河南理工大学

《Python 程序设计》

课程设计报告

（2018 —2019 学年第 一 学期）

题　　目 2048 游戏设计

学生姓名 郑 双 双

专业班级 信管 1602

学生学号 311609030205

教师姓名 徐 文 鹏

成　　绩：

评　　语：

教师签名：

日期：

目录

Python之2048小游戏课程设计

一、设计目的

1.课程设计教学目的:

(1)掌握python语言的程序设计方法，能够在学习专业基础课和《Python程序设计》课程的基础上，编写出具有一定功能的代码实现。

(2)本课程设计旨在加深对Python程序设计的认识，对Python语言及其语言生态有一个进一步的掌握和应用。

(3)学会运用Python标准库及外接相关库来解决实际问题的基本能力，培养和提高学生分析问题、解决问题的能力。提高学生使用Python为开发语言来进行问题描述、交流与思考的能力，提高学生实践论文撰写能力，为毕业设计和以后的工程实践打下良好的基础。

2.本课程设计具体目的：

继Flappy Bird之后，全球最火的一款游戏莫过“2048”了，这是一款看起来异常简单玩于起来却异常虐心的益智小游戏，玩家需要在16个格子中通过数字叠加的方法将最初的数字2凑成数字2048，在玩的过程中锻炼思维能力。学过python语言后，想把理论付诸于实践，所以利用这次python课设将这款游戏用代码实现了。

二、设计要求

1.课程设计教学任务和要求：

(1)课程设计的基本要求是：在课程设计的各个阶段严格、规范地完成相关的文档，例如总体方案报告，详细设计报告、功能说明、数据结构说明、算法说明、程序设计框图、图例和源程序等。要求所写文档结构合理、内容完整、叙述清晰。程序源码要有详细注释，可读性好。更高要求是：有创意、系统界面美观。

(2)由于课程设计项目具有一定的综合性，鼓励具有不同特长和不同能力的学生互相组队。项目小组自己推荐一名组长，实行“组长负责制”。组长组织组员进行项目选题、任务分配、方案确定、方案设计、系统调试测试，组员分工协作。小组成员开展项目讨论，互相支持，形成协作意识。

2.本课程设计具体任务和要求：

本课程设计主要任务是以Python为开发语言完成一个100~300行左右规模的程序项目开发。我们组通过python语言编写一个2048小游戏，能够通过简单的按键实现数字的叠加，带有美观的可视化界面，能够带给用户良好的游戏体验。具体要求是游戏的操作只有上下左右的滑动,每次滑动的结果是数字向该方向滑动，相同的并且相邻的数字相加,数字相加有一定的顺序，方向上靠前的数字先与相邻的数字相加，数字每次滑动只进行一次相加，比如向左滑2， 2，2，2滑动之后是4，4而不是2，4，2或者8。数字之间的计算只在滑动所在的一行或者一列。滑动之后随机在没有数字的地方出现2或者4。如果在某个方向上没办法移动并且没有数字,相加就不会出现数字，所有数字无法移动之后结束游戏或者出现2048这个数字。实现积分制，结束游戏之后可以自动重新开始。

三、总体设计

1．小组任务分配情况：

我们组通过python语言编写一个2048小游戏，能够通过简单的上下左右按键实现数字的叠加，能够锻炼玩家思维能力，带有美观的可视化界面，带给用户良好的游戏体验。

李杨（组长）：绘制界面，绘制图案和显示文字

郑双双（组员甲）：捕捉用户输入（上下左右按键，或者对应的滑动）

2．2048游戏设计思想：

（1）大家都玩过2048，我们可以认为4*4的方块是个矩阵，开始是4*4的零矩阵。游戏开始在任意地方出现2或4，以后每次出现的数字都是2或者4。然后我们可以上下左右移动，移动的规则是例如向左动，某一行(左移只需要考虑每一行)的数比如是[2,4,0,2]向左移动,移动后变成[2,4,2,0],移动后不允许(每行或者每列，与移动方向有关)两个非0数字之间有0的存在。移动前相邻两个数相同的话会合并，例如[2,2,4,4]会合并成[4,8,0,0]。

（2）移动合并完后，会在所有为0的位置随机挑选出一个位置填上2或者4，记住是先移动合并完后才会随机填上2或者4。

（3）有些时候我们发现往某个方向无法移动，向左和向下都无法移动，向左移动的话每一行移动后还是原来老样子，因为每一行任意相邻两个非0数之间不存在间隔且无法合并。向下移动的话每一列还是原来老样子，因为每一列任意相

邻两个非 0 数之间不存在间隔且无法合并。在无法移动(即移动后还是老样子的情况下)不会在随机 0 位置处添加随机数 2 或 4。只有移动后改变了矩阵的原来样子且矩阵最小值为 0 才会在随机 0 位置出添加随机数 2 或 4。

（4）有两种情况移动后不会添加随机数 2 或 4，第一种情况是上面这种情况，往一个方向移动没有效果。另外一种是矩阵都为非 0 数，没有位置添加随机数 2 或 4。

（5）游戏结束的情况，当矩阵没有 0 而且每行每列任意两个相邻数无法合并。

3．软件功能图：

会在空白位置出现一个 2 或 4，每列两个相同数字合并，没有相同数字则不变再填充到上面的方格中，相对位置上面的方格里显示叠加后的数值，下面的变为空，然后其他没有数字的方格中再随机出现一个 2 或 4.

