广州大学纺织服装学院

《Python 课程设计》报 告

姓 名：曹传宽　　指导教师：王锦

班 级：21 计应 1 班　　学　　号：2131131103

地 点：1-215

时 间：2023 年 6 月 19 日至 2023 年 6 月 23 日

广州大学纺织服装学院教务处

基于 Python 开发 2048 小游戏

1.目的

本课程设计的目的是检验本学期所学 python 的知识点以及知识点的迁移能力，还有锻炼编程思想和编程算法。通过此课程设计，我将对游戏的概况、具体内容以及其重要意义进行说明。此外，我还将分析问题、思考解决过程，并总结自我评价、收获和努力方向。

2.概况

2048 是一款数字益智游戏，玩家通过移动方块，使相同数字的方块合并，最终得到数字 2048 的方块。我这个小游戏是一个 4x4 版的 2048，玩家可以使用键盘上的箭头键来控制方块的移动方向。

3.具体内容

该代码包含以下主要部分：

（1）窗口和画笔的初始化：设置游戏窗口的大小，并创建一个 Turtle 对象作为画笔。

```
# 设置窗口大小
window_size = 600
# 设置面板行数列数
board_size = 4
# 设置面板大小
cell_size = window_size // board_size

# 初始化窗口和画笔
window = turtle.Screen()
window.title("2048(4x4 版)小游戏\tby:曹传宽")
window.setup(window_size, window_size)
window.tracer(0)

pen = turtle.Turtle()
pen.speed(0)
pen.penup()
pen.hideturtle()
```

（2）游戏板的创建和绘制：使用二维列表创建一个 4x4 的游戏板，并通过绘制方块来展示当前游戏板的状态。

```
# 创建游戏板
board = [[0] * board_size for _ in range(board_size)]
```

（3）随机数字的添加以及方块的创建和游戏界面创建：在空白方块中随机选择一个位置，并在该位置上添加数字 2 或 4。

```
# 添加随机数字
def add_random_number():
    empty_cells = [(i, j) for i in range(board_size) for j in
range(board_size) if board[i][j] == 0]  # 列表推导式
    if empty_cells:
        row, col = random.choice(empty_cells)
        board[row][col] = random.choice([2, 4])


# 绘制方块
def draw_square(row, col, value):
    # 游戏框坐标
    x = (col - board_size / 2 - 0.5) * cell_size + cell_size / 2
    y = (board_size / 2 - row + 0.5) * cell_size - cell_size / 2

    pen.goto(x, y)
    pen.pendown()
    pen.fillcolor(get_color(value))
    pen.begin_fill()

    # 笔记:
    # 在这个特定的示例中，for _ in range(4) 循环的意义可能不太明显，因为在循环体中并没有使用到变量 _。这种情况下，可以使用任何变量名替代 _，例如 for i in range(4)。
    # 然而，有时候在编程中，我们可能只关心循环的次数，而不需要使用循环变量的值。在这种情况下，使用_作为循环变量可以传达一个清晰的信号，告诉其他人或自己，这个循环变量的值并不重要。
    # 此外，使用_作为循环变量还可以避免产生未使用的变量警告，这在某些编程语言或开发环境中是一个常见的问题。
    # 总的来说，使用_作为循环变量在这个示例中可能没有特别的意义，但在其他情况下，它可以提供一种简洁和清晰的方式来表示循环次数不重要或不需要使用循环变量的情况。
    for _ in range(4):
        pen.forward(cell_size - 2)
        pen.right(90)
```

```
    pen.end_fill()
    pen.penup()

    if value != 0:
        # 数字坐标
        pen.goto(x + 70, y - 120)
        pen.write(str(value), align="center", font=("Arial", 50,
"normal"))


# 获取方块颜色
def get_color(value):
    colors = {
        0: "#CDC1B4",
        2: "#EEE4DA",
        4: "#EDE0C8",
        8: "#F2B179",
        16: "#F59563",
        32: "#F67C5F",
        64: "#F65E3B",
        128: "#EDCF72",
        256: "#EDCC61",
        512: "#EDC850",
        1024: "#EDC53F",
        2048: "#EDC22E"
    }
    return colors.get(value, "#CDC1B4")


# 绘制游戏界面
def draw_board():
    pen.clear()
    for row in range(board_size):
        for col in range(board_size):
            value = board[row][col]
            draw_square(row, col, value)
    window.update()
```

