第20课：提示下一方块

步骤目标

本文的目标是为玩家提示下一方块是什么。如图1所示，窗口右上角提示下一方块是一个 I 型方块。玩家借助这一提示，能够规划消行的策略，从而能够一次多消几行，得到更高的分数。

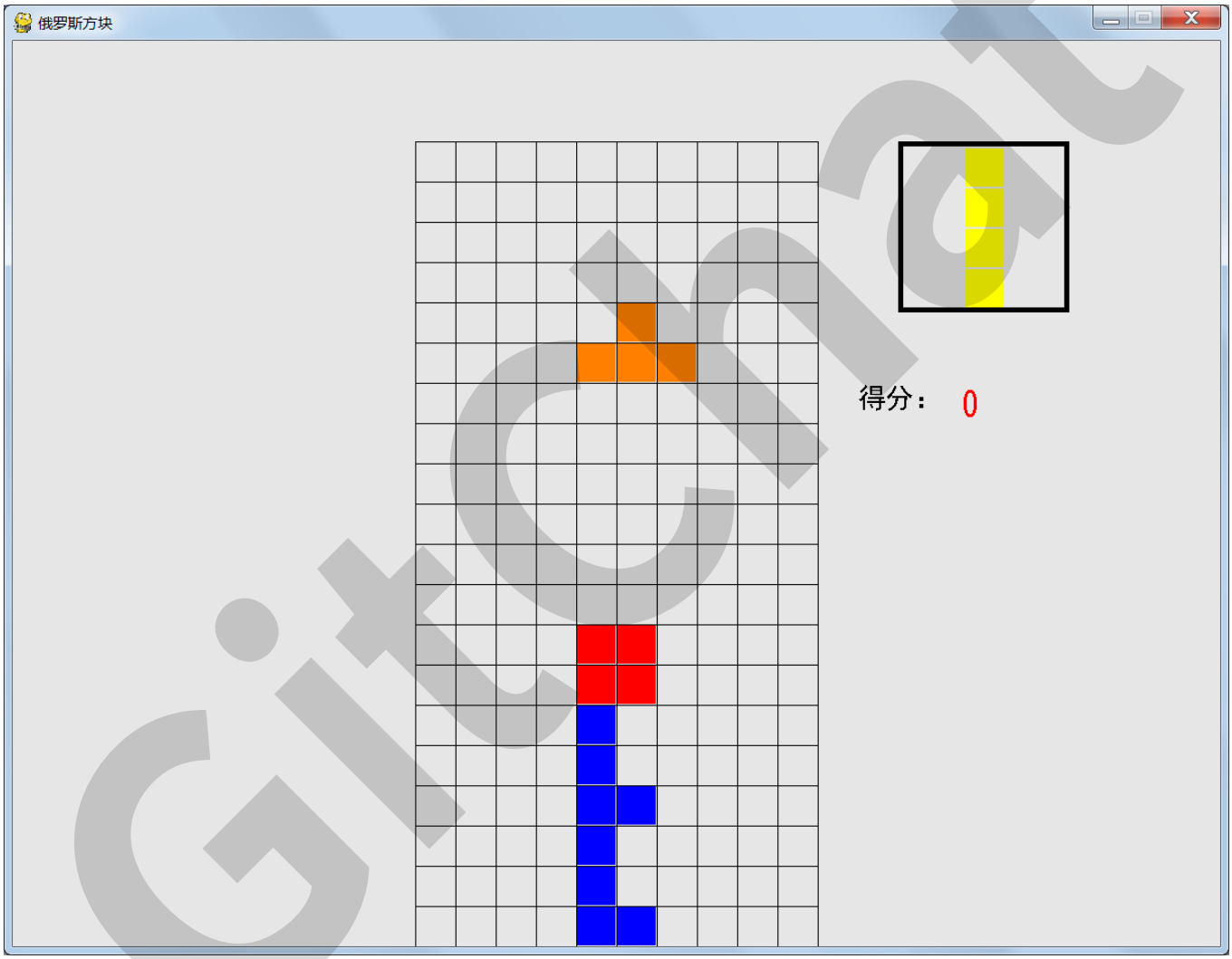

图1 提示下一方块

本步骤的任务是：

1. 生成下一方块。后来，它会变成当前方块。
2. 在窗口右上角显示下一方块。

生成下一方块

实现这一功能要做的事情包括：

1. 游戏状态对象 `game_state` 需要记住下一方块。
2. 玩家按下 s 字母键开始游戏的时候，要生成两个方块，一个是当前方块，一个是下一方块。
3. 当前方块触底时，下一方块变成当前方块，接着生成新的下一方块。

