第05课：绘制方块

步骤目标

本文将实现的目标是：

1. 给出各种方块的表示方法；
2. 在游戏区域顶部绘制方块。

下一篇课程将实现通过键盘方向键来移动方块。

游戏区顶部绘制方块的效果如图1、图2所示。

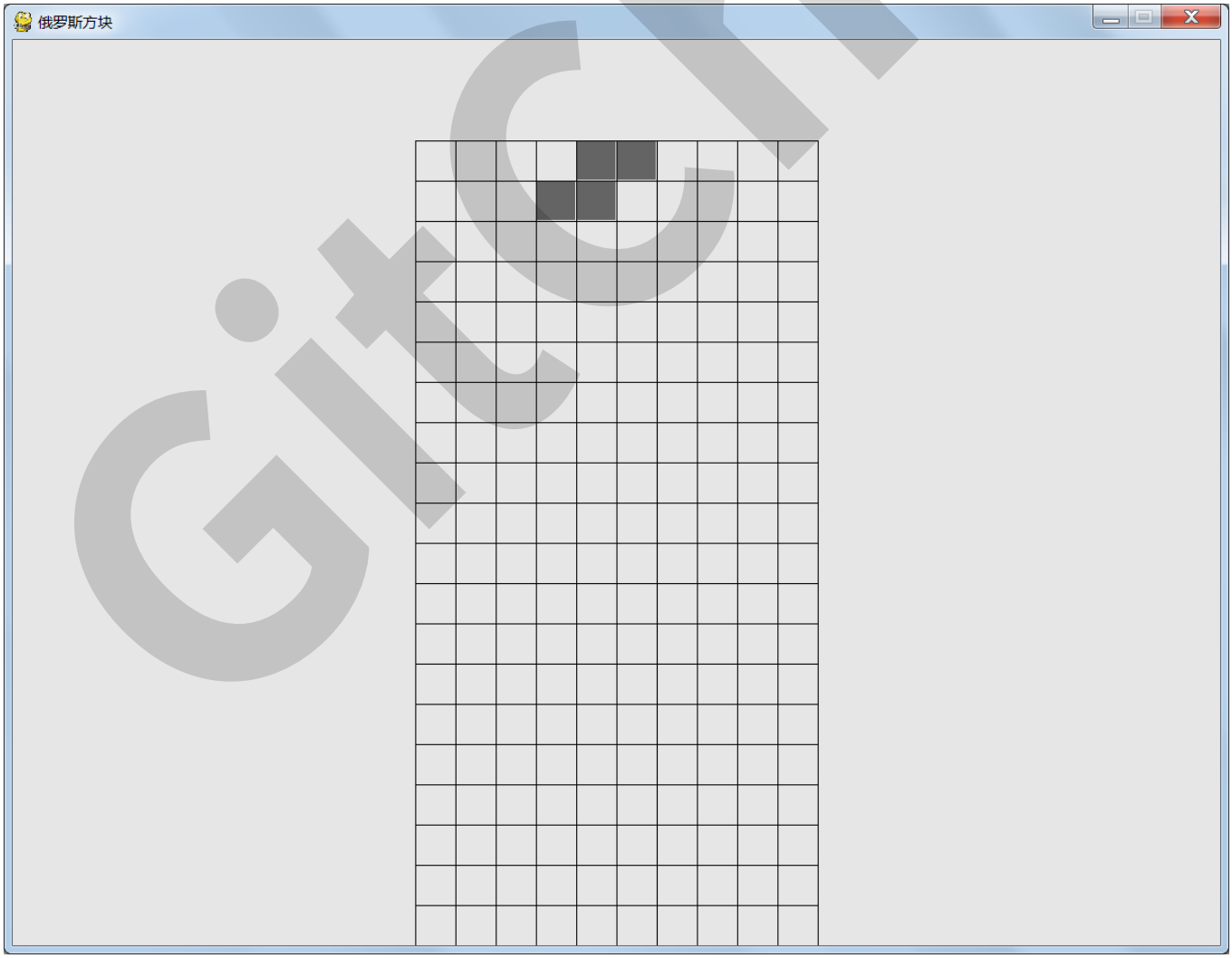

图1 游戏区顶部绘制 S 型方块

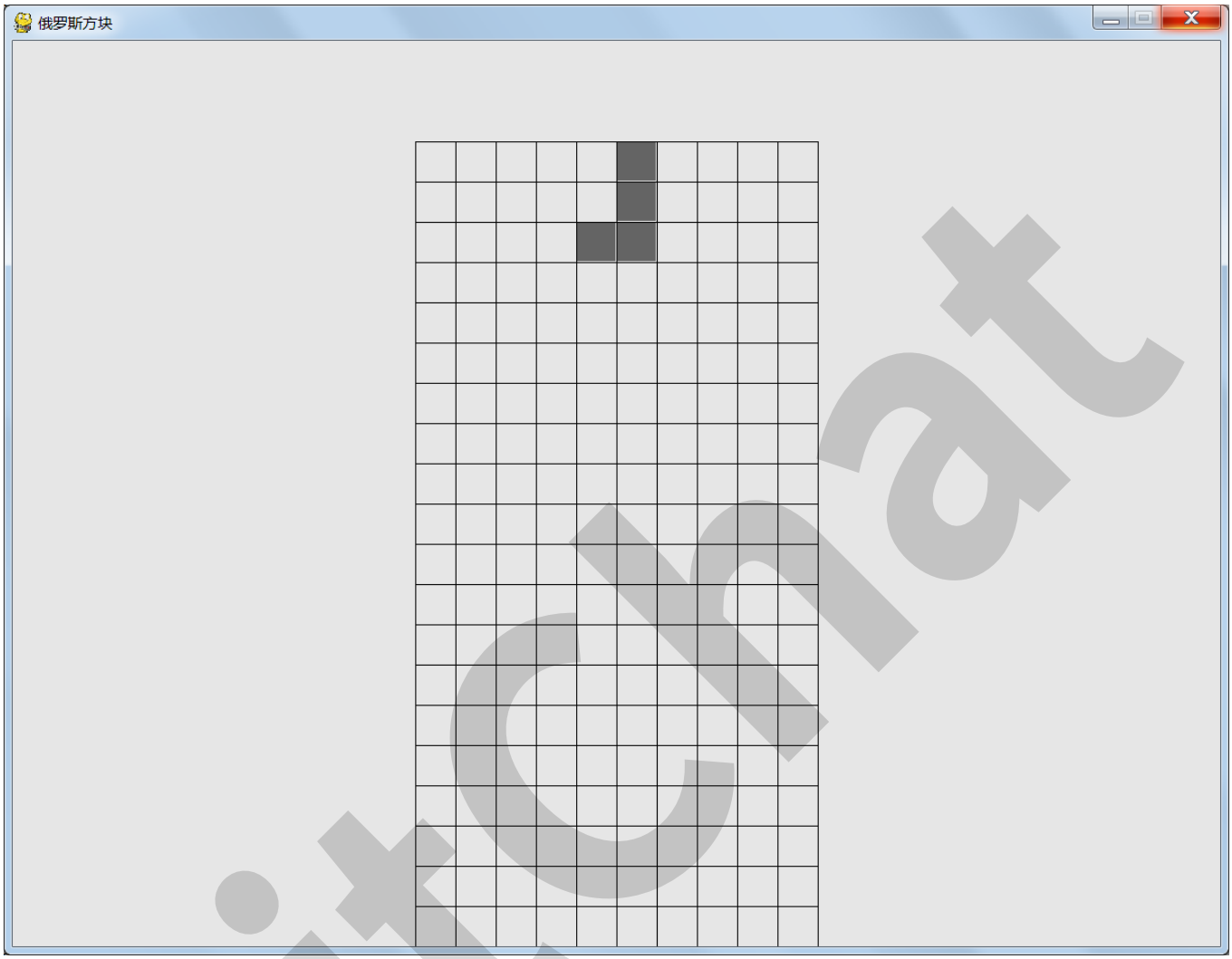

图2 游戏区顶部绘制 J 型方块

看着上面的效果图，你会觉得灰黑色的方块颜色不好看。你说对了，以后我们会把它的颜色设成彩色的。现在先关注形状好了。

方块的表示方法

俄罗斯方块游戏中一共有7种方块，各自的形状如图3所示，依次被叫做 I 型、J 型、L 型、O 型、S 型、T 型和 Z 型方块。

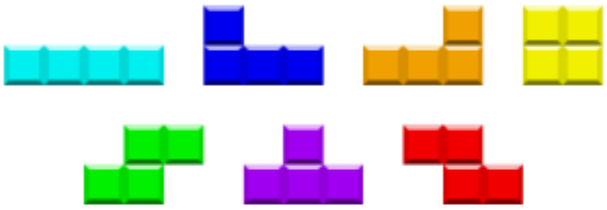

图3 俄罗斯方块的形状

在我们的游戏程序内部，该如何表示一个俄罗斯方块呢？我们知道，这一表示方法需要考虑到：

1. 方块有不同的形状。
2. 方块会移动。
3. 方块会翻转。翻转功能将在后面的课程中实现。本文先不予考虑。

我们采用的方法是：用一个方块对象（Piece 类的实例）来表示方块。

1. 对象的 shape 属性记住方块的形状。
2. 对象的 x、y 属性记住方块在游戏区域内的位置。

这样，Piece 类的数据属性定义如下面代码（代码1）所示。类的定义放置在 piece.py 文件内。

```