（4）方块的移动和合并：根据玩家的输入移动方块，并在移动过程中合并相同数字的方块。

```
# 移动游戏板
def move(dir):
    if dir == "up":
        for col in range(board_size):
```

```
            merge_column_up(col)
    elif dir == "down":
        for col in range(board_size):
            merge_column_down(col)
    elif dir == "left":
        for row in range(board_size):
            merge_row_left(row)
    elif dir == "right":
        for row in range(board_size):
            merge_row_right(row)
    add_random_number()
    draw_board()


# 合并列（向上移动）
def merge_column_up(col):
    for row in range(1, board_size):
        if board[row][col] != 0:
            for k in range(row, 0, -1):
                if board[k - 1][col] == 0:
                    board[k - 1][col] = board[k][col]
                    board[k][col] = 0
                elif board[k - 1][col] == board[k][col]:
                    board[k - 1][col] *= 2
                    board[k][col] = 0
                    break


# 合并列（向下移动）
def merge_column_down(col):
    for row in range(board_size - 2, -1, -1):
        if board[row][col] != 0:
            for k in range(row, board_size - 1):
                if board[k + 1][col] == 0:
                    board[k + 1][col] = board[k][col]
                    board[k][col] = 0
                elif board[k + 1][col] == board[k][col]:
                    board[k + 1][col] *= 2
                    board[k][col] = 0
                    break


# 合并行（向左移动）
def merge_row_left(row):
```

```
    for col in range(1, board_size):
        if board[row][col] != 0:
            for k in range(col, 0, -1):
                if board[row][k - 1] == 0:
                    board[row][k - 1] = board[row][k]
                    board[row][k] = 0
                elif board[row][k - 1] == board[row][k]:
                    board[row][k - 1] *= 2
                    board[row][k] = 0
                    break


# 合并行（向右移动）
def merge_row_right(row):
    for col in range(board_size - 2, -1, -1):
        if board[row][col] != 0:
            for k in range(col, board_size - 1):
                if board[row][k + 1] == 0:
                    board[row][k + 1] = board[row][k]
                    board[row][k] = 0
                elif board[row][k + 1] == board[row][k]:
                    board[row][k + 1] *= 2
                    board[row][k] = 0
                    break
```

（5）注册并处理按键事件，以及开始游戏

```
# 处理按键事件
def handle_key(key):
    if key == "Up":
        move("up")
    elif key == "Down":
        move("down")
    elif key == "Left":
        move("left")
    elif key == "Right":
        move("right")

    if is_game_over():
        # 用于关闭界面
        # window.bye()
        pass


# 初始化游戏
def init_game():
```

```
    add_random_number()
    add_random_number()
    draw_board()


# 注册按键事件
window.listen()  # 事件监听
window.onkey(lambda: handle_key("Up"), "Up")
window.onkey(lambda: handle_key("Down"), "Down")
window.onkey(lambda: handle_key("Left"), "Left")
window.onkey(lambda: handle_key("Right"), "Right")

# 启动游戏
init_game()
turtle.done()
```

