第03课：绘制游戏区域

步骤目标

本文将实现的目标是：在游戏窗口内绘制游戏区，如图1所示。游戏区是20x10个单元格组成的。游戏区顶部、左侧和右侧都留有空白。

该目标分成3小步实现：

1.在游戏窗口内绘制一条直线。这一小步的目的是让我们逐步熟悉 Pygame 的用法。
2.在游戏窗口内绘制游戏区的外边界。这一小步会有讲解，并提供源代码。
3.在游戏区域内绘制单元格分割线。需要你自己来完成这一小步。参照第2小步，难度不大。

最后将提供源代码，供你借鉴。记住，一定要自己先思考；最后要吃透。

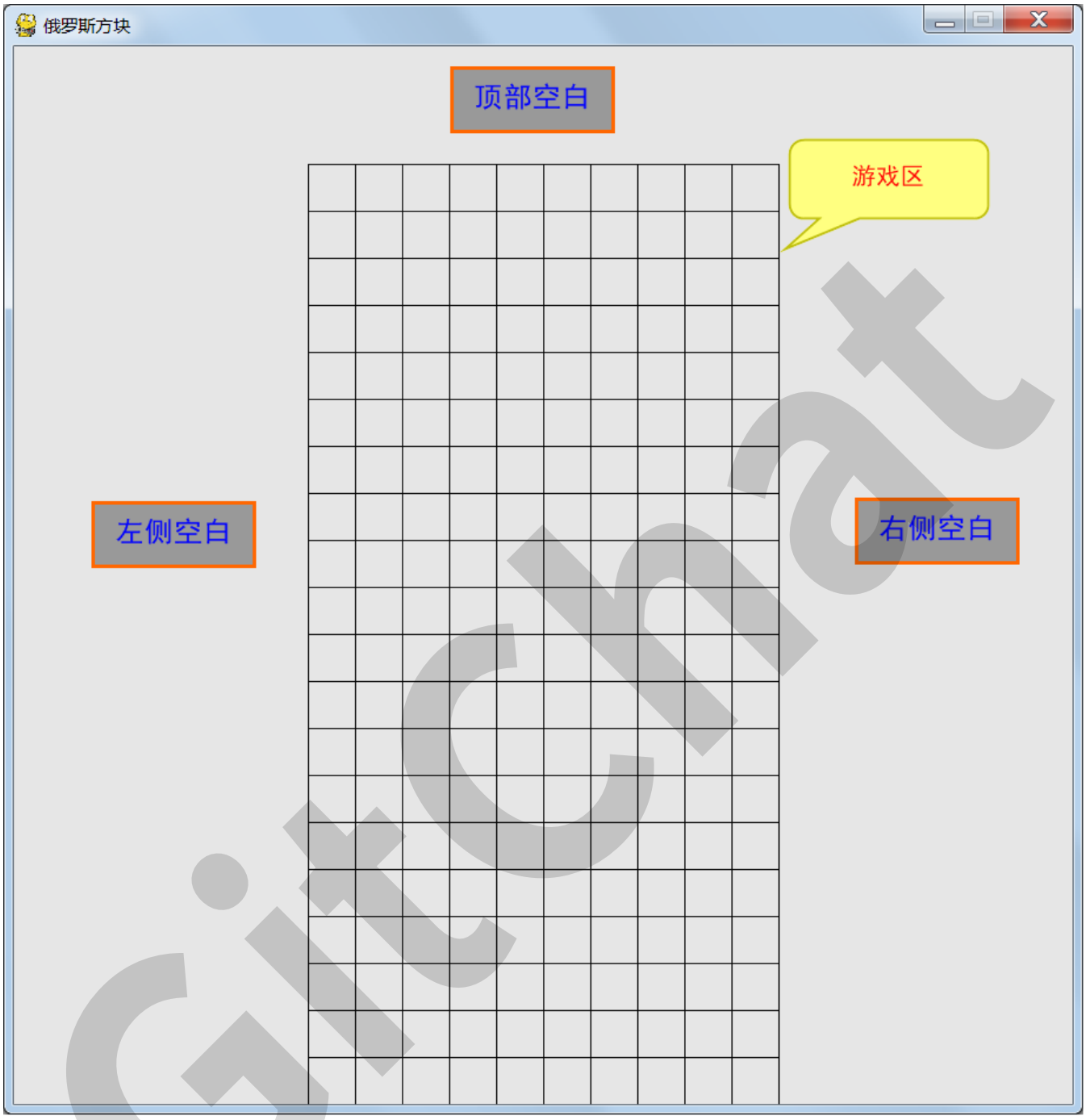

图1 游戏窗口中间区域，20*10个单元格组成游戏区

第1小步：绘制直线

绘制直线功能由 `draw_game_area()` 函数来完成。调用该函数的代码要加入到游戏主循环中，如下面代码第28、、29行代码所示。`draw_game_area()` 函数定义如 main.py 文件的34~37 行所示。

