西北工业大学

课程设计（大作业）报告

课程名称： 《Python 程序设计》

课程编号： U08M11077.01

设计题目： 目标检测实例的复现

组员名单： 敖冠舒 唐中磊 王骏松 王一帆

联系方式： 134 0324 7575

设计时间： 2022 年 12 月

目录

1. 设计概述

1.1 设计目的

本项目通过复现开源项目基于目标检测工地安全帽和禁入危险区域识别系统，实现工地安全帽和禁入危险区域识别系统的目标检测功能，并学习其代码实现的方法，掌握目标检测的基本原理和实现方法，提高自己的编程能力和算法能力。

1.2 设计内容

该项目是使用 YOLOv5 的程序来训练在智能工地安全领域中头盔目标检测的应用，包括数据集的准备、训练、测试、部署等步骤。

1.3 应用平台

表 1-1 硬件、软件环境一览表

	指标	版本参数
硬件环境	CPU	AMD R7-5800H
	RAM	16 GB
软件环境	操作系统	Windows 11 Pro 22H2
	Python	Python 3.8.15

1.4 开发工具

表 1-2 开发工具一览表

工具	版本	用途
PyCharm	2022.3	代码编写
Anaconda	2022.11.1	Python 环境管理

1.5 软件库

表 1-3 软件库一览表

库名	版本	用途
pygame	2.1.2	游戏界面的显示等
numpy	1.24.0	数组的处理

2. 详细设计

2.1 总体方案

本项目采用模块化和面向对象的方法，将程序分为多个类，每个类负责一个功能，类与类之间通过接口进行通信，类与类之间的通信方式采用函数调用的方式，类与类之间的数据传递采用参数传递的方式，类与类之间的数据共享采用全局变量的方式。

具体来说，本项目采用的类有：Main 主函数类、按钮类、Ai 类等，用于实现游戏界面的显示、开始、暂停、结束、重新开始、分数统计等功能。

本项目使用四个.py 文件实现上述功能，分别是 main.py、config.py、ai.py 和 game.py，其中 main.py 是主函数，用于调度各个模块以实现功能；config.py 是配置文件，主要负责游戏参数的设置；ai.py 是 AI 算法文件，用于实现游戏的 AI 模式，game.py 是游戏文件，用于绘制游戏界面，实现具体的游戏功能。

2.2 功能实现

2.2.1 游戏基础配置

在 config.py 文件中，实现了一些基础配置的设置。如：

- 游戏界面的大小
- 方块和背景的颜色
- 方块的阶数
- 游戏帧率（默认为 60）
- AI 模式的操作速度（默认为快）

2.2.2 主函数的实现

初始化一个游戏的主类，准备开始运行游戏。

这段代码是一个 Python 程序的主函数，它定义了一个名为 Main 的类。在这个类中，定义了一个名为 init 的特殊方法，这个方法会在创建 Main 类的实例时被调用。

在 init 方法中，首先调用了 pygame 库的初始化函数，然后设置了窗口的标题和大小，设置了游戏的帧率，创建了一个游戏的实例，创建了一个 AI 类的实例。

在这个方法中还有一些其他的变量，比如 self.state、self.catch_n 和 self.step_time 等，这些变量在程序的其他地方也会被使用。

```
class Main():
def __init__(self):
    global FPS
    pygame.init()
    os.environ['SDL_VIDEO_WINDOW_POS'] = "%d,%d" % (100, 50) # 设置窗口位置
    self.set_win_wh(WINDOW_W, WINDOW_H, title='2048') # 设置窗口大小和标题
    self.state = 'start'
    self.fps = FPS
    self.catch_n = 0
    self.clock = pygame.time.Clock() # 创建一个Clock对象
    self.game = Game(SIZE)       # 创建游戏对象
    self.ai = Ai()              # 创建AI对象
    self.step_time = config.STEP_TIME # 每步的时间间隔
    self.next_f = ''
    self.last_time = time.time() # 上一步的时间
    self.jm = -1 # 用于记录上一步的方向
```