（6）游戏结束的检测：检查游戏是否结束，如果无法再进行移动，则游戏结束。

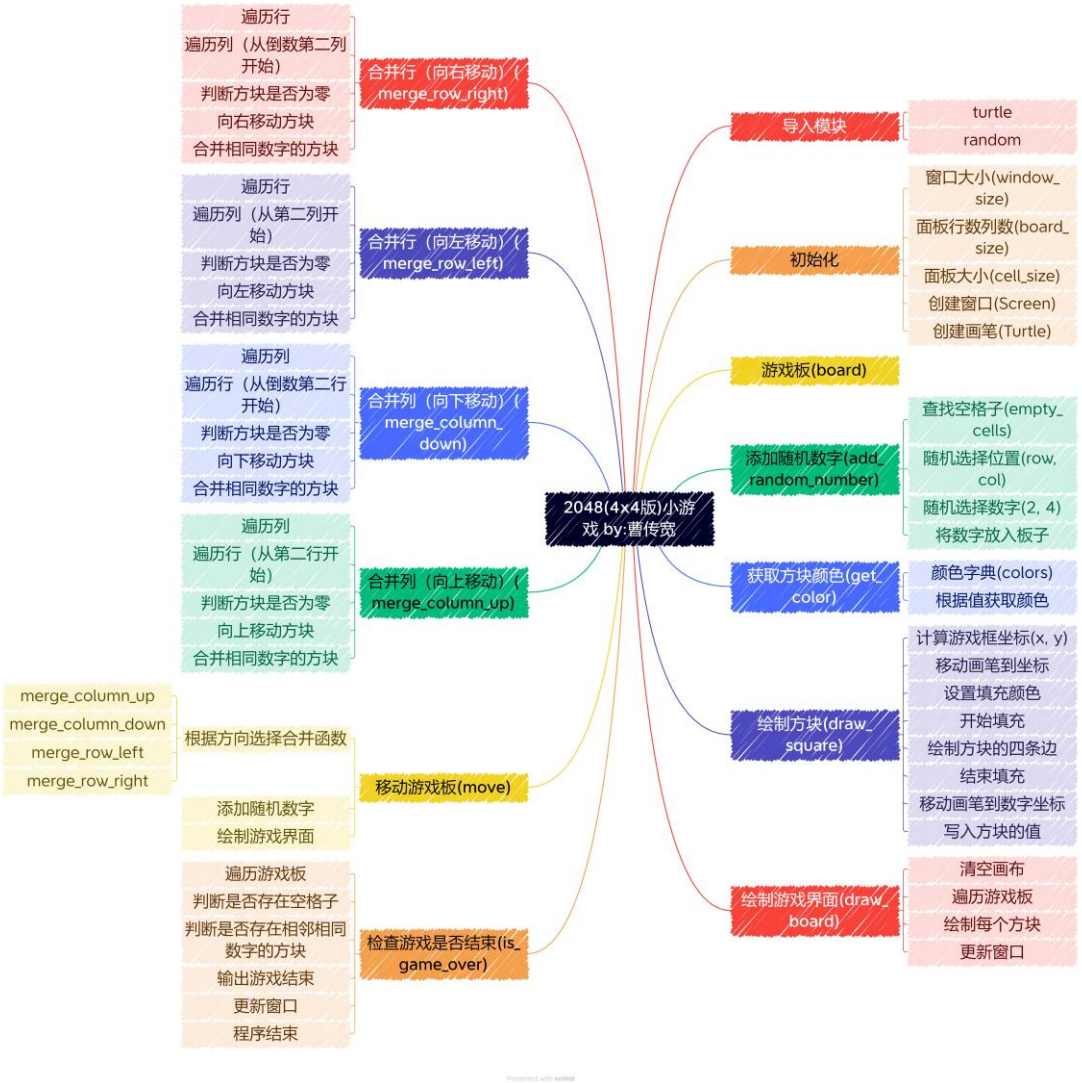
```
# 检查游戏是否结束
def is_game_over():
    for row in range(4):
        for col in range(4):
            if board[row][col] == 0:
                return False

    for row in range(3):
        for col in range(3):
            if board[row][col] == board[row + 1][col] or board[row][col]
== board[row][col + 1]:
                return False

    pen.goto(0, 0)
    pen.write("游戏结束", align="center", font=("黑体", 50, "normal"))
    window.update()

    return True
```

整个游戏设计逻辑的思维导图：



4. 重要意义

（1）娱乐性：游戏提供了一种轻松有趣的方式来消磨时间，并给玩家带来乐趣和挑战。

（2）逻辑思维：游戏需要玩家根据当前的方块状态做出决策，培养了玩家的逻辑思维能力和策略规划能力。

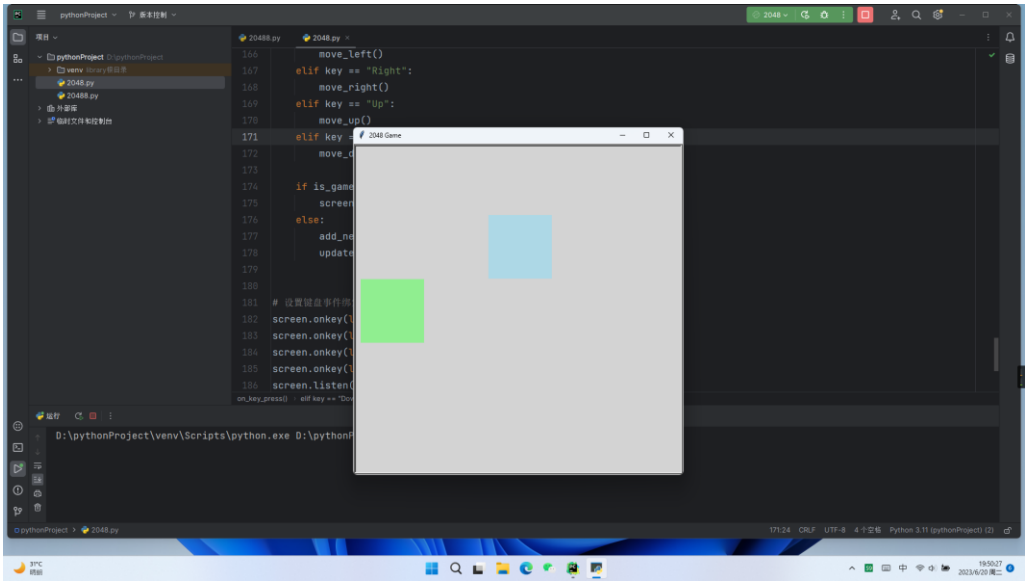
（3）数字理解：游戏中的数字合并过程涉及数字计算和理解，有助于提高玩家的数字意识和计算能力。

