

《pythOn程序设计基础》系统开发说明

项目：游戏《吃定你了》

设计时间：6月1号~6月26日

开发人员：陈丹琪、黎可馨、林恺怡、曾晓琳、张澈然

组长：张澈然



**一、产品设计方案**

**1.1 产品目标**

参考综艺《名侦探学院》中的同名游戏，我们设计了《吃定你了》双人益智小游戏，旨在提高使用者的临场反应能力及逻辑思维，在游戏中收获快乐的同时锻炼思维、活动大脑。同时，制作者亦能在其中将编程落于实践，对游戏编程有更深入的学习和了解，提高个人思考能力和小组合作能力。

**1.2 产品内容总策划：**

**1.2.1 流程规划**

（1）确定编程方向及小组成员分工

编程方向：

益智游戏、pygame第三方库的学习实践、本地双人游戏

小组成员分工：

游戏设计框架讨论：全组成员

主程序：张澈然，林恺怡

落子模块：张澈然

双人游戏模块：林恺怡，陈丹琪

输赢判断：黎可馨

程序打包：黎可馨

音乐版块：曾晓琳

美工：张澈然

游戏底层：张澈然

游戏测试：陈丹琪

汇报ppt：黎可馨

小组报告方案制作：曾晓琳

小组汇报：全组成员

（2）预测可能需要的第三方库及编程知识

第三方库：pygame numpy PIL

编程知识：函数的自定义及调用、面向对象编程、程序的分支结构 对象创建、程序的循环结构、页面输出、音乐的导入及循环播放、列表类型、排列组合分析等

（3）各自学习并编写运行负责版块的程序

（4）整合所有成员的程序代码

（5）运行并进行调整试玩，完善后续操作

（6）上交程序并完成报告及PPT，模拟答辩

**1.2.2 设计与测试规范**

1.2.2.1设计思路：

1、用面向对象编程和自定义函数来设计游戏底层

2、使用数组储存棋盘和棋子状态

3、接收用户的输入信息，或鼠标选择的位置，给出用户反馈

4、游戏底层中，有如下自定义函数

Chessboard 表示棋盘状态

Winer\_check 检查当前回合胜利方

Check chess 检查行、列、对角线棋子状态

Set\_chess 检查棋子类型是否充足，判断当前位置的棋子状态

Remove\_chess 删除当前位置棋子

Move\_chess 移动棋子

Check\_chess\_box 检查当前棋子类型是否剩余

Refresh\_chess\_box 更新棋子库

Print\_chessboard 打印棋盘

3、结束游戏，判断游戏中的胜利方

1.2.2.2 测试规范：

运行环境：pycharm

解释器：python3

顺利运行：运行程序，随即进入A队回合，输入1实现落下棋子，接下来输入0-8实现棋子在九宫格的落下的位置，输入1-3决定落下棋子的大小；进入B队回合，使用同样方法落下B队的棋子。进入A队回合，输入1或2决定落下棋子或移动棋子。如果选择2：移动棋子，则输入棋子原来位置和移动后的位置代表的数字，中间用空格隔开。A、B队轮流落子，直至一队的棋子首先在九宫格连成一条线，则系统判断该队伍取得胜利。

非正常情况：界面闪退，显示“输入错误”（但游戏仍能正常进行）

**1.3 开发日程表**

|  |  |  |
| --- | --- | --- |
| 时间 | 安排 | 细则 |
| 6.1—6.5 | 确认编程方向及游戏规则，安排成员分工 | 创建会议室共同讨论，权衡能力利弊，确定制作方向及分工 |
| 6.5—6.24 | 制作方案，编写程序 | 网络自行学习负责版块所需内容，，创建自己的GitHub账号 |
| 6.24—6.25 | 整合小组成员程序，运行并处理Bug | 各自程序以压缩包形式上交 |
| 6.25—6.26 | 完善程序，进行检测与试玩 | 由组长整合所有人的程序 |
| 6.26 | 制作报告及PPT，模拟答辩 | 所有人参与 |
| 6.26 | 上交程序至GitHub |  |