1.
2.  # TetrisGame/piece.py
3.  10. class Piece():
4.  11.     def __init__(self, shape, screen):
5.  12.         self.x = 3
6.  13.         self.y = 0
7.  14.         self.shape = shape     #值为一个字符。S,  I, J, L, O, T, Z等字母
    中的一个
8.  15.         self.screen = screen    #屏幕对象。绘制方块的方法会用到。
```

代码1 Piece 类的属性定义（左侧数字为 piece.py 文件内的行号）

记住方块的形状

玩家在玩俄罗斯方块游戏过程中，游戏区域出现一个方块。我们把这个方块叫做当前方块。下面来讲我们的游戏程序是如何记住当前方块的形状的。

上篇文章的“作业”一节中，我们把“如何记住方块的形状”这一问题分成了三个子问题。

1. 如何记住方块的类型？
2. 如何根据方块的类型得出方块的形状？
3. 如何在游戏区域内定位方块？

以下依次回答上述三个子问题。

1.记住方块的类型。

做法是用7个字母 S、I、J、L、O、T、Z 来表示7种方块；在生成方块时指定它的类型，接着在方块内部用 shape 属性记住类型值。在本文这一步骤中，main.py 的 main 函数内，我们用以下语句生成方块对象，用 piece变量来引用它。

```
