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 Python 之 2048 小游戏课程设计

学生姓名　　李　杨

专业班级　　信管 1602 班

学生学号　　311609030203

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 小游戏程序设计思想：

（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 且每行每列任意两个相邻数无法合并。

3．软件功能图：

(1)界面首先会随机出现两个 2，按“↑”可以使数字整体上移，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。

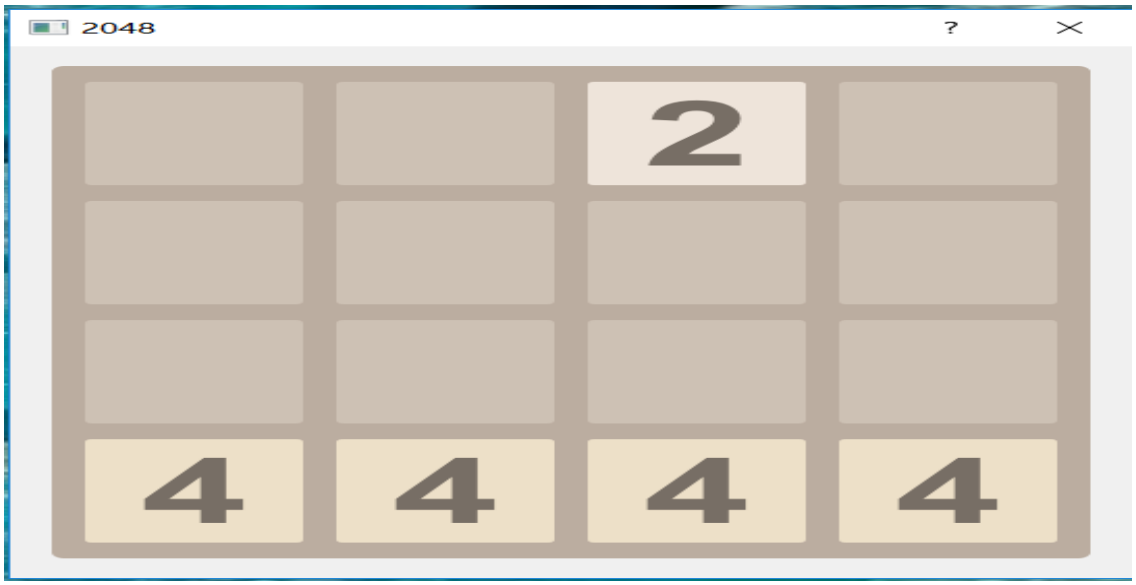

图 3-2

（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。

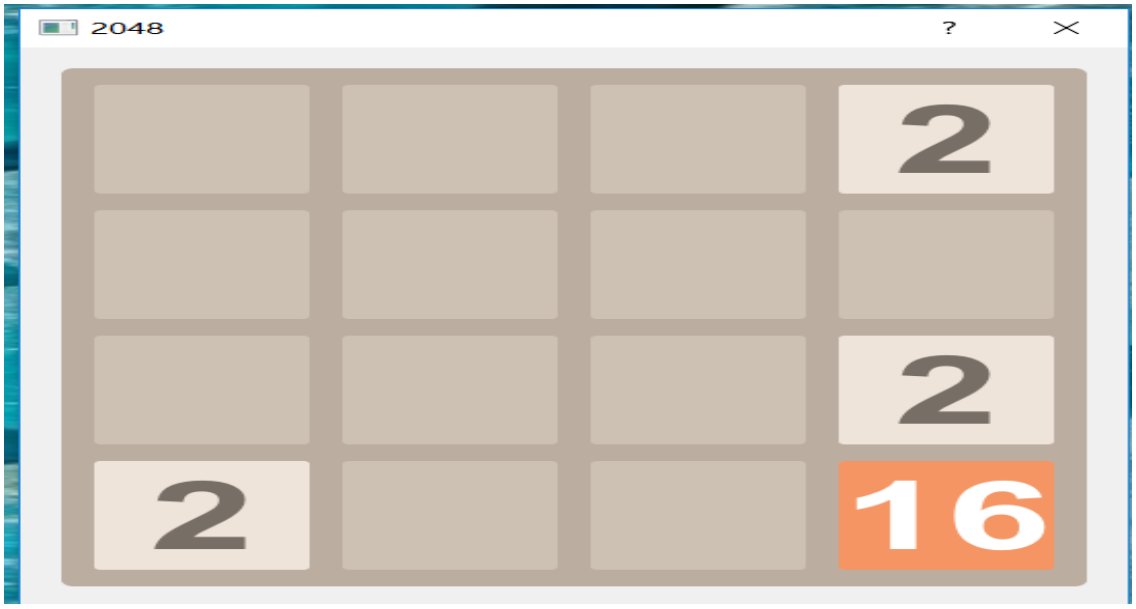

图 3-3

（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。

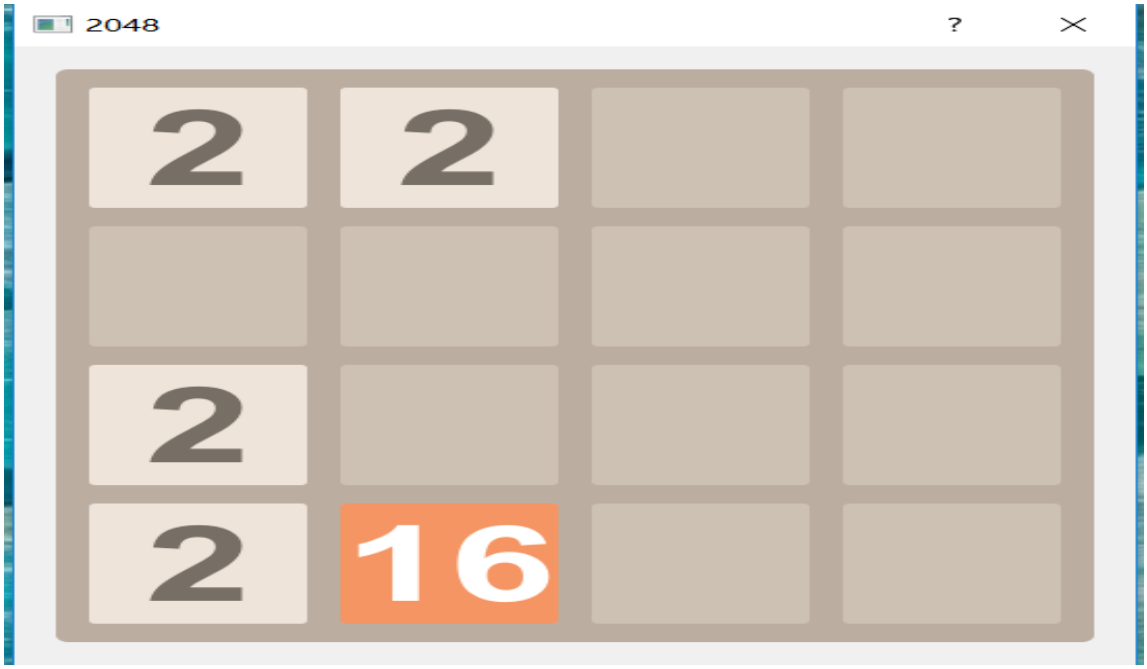

图 3-4

（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

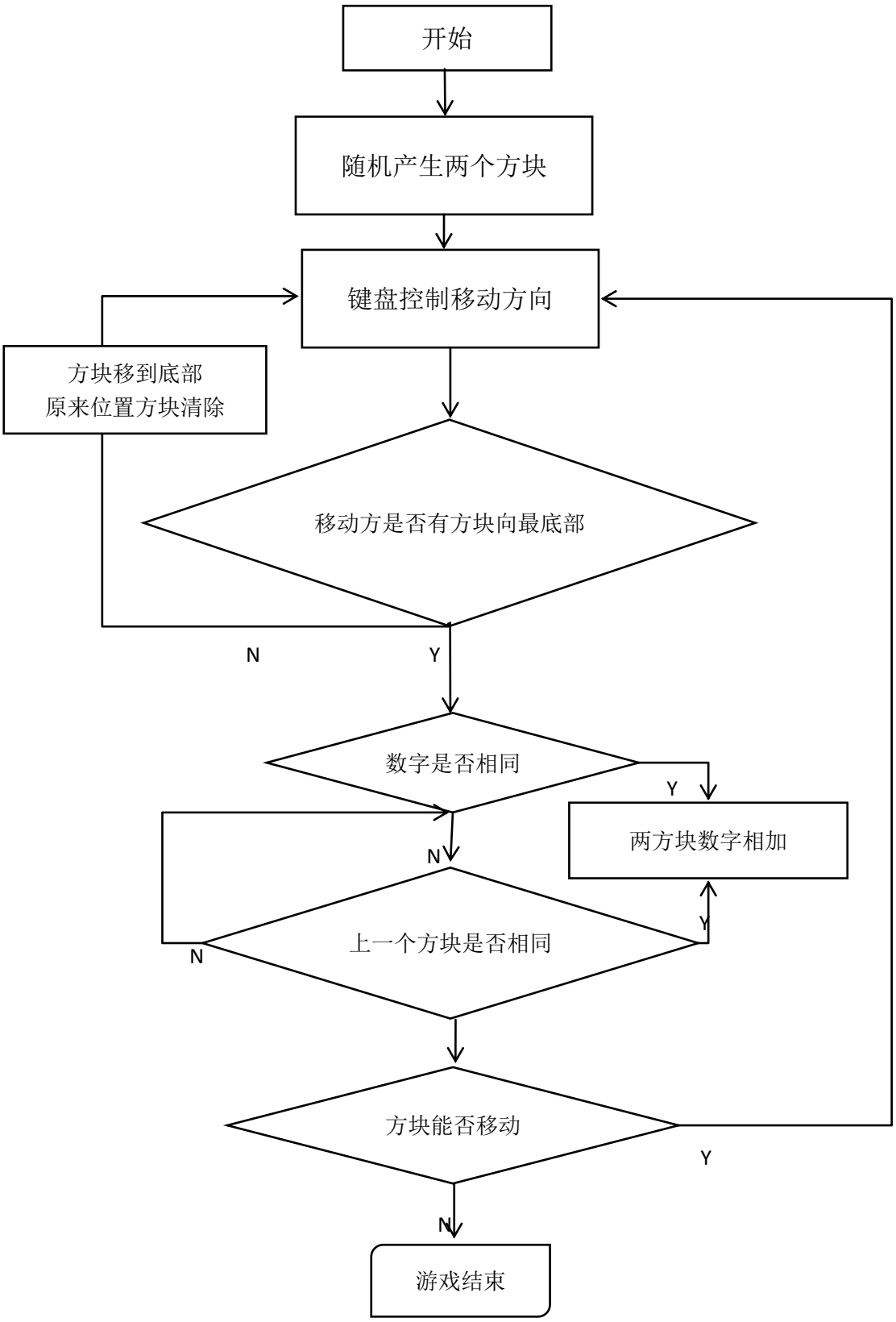