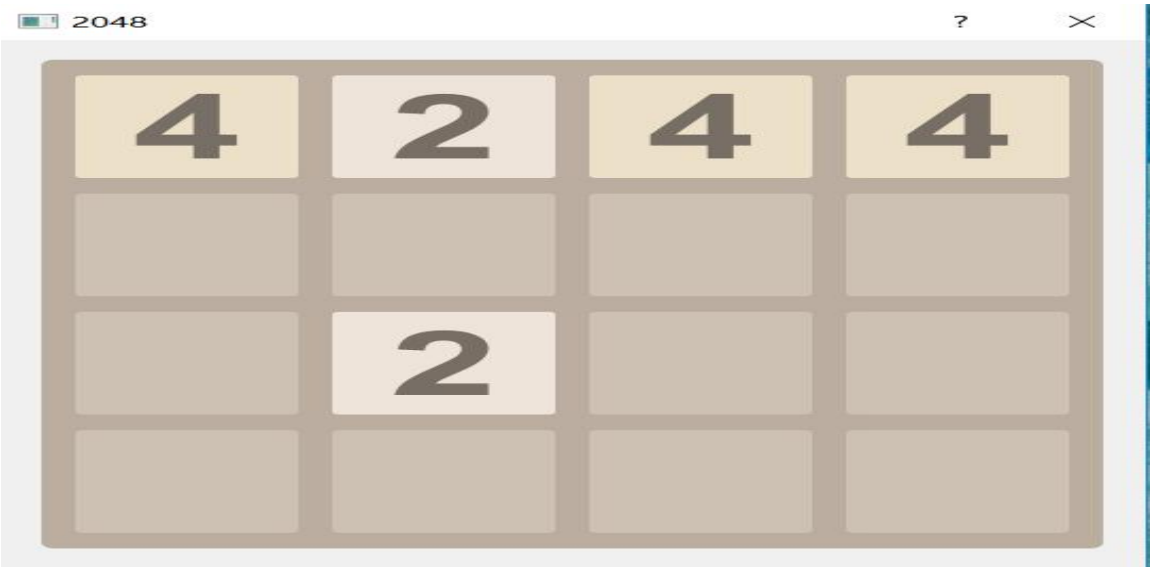

图 3-1

（2）按“ ↓”可以使数字整体下移，每按一次都会在空白位置出现一个 2 或 4，每列两个相同数字合并，没有相同数字则不变填充到下面的方格中，相对位置下面的方格里显示叠加后的数值，上面的变为空，然后其他没有数字的方格中再随机出现一个 2 或 4。

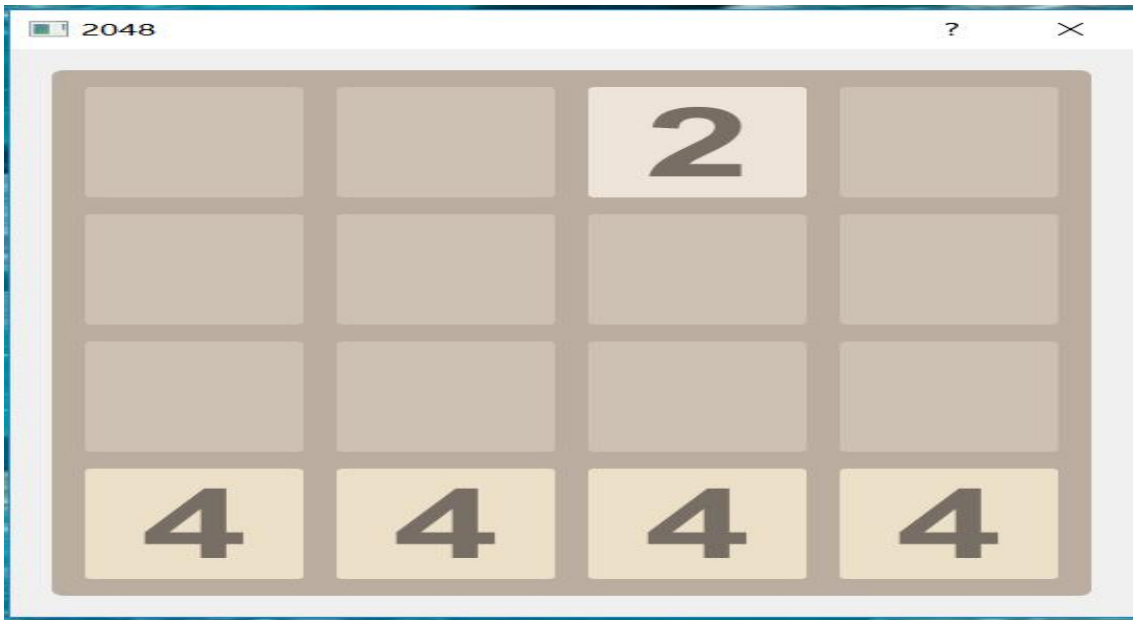

图 3-2

（3）按“⟶ ”可以使数字整体左移，每按一次都会在空白位置出现一个 2 或 4，每行两个相同数字合并，没有相同数字则不变依次填充到右边面的方格中，相对位置右面的方格里显示叠加后的数值，左面的变为空，然后其他没有数字的方格中再随机出现一个 2 或 4。