piece = Piece('S', screen)
```

这里，`'S'` 代表的是方块的类型；screen 是绘制方块时会用到的屏幕对象。我们显然可以把 S 替换为I、J、L、O、T、Z。下面我们以 S 型方块为例讲解做法。

从代码1中 Piece 类的构造方法可知，piece 变量的 shape 属性值将设为 `'S'`。也就是说，我们是用方块对象的 shape 属性来记住方块的类型。它的值是一个字母。

2.根据方块类型得出方块形状。

定义一个字典 PIECES，以表示方块类型的字母为关键字，表示方块形状的矩阵为值。字典的定义如下面代码2所示。以 S 型方块为例，根据 `'S'` 这一类型，可以查到 S 型方块的形状。S 型方块的形状用 `S_SHAPE_TEMPLATE` 常量来定义。该常量的值是表示形状的矩阵。

```
# TetrisGame/settings.py
83. PIECES = {'S': S_SHAPE_TEMPLATE,
84.           'Z': Z_SHAPE_TEMPLATE,
85.           'J': J_SHAPE_TEMPLATE,
86.           'L': L_SHAPE_TEMPLATE,
87.           'I': I_SHAPE_TEMPLATE,
88.           'O': O_SHAPE_TEMPLATE,
89.           'T': T_SHAPE_TEMPLATE}
```

