目录

【Pygame 第 1 课】 hello pygame.....1
【Pygame 第 2 课】 游戏的本质.....4
【Pygame 第 3 课】 游戏中的事件.....5
【Pygame 第 4 课】 获取鼠标位置.....7
【Pygame 第 5 课】 游戏中的运动.....9
【Pygame 第 6 课】 面向对象的游戏设计.....14
【Pygame 第 7 课】 多变的宿敌.....17
【Pygame 第 8 课】 火力全开.....21
【Pygame 第 9 课】 一大波飞机.....25
【Pygame 第 10 课】 命中目标.....27
【Pygame 第 11 课】 GAME OVER.....29
【Pygame 第 12 课】 屡败屡战.....31

【Pygame 第 1 课】 hello pygame

我们已经把 python 的基本内容讲得差不多了，所以从今天起，尝试一下新的方面：pygame -- 用 python 来写游戏。

pygame 是一个 python 的游戏库，借助它，我们可以用 python 写一些小游戏。虽然你想用它写出一个魔兽世界那样的游戏是不大可能的，但它的确适合 python 学习者入手游戏开发。

安装 pygame

python 标准库里是没有包含 pygame 的，所以我们需要去下载安装它。去 www.pygame.org 上的 downloads 找到对应你 python 版本的安装包下载并安装。Mac 用户要注意一下，可能你 mac 里默认的 python 版本无法于 pygame 兼容，需要去 puthon.org 重新下载安装 python2.7。

安装完之后，可以在你的 python shell 里验证一下：

```
>>>import pygame
>>>pygame.ver
'1.9.1release'
```

pygame 的 hello world

照例，我们要用一个 hello world 程序来开始我们的学习。

在写代码之前，先去找一张图片，确定图片的长宽值。我们要用它来做为背景图片。

```
# -*- coding: utf-8 -*-
import pygame
#导入 pygame 库
from sys import exit
#向 sys 模块借一个 exit 函数用来退出程序
pygame.init()
#初始化 pygame,为使用硬件做准备
```

```
8. screen = pygame.display.set_mode((600, 170), 0, 32)
9. #创建了一个窗口,窗口大小和背景图片大小一样
10. pygame.display.set_caption("Hello, World!")
11. #设置窗口标题
12. background = pygame.image.load('bg.jpg').convert()
13. #加载并转换图像
14. while True:
15. #游戏主循环
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            #接收到退出事件后退出程序
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    #将背景图画上去
    pygame.display.update()
    #刷新一下画面
```

运行代码。幸运的话，你会看到一个有图片背景的窗口，不再是黑乎乎或者白花花的控制台了。以后，我们的游戏就会出现在这个窗口里。

hello.py

```
# -*- coding: utf-8 -*-
import pygame
#导入pygame库
from sys import exit
#向sys模块借一个exit函数用来退出程序
pygame.init()
#初始化pygame,为使用硬件做准备
screen = pygame.display.set_mode((600, 170), 0, 32)
#创建了一个窗口,窗口大小和背景图片大小一样
pygame.display.set_caption("Hello, World!")
#设置窗口标题
background = pygame.image.load('bg.jpg').convert()
#加载并转换图像
while True:
#游戏主循环
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            #接收到退出事件后退出程序
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    #将背景图画上去
    pygame.display.update()
    #刷新一下画面
```

Python ▾ Tab Width: 8 ▾ Ln 21, Col 35 INS

```
crossin@nono:~/Desktop$ python hello.py
there is no soundcard
```

【Pygame 第 2 课】 游戏的本质

你小时候有没有玩过这样一种玩具：一块硬纸，一面画着一只鸟，一面画着一个笼子。硬纸下粘上一根细棒。用手来回转动细棒，让硬纸的两面快速交替出现，就会看见鸟被关在了笼子里。

这种现象被称为视觉暂留，又称余晖效应。人眼的性质使得光信号在进入之后，会保持一小段时间，这段时间大约是0.1~0.4秒。电影、动画便是利用这种现象得以实现，把一幅幅静态画面快速连续播放，形成看上去连续的活动画面。游戏也不例外。

