第09课：翻转方块

步骤目标

本文将实现翻转方块功能。有了这一功能后，玩家按下向上方向键，方块将顺时针旋转90度，如图1所示。图1中的网格是游戏区域的一部分，下同。

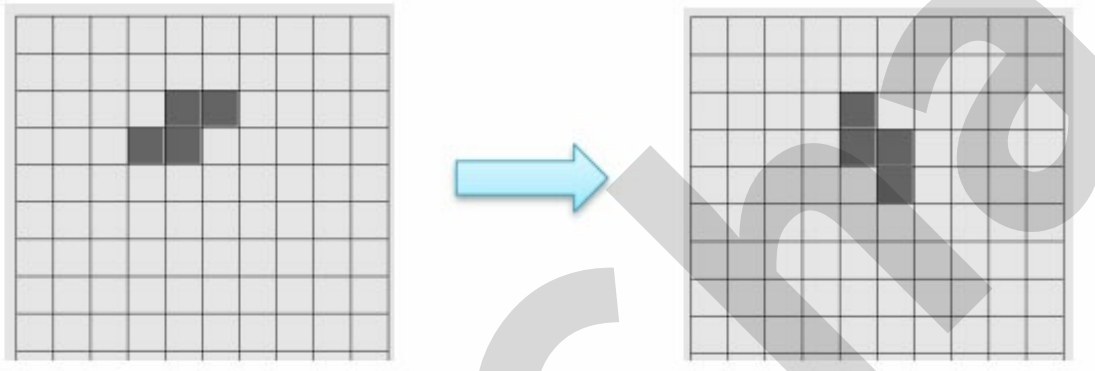

图1 翻转方块（左图是翻转前的姿态，右图是翻转90度后的姿态）

本文的任务是：

1. 阐述“方块的姿态”这一概念。
2. 描述表示方块姿态的方法。
3. 描述如何记住和绘制方块的姿态。
4. 描述如何防止方块因翻转而出界。

方块的姿态

玩家按下向上方向键，方块将顺时针旋转90度，也就是说方块的姿态发生变化。如图1所示，左图是 S 型方块的初始姿态，右图是顺时针翻转90度后的姿态。右图所示的姿态再翻转90度，将回到左图所示的姿态，所以 S 型方块一共有两种姿态。我们不妨把左图所示的姿态叫做 S 型方块的第1个姿态，右图所示的姿态叫做 S 型方块的第2个姿态。

T 型方块有四种姿态，如图2所示。第1个姿态是初始姿态，也就是方块“刚出生”时的姿态。方块顺时针旋转90度，第1个姿态变为第2个姿态，再旋转90度，变成第3个姿态，再旋转90