图 3-6 2048 小游戏程序流程图

四、设计实现

1 最终实现结果：

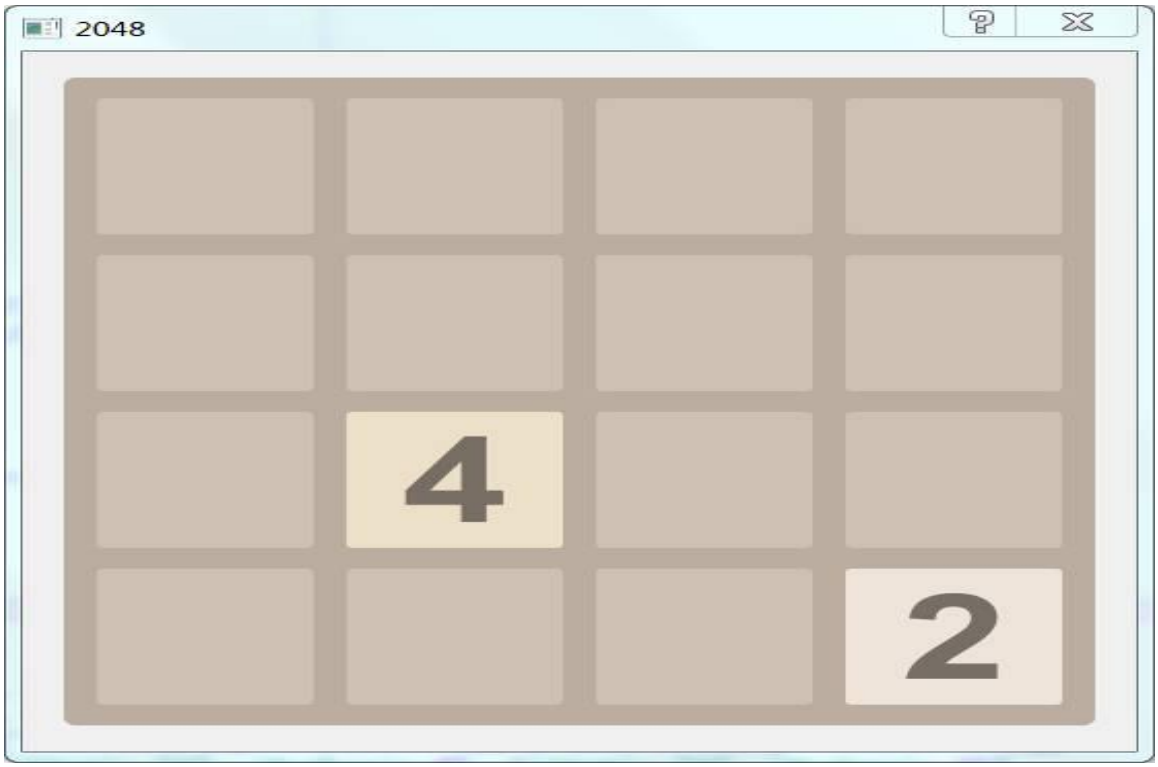

图 4-1 游戏启动界面

图 4-1 游戏结束界面

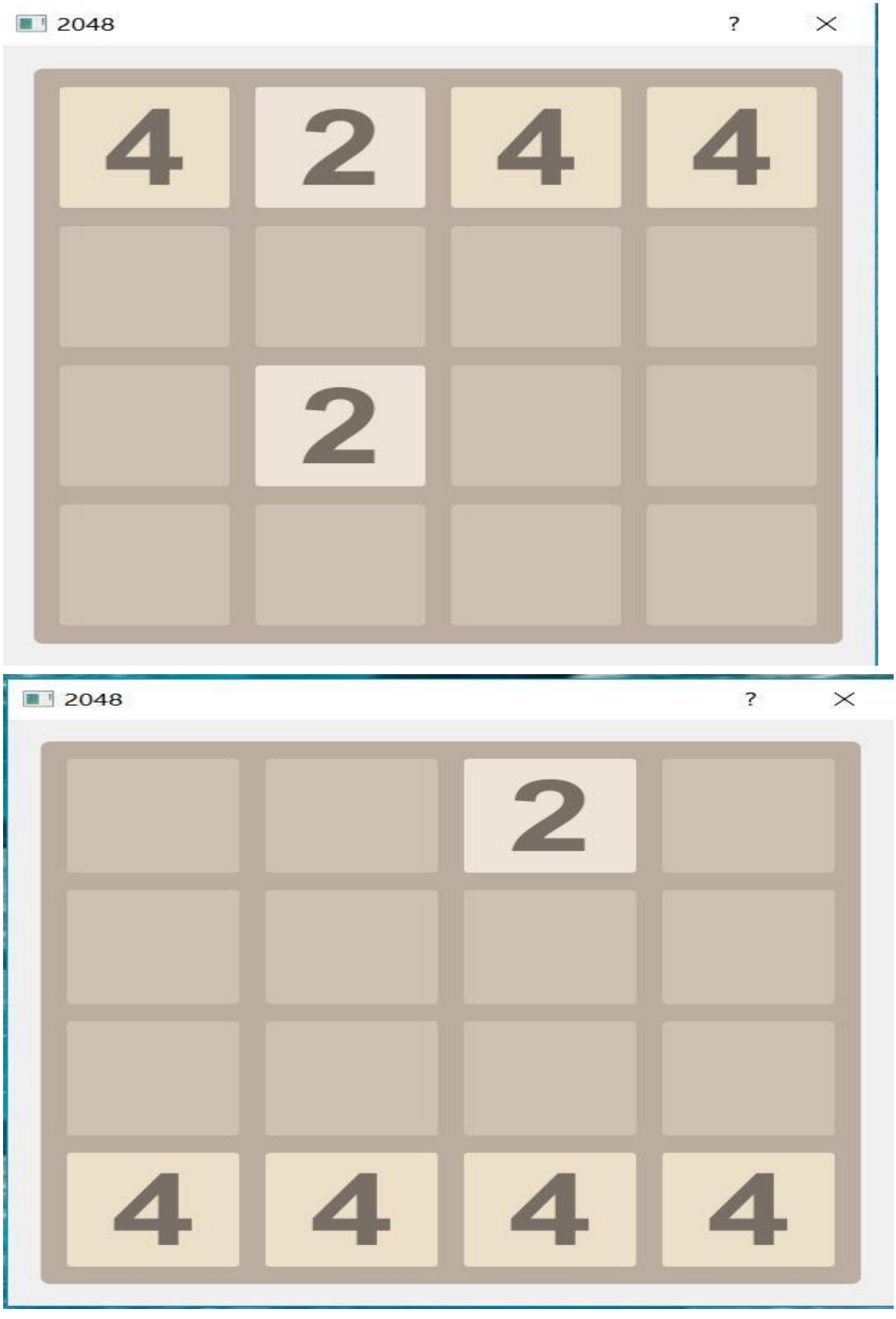

图 4-3 向下运行游戏界面

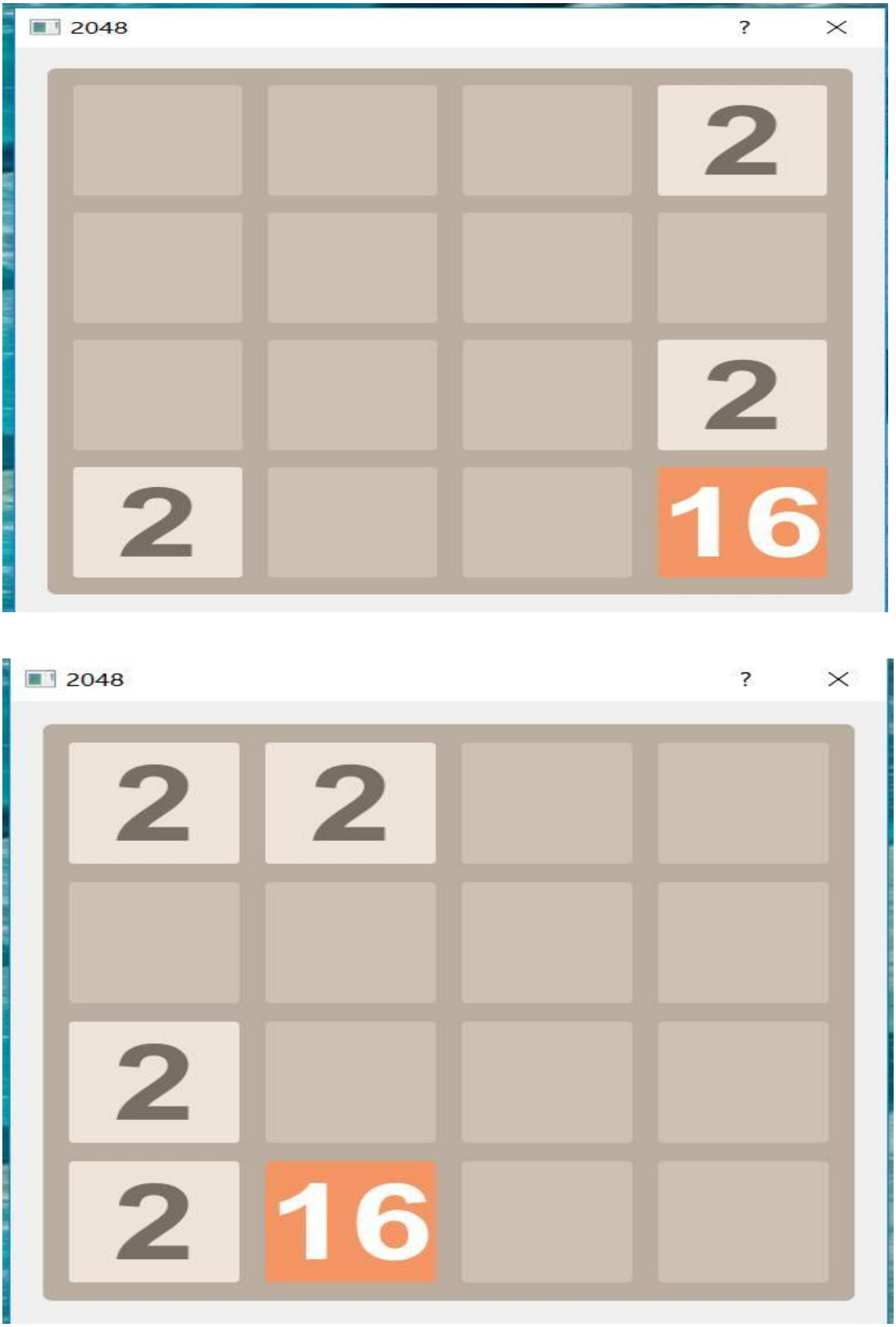

图 4-4 向左运行游戏界面

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

1、游戏总设计逻辑

1）.游戏开始时，游戏界面有两个数字，2 或者 4，2 出现的概率更高

2）.每次滑动（一次操作），数据先进行处理，再在空白处生成一个数字，2 或 4，2 出现的概率更高

3）.数据处理的逻辑

a)数据沿着滑动方向运动

b)遇见 0 便移位，并判断下一位是零，或者与下一位是否相等

c)遇见相同数字便相加

d)不满足 b，c 条件，不移动

4）.数据保存

游戏被抽象成为一行四个数据，[0,0,0,0]

使用可变有序的列表储存数据,16 个数字就是 4 组列表

[[0,0,0,0], [0,0,0,0],[0,0,0,0] ,[0,0,0,0]]

程序由以下四个模块编写

main_dialog	（主函数，包括运行逻辑，调用函数和棋盘初始化）
number_rect.py	（游戏可变化方格动画设置）
game_calculate.py	（游戏的计算类）
pygame_canvas.py	（游戏主面板）