2.2.3 棋盘和方块的绘制

为了实现游戏界面的绘制，定义了一个名为 draw_map 和一个 draw_block 的方法，用于绘制棋盘和方块。

在 draw_map 方法中，首先使用两层循环来遍历棋盘上的每一个格子，并调用 draw_block 函数来绘制每个格子；然后检查当前的游戏状态，如果游戏已经结束（即 state 变量为 over 或 win），则绘制一个半透明的黑色矩形，并调用一个名为 draw_text 的函数来在棋盘上绘制文本。文本内容根据当前的游戏状态而定，如果游戏已经结束则显示“Game Over!”，如果游戏胜利则显示“Victory!”。

```
for y in range(SIZE):
    for x in range(SIZE):
        self.draw_block((x, y), self.game.grid.tiles[y][x])
```

```
if self.state == 'over':
    pygame.draw.rect(self.screen, (0, 0, 0, 0.5),
                     (0, 0, GAME_WH, GAME_WH))
    self.draw_text('Game Over!', (GAME_WH / 2, GAME_WH / 2), size=25,
        center='center')

elif self.state == 'win':
    pygame.draw.rect(self.screen, (0, 0, 0, 0.5),
                     (0, 0, GAME_WH, GAME_WH))
    self.draw_text('Victory!', (GAME_WH / 2, GAME_WH / 2), size=25,
        center='center')
```

在 draw_block 方法中，使用 xy 表示一个元组，表示方块的位置，number 是一个整数，表示方块上的数字。draw_block 函数计算出每个方块的大小，并使用 Pygame 库绘制一个矩形。矩形的颜色由 number 参数决定，如果 number 小于等于 2048 则使用 COLOR 字典中的值，否则使用蓝色. 如果方块上的数字不为 0，则调用 draw_text 函数在方块中间绘制数字。

```
def draw_block(self, xy, number):
one_size = GAME_WH / SIZE
dx = one_size * 0.05
x, y = xy[0] * one_size, xy[1] * one_size
color = COLOR[str(int(number))] if number <= 2048 else (0, 0, 255)
pygame.draw.rect(self.screen, color,
                 (x + dx, y + dx, one_size - 2 * dx, one_size - 2 * dx))
color = (20, 20, 20) if number <= 4 else (250, 250, 250)
if number != 0:
    ln = len(str(number))
    if ln == 1:
        size = one_size * 1.2 / 2
    elif ln <= 3:
        size = one_size * 1.2 / ln
    else:
        size = one_size * 1.5 / ln
```

```
self.draw_text(str(int(number)), (x + one_size * 0.5, y + one_size * 0.5 -
    size / 2), color, size, 'center')
```

2.2.4 方块移动的控制

为了实现对方块移动的控制，定义了一个名为 get_grid 的函数。get_grid() 函数接受两个参数：tiles 和 directions。tiles 参数表示当前棋盘的状态，是一个二维数组，存储了每个格子的值。directions 参数表示要进行的移动方向序列，是一个字符串的列表，其中每个字符表示一个方向，U 表示向上移动，D 表示向下移动，L 表示向左移动，R 表示向右移动。函数的作用是模拟移动棋盘，首先它会创建一个 Grid 类的实例，然后把当前的棋盘状态复制给这个实例的 tiles 属性，然后按照 directions 参数中的顺序对棋盘进行移动，每次移动后调用 add_random_tile() 方法在棋盘上随机添加一个新的格子。最后返回棋盘的状态。例如，调用 get_grid([[2, 4, 8, 16], [32, 64, 128, 256], [512, 1024, 2048, 4096], [8192, 16384, 32768, 65536]], [”U”, ”L”, ”D”, ”R”]) 将会模拟向上、左、下、右四个方向依次移动，并在每次移动后随机添加一个新的格子，最终返回移动后的棋盘状态。

```
def get_grid(tiles, directions):
g = Grid(config.SIZE)
g.tiles = tiles.copy()
for direction in directions:
    g.run(direction)
    g.add_random_tile()
return g.tiles
```

2.2.5 AI 功能的实现

在 ai.py 这个文件中，定义了一个名为 Ai 的类，用以解决游戏自动操作的问题。首先，Ai 类包含了一个名为 get_next() 的方法，它用于获取下一步的最优移动方向。它接受一个参数 tiles，表示当前棋盘的状态。首先调用 get_num() 方法计算当前棋盘上有多少个格子是已经有数字的，如果超过棋盘的一半就随机返回 U 或 D 两个方向之一。然后 gen_next 首先会计算出一个数值 kn，它是当前棋盘的已有数字数量的平方与 20 之间的最小值，但是不能超过 40。