注意：调用 `draw_game_area()` 函数的语句必须放在 `screen.fill(bg_color)` 之后。这就像在墙上画直线须在给墙刷完油漆之后进行。顺序反过来的话，就会看不到直线。

下面是绘制一条直线的代码片段：

```
#TetrisGame/main.py
…    ……    #与前一版本相比，1~17行代码无变化。
18.      #游戏主循环
19.      while True:
20.           #监视键盘和鼠标事件
21.           for event in pygame.event.get():
22.              if event.type == pygame.QUIT:  #关闭窗口的事件
23.                   sys.exit()    #退出程序
24.
25.          # 填充屏幕背景色
26.          screen.fill(bg_color)
27.
28.          #绘制直线
29.          draw_game_area(screen)
30.           #让最近绘制的屏幕可见
31.           pygame.display.flip()
32.
33.
34. def draw_game_area(screen):
35.     '''绘制游戏区域'''
36.     pygame.draw.line(screen, (0, 0, 0), (100, 100), (200, 200))
37. #绘制一条线。第二个参数(0, 0, 0)决定颜色是黑色。起点坐标是(100, 100)，终点是
(200, 200)
38.
39. if __name__ == '__main__':
40.     main()
```

借助上述代码的注释，我们能够知晓 draw 模块的 line 函数的用法。draw 模块隶属于 Pygame 模块。欲了解更多用法，参阅以下两份文档。

1. Pygame 官方文档
2. Pygame 学习笔记

> 知识点：窗口的坐标系
>
> 窗口内部，左上角是坐标系原点（0, 0），如图2所示。水平向右，x 坐标增大；每经过

> 一个像素点，坐标值增1。垂直向下，y 坐标增大；每经过一个像素点，坐标值增1。这样，窗口上的每一个像素点都有一个坐标，由一对整数组成，比如（100, 200）。如果你对像素点这个概念有疑惑，请上网搜索相关解释。