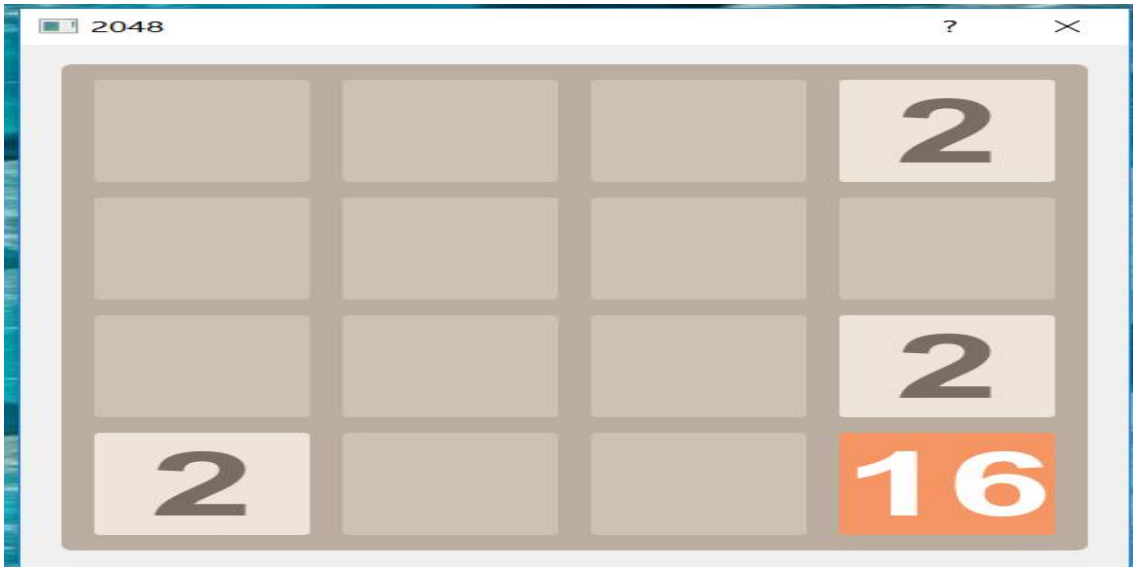

图 3-3

（4）按“ ←”可以使数字整体左移，每按一次都会在空白位置出现一个 2 或 4，每行两个相同数字合并，没有相同数字则不变再一次填充到左面的方格中，相对位置左面的方格里显示叠加后的数值，右面的变为空，然后其他没有数字的方格中再随机出现一个 2 或 4。