然后使用 itertools 库的 product() 方法生成所有可能的移动序列，对于每个序列都调用 get_grid() 方法和 get_score() 方法来计算分数，并将结果存储在一个列表中。每一种操作由两个方向（U、L、R、D）组成，分别表示上、左、右、下四个方向。然后，使用 get_grid() 函数模拟执行这两步操作，并计算每一种操作的得分。最后，将所有得分按照从小到大的顺序排序，并返回得分最高的操作。

在这段代码中，计算得分的方式是使用 get_score() 函数，该函数又使用了另外两个函数：get_bj2__4() 和 get_bj__4()。前者用于计算棋盘上有多少对相邻的数字相等，后者用于计算棋盘上有多少个数字为 4。最后，将这两个数值相加并乘以一个常数，得到最终的得分。

另外，这段代码中还有一个名为 debug() 的方法，它用于模拟执行单步操作，并输出每一步操作的结果。这可以帮助我们理解自动玩 2048 游戏的原理。

还有，这段代码中还有一些其他的方法，例如：

- get_num() 函数用于计算当前棋盘上有多少个格子是已经有数字的。
- H() 函数用于计算某个数字的“深度”，也就是该数字的对数。这可以帮助我们评估某个数字在棋盘上的位置。

最终得到的最优方向是得分最高的操作中的第一个方向。具体来说，get_next() 方法会使用 itertools.product() 函数枚举所有可能的两步操作，然后计算每一种操作的得分。最后，会将所有得分按照从小到大的顺序排序，并返回得分最高的操作中的第一个方向。

例如，假设某一次调用 get_next() 方法时，计算出了以下几组操作：

- (”U”, ”L”) -> 得分 = 10
- (”U”, ”R”) -> 得分 = 20
- (”L”, ”D”) -> 得分 = 15
- (”R”, ”D”) -> 得分 = 25

在这种情况下，最终得到的最优方向就是 R。因为在所有计算出的操作中，得分最高的操作是 (”R”, ”D”)，并且这个操作的第一个方向就是”R”。

```
class Ai:
  def __init__(self):
     self.g = Grid(config.SIZE)
```

```
def get_next(self, tiles):
    score_list = []
    tn = self.get_num(tiles)
    if tn >= self.g.size ** 2 / 3:
        return "RD"[np.random.randint(0, 2)], 0
    kn = min(max(tn ** 2, 20), 40)
    for directions in itertools.product("ULRD", repeat=3):
        fen = []
        for i in range(kn):
            t_g = get_grid(tiles, directions)
            fen.append(self.get_score(t_g))
        print(directions, min(fen))
        score_list.append([directions, min(fen)])
    score_list = sorted(score_list, key=(lambda x: [x[1]]))
    for d in score_list[::-1]:
        self.g.tiles = tiles.copy()
        if self.g.run(d[0][0], is_fake=False) != 0:
            return d[0][0], d[1] / kn
    self.g.tiles = tiles.copy()
    return score_list[-1][0][0], score_list[-1][1] / kn
```

2.2.6 其他功能的实现

在 game.py 中，我们还实现了一些其他的功能，例如：

- is_full() 方法用于判断棋盘是否已满，它会遍历棋盘的每个格子，空白返回 False，否则 True
- get_random_xy() 方法用于获取一个随机的空白格子的坐标。它会调用 is_zero() 方法判断某个位置的数字是否为 0，如果是，就返回这个位置的坐标，否则就继续随机生成坐标
- set_tiles() 方法用于设置棋盘上某个位置的数字。它接受两个参数，一个是位置的坐标，另一个是数字，然后通过坐标索引设置棋盘上对应位置的数字

- add_random_tile() 方法，在棋盘上随机添加两个数字
- add_tile_init() 方法用于初始化棋盘
- __init__ 方法：初始化了类的 size 和 tiles 属性
- is_zero 方法：判断指定位置的砖块是否为 0
- get_random_xy 方法：获取一个随机的坐标
- add_tile_init 方法：初始时设置两个砖块
- move_hl 方法：移动某一行或某一列
- is_over 方法：判断是否结束