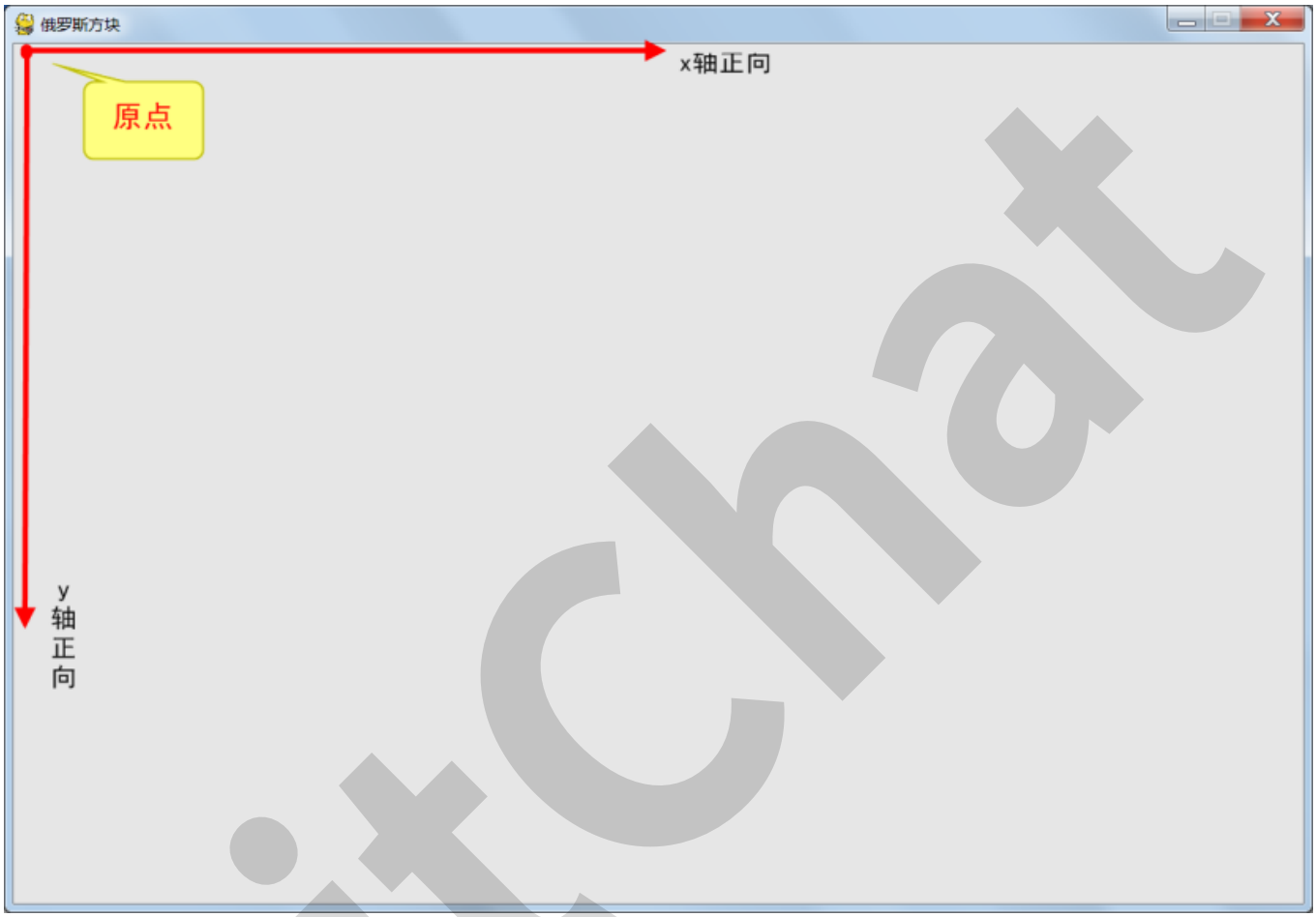

图2 窗口坐标系统

第2小步：绘制游戏区边界线

第2小步完成后，显示效果如图3所示。

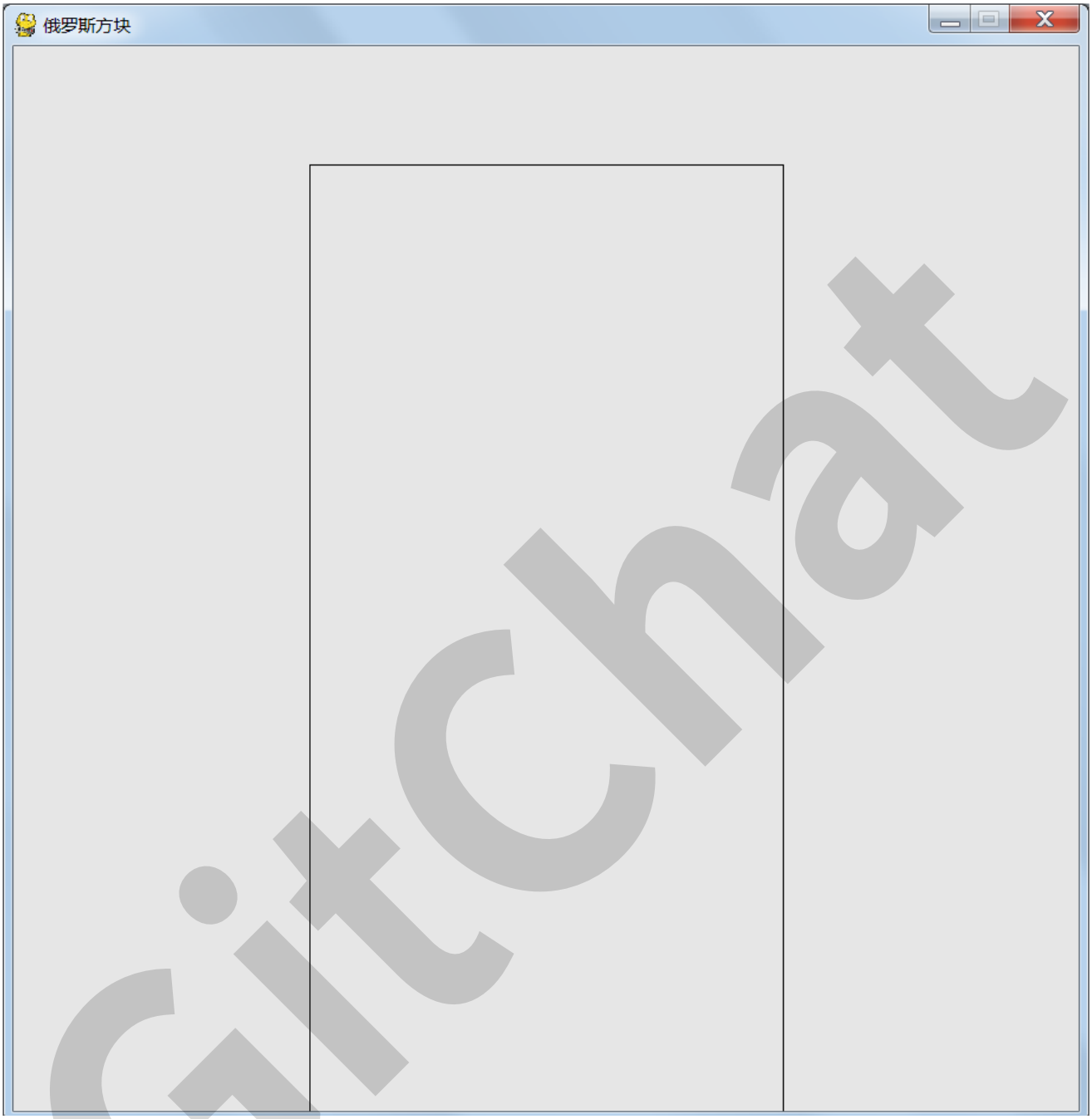

图3 游戏区边界线

图3中，底部的边界线与窗口边界重叠，所以不容易看出来。

游戏区的边界由4条线段组成。水平的两条，即顶部和底部；垂直的两条，即左侧和右侧。绘制边界线要做的就是绘制这4条线。只要修改 `draw_game_area` 函数的定义就能实现绘制边界线的功能。代码如下：

```
1.  def draw_game_area(screen):
```

```
    #绘制顶部边界线
pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT, GAME_AREA_TOP),
                 (GAME_AREA_LEFT + GAME_AREA_WIDTH, GAME_AREA_TOP) )
#绘制底部边界线
pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT, GAME_AREA_TOP + 20 * CELL_WIDTH),
                 (GAME_AREA_LEFT + GAME_AREA_WIDTH, GAME_AREA_TOP + 20 * CELL_WIDTH) )
#绘制左侧边界线
pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT, GAME_AREA_TOP),
                 (GAME_AREA_LEFT, GAME_AREA_TOP + 20 * CELL_WIDTH) )
#绘制右侧边界线