回顾一下昨天的代码，你会注意到有一个while True的循环，注释为“游戏主循环”。这就是游戏的主体部分。每次循环都相当于是一张静态的画面，程序一直运行，画面就有了动态的效果。这个程序中还看不出，因为始终只有一张固定不动的背景图片。

与动画不同，游戏中不仅要把一幅幅画面播放出来，还需要处理玩家的操作与游戏中内容的交互。所以在这个while循环中，还要去接收玩家的输入，以及处理游戏中的各种逻辑判断、运动、碰撞等等。

在我们程序的主循环里，做了对退出事件的响应：

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
    #接收到退出事件后退出程序
        pygame.quit()
        exit()
```

然后把图像绘制到窗口中：

```
1. screen.blit(background, (0,0))
```

最后，把整个窗口画面更新：

```
1. pygame.display.update()
```

如果你看过泥土动画,那么就可以把整个游戏过程想象成拍摄泥土动画的过程:每一次,screen.blit()相当于去把人偶、布景移动一点点位置，pygame.display.update()则是按下快门拍下一帧新的画面。而 if event.type == pygame.QUIT:就是在判断，导演是不是喊停收工了。这一切都由计算机在很短的时间内处理，以至于玩家感觉是连贯的。

尽管我们只是从简单的游戏做起，但在本质上，绝大多数的游戏都是相通的。

有些大型3D游戏需要在一次循环内做很多事情，需要进行复杂的物理运算、计算光线的照射效果、处理大量电脑角色的智能、网络信息通讯等。在配置不高的电脑上，这些计算花费的时间就多，游戏刷新画面的频率就变慢了。这也就是我们常听说“一个游戏的帧率低”的原因，这种时候你就会感到游戏不流畅。

【Pygame 第3课】 游戏中的事件

有人问，为什么突然讲游戏了？有人问，为什么不继续讲python的基础？有人问，为什么不讲爬虫？有人问，为什么不讲算法？……

因为有很多内容，每一块都有想听的人。因为同时不可能推送很多内容。因为我喜欢游戏开发，制作一款游戏的过程很有趣。我会尽量在论坛上补充更多方面的内容。微信上的推送有天生的限制，不能让所有人满足，大家见谅。

上次课讲了游戏最根本的框架，说到在每次循环中会接收玩家的操作。这是游戏中很重要的一个环节

--事件响应。

玩家的操作会触发程序中的事件，常见的事件包括：关闭程序、按下键盘、移动鼠标、按下鼠标等等。今天我们挑其中一个来举例说明：“鼠标按下”事件（MOUSEBUTTONDOWN）。

还记得上次课中的这段代码吗：

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
    #接收到退出事件后退出程序
        pygame.quit()
        exit()
```

它就是一段事件响应的处理代码。pygame.event.get()会接收所有程序中的事件。当判断这个事件是一个关闭程序（QUIT）的事件时，就将程序关闭。

现在，我们要增加一个事件响应：当玩家点击了鼠标之后，就换一张背景图。

在 for 循环中新增一段 if 语句：

```
if event.type == pygame.MOUSEBUTTONDOWN:
#接收到鼠标按下事件后更换背景
    background = pygame.image.load('bg2.jpg').convert()
```

pygame.image.load().convert()是将图片文件读入程序，后面的.convert()可以省略。

运行程序，在窗口上点击鼠标，背景会变成 bg2.jpg 的图案。为了显示效果，最好使用和 bg.jpg 长宽一样的图片。

*Untitled Document 1 hello.py

```
# -*- coding: utf-8 -*-
import pygame
#导入pygame库
from sys import exit
#向sys模块借一个exit函数用来退出程序
pygame.init()
#初始化pygame,为使用硬件做准备
screen = pygame.display.set_mode((600, 170), 0, 32)
#创建了一个窗口,窗口大小和背景图片大小一样
pygame.display.set_caption("Hello, World!")
#设置窗口标题
background = pygame.image.load('bg.jpg').convert()
#加载并转换图像
while True:
#游戏主循环
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            #接收到退出事件后退出程序
            pygame.quit()
            exit()
        if event.type == pygame.MOUSEBUTTONDOWN:
            #接收到鼠标按下事件后更换背景
            background = pygame.image.load('bg2.jpg').convert()
    screen.blit(background, (0,0))
    #将背景图画上去
    pygame.display.update()
    #刷新一下画面
```