3. 完成情况

3.1 程序使用说明和运行结果

- 运行 main.py 文件，即可开始游戏。此时终端会显示程序开发的相关信息，如下图所示：

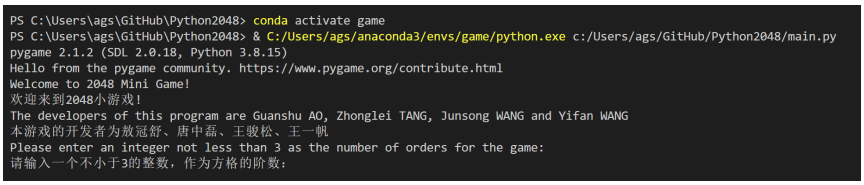

图 3-1 控制台显示的内容

- 键入一个整数并回车（此处以 4 为例），即可开始游戏。如下图所示：

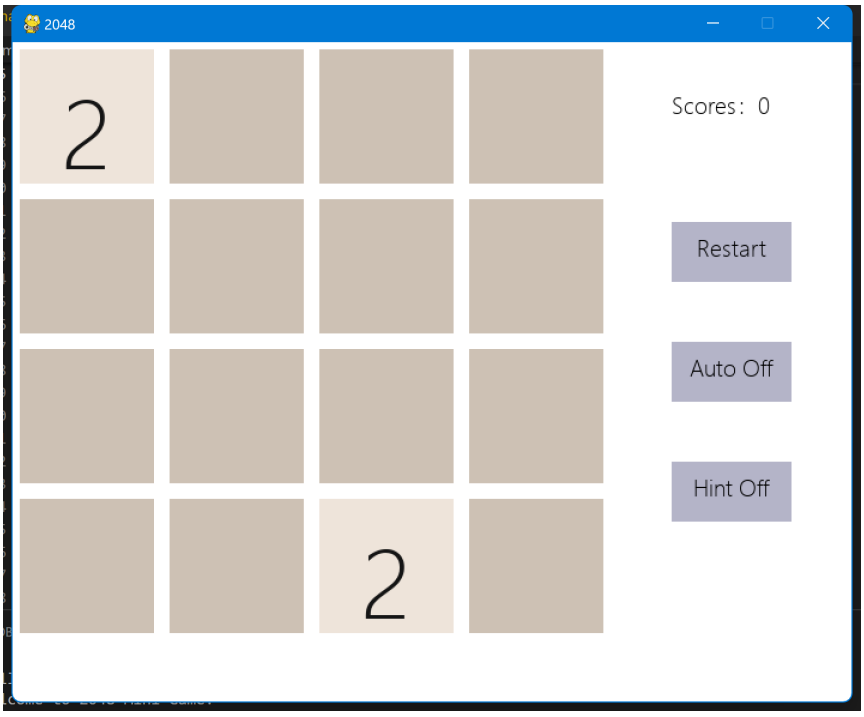

图 3-2 四阶方格的 2048 游戏

- 此时可以通过键盘上的上下左右键来控制方块的移动，游戏界面的右上角会显示当前的得分。如下图所示：

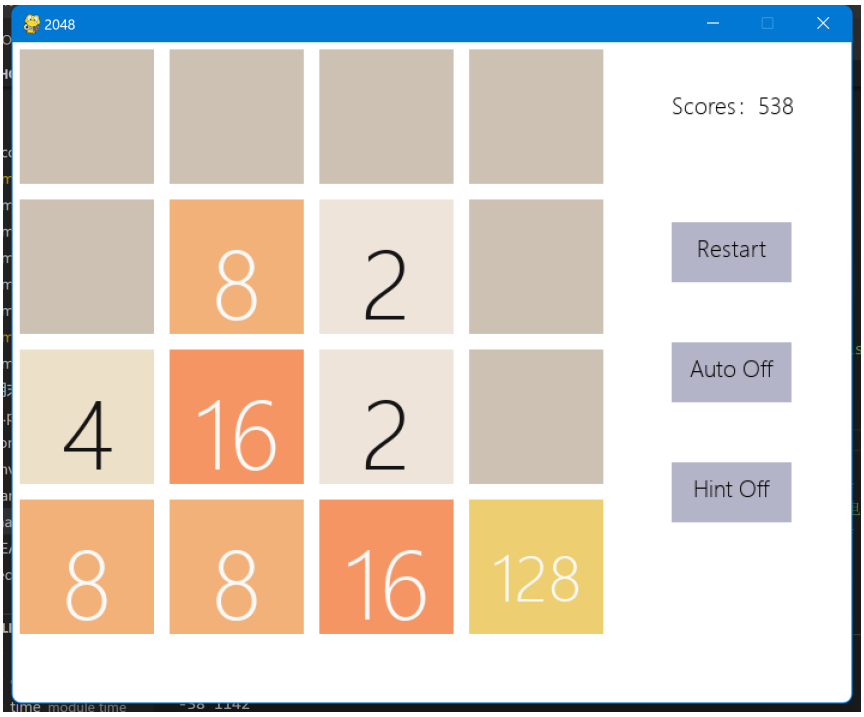

图 3-3 正常模式

- 当游戏结束时，会显示游戏结束的提示，如下图所示：

图 3-4 游戏结束