图 3-4

（5）当棋盘满或者不能移动的时候游戏结束。

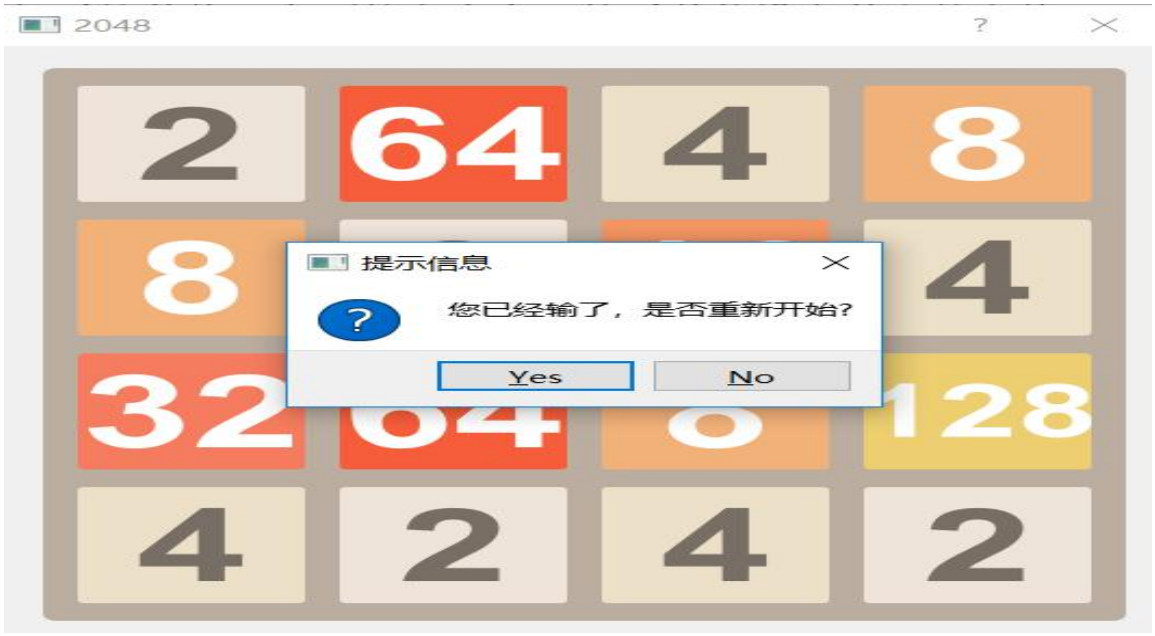

图 3-5

4．程序流程图：

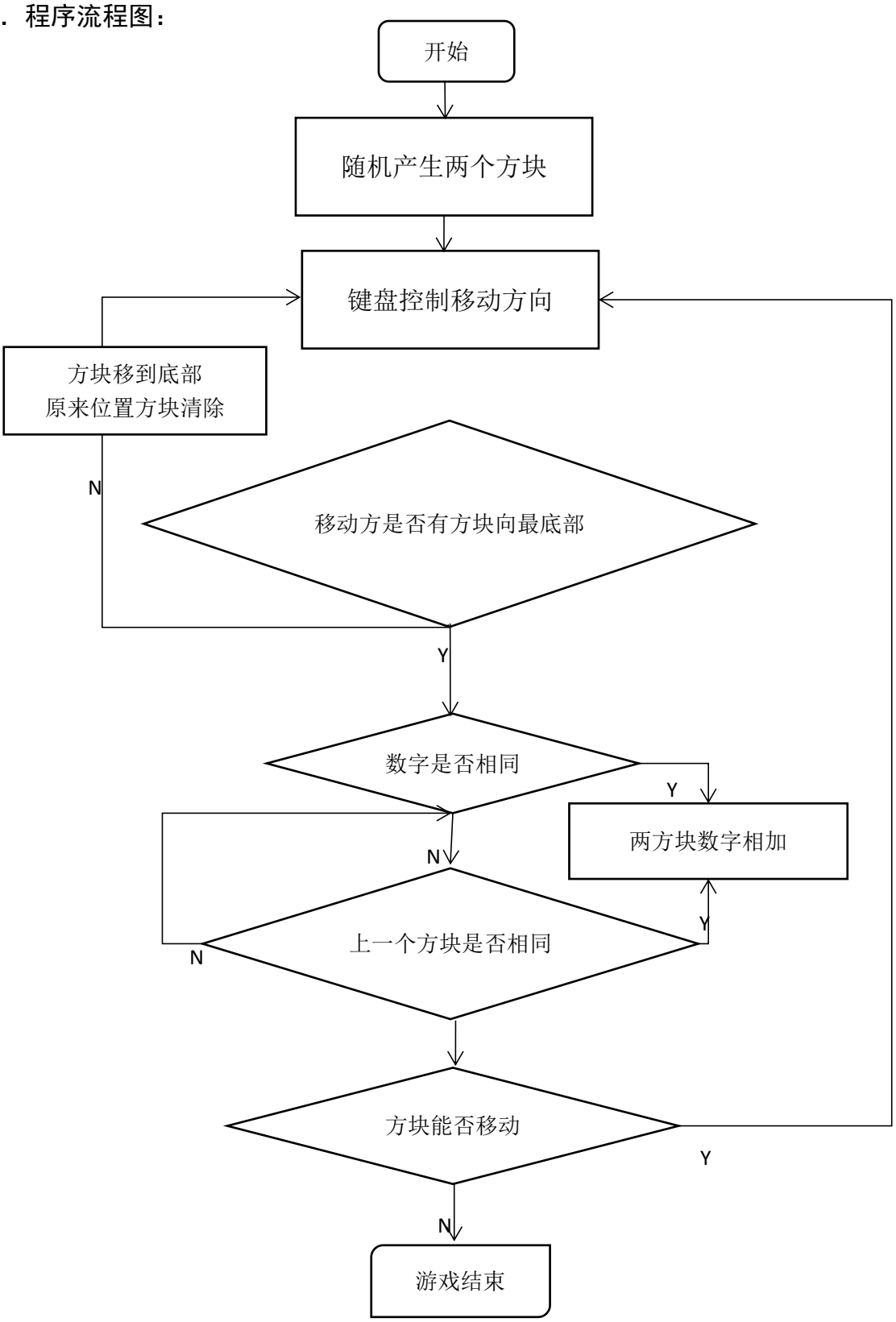

图 3-6 2048 小游戏程序流程图

四、设计实现

1. 最终实现结果：

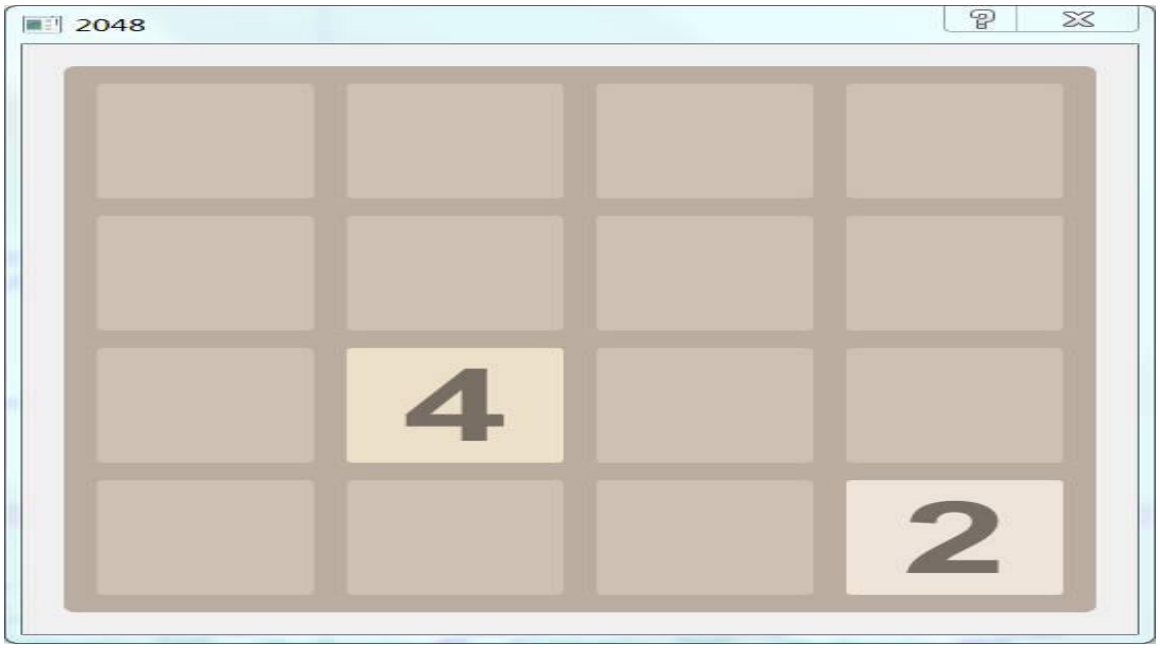

图 4-1 游戏启动界面

图 4-2 游戏结束界面

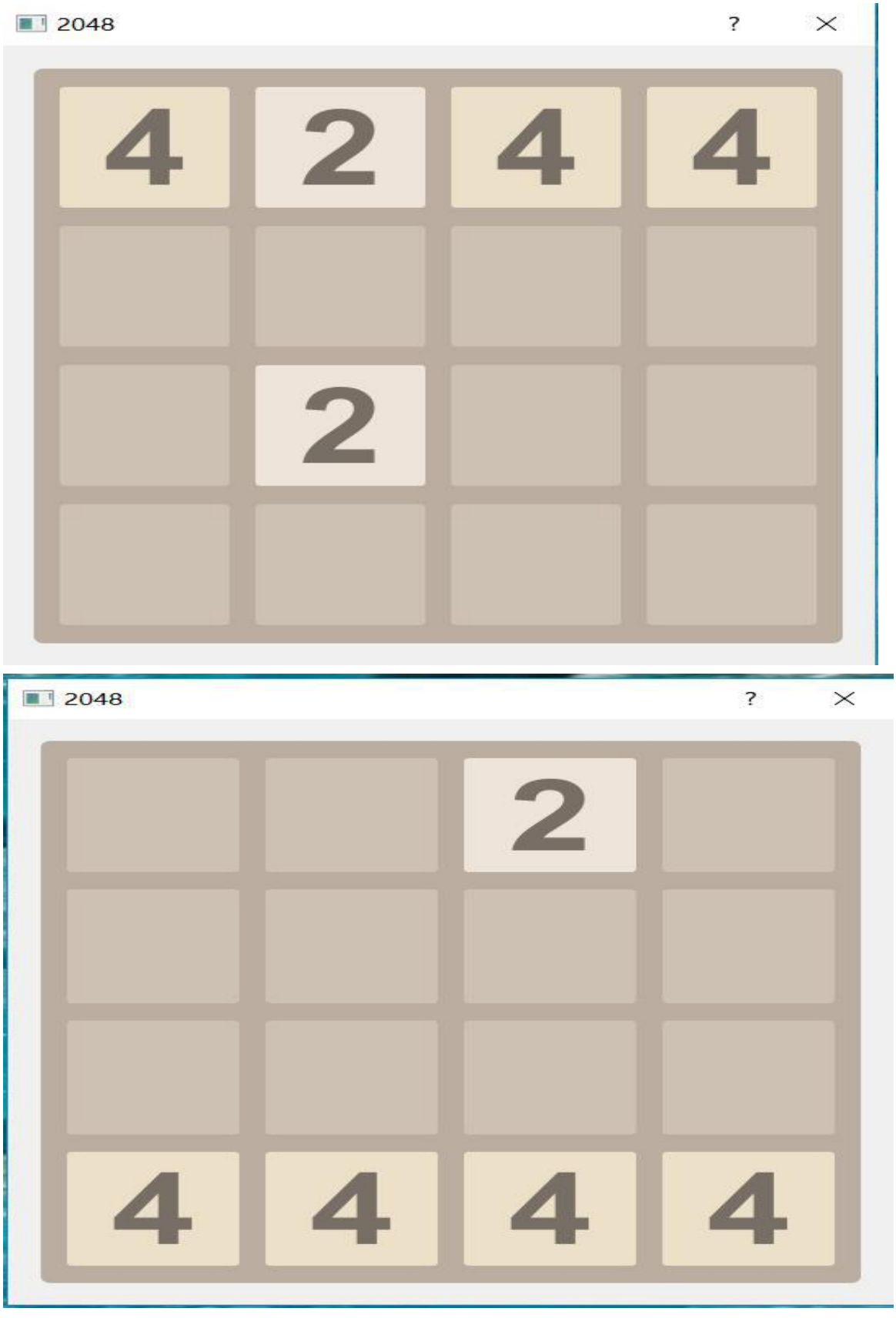

图 4-3 向下运行游戏界面

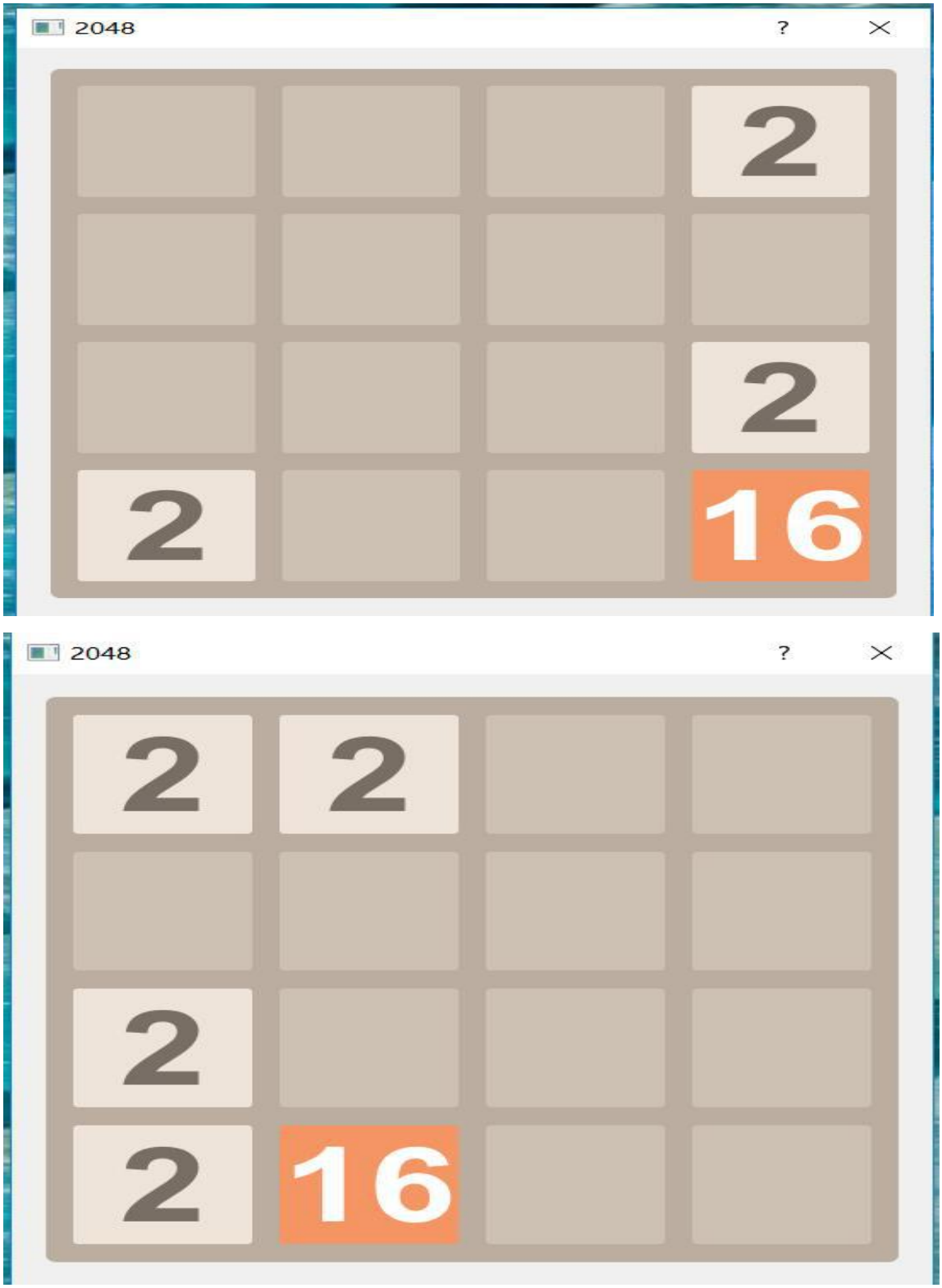

图 4-4 向左运行游戏界面

2. 实现结果评价：

我们组通过 python 语言实现一个 2048 小游戏，基本实现了最初讨论后想要实现的功能。

能够通过简单的按键实现数字的叠加，带有美观的可视化界面，给用户良好的游戏体验。通过上下左右的滑动实现数字向该方向滑动，相同的并且相邻的数字相加,数字相加有一定的顺序，方向上靠前的数字先与相邻的数字相加，数字每次滑动只进行一次相加，比如向左滑 2， 2，2，2 滑动之后是 4，4 而不是 2，4，2 或者 8。数字之间的计算只在滑动所在的一行或者一列。滑动之后随机在没有数字的地方出现 2 或者 4。如果在某个方向上没办法移动并且没有数字,相加就不会出现数字，所有数字无法移动之后结束游戏或者出现 2048 这个数字。

