第16课：玩家指示开始游戏

步骤目标

到上一篇为止，游戏程序一运行，游戏就开始了——方块从顶部往下落。本文要达成的目标是实现“由玩家按下 s 字母键来开始游戏”的功能。

本步骤要完成的任务是：

1. 游戏程序启动后，向玩家提示“按 s 字母键开始” ，如图1所示。
2. 玩家按下 s 字母键后，游戏正式开始。

图1 程序启动后提示玩家“按s字母键开始”

技术上讲，本步骤要做到的事情是：

1. 修改前一步骤的代码版本，使得程序启动阶段不生成第一个方块，也不设置触发方块自动下落的定时器。我们把这些修改叫做“程序启动阶段的修改”。
2. 玩家指示开始游戏后，生成第一个方块，启动定时器。
3. 显示图片。图1中，游戏区域内显示的“按 s 字母键开始”是一幅图片。在游戏停止期间，将显示图片。游戏进行过程中，不显示图片。

程序启动阶段的修改

上面已经讲到，程序启动阶段，不要生成第一个方块，也不要启动触发方块自动下落的定时器。从第13课开始，程序启动后执行创建游戏状态对象 `game_state` 的语句，期间调用 GameState 类的构造方法，在方法内生成游戏的第一个方块，并启动定时器。我们要修改 GameState 类的构造方法，使得程序启动阶段不做这两件事：生成游戏的第一个方块，和启动定时器。

我们要修改的是 gamestate.py 文件内 GameState 类的构造方法，如代码1和代码2所示。代码1是上一步骤完成后的版本，也即修改前的版本，代码2是修改后的版本。修改之处请看代码中的注释说明。代码2中还有第三处修改，即定义 stopped 属性，这里把它设值为 True，表示游戏处于停止状态。后面会用到这一属性，到时候再解释。

```
class GameState():
    def __init__(self, screen):
        self.screen = screen
        self.wall = GameWall(screen)
        self.piece = Piece(random.choice(PIECE_TYPES), screen, self.wall)
        self.timer_interval = TIMER_INTERVAL   #1000ms
        self.set_timer(self.timer_interval)
        self.game_score = 0
```

代码1 GameState 类的构造方法修改之前

```
class GameState():
    def __init__(self, screen):
        self.screen = screen
        self.wall = GameWall(screen)
```

```
            self.piece = None   # 此处有修改
            self.timer_interval = TIMER_INTERVAL    #1000ms
            # self.set_timer(self.timer_interval)   #本行可删除
            self.game_score = 0
            self.stopped = True   #游戏停止了吗？
```

代码2 GameState 类的构造方法修改之后

玩家指示开始游戏后，将生成第一个方块和启动定时器。下一节我们安排一个函数来做这两件事。

玩家指示开始游戏

玩家按下 s 字母键指示开始游戏后，要做三件事：

1. 响应按键事件；
2. 生成第一个方块；
3. 启动定时器。

后两件事是响应玩家按 s 字母键而进行的处理。我们用 GameState 类的 `start_game` 方法来完成后两件事。在 gamestate.py 文件内定义的 GameState 类的 `start_game` 方法如代码3所示。

```
TetrisGame/gamestate.py
26      def start_game(self):
27          self.stopped = False
28          self.set_timer(TIMER_INTERVAL)
29          self.timer_interval = TIMER_INTERVAL
30          self.piece = Piece(random.choice(PIECE_TYPES), self.screen,
self.wall)
```

代码3 `start_game` 方法

下面简要说明 `start_game` 方法的代码：

1. 第27行是把 stopped 属性设为 False，表示游戏处于进行中状态，即玩家指示开始后的游戏运行状态。
2. 第28行启动定时器。定时器的闹铃间隔时间是 `TIMER_INTERVAL`，常量对应的 1000ms。

3. 第29行是把记录闹铃间隔时间设置为 1000ms。玩家玩游戏过程中，得分越来越多，难度级别随之提高，这个间隔时间将相应地缩短。
4. 第30行是生成第一个方块。

我们在玩家按下 s 字母键后调用 `start_game` 方法。调用层次是：main.py -> main() -> `check_events()` -> `on_key_down()` -> `start_game()`。`on_key_down` 函数内调用 `start_game` 方法的代码如代码4中最后两行代码行所示。

```
1.  # TetrisGame/main.py
2.  64  def on_key_down(event, game_state):
3.  65      if event.key == pygame.K_DOWN:
4.  66          # print("向下方向键被按下")
5.  67          if game_state.piece:
6.  68              game_state.piece.move_down()
7.  69      elif event.key == pygame.K_UP:
8.  70          # print("向上方向键被按下")
9.  71          if game_state.piece:
10. 72              game_state.piece.turn()
11. 73      elif event.key == pygame.K_RIGHT:
12. 74          # print("向右方向键被按下")
13. 75          if game_state.piece:
14. 76              game_state.piece.move_right()
15. 77      elif event.key == pygame.K_LEFT:
16. 78          # print("向左方向键被按下")
17. 79          if game_state.piece:
18. 80              game_state.piece.move_left()
19. 81      elif event.key == pygame.K_f:
20. 82          if game_state.piece:
21. 83              game_state.piece.fall_down()
22. 84      elif event.key == pygame.K_s and game_state.stopped:
23. 85          game_state.start_game()
```