我主要负责前两个模块，在以下部分进行着重解释说明。

2、部分数据流图

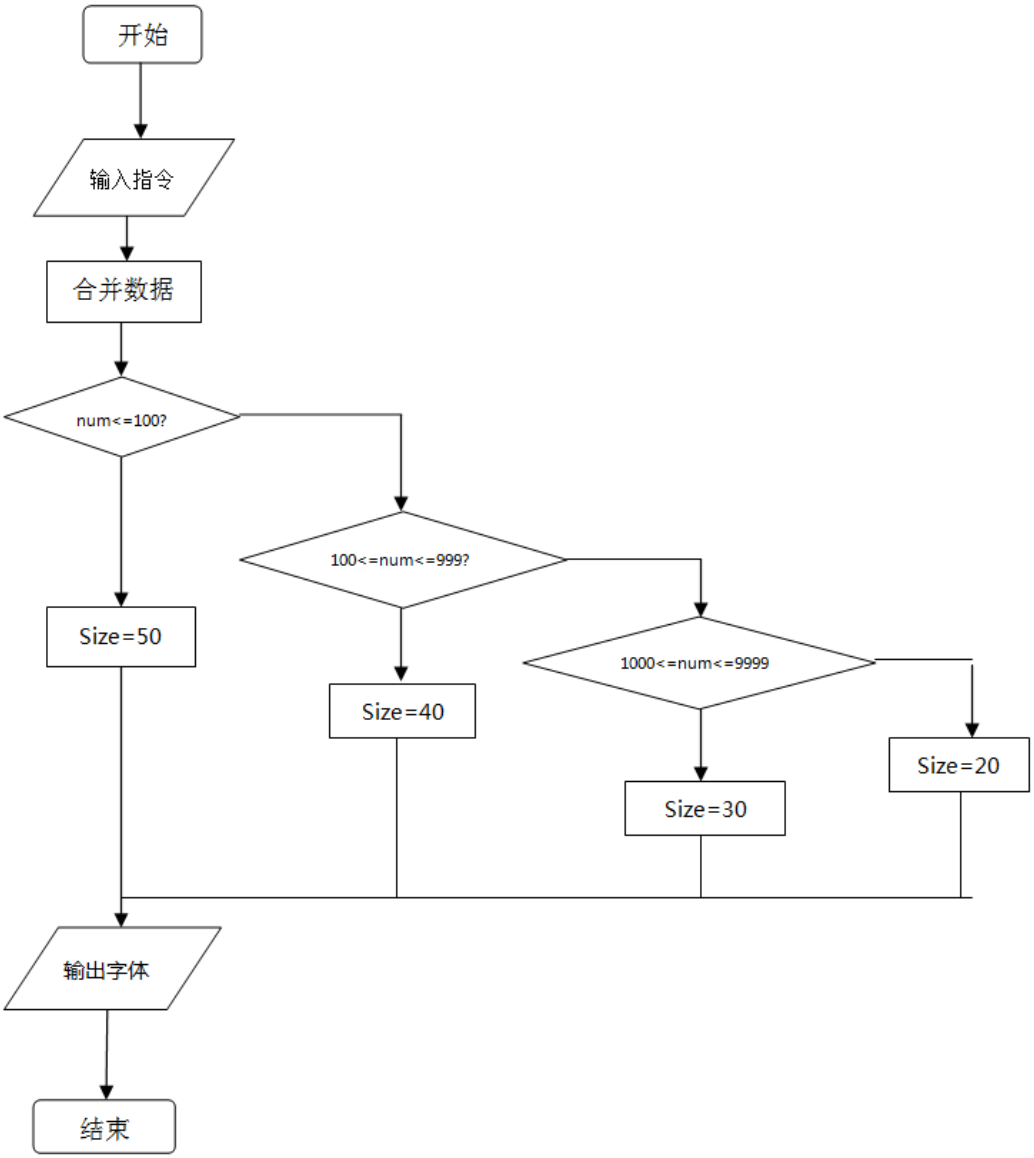

图 5-1

3、算法详细说明

输入输出

在主函数中定义一个二维列表作为初始界面方格，进行窗口初始化。

```
d = [
    [{'Item': None, 'Number': 0}, {'Item': None, 'Number': 0}, {'Item': None, 'Number': 0}, {'Item': None, 'Number': 0}],
    [{'Item': None, 'Number': 0}, {'Item': "A", 'Number': 16}, {'Item': None, 'Number': 0}, {'Item': None, 'Number': 0}],
    [{'Item': "A", 'Number': 8}, {'Item': "A", 'Number': 8}, {'Item': None, 'Number': 0}, {'Item': None, 'Number': 0}],
    [{'Item': "A", 'Number': 2}, {'Item': "A", 'Number': 2}, {'Item': "A", 'Number': 4}, {'Item': "A", 'Number': 2}]
]                    # 定义一个二维列表，窗口初始化
```

图 5-2

通过调用游戏计算类 game_calculate，对 d 进行计算，每当程序捕捉输入指令并进行计算后，输出经计算后的 d.

```
c = GameCalculate(d)
c.calculate(4)
d = c.data
for d in d:
    print(d)
```

图 5-3

创建棋盘

定义 gameCanvas 函数，设置 16 个只读小方格，定义 NumberRect 函数，实例化初始化棋盘的参数，指定棋盘的高和宽度以及游戏背景颜色

```
class NumberRect(QLabel): #QLable类主要用来文本和图像的显示没有提供交互功能 实例化qlabel
    animation = None
#动画默认值为空
    color_dict = {"2": "#eee4da", "4": "#ede0c8", "8": "#f2b179", "16": "#f59563", "32": "#f67c5f",
                  "64": "#f65e3b", "128": "#edcf72", "256": "#edcc61", "512": "#EFCB52",
                  "1024": "#EFC739", "2048": "#EFC329", "4096": "#FF3C39"}
#通过color_dict字典设置每个数字背景色
    font_color_dict = {"2": "#776e65", "4": "#776e65", "8": "#ffffff", "16": "#ffffff", "32": "#ffffff",
                       "64": "#ffffff", "128": "#ffffff", "256": "#776e65", "512": "#776e65",
                       "1024": "#776e65", "2048": "#776e65", "4096": "#ffffff"}
#通过color_dict字典设置每个数字字体颜色
    def __init__(self, parent, width, ds):     #__init__是pytnon的构造方法，用其初始化新创建对象之后，可以
        super(NumberRect, self).__init__(parent)  #super类似于嵌套的一种设计，子类在父类前，所有类不重复调
        self.ds = ds
        self.w = width  #窗口宽度
        self.resize(width, width)  #调整窗口大小
        self.setFont(QFont("\"Clear Sans\", \"Helvetica Neue\", Arial, sans-serif", 55, QFont.Bold))  #
        self.setAlignment(Qt.AlignCenter)  #设置对其方式（居中对其）
        self.refresh_ds(ds) #恢复函数
```

```
class GameCanvas(QLabel):
    # 只读的背景16个方格
    item_bgs = []
    # 背景16个方格的xy位置也是只读的
    item_bg_pos = []
    item_data = []

    parent_x = 20
    parent_y = 20
    split_width = 16
    rect_width = 105
```

图 5-4

声明_init_rect 函数，利用 random 库随机生成

```
def set_init_rect(self):
    self.random_rect_item()      #指定生成16个小方格
    self.random_rect_item()       #random随机生成库
```

图 5-5

创建窗口

设置弹出可视化窗口大小以及题目

```
class MainDialog(QDialog):
    def __init__(self, parent=None):          # __init__是pytnon的
        super(MainDialog, self).__init__(parent)      #自动查找
        self.setWindowTitle("2048")          #设置窗口题目为2048
        self.resize(540, 540)               #窗口大小赋值
        self.set_main_canvas()             #设置油布
```