对比之前设置的要求和任务，我们仍存在一些功能尚未实现：

没有实现积分制，结束游戏之后不能自动重新开始，需要退出程序重新启动才能重新开始。总体来说我们的程序 80%功能或要求均已实现，设为良好等级。

五、详细设计

我主要负责捕捉用户输入来对游戏进行相应的计算以及游戏结果的判定。

1. 计算部分代码的解析：

```
class GameCalculate:#定义一个 GameCalculate 类
    def __init__(self, item_data):#当调用类的实例化方法时__init__()方法被调用进行初始化
        self.data = item_data

    def calculate(self, direction):#定义一个 calculate()方法执行矩阵各个方向的操作，根据移动方向，计算矩阵的状态值
        if len(self.data) <= 0:
            return 0
        if direction == 1:
            self.up_run()#方向 1 代表向上滑动，调用 up_run()方法
        if direction == 2:
            self.down_run()
        if direction == 3:
            self.left_run()
        if direction == 4:
            self.right_run()

    def get_data(self):#定义 get_data()方法返回当前方格中的数值
```

```
        return self.data
#四个方向上操作基本相同，选择向左滑动时的计算进行描述
def left_run(self):#定义方法捕捉用户向左滑动时所要执行的操作
        for i in range(0, 4):#遍历每一行
            k = None
            for j in range(0,4):#遍历每一列
                l = self.data[i][j]#将遍历到的数字存入二维列表中
                if k and l["Number"] != 0:#如果 k 不是 0 且列表非空
                    k_v = self.data[k["i"]][k["j"]]#将 k 中的数字复制给
一个新列表
                    if k_v["Number"] == l["Number"]:#如果相邻两数相同
                        self.data[k["i"]][k["j"]]["Number"]          =
k_v["Number"] + l["Number"]#合并
                        self.data[i][j] = {"Item": None, "Number": 0}
                        k = None
                    else:
                        k = {"i": i, "j": j}#若相邻两个数不相同，不合并，
数不变
                else:
                    if l["Number"] != 0: k = {"i": i, "j": j}
        for i in range(0, 4):
            for j in range(0, 4):
                if self.data[i][j]["Number"] != 0:
                    if j != 0:
                        for k in range(0, j):
                            if self.data[i][k]["Number"] == 0:
                                swap = self.data[i][k]
                                self.data[i][k] = self.data[i][j]
                                self.data[i][j] = swap
                                Break
```

#每行每列遍历，将第一个非零的数存入数组，继续遍历，当遇到第一个空格时将两个方格交换，依次循环，最后实现整体向左滑动的效果。

(1)输入输出分析：控制台输入有玩家按键输入，内部参数输入有当方向是1,2,3，4 的时候程序自动调用 up_run()，down_run()，left_run()，right_run()方法。