代码4 玩家按下 s 字母键后调用 `start_game` 方法（左侧数字是代码行号）

第84行代码使用了游戏状态对象的 stopped 属性。该属性为 True 的话，表示游戏处于停止状态，为 False 表示游戏处于进行中状态。它在以下情形下被修改：

1. 程序一开始，stopped 属性值设为 True。
2. 游戏开始后，stopped 属性值设为 False。
3. 游戏结束后，stopped 属性值设为 True。以后的步骤将会做这一赋值。

提示“按 s 字母键开始”

我们通过以下做法给出提示。在游戏区域中央绘制一幅图片，上标“按 s 字母键开始”这段文字。这幅图片是利用在线字体转换器网站生成的。下面讲解如何加载显示图片。

首先，把图片放到工程文件夹下。我们的做法是在 TetrisGame 文件夹下创建一个名叫 images 的子文件夹，专门用来放置图片文件。上述图片的文件名是 `press-s-newgame.png`，它位于 images 文件夹内，如图2所示。

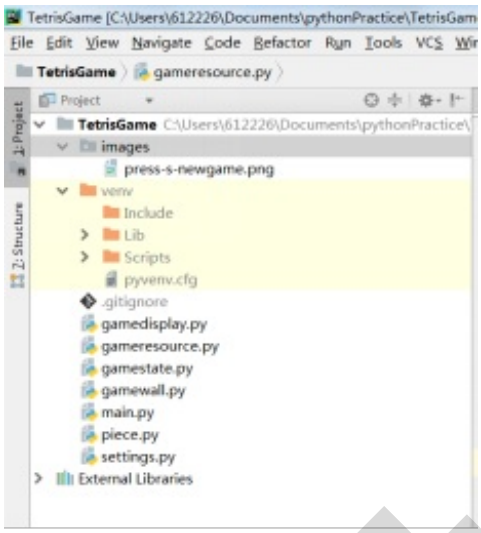

图2 images 文件夹放置图片文件

接着，定义 GameResource 类来负责加载图片。以后 GameResource 类将加载音频文件。图片和音频都属于资源。代码5示范了如何加载“按 s 字母键开始”图片（见图1）。GameResource 的构造函数中，`newgame_img` 属性被设值为 None。该属性用来引用“按 s 字母键开始”图片对象。设值为 None 表明在 GameResource 实例刚创建的时候，没有加载图片。

那什么时候加载图片呢？要用的时候，使用者调用 `load_newgame_img` 方法来获得图片对象。`load_newgame_img` 方法首先检测 `newgame_img` 属性是否为 None 值。若为 None 值（等同于 False），则 `not self.newgame_img` 为真，此时调用 `pygame.image.load()` 方法来加载图片，见第13行。有了第12代码给出的判断，图片不会重复加载。第13行还调用了 `convert_alpha()` 方法，作用是启用透明效果通道。注意，只有图片本身有透明属性值，才会有透明效果。具有透明效果的图片，一个像素点要用四个字节表示，三个字节是 RGB，第四个字节是 Alpha 通道值，即表示透明程度的整数值。

```
TetrisGame/gameresource.py
5   import pygame
6   class GameResource():
7       def __init__(self):
8           self.img_path = 'images/'
9           self.newgame_img = None
10
11      def load_newgame_img(self):
12          if not self.newgame_img:
13              self.newgame_img = pygame.image.load(self.img_path + "press-s-newgame.png").convert_alpha()
14          return self.newgame_img
```

代码5 加载“提示开始游戏图片”