图 5-6

调用函数

调用定义好的三个类

```
#调用这三个模块中的三大类
from game_calculate import GameCalculate
from game_canvas import GameCanvas
from number_rect import NumberRect
```

图 5-7

设置变化字体

定义 refresh 函数，在游戏进行中数值不断增加，字体颜色和字体大小随数字不同而进行变化。

```
def refresh_ds(self, ds):
    self.ds = ds
    self.setText(str(ds["num"]))    #设置文本格式
    self.move(ds["x"], ds["y"])    #移动位置
    color = self.color_dict[str(ds["num"])]          #背景颜色
    font_color = self.font_color_dict[str(ds["num"])]          #字体颜色  把定义的字体颜色列表插入字典
    self.setStyleSheet("QLabel{background-color:" + color + ";color:" + font_color + ";border-radius:3}") #设置
    font_size = 55                                    #字体大小55号
    if 100 <= ds["num"] <= 999: font_size = 40                  #根据数字位数适当调整字体大小
    if 1000 <= ds["num"] <= 9999: font_size = 30
    if 10000 <= ds["num"] <= 99999: font_size = 20
    self.setFont(QFont("\"Clear Sans\", \"Helvetica Neue\", Arial, sans-serif", font_size, QFont.Bold))
```

图 5-8

设置动画

定义 animation 函数，设置移动动画，包括出现速度、出现位置，以线性移动方式，并在动画结束后清空原来位置的数字。（同后合并提交动画）

```
def move_animation(self):    #移动动画          #位置
    self.animation = QPropertyAnimation(self, "pos".encode())      #动画Qt属性的类 给予我们极大的自由
    self.animation.setDuration(150)      #窗口出现速度0.15秒
    self.animation.setStartValue(QPoint(0, 0))
    self.animation.setEndValue(QPoint(300, 300))      #窗口出现位置
    self.animation.setEasingCurve(QEasingCurve.Linear)    #让动画按照线性移动
    self.animation.start(QAbstractAnimation.DeleteWhenStopped)    #动画结束后进行清理原本位置数字
```

图 5-9

六、调试与测试

在后期调试阶段，我使用的是 anaconda 的 Spyder 编译器。作为开源社区贡献者的由 Python 编写的跨平台 IDE，Spyder 以轻便、便捷、高度集成为卖点。Spyder 允许在多种不同的预设模式下工作，例如类似 Matlab 式的科学计算交互界面，以及其他应用工程开发形式的界面环境;在编码过程中 Spyder 可实时提示文档、交互式运行、调试时显示全部变量表，并可一键可视化等，对于数据分析而言具有很好的便利性;同样，它也支持步进跟踪等一系列 PDB 所提供的调试功能。若是说缺点，界面本身不具时尚感。

在最开始运行时候，运行结果只出现窗口，而对窗口设置的格式，背景颜色，大小都不显示。

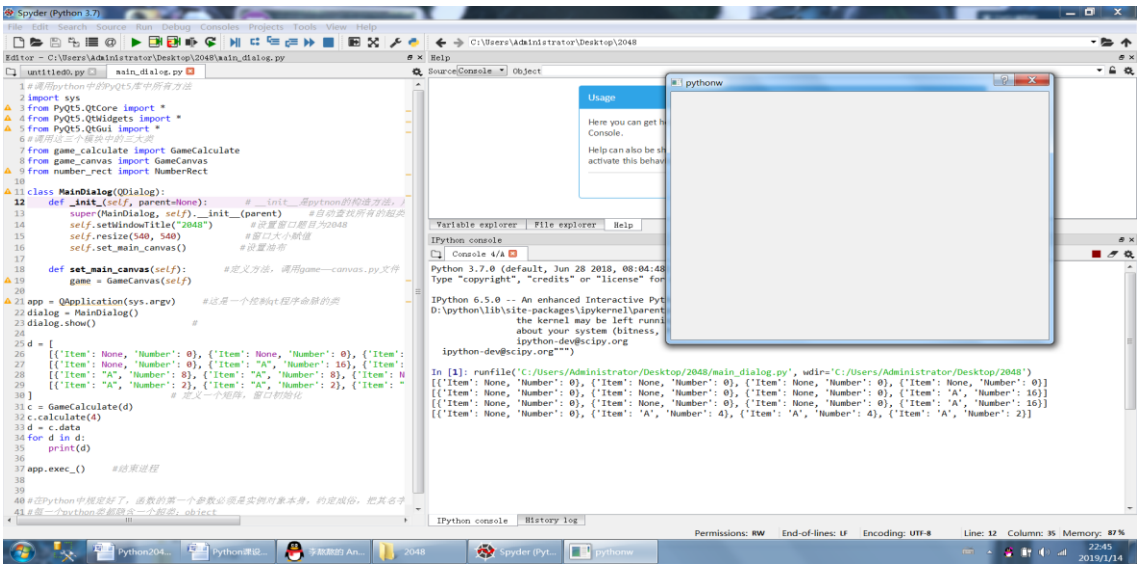

图 6-1

我检查过后并不知道哪里出了问题，于是进行了以下调试。

添加断点

用 Spyder 打断点非常简单，只要在想加断点的那一行行首双击鼠标或者选中语句点击 F12 按钮即可，如图所示，我在第 9、16、19 行建立了断点：

```
 7 from game_calculate import GameCalculate
 8 from game_canvas import GameCanvas
 9 from number_rect import NumberRect
10
11 class MainDialog(QDialog):
12     def __init__(self, parent=None):        # __init__是pytnon的构造方法，用其初始化新创建对象之后，
13         super(MainDialog, self).__init__(parent)      #自动查找所有的超类（父类），继承父类
14         self.setWindowTitle("2048")         #设置窗口题目为2048
15         self.resize(540, 540)               #窗口大小赋值
16         self.set_main_canvas()             #设置油布
17
18     def set_main_canvas(self):         #定义方法，调用game—canvas.py文件
19         game = GameCanvas(self)
```

图 6-2

进入调试模式

在 debug 之前，现在 Spyder 的 ipthon 界面中输入%rest 把工作空间的所有变量清除，以免影响我们接下来的测试。点击 Spyder 工具栏上的 Debug file 按钮，或者使用快捷键 Ctrl+F5 进入调试，程序会在添加的第一个断点之后停止。