（4）编程实践：代码实现了游戏逻辑和图形界面的交互，对我来说是一个很好的编程实践项目，可以加深对Python编程语言和海龟图形库turtle的理解。

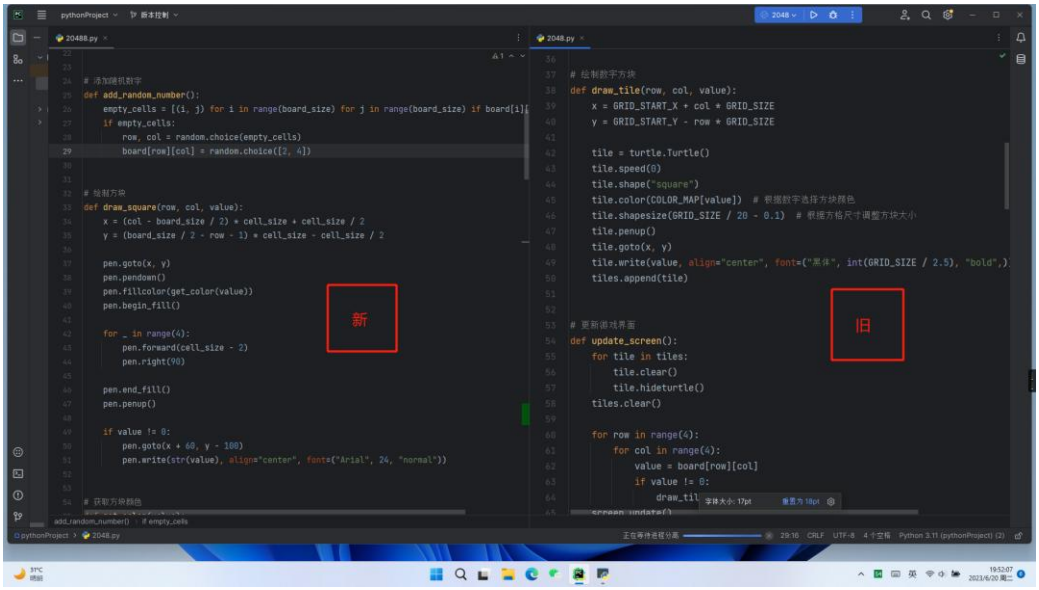
5.发现的问题的分析、思考和解决过程

（1）方格不显示文字

通过分析代码，是由于使用了 turtle 库绘制方格以及数字，但由于两者颜色一致，所以“不显示”数字。经过思考以及参考各种资料，将绘制方格以及数字改成如下图代码，成功解决。