- 在游戏过程中，点击 Restart 按钮，即可清空现有游戏进度重新开始。如下图所示：

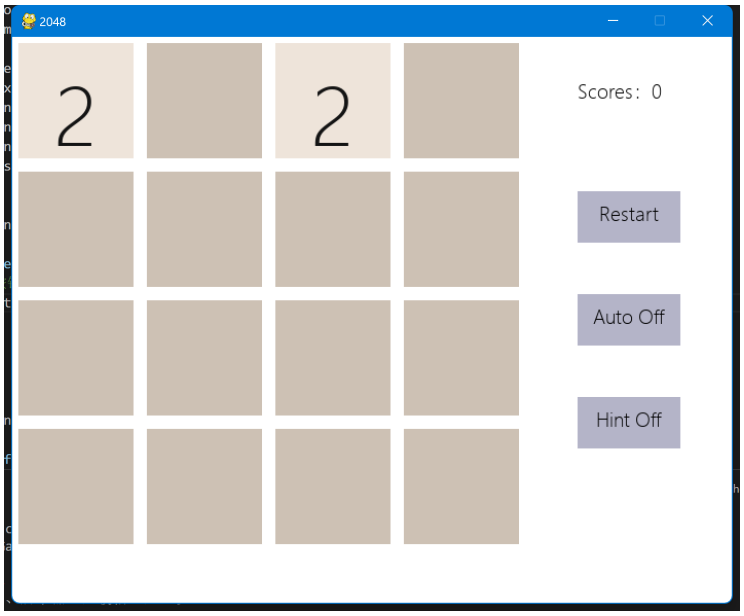

图 3-5 重新开始游戏

- 右侧第二个按钮上显示 Auto Off，表示现在处于手动模式。点击该按钮，即可切换到自动模式，在自动模式下，程序会自动寻找最优解并显示 Auto On。再次点击按钮，即可恢复手动模式。具体情形如下图所示：

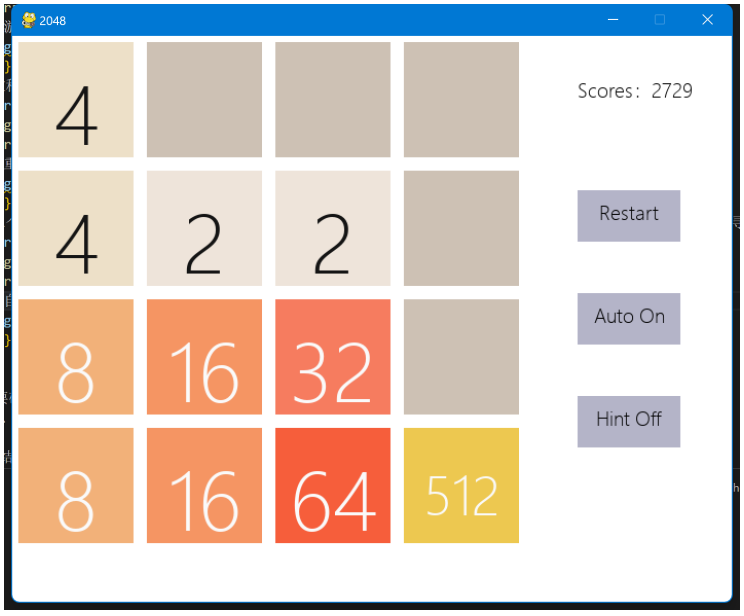

图 3-6 自动模式

- 右侧第三个按钮显示 Hint Off，表示现在未处于提示模式。点击该按钮即可切换到提示模式，在提示模式下，程序会显示当前最优解的方向，但不会自动操作。再次点击按钮，即可关闭提示模式。具体情形如下图所示：