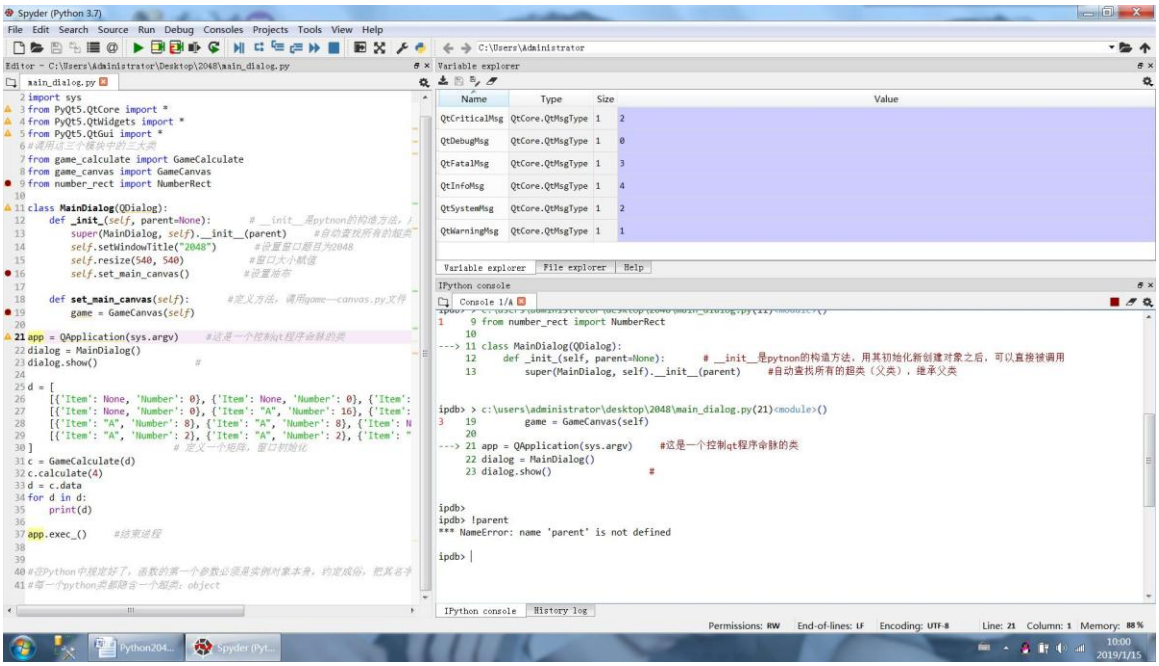

图 6-3

然后在 ipython 界面中会输出如图所示的内容，出现 ipdb 提示符，说明我们已经进入了调试模式。

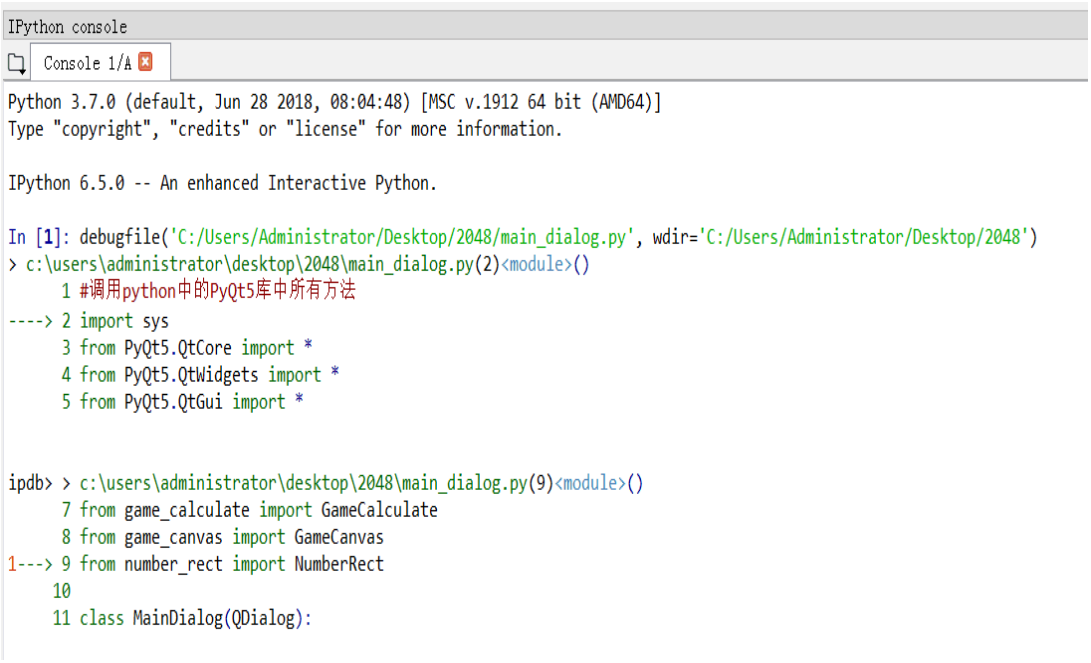

图 6-4

单行调试

直接点击 Ctrl+F10 尽可以在设置的断点之后进行单步调试

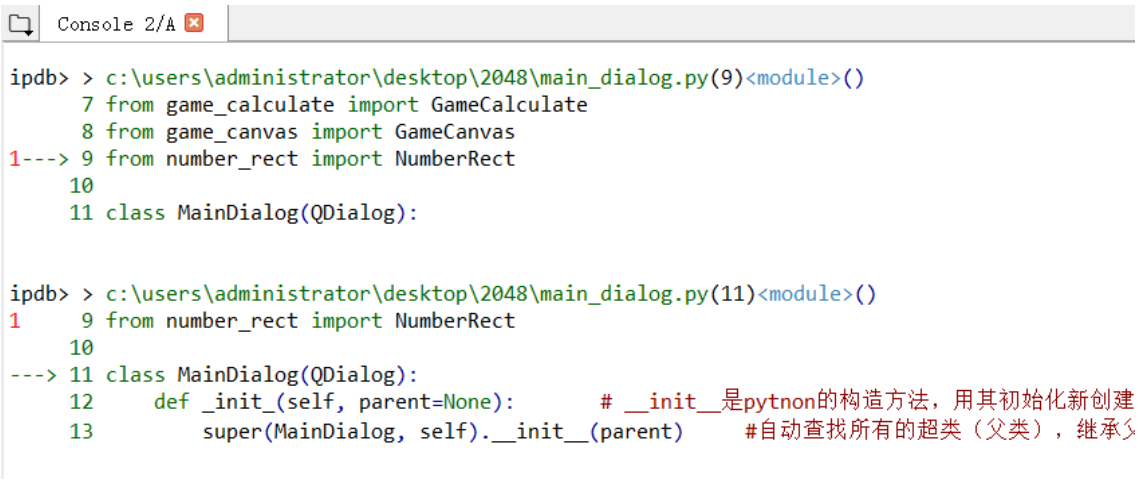

图 6-5

在 variable explorer 中提供了调试面板。

Variable explorer

Name	Type	Size	Value
QtCriticalMsg	QtCore.QtMsgType	1	2
QtDebugMsg	QtCore.QtMsgType	1	0
QtFatalMsg	QtCore.QtMsgType	1	3
QtInfoMsg	QtCore.QtMsgType	1	4
QtSystemMsg	QtCore.QtMsgType	1	2
QtWarningMsg	QtCore.QtMsgType	1	1

图 6-6

在 ipython 界面输入！parent 可以查看具体定义行执行情况，ipthon 界面显示”name ‘parent’ is not defined”,但 12 行确实定义也执行了。

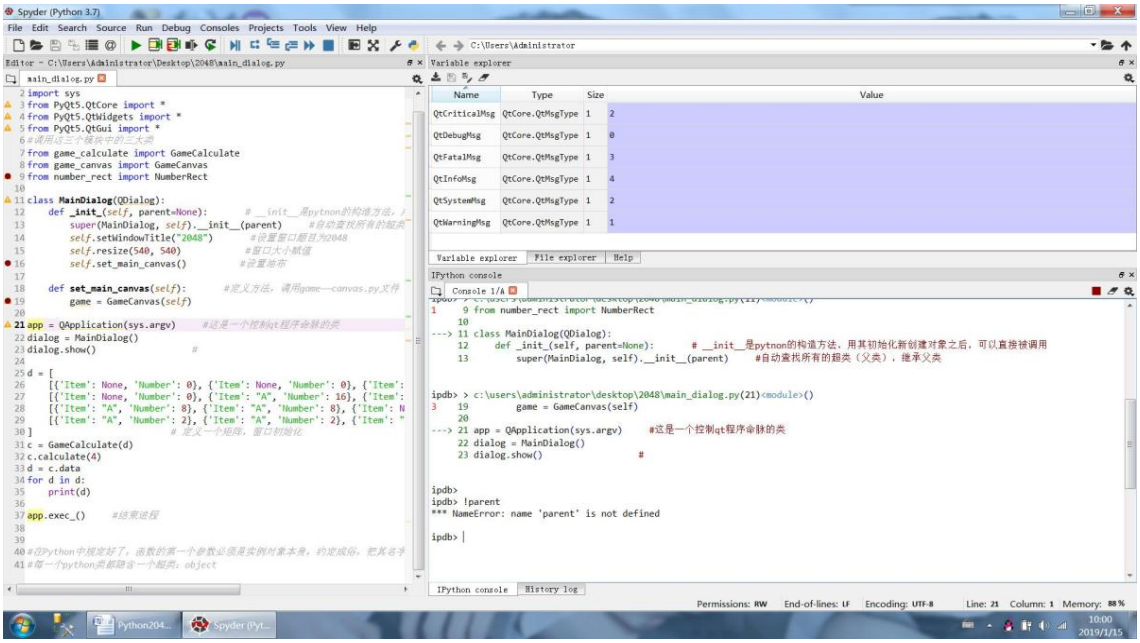

图 6-7

在认真检查检查代码之后，我发现__init__方法写错，左右下划线是两个，而我由于粗心写成了一个，导致没有出现想要的结果，这也说明了在敲代码时候一定要细心。

六、设计总结