代码2 字典 PIECES，以方块类型为关键字

`S_SHAPE_TEMPLATE` 表示 S 型方块形状的矩阵，是一个3x4字符矩阵（也可以定义成3x3的）。其中，`'.'` 表示此位置为空；`'O'` 字母（不是数字 0，你可以使用其他字符来表示不为空）表示此位置被占据，不为空。我们把表示方块形状的矩阵叫做方块的形状矩阵，如代码3所示。

```
1.
2.  # TetrisGame/settings.py
3.  18. S_SHAPE_TEMPLATE = [  '.OO.',
4.  19.                       'OO..',
5.  20.                       '....'  ]
```

代码3 S 型方块的形状矩阵

3.在游戏区域内定位方块。

上述3x4的形状矩阵将映射到游戏区域中的一块子区域，即3行4列的单元格组成的子区域。在这个子区域中，不为空的单元格内绘制小方块（上一篇文章讲了如何绘制），为空的单元格不绘制小方块。

这个3x4子区域的具体位置是由方块对象的 x、y 属性决定的。x、y 的值是正整数，x 的范围是 [0, 9]，y 的范围是 [0, 19]。假设一个 S 型方块的坐标值是：x=3、y=0，那么方块的定位点是第1行第4列。这意味着，对于 S 型方块，将在以下四个单元格上绘制小方块：第1行第5列，第1行第6列，第2行第4列，第2行第5列。绘图效果见图1。

以上，给出了方块的表示方法。移动方块和翻转方块的功能实现在后面的课程中详细阐述。

绘制方块功能实现

添加 settings.py 文件

settings.py 文件的作用是定义程序使用的常量。它包含三个部分：

1. 到上篇文章为止，我们定义的常量有 `SCREEN_WIDTH`， `BG_COLOR` 等。前文，这些常量在 main.py 中定义。现在，把它们移到 settings.py 文件内。我们需要在 main.py 文件内导入 settings 模块定义的常量，完成导入常量的语句是：

```
1.  from settings import *
```

2. 7种方块的形状矩阵。代码3是一个具体的例子。你需要理解方块的形状矩阵是什么，上一篇做出了详细解释。在理解的基础上，你可以访问文末给出的链接，打开 settings.py 文件，复制定义7种方块的形状矩阵的代码。你无须自己从头敲入定义各种方块的形状矩阵的语句。
3. PIECES 字典，即代码2给出的代码。

定义方块类

方块类 Piece 的完整定义如代码4所示。

```
1.  # TetrisGame/piece.py
2.  from settings import *
3.  import pygame
4.
5.  class Piece():
6.      def __init__(self, shape, screen):
7.          self.x = 3
8.          self.y = 0
9.          self.shape = shape
10.         self.screen = screen
11.
12.     def paint(self):
13.         shape_template = PIECES[self.shape]
14.
15.         for r in range(len(shape_template)):
16.             for c in range(len(shape_template[0])):
17.                 if shape_template[r][c] == 'O':
18.                     self.draw_cell(self.x + c, self.y + r)
19.
20.     def draw_cell(self, x, y):
21.         cell_position = (x * CELL_WIDTH + GAME_AREA_LEFT + 1,
22.                          y * CELL_WIDTH + GAME_AREA_TOP + 1)
23.         cell_width_height = (CELL_WIDTH - 2, CELL_WIDTH - 2)
24.         cell_rect = pygame.Rect(cell_position, cell_width_height)
25.         pygame.draw.rect(self.screen, CELL_COLOR, cell_rect)
```

代码4 Piece 类的定义

在生成方块对象的时候，将自动执行上面第11~16行定义的构造方法。

1. 第11、12行设定方块的初始位置。self.x=3 设定横坐标的值，即第4列，self.y=0 设定纵坐标的值，即第一行。这使得方块最初位于游戏区域顶部居中的位置。方块的位置实际上就是形状矩阵的定位点。
2. 方块对象的 shape 属性记住方块的类型，值为一个字母。