pygame.draw.line(screen, EDGE_COLOR, (GAME_AREA_LEFT + 10 * CELL_WIDTH, GAME_AREA_TOP),
                 (GAME_AREA_LEFT + 10 * CELL_WIDTH, GAME_AREA_TOP + 20 * CELL_WIDTH) )
```

上述代码中，由大写字母和下划线组成的变量用作常量，定义如下。这些代码行放在 main 函数定义之前，导入模块的语句（第6行）之后。以后会单独用一个文件来存储这些常量。

```
SCREEN_WIDTH = 1200     #窗口宽度
SCREEN_HEIGHT = 900     #窗口高度
CELL_WIDTH = 40          #方块在20x10个单元格组成的游戏区内移动。每个单元格的边长是40个像素。
GAME_AREA_WIDTH = CELL_WIDTH * 10        #一行10个单元格
GAME_AREA_HEIGHT = CELL_WIDTH * 20       #一共20行
GAME_AREA_LEFT = (SCREEN_WIDTH - GAME_AREA_WIDTH) // 2       #游戏区左侧的空白区的宽度
GAME_AREA_TOP = SCREEN_HEIGHT - GAME_AREA_HEIGHT           #游戏区顶部的空白区的宽度
EDGE_COLOR = (0, 0, 0)            #游戏区单元格边界线的颜色。今后，网格线会被去除。
CELL_COLOR = (100, 100, 100)     #单元格填充色。
BG_COLOR = (230, 230, 230)       #窗口背景色
```

main 函数内，创建屏幕对象处的代码修改为：

```
#创建屏幕对象
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT) )  #分辨率是1200*900
```