经过一周左右的 python 课程设计，我和队员终于把这项艰巨的任务完成了。通过这次课设，加深学习了 python 语言的类型与操作符、基本语句、函数、模块、类以及异常，提高了我们的动手实践能力。从最开始的选题，开题到分析设计，绘图到完成代码，再到后来调试测试，期间我们查找资料，与同学讨论，反复修改，每一个过程都是对自己能力的检验和充实。

但是课程设计也暴露出自己基础知识的很多不足之处。比如缺乏综合应用知识的能力，对所使用库不够了解，对设计涉及到的规范要求不熟悉等等，需要在做的过程中不断翻阅相关资料和书籍，这降低了自己编程的速度和小组设计的进度，但这个但这个过程对我来说是对自己知识不足的一个很好的补充，也是对学过知识的一个巩固。

经过和队友商量，我们选定了一款在手机端很火的小游戏—2048，在游戏设计之初，2048 是不能再 PC 端玩的，我和队友就想尝试自己动手实现其基本功能。这个游戏操作简单，界面简洁，非常锻炼玩家的思维能力，并且我们也非常喜欢玩。

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，或者对函数定义不太熟悉，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。我们使用的 pyqt5 库也较大，常常在机房电脑下载不了，队友电脑也不太支持，编程过程中遇到了不少问题，走了不少弯路。

经过努力，我们基本实现最初设想的功能，整体来说，界面看起来舒服，玩起来也很顺手，只要操控键盘即可。但也不可避免的留下了一些没有实现的部分。最开始的设想里积分部分，再后来编写过程中未能成功实现，降低了游戏体验感。另外在手机端游戏中，只有棋盘不能再移动时，游戏结束，而我们的游戏只要当数字布满棋盘，游戏就结束，并且游戏结束后不能自动重新载入开始游戏。我和队友在几经调试后仍未解决，在时间范围内我们无法再过多的进行修改学习，尚留遗憾，我们会继续好好学习，保持对 python 的热情