游戏分享：手把手教你用 Python 编写战斗机游戏

2018.1.23

游戏制作

今天我们要分享的是战斗机游戏！在这个游戏中，飞机（游戏主角）需要躲避迎面飞来的炮弹（敌人）。

下面让我们一起来看一下如何编写这个小游戏吧！

初始设置

首先，创建一个新的.py 文件然后置入下面的代码:

```
import pygame
from pygame.locals import *
pygame.init()
```

和其他所有 python 程序一样，我们要首先导入（import）我们所要使用的模块（module）。在今天这个例子里，我们要导入 pygame 以及 pygame.locals，以用于后面的内容。最后一行 pygame.init()将所有的 PyGame 模块初始化，在我们正式开始写 PyGame 的“正文内容”之前，都要先加上这一行。

屏幕对象

首先，我们需要为所有的物体创建一个画布窗口，我们把这个变量叫做 **screen**。为了创建这个变量，我们需要调用 pygame.display 下的 set_mode（）方法（method），并向 set_mode()传递一个元组（tuple）来表示窗口的高和宽（在这里我们设置一个高 800 宽 600 的窗口）。

```
import pygame
```

```
from pygame.locals import *
pygame.init()

screen = pygame.display.set_mode((800, 600))
```

现在，如果你运行程序，你将会看到一个窗口突然出现，并在我们退出程序时马上消失。看来我们已经成功创建了画布！在下一部分，我们将加入游戏的 main 循环，这样我们就能确保只有我们输入正确的退出命令时，程序才会退出。

游戏循环

游戏的主循环内包含了所有要发生的事件。当我们玩游戏时，这段程序一直在运行。它能够不断更新游戏的状态，渲染游戏屏幕效果，并且收集玩家输入的指令。除了让主循环在游戏进行中不断执行，我们也需要让循环在我们退出程序时停止。因而，我们需要引入一些相应的指令，让程序能够在这些指令输入（input）时作出相应的反应。所有输入的指令（以及之后一些其他的事件）都会进入 PyGame 的事件队列中，我们可以通过 pygame.event.get()来得到这些事件。这个命令会返回队列中的事件列表，在游戏中，程序将对这些事件进行循环，并且根据事件的类别作出相应的反应。在下一段代码中，我们要处理的是 KEYDOWN 和 QUIT 事件:

```
# 用于让主循环不断运行的变量
running = True

# 主循环
while running:
# 针对事件队列的 for 循环
    for event in pygame.event.get():
# 检测是否有 KEYDOWN 事件，KEYDOWN 是一个在 pygame.locals 中定义了的事件常量，这个库我们之前导入过
        if event.type == KEYDOWN:
# 如果按了 ESC 键，把 running 的值设为 False，并退出主循环
            if event.key == K_ESCAPE:
                running = False
# 检测是否有 QUIT 事件，如果有的话，把 running 的值设为 Flase
        elif event.type == QUIT:
```

```
        running = False
```

把这几行代码加到之前的代码之后，然后执行程序。你应该会看到一个空白的窗口。这个窗口只会在你按下 ESC 键以激活 QUIT 事件的时候才会关闭。

Surface 和 Rect 对象

Surface 和 **Rect** 是 PyGame 里的基础模块。我们可以把 surface 理解为一张白纸，我们可以在这张白纸上画任何想画的东西。我们的 **screen** 变量也是一个“surface“，因为它也是用于承载图像的。Rect 代表 surface 内的一个矩形区域。

现在，让我们创建一个 50 像素 x 50 像素的 **surface**，用 **surf** 变量来代表这个 surface，然后给这个 surface 对象填充颜色。由于窗口的默认背景色是黑色，我们将把这个 surface 对象填充为白色，这样在颜色对比之下就更明显一些。然后我们要对这个 surface 调用 get_rect()方法，然后得到这个 surface 代表的矩形区域以及 surface 的 x、y 坐标。

```
# 创建 surface 对象，将长和宽以元组的形式传递
surf = pygame.Surface((50, 50))
# 用白色填充 surface 对象，让它与背景有所区别
surf.fill((255, 255, 255))
rect = surf.get_rect()
```

Blit 和 Flip

要使我们创建的 surface 显示在屏幕上，仅仅创建它是不够的，我们还需要用 **blit** 把这个 surface 放到另一个 surface（screen）上。在 PyGame 中，我们可以把 blit 理解为“画”。我们可以用 blit 把一个 surface 画在另一个 surface 上，在这里，我们的另一个 surface 就是我们的大画布 **screen**。下面的代码显示如何将 surface 变量 **surf** 画在 screen 上：

```
# 这行的意思是“在画布 screen 横坐标 400 纵坐标 300 的地方画一个 surf”
```

```
screen.blit(surf，(400，300))
pygame.display.flip()
```

blit（）带有两个参数：一个是我们要画上去的 surface，另一个是我们要画的位置的坐标。在这里，（400，300）代表了 **screen** 画布的中心，但是当你真正执行程序的时候你会发现，**surf** 并没有出现在屏幕的正中央。这是因为 blit（）都是从左上角开始画图的，（400，300）代表了 surface 的左上角。

需要注意的是，我们在 blit 后调用了 pygame.display.flip()。Flip 会在每一次循环时刷新屏幕，从而显示在上一次循环后屏幕更新后的状态。如果不调用 flip（）的话，这些更新是不会显示的。

（未完待续）