输出是以计算机屏幕为输入目标，当玩家执行上下左右按键时，棋盘中的数字做相应的变化，保证游戏的正常进行。

(2)算法的详细说明：

class GameCalculate:定义一个 GameCalculate 类，便于子类继承和调用。

def __init__(self, item_data):当调用类的实例化方法时__init__()方法被调用进行初始化，实例化。

def calculate(self, direction):定义一个 calculate()方法执行矩阵各个方向的操作，根据移动方向，计算矩阵的状态值

def left_run(self):#定义方法捕捉用户向左滑动时所要执行的操作。

（3）计算流程图

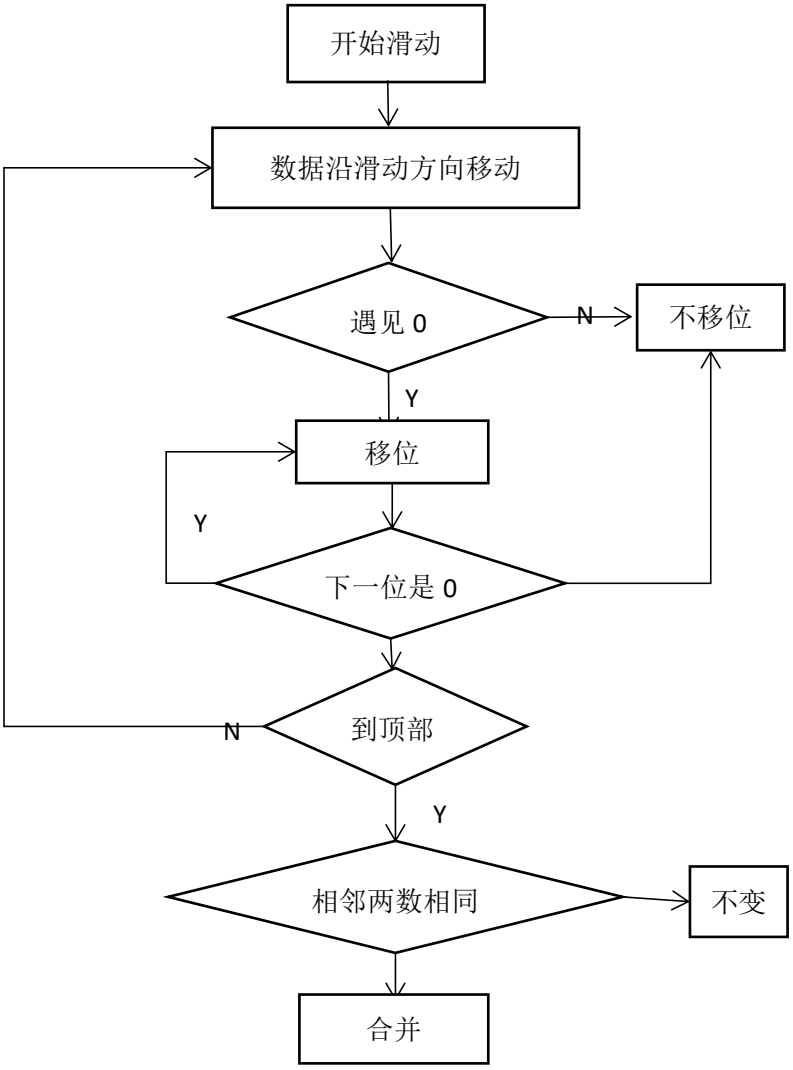

图 5-1 计算流程图

2.游戏结果判定代码描述：

```
def random_rect_item(self):#产生随机数坐标
        zero_ds = []#定义一个空列表
        for i in range(0, 4):
            for j in range(0, 4):
                if self.item_data[i][j]["Number"] == 0:#如果出现空格
                    zero_ds.append({"i": i, "j": j})##将第 i 行 j 列的值
加入到列表中

        if len(zero_ds) > 0:                    #列表非空
            rnd = int(random.uniform(0, len(zero_ds)))#在列表长度范围内
随机产生一个整数
            k = int(random.uniform(0, 10))#k 为 0 到 10 之间的随机整数
            number = 2                          #初始化 number
            if k == 2 or k == 10:
                number = 4#产生随机数
            d = zero_ds[rnd]#将产生的随机数赋值给 d
            d["num"] = number
            d["x"] = self.item_bg_pos[d["i"]][d["j"]]["x"]
            d["y"] = self.item_bg_pos[d["i"]][d["j"]]["y"]
                                                #得到随机数的坐标位置
            rect = NumberRect(self.parent, self.rect_width, d)
            self.item_data[d["i"]][d["j"]]["Number"] = number
            self.item_data[d["i"]][d["j"]]["Item"] = rect
            rect.show()   #得到相对应的数字颜色，方格背景，最后显示出来

            return self.item_data[d["i"]][d["j"]]
        else:
            reply = QMessageBox.question(self.parent, '提示信息',
                                         "您已经输了，是否重新开始?",
QMessageBox.Yes, QMessageBox.No)
            if reply == QMessageBox.Yes:
                print("OK")
            else:
                print("NO")
        return None
#没有空格，棋盘已满，输出提示窗口游戏失败
      def keyPressEvent(self, QKeyEvent):#定义 keyPressEvent()函数获取
按键信息
        if QKeyEvent.key() == Qt.Key_Up:
            self.reset_rect(1)#调用 up_run()方法
        elif QKeyEvent.key() == Qt.Key_Down:
            self.reset_rect(2)
```

```
        elif QKeyEvent.key() == Qt.Key_Left:
            self.reset_rect(3)
        elif QKeyEvent.key() == Qt.Key_Right:
            self.reset_rect(4)

    def reset_rect(self, direction):#重置棋盘函数
        print("方向:" + str(direction))#打印方向+数字
        items = []
        for i in range(0, 4):
            for j in range(0, 4):
                item = self.item_data[i][j]["Item"]
                if item:
                    items.append(item)#将之前产生的随机数加入棋盘中

        calculate = GameCalculate(self.item_data)#调用 GameCalculate 类
进行当前数字的合并计算
        calculate.calculate(direction)#得到方向确定应调用的函数
        self.item_data = calculate.get_data()

        for d in self.item_data:
            print(d)
        for p in self.item_bg_pos:
            print(p)

        self.redraw_rect(items)
        self.check_is_win()
        self.random_rect_item()

    def redraw_rect(self, items):#重新定义画布函数
        for i in range(0, 4):
            for j in range(0, 4):
                item = self.item_data[i][j]["Item"]
                if item:
                    items.remove(item)
                    x = self.item_bg_pos[i][j]["x"]
                    y = self.item_bg_pos[i][j]["y"]#得到当前数字坐标
                    ds = item.ds
                    ds["x"] = x
                    ds["y"] = y
                    ds["num"] = self.item_data[i][j]["Number"]
                    item.refresh_ds(ds)#得到更新后的数字颜色
        for k in items:
            if k: k.hide()
```

```
        del items

    def check_is_win(self):#再次检查游戏是否胜利
        for i in range(0, 4):
            for j in range(0, 4):
                item = self.item_data[i][j]["Item"]
                if item and item.ds["num"] == 2048:#每行每列遍历，如果存在方格中的数等于2048，
                    reply = QMessageBox.question(self.parent, '提示信息',
                                                 "真厉害，你已经赢了是否继续往上挑战？", QMessageBox.Yes, QMessageBox.No)
                    if reply == QMessageBox.Yes:
                        print("OK")
                    else:
                        print("NO")#输出提示信息
```

(1)输入输出分析：输入是棋盘上当前存在的所有数字，当遍历后棋盘满且最大数字＜2048，输出是游戏失败的提示窗口。当输入是最大数字为2048时输出窗口提示信息游戏胜利。

(2)有关算法的详细说明：

random_rect_item(self):产生随机数坐标

random.uniform()调用random库的uniform函数产生随机整数

keyPressEvent():keyPressEvent()函数获取按键信息，若界面上存在按钮，界面焦点默认在按钮上，此时空格键，回车键，方向键以及tab键均无法获取。