**1.4 技术解决方案**

**1.4.1 遇到的问题**

1、游戏界面的按键实现的条件设置无法达到点击按键的预期情况，

2、联网需要训练AI，人机对战需要获取ip地址。但以上两项对编程能力要求较高，我们在短时间内还无法完成。

**1.4.2 对应的解决方法**

1、为了符合游戏界面按键设置条件，现选择添加设置鼠标光标位置的代码，用来达到预期的点击效果。

2、将游戏简化成本地双人游戏

**二、产品实现方案**

**2.1 系统的主要功能**

1、将课堂所学知识运用到实践中，对python编程有更深刻的理解和学习

2、利用pygame进行电脑小游戏程序编写，控制游戏进行

3、活动使用者大脑，在游戏中锻炼思维

**2.2 关键技术和技术难点**

**2.2.1 关键技术**

1、所需的第三方库

2、分支结构处理游戏界面切换问题

3、自定义函数实现游戏的棋盘和棋子状态存储

4、某些第三方库的使用方法

5、处理编程Bug，如落子程序的修改完善

6、对棋盘上棋子存在情况进行排列组合分析

7、游戏主程序的编写完善

**2.2.2 技术难点**

1、相关编程知识不熟悉，编写时部分bug难以解决

2、程序在分部分编写时局部效果不明显且所有程序整合成同一份代码时出现的问题需要依次解决，不断完善

3、写游戏底层的逻辑和用pygame添加界面的逻辑有些区别，导致在有限的时间内未能完成所有部分的界面处理

4、游戏界面按键设置

5、游戏的视觉效果体验及上手试玩的直观感受

**2.3 存在的问题和已完成的改进**

**2.3.1 存在的问题**

1、在游戏界面中无法实现将棋子移动至棋盘

2、游戏菜单中两处选项后的界面重合

3、双人游戏程序进行时游戏界面卡顿或闪退

4、游戏一方已胜出时，另一方仍然显示游戏继续

5、胜利队伍判断错误

6、联网人机对战难度超出预期设想

7、成功将py文件打包成exe应用程序，但是无法将其运行

**2.3.2 已完成的改进**

1、游戏菜单中两处选项后的界面重合

2、修改程序后棋子能顺利移动至棋盘进行游戏

3、调整联网人机对战模式为本地双人对战模式

4、系统能正确判断游戏胜利方

**三、测试大纲和测试报告**

**3.1 测试大纲**

**3.1.1 测试目的**

1、测试已有的游戏界面切换是否正常

2、测试游戏底层是否能按规则正常运行，并且不会出现输赢判断错误

**3.1.2 测试环境**

Pycharm

解释器 python3

**3.1.3 测试项目**

1、系统的安装与卸载

2、软件功能测试

3、安全可靠性

4、用户界面

5、文字符合性

**3.2 测试报告**

**3.2.1 运行程序**

（1）测试程序是否可以运行；

（2）测试A，B两方棋子是否可以按照指令在指定区域内放置和移动棋子；

（3）测试棋子在九宫格的水平，垂直，对角线连成一条线是否可以判断游戏结束，得出一方胜利的结论；

（4）测试棋子是否可以被比它大的棋子吃掉；

（5）反复测试两队游戏的各种胜利情况，确认程序是否将所有情况兼顾到；

**3.2.2 发现问题**

（1）左上到右下这一条对角线成功连成，但程序没有判断游戏结束；

（2）A，B两队得出结果，但系统判断的胜利一方实为失败一方；

**3.2.3 后续操作**

将问题反馈给组员，改正程序中的错误，继续按原来方法测试，直至游戏的所有环节都不出现错误。

**四、产品安装和使用说明**

**4.1 产品安装**

下载压缩包，解压文件，打开python程序即可运行

**4.2 使用说明**

**4.2.1 游戏启动：**

下载压缩包后解压，将文件夹中的所有py文件导入pycharm中打开，运行main.py程序即可启动游戏界面。运行game.py可测试游戏底层。

**4.2.2 游戏规则：**

游戏双方各拥有大中小号箱子各两个。 双方轮流在九宫格棋盘中放入或移动一个箱子。每个箱子都可以盖住任意比他小的箱子并将其吃掉。 若其中一方的箱子率先在棋盘上连成一条直线，则该方获胜。游戏提供红、绿两种颜色的箱子。选择红色箱子的一方为先手。

**4.2.3 游戏须知：**

请勿单独提取python文件或将文件夹删除，否则游戏将无法运行。