度，变成第4个姿态，再旋转90度，回到第1个姿态。旋转4次，合计旋转360度。

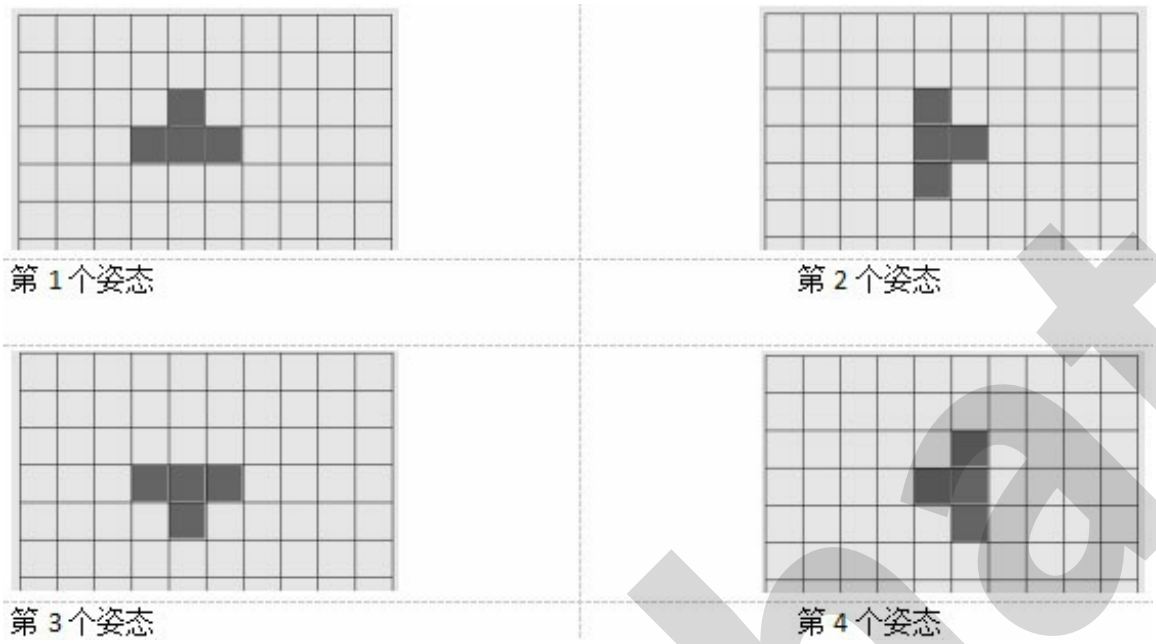

图2 T 型方块有4个姿态

归纳起来，各种方块的姿态数目是：

1. S、Z、I 型方块有两种姿态。
2. T、J、L 型方块有四种姿态。
3. O 型方块有1种姿态。

表示方块姿态的方法

前面的实现步骤中，我们用形状矩阵来表示方块的形状。比如，S 型方块的形状矩阵是：

```
S_SHAPE_TEMPLATE = ['.OO.',
                    'OO..',
                    '....']
```

如何表示 S 型方块的各个姿态？答案是把表示方块形状的矩阵（称之为形状矩阵）扩充为表示方块姿态的一组矩阵（称之为姿态矩阵列表）。

比如，S 型方块的姿态矩阵列表是：

```
S_SHAPE_TEMPLATE = [['.OO.',
                     'OO..',
                      '....'],
                    ['.O..',
                     '.OO.',
                      '..O.']]
```

我们可以看到，S 型方块的姿态矩阵列表一共有两个元素。这两个元素都是矩阵。第1个元素表示的是 S 型方块的第1个姿态，第2个元素表示的是第2个姿态。表示第1个姿态（即初始姿态）的矩阵，正是原来的形状矩阵。

I 型方块的姿态矩阵列表是：

```
I_SHAPE_TEMPLATE = [['.O..',
                     '.O..',
                     '.O..',
                     '.O..'],
                    ['....',
                     'OOOO',
                     '....',
                     '....']]
```

现在，你应该猜到 I 型方块的姿态矩阵为什么是4x4的。因为初始姿态占4行，第2个姿态占4列。姿态矩阵的高度和宽度是固定的，不随着姿态的变化而变化。是姿态矩阵内的元素内容的变化体现了姿态的变化。

T 型方块的姿态矩阵列表是：

```
T_SHAPE_TEMPLATE = [['.O..',
                     'OOO.',
                      '....'],
                    ['.O..',
                     '.OO.',
                      '.O..'],
                    ['....',
                     'OOO.',
                      '.O..'],
                    ['..O.',
                     '.OO.',
                      '..O.']]
```

T 型方块的姿态矩阵是3x4的。4个姿态矩阵的最后1列都为空，因此删除最后1列，设定为3x3也是可以的。

需要指出的是，方块的翻转，即顺时针旋转90度，并不是以某个方块为中心点进行旋转的。你不要死扣这个中心点。只要旋转的效果符合玩家的期望就可以了。

7种方块的姿态矩阵列表在 settings.py 文件内定义。本文末尾给出了整个工程的源代码链接，你可以浏览或下载。在理解方块姿态表示方法的前提下，可以复制各种方块的姿态矩阵列表的定义。

