayerInfo(models.Model):
2.
3.      name = models.CharField(max_length=128, unique=True) #用户名
4.      passMapNumber = models.IntegerField() #已过关卡数
```

## 4.3. 玩家已过关卡历史信息

```
1.  class HistoryInfo(models.Model):
2.
3.      name = models.CharField(max_length=128, unique=True) #用户名
4.      mapNum = models.IntegerField() #第几关
5.      score = models.IntegerField() #得分
6.      code = models.TextField() #过关代码
```

# 5. 数据交换格式

## 5.1. 登录

要求提供包含用户 id，用户名，用户级别的 JSON 信息。

## 5.2. 地图

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
    }
}
```

## 5.3. 代码

### 5.3.1. 代码分析函数

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
    }

}
```

### 5.3.2. 动作序列

由后端发给前端

```
{
    "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
}
```

## 5.4. 地图编辑器

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
        }
```

```
29.     }
30. }
```