reset_rect()重置矩形

calculate.calculate(direction)调用calculate类的calculate()方法进行数的合并

redraw_rect()重置画布函数，将更新后的数字与棋盘颜色背景和字体一一对应

check_is_win()自定义检查游戏胜利函数，获取游戏结果

(3)程序流程图：

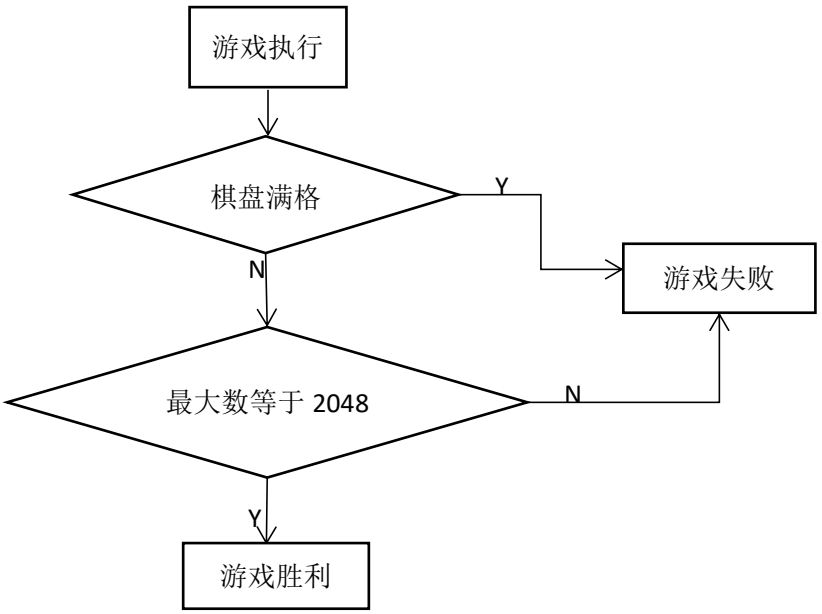

图 5-2 游戏结果判定流程图

六、调试与测试

一方面是对代码测试，最简单粗暴的方式就是用 print 所有可能的问题变量打印出来逐个查看，但这种方式到后来还要全部删掉，整个程序到处都是 print，有时候会因为不细心造成代码更大的错误。为了提高调试效率，我在室友介绍和网上搜索得知运用 pycharm 进行多线程调试，由于刚开始用，操作还不熟练，中间出现很多硬性问题，请教同学和百度最终顺利解决了。

首先在调试之前设置断点，断点可以设置在循环或者条件判断的表达式处或者程序的关键点。在代码编辑框中将光标移动到需要设置断点的行，然后直接按 Ctrl+F8 或者选择菜单"Run"->"Toggle LineBreakPoint",当调试开始的时候，当前正在执行的代码会直接显示为蓝色。在调试过程中有的时候需要追踪一些表达式的值来发现程序中的问题，可以通过选中该表达式，然后选择“Run”->” Evaluate Expression’在出现的窗口中直接选择 Evaluate 便可以查看。Pychar 同时提供了 Variables 和 Watches 窗口，其中调试步骤中所涉及的具体变量的值可以直接在 variable 一栏中查看。如果要动态的监测某个变量可以直接选中该变量并选择菜单”Run””->”Add Watch’添加 watches 栏中。当调试进行到该变量所在的语句时，在该窗口中可以直接看到该变量的具体值。在测试过程中一直出现 missing parentheses in call to ‘print’的问题，搞得很头大，后来找度娘才知道是因为版本不支持，下载了 python3 重新调试问题得到解决。解决之后运行有出现 importerror:no module named pyqt5,仍然不出现界面，问了同学才知道这是 python 的第三方库，需要单独安装，在控制台使用 Pip3 intall pyqt5 指令安装，后来解决了这个问题。重新运行代码出现游戏界面，游戏可以操作，基本完成设定的功能，说明代码本身没有大问题。

一方面是测试游戏能否在不同的计算机系统中运行，通过在 win 和 unix 系统上安装上 pyqt5 库进行测试，游戏可以在不同的计算机系统上运行并且操作流畅。没有出现什么问题。另一方面测试数字能否跟随按键方向而向相应的方向移动，通过按键观察数字移动方向发现跟按键方向一致，测试成功。

在本次程序的测试调试中总体比较顺利，对于 python 第三方库的基本知识还要多加理解和掌握，避免以后再出现类似的“低级”错误。

七、设计总结

经过一周左右的 python 课程设计， 终于把这项艰巨的任务完成了，我自己感受是挺辛苦的，从定考试时间起就一直有很多人吐槽，因为是第一次机考可能很多人排斥，一直就特别担心考试跟课设，徐老师工作真的特别认真，小组展示到晚上都顾不上吃饭，还让我们先去吃饭，也是对我们负责，想让我们学到实打实的本领。在这次课设中，我跟我的组长李杨同学按照老师制定的进度安排分组定题目，中间也换过好几次课题，由于技术不精，不想跟别人重复，毕竟人外有人，天外有天，人比人，气死人，哈哈。待在机房时间毕竟有限，其他时间待在寝室做课设，有时候交流沟通不太方便，就会降低完成进度和效率。

最后经过商量我们选定了在手机端很火的一个小游戏 2048，这个游戏操作简单界面简洁，很锻炼玩家的思维能力，我们本人也很喜欢玩。我们俩人花了大概两天时间把代码完工，我体会到团队协作很重要，因为 python 是面向对象编程的解释性语言，模块内部要求耦合性很大，有时候没有交流好去写，结果因为变量名或者一些函数名错误白白增加了很多错误，加重了任务量。

由于我电脑上 python3.7 突然出现了问题，我将代码放在新安装的 pycharm 软件中进行的运行调试，对这个软件不太熟悉的原因，中间走了很多弯路，不过最后还是调试成功了。写出来的游戏有一点不足就是没有积分显示，我觉得挺大问题，有点降低用户体验感，还有一个 bug，常理是当不能移动时游戏失败，但我们的游戏只要布满整个棋盘就显示游戏失败，这是一个需要改进的地方。如果再设计个作弊的小方法就更好了，这也是代码之后需要完善的地方。

整体来说，界面挺好，玩起来也很顺手，只要操控键盘即可。

两个月左右的 python 基础知识和一周左右的课设学习，我觉得比我之前学的 Java 还要 c 语言要简单好多，很容易入门，本来觉得自己不适合计算机，编程很枯燥乏味，但我觉得 python 好像有与众不同的“魅力”，我对它有浓厚的学习兴趣，希望自己接下来能保持这份热情，好好发掘 python 无穷的力量。