记住和绘制方块的姿态

记住方块的姿态

通过记住方块的翻转次数，即可记住方块的姿态。为此，我们在 Piece 类的构造方法（代码见下）内添加了 `turn_times` 属性，用来记录翻转次数。最开始，翻转次数为0，故初始化为0。

```
class Piece():
    def __init__(self, shape, screen):
        self.x = 3
        self.y = 0
        self.shape = shape
        self.turn_times = 0    #翻转了几次，决定方块的姿态
        self.screen = screen
```

玩家每翻转方块1次，方块对象的 `turn_times` 属性增1。下面我们来看看如何实现这一特性。

玩家按下向上方向键，意味着玩家指示翻转方块。于是，我们首先要响应向上方向键按下事件。实现这一响应的代码如代码1第54~56行所示。第56行调用了方块对象的 turn 方法。你能够想到方法内部将把 `turn_times` 属性增1。

```
# TetrisGame/main.py
45  def check_events(piece):
46      '''捕捉和处理键盘按键事件'''
47      for event in pygame.event.get():
48          if event.type == pygame.QUIT:
```

```
49                 sys.exit()
50             elif event.type == pygame.KEYDOWN:
51                 if event.key == pygame.K_DOWN:
52                     # print("向下方向键被按下")
53                     piece.move_down()
54                 elif event.key == pygame.K_UP:
55                     # print("向上方向键被按下")
56                     piece.turn()
57                 elif event.key == pygame.K_RIGHT:
58                     # print("向右方向键被按下")
59                     piece.move_right()
60                 elif event.key == pygame.K_LEFT:
61                     # print("向左方向键被按下")
62                     piece.move_left()
```

代码1 响应向上方向键，翻转方块

Piece 类中，添加 turn 方法。方法定义如下:

```
1.     def turn(self):
2.         shape_list_len = len(PIECES[self.shape])
3.         self.turn_times = (self.turn_times + 1) % shape_list_len
```

你会问，怎么不是增1呢，怎么搞这么复杂？容我解释一下。先问一个问题，翻转次数超过方块的姿态列表的长度时，该怎么确定方块的姿态？比如说，S型方块的姿态列表长度为2，而翻转次数为5，怎么确定方块的姿态？答案是进行求余运算，5 % 2 => 1，因此绘制方块时该用第2个姿态（记住，列表的下标是从0开始的。我们说的第1个对应0号下标）。

上面的 turn 方法定义体内，第1条语句中，self.shape 是方块的类型（值为字母，如 `S`、`T`）。PIECES 是字典，PIECES[self.shape] 得出方块的姿态矩阵列表。len(PIECES[self.shape]) 是求出姿态矩阵列表的长度，也就是说 `shape_list_len` 变量是姿态列表的长度。以 S 型方块为例，`shape_list_len` 变量的值是2。

第2条语句中，`self.turn_times + 1` 把翻转次数增1。然后对 `shape_list_len` 变量求余，这将得出翻转次数对应的下标。这一下标是指方块的姿态矩阵列表中的下标。也就是说，方块对象的 `turn_times` 属性不是直接记住翻转次数，而是记住翻转次数对应的下标。

绘制方块的姿态

接下来，我们修改Piece类的paint方法，以实现绘制方块的功能。修改前、后如代码2、代码3所示。

```
def paint(self):
    shape_template = PIECES[self.shape]

    for r in range(len(shape_template)):
        for c in range(len(shape_template[0])):
            if shape_template[r][c] == 'O':
                self.draw_cell(self.x + c, self.y + r)
```

代码2 修改前的paint()

```
def paint(self):
    shape_template = PIECES[self.shape]
    shape_turn = shape_template[self.turn_times]

    for r in range(len(shape_turn)):
        for c in range(len(shape_turn[0])):
            if shape_turn[r][c] == 'O':
                self.draw_cell(self.x + c, self.y + r)
```

代码3 修改后的paint()

下面的代码行：

```
shape_turn = shape_template[self.turn_times]
```

作用是得出表示方块姿态的矩阵。以 S 型方块为例，若翻转次数合计为5，则 `self.turn_times` 等于1，那么 `shape_turn` 的值是：

```
['.O..',
 '.OO.',
 '..O.']
```