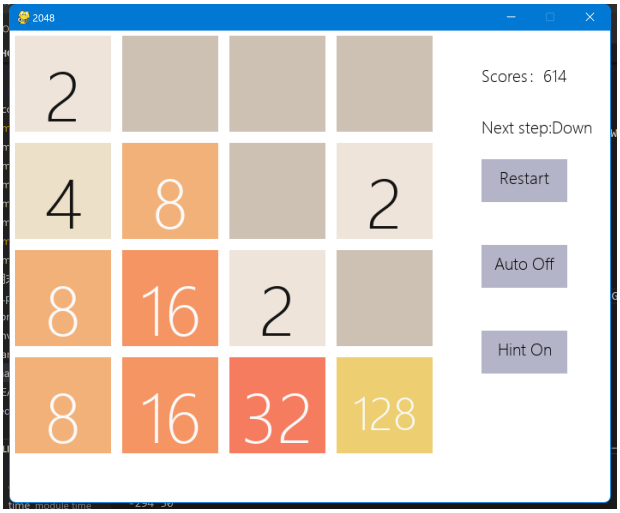

图 3-7 提示模式

3.2 特色功能

相较于其他传统 2048 游戏，本程序新增了高阶方格下的游戏模式。只需在游戏开始时在终端输入想要的阶数，即可开始游戏。如下图所示，分别为 6 阶、10 阶、20 阶、40 阶。其余功能与 4 阶方格下的游戏模式相同，不再赘述。

图 3-8 高阶方格下的游戏模式

3.3 主要研究过程

- 首先，我们需要了解 2048 游戏的基本规则，即每次操作后，所有数字方块都会向一个方向移动，如果有两个相同的数字方块相邻，那么这两个数字方块会合并，合并后的数字方块的值为两个相加的和。如果有两个不同的数字方块相邻，那么这两个数字方块不会合并。如果有空白的位置，那么数字方块会移动到空白的位置。如果有两个数字方块相邻，中间有空白的位置，那么数字方块会移动到空白的位置。如果有两个数字方块中间有其他数字方块，那么这两个数字方块不会合并。

- 其次，我们需要了解 2048 游戏的算法。2048 游戏的算法主要分为两个部分，一是生成新的数字方块，二是移动数字方块。

  生成新的数字方块。在游戏开始时，会随机生成两个数字方块，每个数字方块的值为 2 或 4。在游戏过程中，每次操作后，会随机生成一个数字方块，该数字方块的值为 2 或 4。该数字方块会随机出现在空白的位置。

  移动数字方块。从键盘读取用户的操作，根据键盘的操作，将数字方块向该方向移动，并生成新的状态矩阵。

- 最关键的是 AI 算法的实现。在本程序中，gen_next 首先会计算出一个数值 kn，它是当前棋盘的已有数字数量的平方与 20 之间的最小值，但是不能超过 40。然后使用 itertools 库的 product() 方法生成所有可能的移动序列，对于每个序列都调用 get_grid() 方法和 get_score() 方法来计算分数，并将结果存储在一个列表中。接着将所有序列的分数按照从小到大的顺序排序，然后从最高分的序列开始遍历，如果能够找到一个序列的第一个方向能够使得棋盘发生变化，就返回这个序列的第一个方向和平均分。如果所有序列都无法使棋盘发生变化，就返回最后一个序列的第一个方向和平均分。

- 在 AI 算法的基础上便可很容易地开发出提示功能。将 AI 算法的返回值改为提示的方向，显示在界面上即可实现提示功能。

4. 设计总结

4.1 成员分工

四位成员均来自电子信息学院 08062002 班，其分工信息如下表所示：

表 4-4 成员分工一览表

姓名	学号	分工
敖冠舒	2020301928	总体方案设计、撰写报告、维护 GitHub 项目、AI 算法实现、验收答辩
唐中磊	2020301923	游戏界面设计
王骏松	2020301930	软件功能测试
王一帆	2020301927	方块移动模块的实现

4.2 当前程序不足之处

总体来看，本程序的功能实现较为完整，较好地实现了指定的要求，并能在基本功能上进行扩展，如高阶方块模式，很好地增强了游戏的趣味性和丰富性。但由于时间和能力的限制，仍存在改进之处：