> 知识点：
> Python 语言的类定义内部，赋值给属性 t 或者访问属性 t 的值，都要写作 self.t。调用方法 m，要写作 self.m。

代码4中，paint() 方法实现方块绘制形状功能。下面作简要说明：

1. 第17行中，PIECES 是字典。self.shape 属性记住了方块的类型（是 `'S'`，`'L'` 还是其他）。该行代码的作用是令 `shape_template` 变量记住形状矩阵。

2. 第19~22行代码的作用是：先行后列地遍历形状矩阵，如果 r 行 c 列不为空（这里的 r 和 c 是形状矩阵内部的行号和列号），则在游戏区域的 `self.y+r` 行 `self.x+c` 列绘制小方块。第22行调用 `draw_cell` 方法干了这件事。`self.y` 和 `self.x` 正是方块的定位点。

3. 第24~29行定义的 `draw_cell` 方法与上文所描述的 `draw_cell` 函数（main.py 内定义了它）功能是一样的，都是为单元格填充颜色。不过，两者的参数不一样。这里，`draw_cell` 方法的第一个参数是单元格的列号，第二个参数是单元格的行号。这两个参数确定了单元格在游戏区域内的网格坐标，方法内部需要把网格坐标转换为窗口像素坐标。列号 x（范围是 [0, 9]）转换为窗口像素坐标的公式是：

> x * CELL_WIDTH + GAME_AREA_LEFT

其中，`CELL_WIDTH` 是单元格宽度，`GAME_AREA_LEFT` 是游戏区离窗口左边界的距离。

4. `draw_cell()` 方法的第25~26行代码中，横坐标和纵坐标为什么要额外加1呢？这是为了使单元格内填色区域离单元格边界有1个像素的距离。给 `cell_width_height` 赋值的时候，减去2也是为了同样的目的。这使得相邻小方块之间有2个像素的间隔。你可以去掉“加1”和“减2”看看显示效果会有什么变化。

调用绘制方块的方法

要在 main 函数的主循环（也即程序主循环）内调用绘制方块的方法，如下代码5所示。第7行从 settings 模块导入常量。第8行从 piece.py 导入 Piece 类。第20行代码生成了 S 型方块。如果你想测试其他类型的方块，比如 J 型方块，那么把 `'S'` 改为 `'J'` 即可（注意：是大写的字母，而不是小写的）。第34行调用了 piece 对象的 paint 方法，达成在游戏区顶部中央绘制方块的效果。

```
# TetrisGame/main.py
5   import sys
6   import pygame
7   from settings import *
8   from piece import Piece
9
10  def main():
……      #相比以前，代码未改变。
19      #生成方块对象
20      piece = Piece('S', screen)
21
22      #游戏主循环
23      while True:
…           ……  #相比以前，代码未改变。
31          #绘制直线
32          draw_game_area(screen)
33          #绘制方块
34          piece.paint()
35          #让最近绘制的屏幕可见
36          pygame.display.flip()
……      #相比以前，代码未改变。
```

代码5 调用绘制方块方法的代码

小结

绘制方块的做法是：

1. 用字母 S、I、J、L、O、T、Z 代表方块的类型，生成方块时指定方块的类型。
2. 用字典 PIECES，根据方块类型得出方块的形状矩阵。字典 PIECES 的定义代码2。
3. 用方块对象的 x、y 属性记住方块在游戏区域中的定位点。
4. 绘制方块。假设定位点是 x=3、y=0，以 S 型方块的3x4形状矩阵为例，将在 [x=4, y=0]、[x=5, y=0]、[x=3, y=1]、 [x=4, y=1] 这四个单元格内绘制小方块。这里，行号、

列号是矩阵的行下标、列下标，从0开始编号。

5. 在 main.py 内生成了方块对象，并在主循环内调用绘制方块的方法，见代码5。

实现上述全部功能的代码请访问 Github。

强烈建议你自己多尝试，多试错，真正吃透代码，为后续步骤打下坚实基础。