翻转方块的执行流程

通过上面描述的代码改动，我们实现了翻转方块的功能。这些改动是：

1. settings.py 文件内，定义7种方块的姿态矩阵。
2. piece.py 文件内，增加 `turn_times` 属性，用来记住翻转次数。增加 turn 方法，用来把翻转次数增1。修改 paint 方法，实现绘制方块的姿态的特性。
3. main.py 文件内，`check_events` 函数内添加响应向上方向键按下事件，处理动作是调用方块对象的 turn 方法。

我们来梳理一下翻转方块的执行流程。

1. 玩家按下向上方向键。键盘事件产生，在程序的事件队列中加入事件对象。
2. 程序再一次执行主循环。主循环内，首先调用 `check_events` 方法，于是会响应处理向上方向键按下事件，调用方块对象的 turn 方法，把翻转次数增1。
3. 主循环内，在 `check_events` 函数执行完毕后，调用方块对象的 paint 方法。该方法执行时，将根据翻转次数得出方块的姿态，接着取到这种姿态对应的姿态矩阵，然后进行绘制。

防止翻转后出界

与移动方块类似，翻转方块也要避免方块出界。实现的思路是：先检查翻转后有没有发生出界的情况，如何有，则不翻转方块；否则翻转方块。

实现代码如代码3所示。第82行代码是新增的。它的作用是调用 `can_turn` 方法检查方块翻转后有没有发生出界的情况。

```
# TetrisGame/piece.py
80  def turn(self):
81      shape_list_len = len(PIECES[self.shape])
82      if self.can_turn():
83          self.turn_times = (self.turn_times + 1) % shape_list_len
84
85  def can_turn(self):
86      shape_list_len = len(PIECES[self.shape])
87      turn_times = (self.turn_times + 1) % shape_list_len
88      shape_mtx = PIECES[self.shape][turn_times]
89      for r in range(len(shape_mtx)):
90          for c in range(len(shape_mtx[0])):
91              if shape_mtx[r][c] == 'O':
92                  if (self.x + c < 0 or self.x + c >= COLUMN_NUM) or (
self.y + r < 0 or self.y + r >= LINE_NUM):
```

```
93                         return False
94     return True
```

代码3 避免翻转后出界的实现代码（左侧数字是文件内的行号）

对 `can_turn` 方法说明如下：

1. 第85~94行定义了 `can_turn` 方法，作用是检查方块翻转后有没有发生出界的情况。如果有出界情况则返回 False，如果没有则返回 True。
2. 第87行，`turn_times` 是一个局部变量，而不是方块对象的 `turn_times` 属性，它的值是增1后的翻转次数（对应的下标）。
3. 第88行，`shape_mtx` 变量的值是翻转次数对应的姿态矩阵。这是 `turn_times` 局部变量对应的姿态矩阵。
4. 第89~90行是遍历姿态矩阵的元素，`len(shape_mtx)` 得到矩阵的行数，`len(shape_mtx[0])` 得到矩阵的列数。
5. 第91~93行的作用是，对于组成方块的小块（一共4个），如果出了水平方向的边界或垂直方向的边界，那么就返回 False，函数结束。
6. 4个小块都没有越界的话，则执行第94行返回 True。

小结

本文所讲解的步骤实现了方块翻转功能，以及防止翻转后出界的功能。你首先要理解方块的姿态这一概念，以及表示方块姿态的方法。接着，着手实现记住翻转次数，据翻转次数得出姿态和绘制方块的姿态的功能特性。最后，实现防止方块翻转后出界的功能特性。

完成以上全部功能的代码请见：Github。

你可以下载下来，进行比对。Pycharm 有文件对比功能。你可以上网搜索文件对比的操作方法。

下一篇所讲解的步骤中，我们将实现随机生成新方块的功能。前面的步骤中，方块的颜色是灰灰的，大家想必有点看法吧。那么，下一实验步骤将改变这一点，让不同类型的方块有不同的颜色。