第07课：移动方块

步骤目标

前面两篇文章中，我们首先绘制了方块，接着捕捉了键盘按键事件。这些都为本文所要实现的目标做铺垫。

本文实现的目标是：玩家按下向左、向右或向下方向键的时候，方块向左、向右或向下移动。方块移动的效果图见图1。下一实验步骤将实现避免方块移出游戏区域的功能，本实验步骤不予实现。

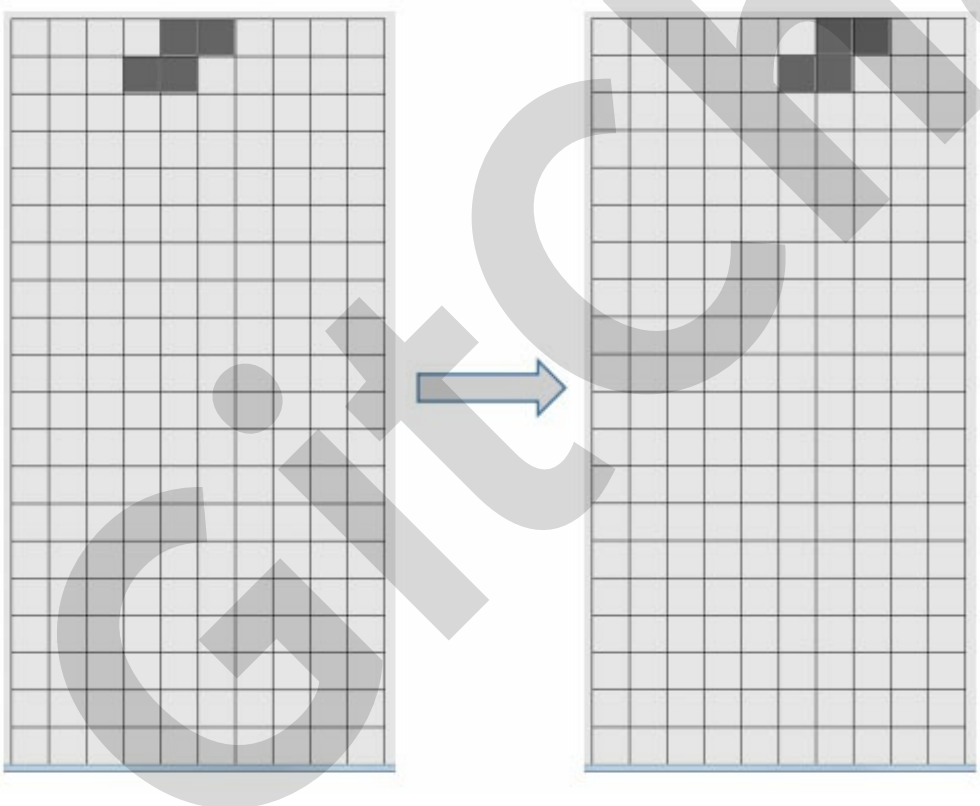

图1 移动方块（按下向右方向键2次，方块向右移动了2格）

实现移动方块

本文，我们首先讲解如何实现向右移动方块的功能。然后，由你亲自实现向左、向下移动方块的功能。

下面分两小节讲解向右移动方块的功能是如何实现的。第一小节给出实现方块向右移动功能的代码。第二小节是解释这一段代码是如何执行的。

实现方块向右移动的代码

基于上一篇的代码实现，我们对其中的三处做下扩充。一是在 Piece 类内定义 `move_right()` 方法。二是在 `check_events()` 函数内调用 `move_right()` 方法。三是调用 `check_events` 函数要传入方块对象。

1. 在 Piece 类内定义 `move_right()` 方法。

定义 `move_right()` 方法的代码如代码1所示。代码1中，第34行之前的代码行未作任何改动，故省略。

```
TetrisGame/piece.py
10  class Piece():
…       ……  #定义了构造方法__init__, paint, draw_cell。与上一版相同。
33      def move_right(self):
34          '''方块向右移动1个单元格'''
35          self.x += 1
```

表1 定义 `move_right()` 方法的代码

我们来看 `move_right()` 方法的定义。里头只有一条语句。属性 self.x 记录的是方块在游戏区域内的定位点的横坐标。横坐标增1，意味着方块向右移动1格。

2.在 `check_events()` 函数内调用 `move_right()` 方法。

调用 `move_right()` 方法的时机是在玩家按下向右方向键的时候。上一篇步骤的描述中讲到 `check_events()` 函数来响应键盘按键事件。因此，要在 `check_events()` 函数内调用 `move_right()` 方法，如代码2所示。这一函数定义在 main.py 文件内45~59行。要注意到函数需要传入一个参数，用于引用当前方块。

```
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                 elif event.key == pygame.K_UP:
54                     print("向上方向键被按下")
55                 elif event.key == pygame.K_RIGHT:
56                     # print("向右方向键被按下")
57                     piece.move_right()
58                 elif event.key == pygame.K_LEFT:
59                     print("向右方向键被按下")
```

代码2 在 `check_events()` 函数内调用 `move_right()` 方法

3. 调用 `check_events` 函数时传入方块对象。

程序主循环内，调用 `check_events` 函数的时候要传入方块对象，如下代码所示。

```
# TetrisGame/main.py
10  def main():
11      #初始化pygame。启用Pygame必不可少的一步，在程序开始阶段执行。
12      pygame.init()
…   ……   #省略部分代码
19      #生成方块对象
20      piece = Piece('S', screen)
21
22      #游戏主循环
23      while True:
24          #监视键盘和鼠标事件
25          check_events(piece)
26
27          #设定屏幕背景色
28          screen.fill(bg_color)
29          #绘制游戏区域网格线
30          draw_game_area(screen)
31          #绘制方块
32          piece.paint()
33          #刷新屏幕
34          pygame.display.flip()
```

以上三处作了修改扩充之后，就能够响应向右方向键按下事件，效果是向右移动方块。你可以运行试试，看是不是达成了意图。

向右移动方块代码的执行流程

为什么做了上面小节描述的三处扩充，就能向右移动方块呢？让我们来分析代码的执行流程，一窥究竟。

程序主循环的执行流程是：

> while True:
>
> ```
> 1.响应处理键盘按键等事件
> 2.填充窗口背景色
> 3.绘制游戏区域网格线
> 4.绘制方块
> 5.刷新窗口
> ```

向右移动方块代码的执行流程是：

1. 当玩家按下向右方向键的时候，将进入主循环第1步，调用 `check_events()` 函数，接着调用方块对象的 `move_right()` 方法，使得方块的定位点向右移动1格。
2. 在主循环第4步“绘制方块”中，根据方块的新定位点来绘制方块，显示的是“向右移动1格”后的方块。

实现向左、向下移动方块

理解实现向右移动方块的代码后，依葫芦画瓢，很容易就可以实现向左或向下移动方块的功能。请你自己试着实现它们。

方块不会向上移动，所以无需实现这一功能。以后，我们将利用向上方向键来翻转方块。

实现向左、向右、向下移动方块的完整代码请见 Github。

你要自己彻底搞懂。这样，实现后面的步骤就轻松多了。加油吧！

在下一步骤中，我们将防止方块出界。