我们要在 `GameDisplay.draw_game_area()` 函数内调用绘制图片的代码，如代码6中第31、32行所示。第31行代码用于保证游戏进行期间不会绘制图片。第32行代码调用了绘制图片的函数 `draw_start_prompt()`，传入图片资源对象 `game_resource`。这一图片资源对象由 `draw_game_area` 传入。`draw_start_prompt` 函数定义见第58~60行。`screen.blit` 函数显示图片，它的第一个参数是图片资源对象，第二个参数是图片显示的位置。

```
TetrisGame/gamedisplay.py
19  @staticmethod
20  def draw_game_area(screen, game_state, game_resource):
21      '''绘制游戏区域'''
22      for r in range(21):
23          pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT, GAME_AREA_TOP + r * CELL_WIDTH),
24                           (GAME_AREA_LEFT + GAME_AREA_WIDTH, GAME_AREA_TOP + r * CELL_WIDTH))
25      for c in range(11):
26          pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT + c * CELL_WIDTH, GAME_AREA_TOP),
27                           (GAME_AREA_LEFT + c * CELL_WIDTH, GAME_AREA_TOP + GAME_AREA_HEIGHT))
28
29      GameDisplay.draw_wall(game_state.wall)
30      GameDisplay.draw_score(screen, game_state.game_score)
31      if game_state.stopped:
32          GameDisplay.draw_start_prompt(screen, game_resource)
```

```
…   ……
57  @staticmethod
58  def draw_start_prompt(screen, game_resource):
59      start_tip_position = (GAME_AREA_LEFT + 2 * CELL_WIDTH, GAME_AREA_TOP + 10 * CELL_WIDTH)
60      screen.blit(game_resource.load_newgame_img(), start_tip_position)
```

代码6 加载“提示开始游戏图片”

主函数 main() 内要作出多处改动。第29行是新增代码行，创建了资源对象 `game_resource`。第45行增加一个参数，即 `game_resource`。新增的第47行使得程序启动后，不会在游戏区域绘制方块，直到玩家指示开始游戏之后才会绘制。

```
TetrisGame/main.py
16  def main():
…       ……
28      game_state = GameState(screen)
29      game_resource = GameResource()
30      #游戏主循环
31      while True:
…           ……
44          #绘制游戏区域网格线和墙体
45          GameDisplay.draw_game_area(screen, game_state, game_resource)
46          #绘制方块
47          if game_state.piece:
48              game_state.piece.paint()
49          #让最近绘制的屏幕可见
50          pygame.display.flip()
```

代码7 主函数 main() 内的修改

其他修改的地方

程序启动后到玩家按下 s 字母键开始玩游戏之前，玩家按下任何方向键将是无效的。为做到这一点，我们需要对响应键盘按键事件的代码作出修改，如代码8、9所示。代码8是修改前的代码，代码9是修改后的代码，修改之处请看代码中的注释说明。在游戏开始之前，`game_state.piece` 的值为 None，用在 if 语句内效果等同于 False，所以“if

`game_state.piece`”被判为不成立，故不会去操作方块对象。而游戏开始后，`game_state.piece` 的值不为 None，视为 True，所以“if `game_state.piece`”成立。

```
def on_key_down(event, piece):
    if event.key == pygame.K_DOWN:
        # print("向下方向键被按下")
        piece.move_down()
    elif event.key == pygame.K_UP:
        # print("向上方向键被按下")
        piece.turn()
    elif event.key == pygame.K_RIGHT:
        # print("向右方向键被按下")
        piece.move_right()
    elif event.key == pygame.K_LEFT:
        # print("向左方向键被按下")
        piece.move_left()
    elif event.key == pygame.K_f:
        piece.fall_down()
```

代码8 main.py 文件内的 `on_key_down` 函数修改之前

```
def on_key_down(event, game_state):
    if event.key == pygame.K_DOWN:
        # print("向下方向键被按下")
        if game_state.piece:   # 此处有修改
            game_state.piece.move_down()
    elif event.key == pygame.K_UP:
        # print("向上方向键被按下")
        if game_state.piece: # 此处有修改
            game_state.piece.turn()
    elif event.key == pygame.K_RIGHT:
        # print("向右方向键被按下")
        if game_state.piece: # 此处有修改
            game_state.piece.move_right()
    elif event.key == pygame.K_LEFT:
        # print("向左方向键被按下")
        if game_state.piece: # 此处有修改
            game_state.piece.move_left()
    elif event.key == pygame.K_f:
        if game_state.piece: # 此处有修改
```

```
20.                 game_state.piece.fall_down()
21.         elif event.key == pygame.K_s and game_state.stopped:
22.             game_state.start_game()
```

代码9 main.py 文件内的 `on_key_down` 函数修改之后

小结

本步骤实现玩家按 s 键后开始游戏的功能。做法是：

1. 修改 GameState 类的构造方法，使其内部不产生第一个方块，不设置定时器。
2. 在 GameState 类内定义 `start_game` 方法，执行该方法会产生第一个方块和设置定时器。
3. 在 main.py 文件的 `on_key_down()` 方法内调用 `start_game` 方法。调用是在检测到 s 键被按下之际发起的。
4. 提示“按 s 字母键开始”。在游戏未开始之际，显示“按 s 字母键开始”的提示。游戏开始后，不显示该提示。
5. 消除副作用。程序启动后到游戏开始之前，要忽略四个方向键按下事件。

完成本步骤全部功能的代码可从以下链接浏览或下载。

- Github

下一步骤中，我们会去实现暂停游戏的功能。以后我们会实现判别游戏结束的功能。