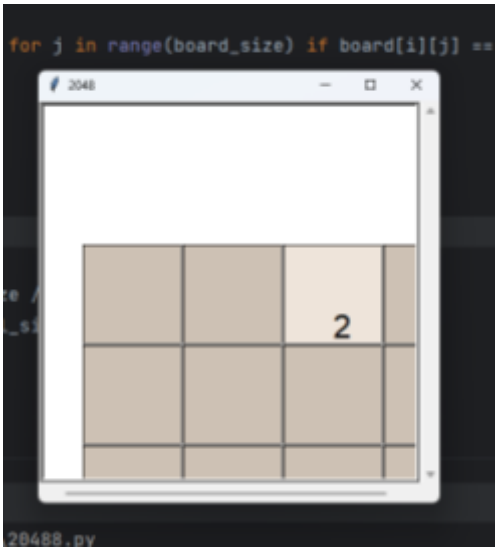


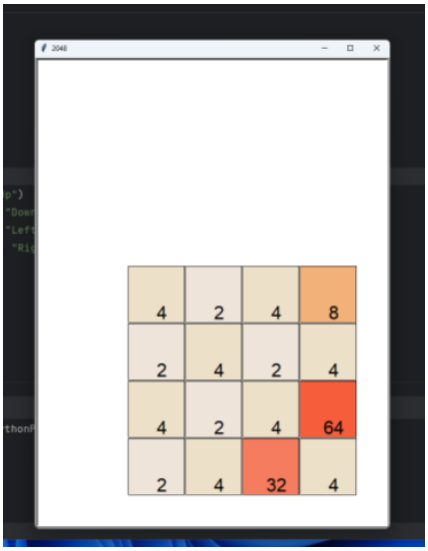
问题图



解决图

（2）窗口显示不全，数字偏移

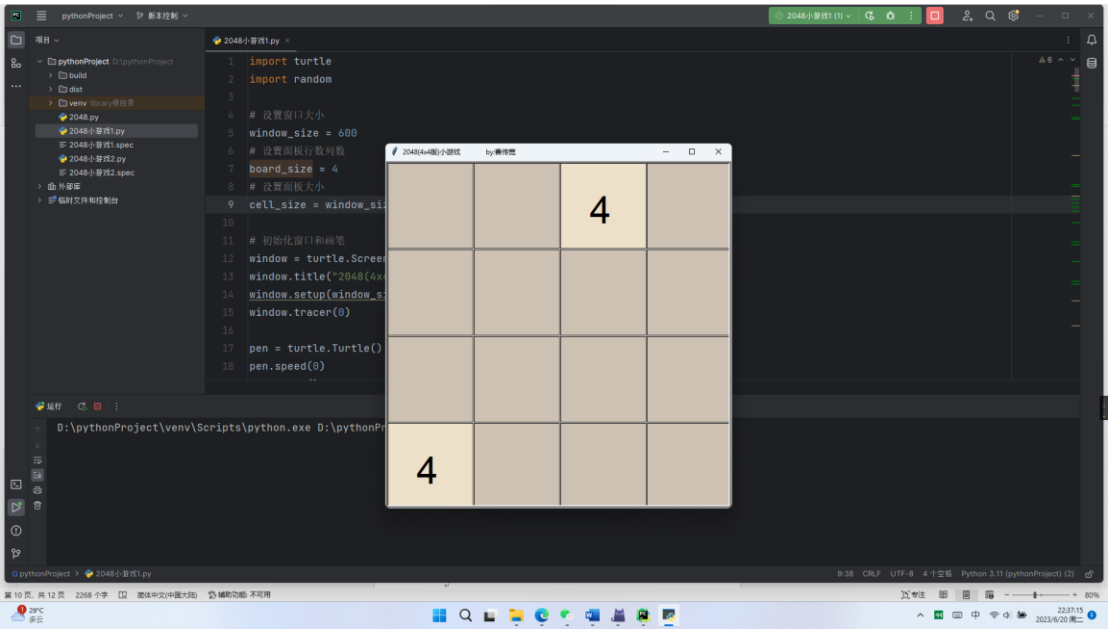




通过分析代码，参考 turtle 库文档，修改相关位置坐标以及长度宽度即可解决。

经过上述问题的解决，基本完成了 2048 小游戏界面显示问题的修复，经过测试验证，本游戏基本已完成。

最终可视化窗口界面如下图：



6.总结或体会

（1）自我评价：通过分析这个小游戏的代码，我对 Python 编程语言和图形库的应用有了更深入的了解，同时也锻炼了我的逻辑思维和问题解决能力。

（2）收获：从代码中学习了如何使用 Turtle 库来创建图形界面，以及如何处理按键事件和绘制方块等功能。同时，也加深了对二维列表和循环结构的理解。

（3）后续工作：可将游戏代码通过打包工具 pyinstaller 可生成 exe 可执行文件，可以分享给朋友玩。

（4）努力方向：在今后的学习中，我将继续探索更多有趣的编程项目，提高自己的编程能力和实践经验。同时，也希望能够加深对 Python 库和算法的理解和应用。

报告人：_____

日期：2023 年 6 月 23 日