完成前两件事的代码如代码1所示。第15行是在 GameState 类内添加 `next_piece` 属性来记住下一方块是谁。第35、36行是生成两个方块。当玩家按下 s 字母键开始游戏是，程序调用 `start_game` 方法使得游戏开始。

```
1.  # TetrisGame/gamestate.py
2.  10  class GameState():
3.  11      def __init__(self, screen):
4.  12          self.screen = screen
5.  13          self.wall = GameWall(screen)
6.  14          self.piece = None
7.  15          self.next_piece = None
8.  16          self.timer_interval = TIMER_INTERVAL   #1000ms
9.  17          self.game_score = 0
10. 18          self.stopped = True
11. 19          self.paused = False
12. 20          self.session_count = 0
13. …       …… #其他方法的定义
14. 31      def start_game(self):
15. 32          self.stopped = False
16. 33          self.set_timer(TIMER_INTERVAL)
17. 34          self.timer_interval = TIMER_INTERVAL
18. 35          self.piece = self.new_piece()  #生成第一个方块。此时
    self.piece=None, self.next_piece引用下一方块对象。
19. 36          self.piece = self.new_piece()  #生成第二个方块，此时self.piece
    引用当前方块对象。
20. 37          self.session_count += 1
21. 38          self.wall.clear()
22. 39          self.game_score = 0
23. 40          self.paused = False
```

方法 `new_piece()` 的定义见下面的代码。它的作用是下一方块变成当前方块，并生成新的下

一方块。

```
def new_piece(self):
    self.piece = self.next_piece
    self.next_piece = Piece(random.choice(PIECE_TYPES), self.screen, self.wall)

    return self.piece
```

对于第三件事，也就是当前方块触底时下一方块变成当前方块，调用 `new_piece` 即可完成。修改后的代码如下所示，有注释的一行是修改后的代码。别的地方都不用修改。

```
    def touch_bottom(self):
        self.wall.add_to_wall(self.piece)
        self.add_score(self.wall.eliminate_lines())
        for c in range(COLUMN_NUM):
            if self.wall.is_wall(0, c):
                self.stopped = True
                break
        if not self.stopped:
            self.piece = self.new_piece()   #此处有修改
            if self.piece.hit_wall():
                self.stopped = True
        if self.stopped:
            self.stop_timer()
```

显示下一方块

要显示下一方块，我们打算在 GameDisplay 类内定义一个 `draw_next_piece` 静态方法。该方法完成绘制下一方块的功能。如图1右上角所示，绘制的内容有边框和下一方块。`draw_next_piece` 方法的定义如代码2所示。

```
# TetrisGame/gamedisplay.py
87  @staticmethod
88  def draw_next_piece(screen, next_piece):
89          '''绘制下一方块'''
90          #绘制边框
91          start_x = GAME_AREA_LEFT + COLUMN_NUM * CELL_WIDTH + MARGIN_WIDTH * 2
92          start_y = GAME_AREA_TOP
```

```
93          GameDisplay.draw_border(screen, start_x, start_y, 4, 4)
94
95          if next_piece:
96              start_x += EDGE_WIDTH
97              start_y += EDGE_WIDTH
98              cells = []     #cells变量存储姿态矩阵中有砖块的单元格
99              #扫描姿态矩阵，得出有砖块的单元格
100             shape_template = PIECES[next_piece.shape]
101             shape_turn = shape_template[next_piece.turn_times]
102             for r in range(len(shape_turn)):
103                 for c in range(len(shape_turn[0])):
104                     if shape_turn[r][c] == 'O':
105                         cells.append( ( c,  r, PIECE_COLORS[next_piece.shape]) )
106             #绘制方块
107             for cell in cells:
108                 color = cell[2]
109                 left_top = (start_x + cell[0] * CELL_WIDTH, start_y + cell[1] * CELL_WIDTH)
110                 GameDisplay.draw_cell_rect(screen, left_top, color)
```