main函数内，设定屏幕背景色的代码改为：

```
#屏幕背景色
bg_color = BG_COLOR
```

完成绘制游戏区域边界线功能的代码已经上传至 Github，你可以下载或浏览 main.py 文件。

第3小步：绘制游戏区网格线

第3小步完成后，效果图如图4所示。这一小步的功能交由你来实现。建议你：

1. 借鉴第2小步的做法。
2. 采用循环来减少调用 draw.line() 的次数。
3. 使用第2小步定义好的常量。

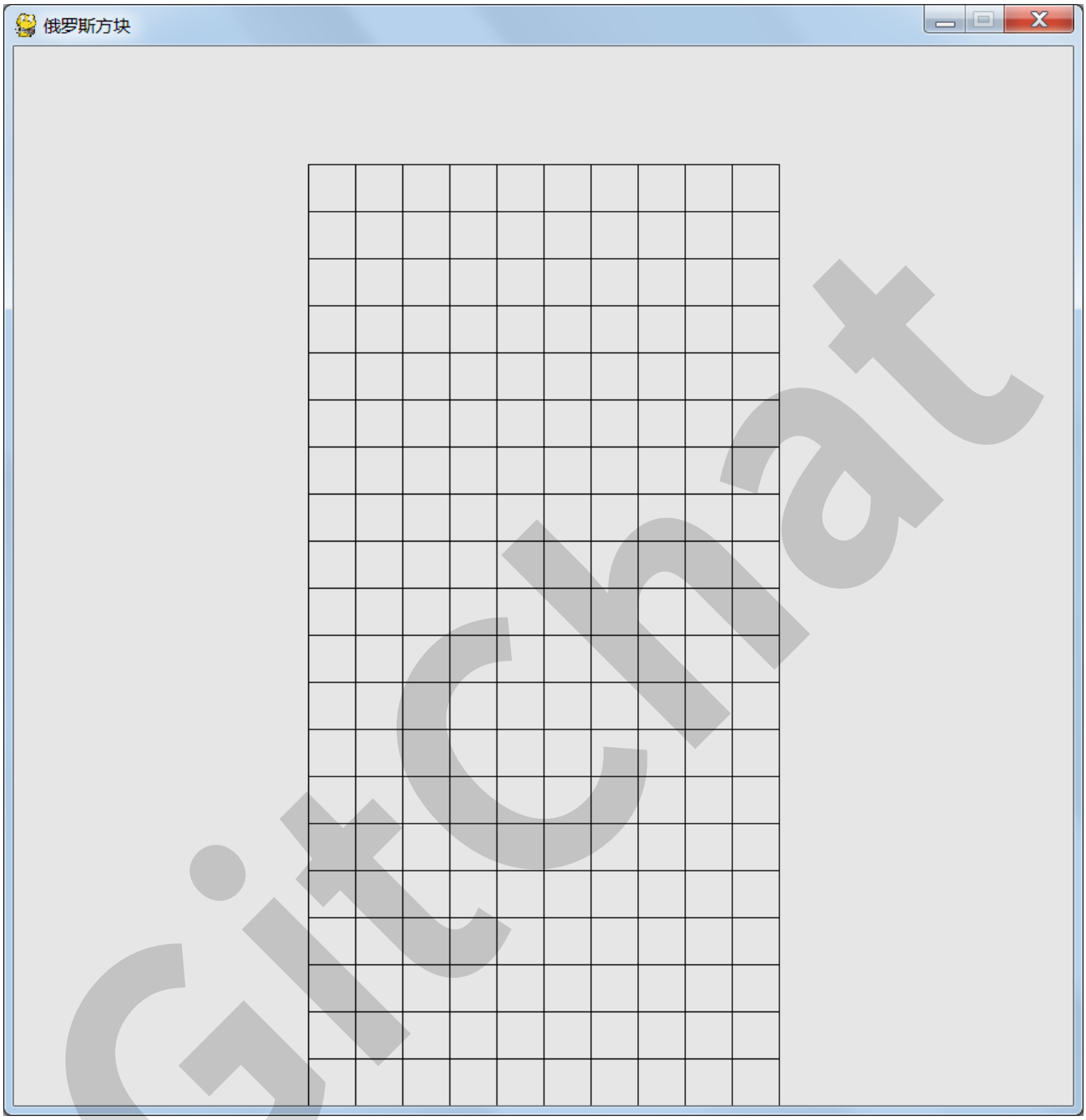

图4 绘制游戏区网格线

小结

本文实现了绘制游戏区的功能。游戏区是一个20x10的网格。方块将限定在游戏区内移动。

还记得上一篇讲述的程序主循环吧？每循环一次，将为整个窗口填充一次背景色，接着在窗口内绘制游戏区的网格线。要注意绘制网格线须在填充背景色之后。

我们有必要熟悉窗口坐标系。左上角是原点(0, 0)，沿水平向右，横坐标 x 递增，每过一个像素点就增1；沿垂直向下，纵坐标 y 递增，每过一个像素点就增1。

你可以到 Github 上下载我发布的、完成本步骤全部功能的代码文件。链接是：
。记住，仅供借鉴，一定要自己先思考，先做尝试；最后要吃透。

在后面的课程中，我们将去除网格线，只留游戏区的边界线。