- 可以将游戏开始时的方块阶数集成至游戏界面中而不是终端，方便用户选择
- 可以适当添加一些游戏音效和动效，增强游戏的趣味性

4.3 改进措施

对于以上存在的问题，小组成员计划采取以下措施进行改进：

- 进一步学习用户界面设计，以期达到更好的用户体验
- 深入学习数据结构等相关知识，进一步优化当前算法，以期能够实现更加复杂的功能

4.4 课程收获

- 敖冠舒

通过本课程的学习，我收获颇丰，受益匪浅。一方面精进了 Python 语言编程技能，对面向对象编程、模块化编程、数据结构、算法等方面的知识有了更深入的了解；另一方面，我也掌握了 PyCharm、Anaconda、VSCode 等开发工具的使用，利用 GitHub 进行团队协作，以及如何利用 LaTeX 进行文档编写等技能；此外，张老师邀请到的讲座嘉宾也令我大受启发。相信这些在今后的学习和工作中都会对我有所帮助，我也希望自己日后成为一名有技能、有格局的工程师，用技术造福更多的人，让每个人享受信息科技带来的便利！

- 唐中磊

在这学期课程学习中，我学习了 Python 的一些语言基础，很多知识与大一所学的 C 语言很相似，但也有很多不同，如 Python 的输入语句是 print，而 c 语言的输入语句是 printf，C 语言一个语句的结束都要用分号隔开，而 Python 则不需要，又如在 Python 中定义变量是不需要先声明变量名及其类型的，而 C 语言则需要。而且，在字符串上的处理，Python 相对于 C 语言更加简洁便利。除了这些基础语言知识，后边也学习了 Python 字典，库的使用，还有文件检索，以及函数的应用。在学习过程中，老师也会布置一些作业，也会组织一些考试，这对我们在 Python 学习的实际应用方面有所帮助，通过做这些作业和考试积累经验，逐步将 Python 的基础知识掌握。以及最后的大作业课程设计更好的帮助我们延申 Python 的学习，从小做起，慢慢熟练掌握 Python 完成更多更难的任务。因此学习 Python 不光要课堂上听老师讲课，课下的练习和实践更为重要，通过不断地学习和编程积累经验，更加深入的了解 Python。

- 王骏松

初次学习 Python，对于这学期的课程学习，学了 Python 的一些语言基础，很多知识与大一第一学期所学的 c 语言很相似，但有很多也是不同，如 Python 的输入语句是 print，而 c 语言的输入语句是 printf，c 语言一个语句的结束都要用分号隔开，而 Python 则不需要，又如在 Python 中定义变量是不需要先声明变量名及其类型的，而 c 语言则需要。通过一学期的学习，我觉得 Python 比起 C 语言来说更为简洁简单，而 c 语言可以说是 Python 的基础。但 Python 虽然功能强大，操作起来也比较简单，但有一点是绝对不能忽视的，这是一门编程语言。学习 Python 之前一定要遵循一定的逻辑，不能急躁且盲目的乱学乱打编程，而是要循序渐进。作为一个初学者，在刚开始的学习中，我会按照课本中出现的代码，多进行实操，多打代码，多多学习别人的代码，希望能通过日积月累的学习不断提高我的编程能力，实现我一开始的学习目标。

- 王一帆

  Python 是一种非常有潜力的高级语言，经过多年来的发展，已经在编程领域有了举足轻重的作用。经过一个学期的学习，我对 Python 有了一定的了解，明白了其强大的库对编程时起到的巨大作用，让编程不在困难。在学习 Python 的第一节课上，其对我的最初的印象就是，相较于我学习过的 C 语言编程，它更加的简洁。所有的变量都不需要像 C 语言编程那样需要提前去定义，这样给了编程者很大的自由空间与方便。对简化程序代码，起了很大作用。

4.5 对课程的建议

希望老师在日后的课程中能够更加注重实践，比如结合具体的程序和工程项目，这样让学生能够更加深刻的理解所学的知识，有助于学生更好地接触工程开发前沿。

5. 附录

5.1 程序源代码

见电子压缩文档 Python2048_AGS.zip 文件，或点击这里查看本项目的 GitHub 仓库。

5.2 其他

作者的 GitHub 主页
作者的个人主页
本报告的 LaTeX 源码

5.3 致谢

感谢张顺老师一学期以来的指导与帮助，您的严谨、亲和，以及专业的教学风格，让我在学习 Python 编程的过程中，不仅学到了知识，更学到了如何学习、如何思考，以及做学问的严谨态度。