代码2 绘制下一方块的代码

对于以上代码，简要说明如下：

1. 第91~93行的三行代码的作用是调用 `draw_border` 方法绘制边框。方法 `draw_border` 的定义见后。边框包围的区域是 4X4 的网格。每个单元格的尺寸与游戏区域单元格的尺寸一样。
2. 第98~105行的代码的作用是得出下一方块的姿态矩阵中有砖块的单元格。一共有4个有砖块的单元格，由 cells 变量来存储。
3. 第107~110行代码的作用是调用 `draw_cell_rect` 方法绘制每一个“小块”。最终，完成绘制下一方块的功能。

GameDisplay 类的 `draw_border` 方法的作用是绘制边框。它是通过绘制4个5像素宽的矩形来做到的。下面的 `EDGE_WIDTH` 常量的值是5。

```
    @staticmethod
    def draw_border(screen, start_x, start_y, line_num, column_num):
        top_border = pygame.Rect(start_x, start_y, 2 * EDGE_WIDTH + column_num * CELL_WIDTH, EDGE_WIDTH)
        pygame.draw.rect(screen, EDGE_COLOR, top_border)
```

```
5.
6.          left_border = pygame.Rect(start_x, start_y, EDGE_WIDTH, 2 * EDG
    E_WIDTH + line_num * CELL_WIDTH)
7.          pygame.draw.rect(screen, EDGE_COLOR, left_border)
8.
9.          right_border = pygame.Rect(start_x + EDGE_WIDTH + column_num *
    CELL_WIDTH, start_y, EDGE_WIDTH,
10.                                     2 * EDGE_WIDTH + line_num *
    CELL_WIDTH)
11.         pygame.draw.rect(screen, EDGE_COLOR, right_border)
12.
13.         bottom_border = pygame.Rect(start_x, start_y + EDGE_WIDTH + lin
    e_num * CELL_WIDTH,
14.                                     2 * EDGE_WIDTH + column_num *
    CELL_WIDTH, EDGE_WIDTH)
15.         pygame.draw.rect(screen, EDGE_COLOR, bottom_border)
```

上述两个方法内使用了两个常量，我们要在 settings.py 内定义它们。它们的单位都是像素。

```
1.  EDGE_WIDTH = 5        #游戏区域外框线宽度
2.  MARGIN_WIDTH = 40     #游戏区域边界离其他窗口元素的间距
```

方法 `draw_cell_rect()` 定义如下，它的作用是绘制“砖块”，也就是填充单元格。第二个参数 `left_top_anchor` 是绘制的矩形的左上角在窗口中的横纵坐标。这里视作锚点，所以用了 anchor 这个词。

```
1.  @staticmethod
2.  def draw_cell_rect(screen, left_top_anchor, color):
3.      left_top_anchor = (left_top_anchor[0] + 1, left_top_anchor[1] + 1)
4.      cell_width_height = (CELL_WIDTH - 2, CELL_WIDTH - 2)
5.      cell_rect = pygame.Rect(left_top_anchor, cell_width_height)
6.      pygame.draw.rect(screen, color, cell_rect)
```

上述代码能够绘制下一方块。不过呢，有点小问题。如图2所示，窗口右上角，I 型方块没有居中显示，不够美观。

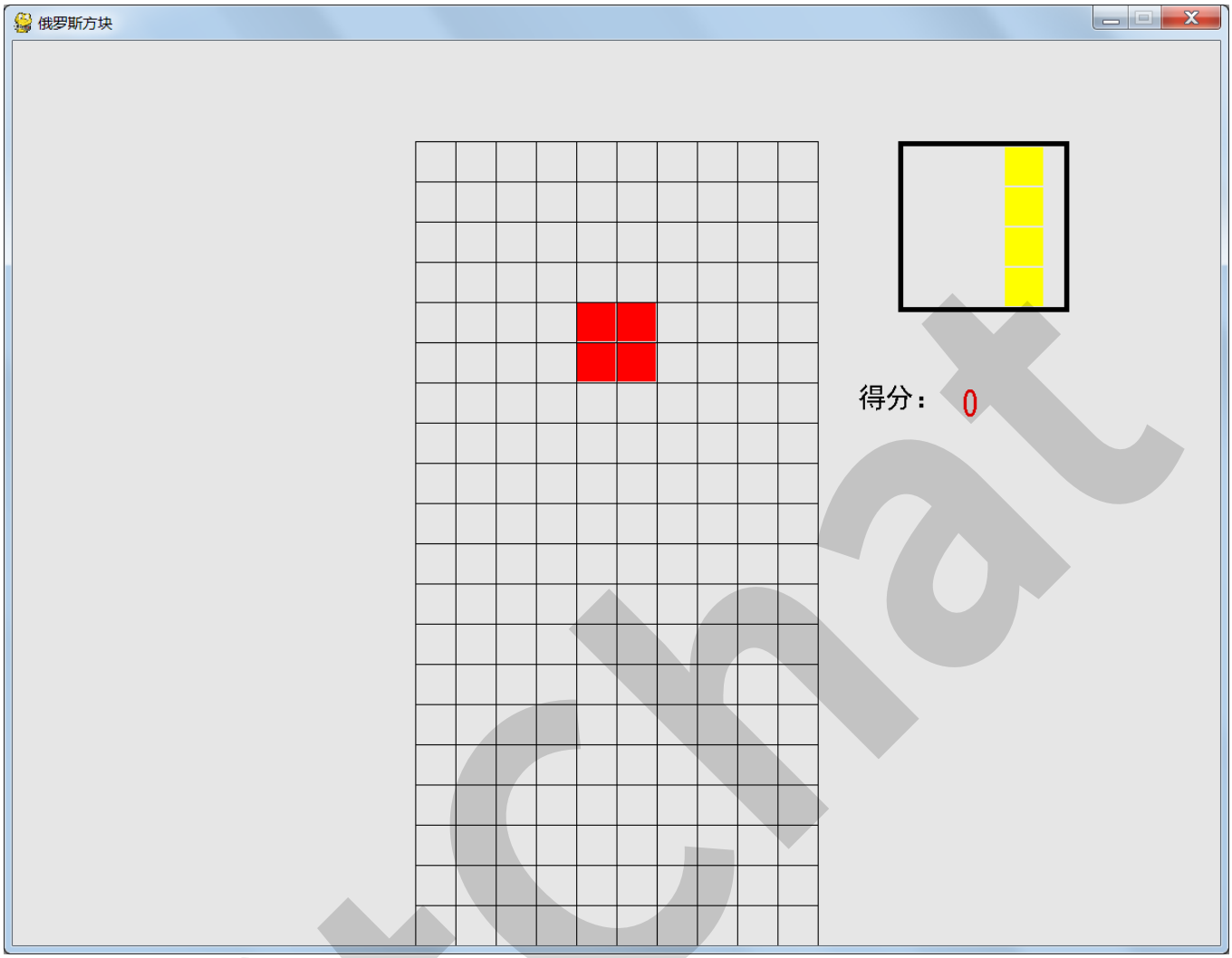

图2 右上角的 I 型方块没有居中显示

要怎么做才能够居中显示下一方块呢？就 I 型方块而言，它需要离左边界1.5个单元格，离右边界也是1.5个单元格。实现水平居中的思路是，首先求出方块有几列（比如 I 型有1列，S 型有3列），接着总列数4减去方块的列数得到空白列数，再接着空白列数除以2，乘以40（单元格的边长，即列宽，单位是像素），最后得到方块与下一方块显示区域左边界的距离。下面的代码按这个思路，使得方块水平居中。

```
1.  max_c = max([cell[0] for cell in cells])
2.  min_c = min([cell[0] for cell in cells])
3.  start_x += round( (4 - (max_c - min_c + 1)) / 2 * CELL_WIDTH)
```

简要说明上述三行代码:

1. `[cell[0] for cell in cells]` 是列表解析式（也叫列表推导式）。它生成一份新列

表，值是方块占据的各列的列号。新列表的元素由 cells 列表的各个元素的第一个子元素（即 `cell[0]`）组成。这里，cell 变量会依次取 cells 列表中的各个元素。据代码2中第105行，`cell[0]` 是单元格的列号（从0开始编号），`cell[1]` 是行号。

2. 变量 `max_c` 是这些列号的最大值。变量 `min_c` 是这些列号的最小值。`max_c - min_c + 1` 是方块的列数。
3. `round( (4 - (max_c - min_c + 1)) / 2 * CELL_WIDTH)` 就是方块与“下一方块显示区域”左边界的距离（单位是像素）。`4 - (max_c - min_c + 1)` 是空白列数。`(4 - (max_c - min_c + 1)) / 2` 是空白列数的一半。要记得，这里的除法是浮点数除法。round 函数的作用是四舍五入取整。
4. 上面第3行代码执行前，`start_x` 的值是下一方块显示区域左边界与窗口左边界的距离。这个值加上方块与下一方块显示区域左边界的距离后，`start_x` 的值变为方块与窗口左边界的距离。这里的方块是指下一方块。
5. 垂直居中的做法与水平居中的做法是类似的。

改写后的 `draw_next_piece` 方法如代码3所示。

```
# TetrisGame/gamedisplay.py
87  @staticmethod
88  def draw_next_piece(screen, next_piece):
89      '''绘制下一方块'''
90      start_x = GAME_AREA_LEFT + COLUMN_NUM * CELL_WIDTH + MARGIN_WIDTH * 2
91      start_y = GAME_AREA_TOP
92      GameDisplay.draw_border(screen, start_x, start_y, 4, 4)
93
94      if next_piece:
95          start_x += EDGE_WIDTH
96          start_y += EDGE_WIDTH
97          cells = []     #记住要绘制的单元格
98          shape_template = PIECES[next_piece.shape]
99          shape_turn = shape_template[next_piece.turn_times]
100         for r in range(len(shape_turn)):
101             for c in range(len(shape_turn[0])):
102                 if shape_turn[r][c] == 'O':
103                     cells.append( ( c,  r, PIECE_COLORS[next_piece.shape]) )
104
105         max_c = max([cell[0] for cell in cells])
106         min_c = min([cell[0] for cell in cells])
```

```
107          start_x += round( (4 - (max_c - min_c + 1)) / 2 * CELL_WIDTH
)
108          max_r = max([cell[1] for cell in cells])
109          min_r = min([cell[1] for cell in cells])
110          start_y += round( (4 - (max_r - min_r + 1)) / 2 * CELL_WIDTH
)
111
112          for cell in cells:
113              color = cell[2]
114              left_top = (start_x + (cell[0] - min_c) * CELL_WIDTH,
115                                    start_y + (cell[1] - min_r) *
CELL_WIDTH)
116              GameDisplay.draw_cell_rect(screen, left_top, color)
```

代码3 `draw_next_piece` 方法

代码3中，第105~107行代码使方块水平居中，第108~110行使方块垂直居中。第114行中，减去 `min_c` 是必要的，因为 `start_x` 的值是方块与窗口左边界的距离，而不是下一方块显示区域与窗口左边界的距离。第115行中，减去 `min_r` 也是必要的，道理类似。

小结

本文实现了提示下一方块的功能。做法是：

1. 游戏开始时，程序生成两个方块，一个是第1个方块（即当前方块），一个是第2个方块（即下一方块）。
2. 当前方块触底时，下一方块变成当前方块，接着生成新的下一方块。
3. 在窗口右上角显示下一方块。这样，玩家就能够规划消行策略。
4. 在下一方块显示区域内居中（包括上下居中和水平居中）显示方块。

完成本实验步骤的全部代码可从以下链接浏览或阅读。

- Github

下一篇，我们将去除游戏区域的网格线。