Python Tab Width: 8 Ln 7, Col 20 INS

不过点击了一次之后，背景就不会再变了。而实际上，在你每次点击的时候，程序都会去读取一遍 bg2. jpg，这是没有必要。把这个程序的改进留给你们：点击鼠标的时候，背景可以在 2 张甚至多张图片间切换，另外最好不要每次都去读文件。

【Pygame 第 4 课】 获取鼠标位置

最近微信上很火的“打飞机”游戏，通过手指在屏幕上触摸的位置来移动你的飞机。在电脑上，我们没法直接用手操作，但可以用鼠标替代手指。

在电脑游戏里，鼠标是个很好用的输入设备。因此在很多游戏中，都需要得到鼠标的位置，以响应用户的操作。

现在，我们要在之前 hello world 的程序上增加一架飞机，并且用鼠标来控制飞机的位置。

得到鼠标位置坐标的方法是：

```
1. pygame.mouse.get_pos()
```

与以往用的函数有些不同，这个函数会返回两个值：鼠标的 x 坐标和 y 坐标。所以你需要两个变量来记录返回值：

```
1. x, y = pygame.mouse.get_pos()
```

然后，在游戏主循环中，把实现准备好的飞机图片画到屏幕上，位置就是(x,y)：

```
1. screen.blit(plane, (x,y))
```

运行程序，你会发现，鼠标移动到哪，飞机就会“飞”到哪。但是，飞机图片始终在鼠标的右下方。这是因为图片的坐标原点是在左上角，原点与鼠标的位置对齐。

如果你想让图片的中心和鼠标位置对齐，则需要再调整一下 x，y 的位置：

```
1. x -= plane.get_width() / 2
2. y -= plane.get_height() / 2
```

get_width 和 get_height 分别是获取图片的宽和高。

你可以挑张合适的背景图和一张边缘透明的飞机图，再把窗口的长宽调整一下，让它看上去更舒服一些。

完整代码：

```
1. # -*- coding: utf-8 -*-
2. import pygame
3. from sys import exit
4. pygame.init()
5. screen = pygame.display.set_mode((600, 170), 0, 32)
6. pygame.display.set_caption("Hello, World!")
7. background = pygame.image.load('bg.jpg').convert()
8. plane = pygame.image.load('plane.jpg').convert()
9. #加载飞机图像
10. while True:
11.     for event in pygame.event.get():
12.         if event.type == pygame.QUIT:
13.             pygame.quit()
14.             exit()
```

```
15.     screen.blit(background, (0,0))
16.
17.     x, y = pygame.mouse.get_pos()
18.     #获取鼠标位置
19.     x -= plane.get_width() / 2
20.     y -= plane.get_height() / 2
21.     #计算飞机的左上角位置
22.     screen.blit(plane, (x,y))
23.     #把飞机画到屏幕上
24.     pygame.display.update()
```

*Untitled Document 1 ✖ 4.py ✖

```
# -*- coding: utf-8 -*-
import pygame
from sys import exit
pygame.init()
screen = pygame.display.set_mode((600, 170), 0, 32)
pygame.display.set_caption("Hello, World!")
background = pygame.image.load('bg.jpg').convert()
plane = pygame.image.load('plane.jpg').convert()
#加载飞机图像
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
    screen.blit(background, (0,0))

    x, y = pygame.mouse.get_pos()
    #获取鼠标位置
    x-= plane.get_width() / 2
    y-= plane.get_height() / 2
    #计算飞机的左上角位置
    screen.blit(plane, (x,y))
    #把飞机画到屏幕上
    pygame.display.update()
```

【Pygame 第 5 课】 游戏中的运动

本来，在上一次 pygame 的教程中，我只是顺手拿了微信“打飞机”里的图来演示用鼠标控制图片位置的操作。后来觉得，这个游戏还算比较适合用来做例子，也有朋友反馈说想做这个游戏，那不如就以“打飞机”为例来说 python 游戏开发好了。

今天，就再进一步：既然要打飞机，那得能发射子弹才行。所以从最简单的做起，来给游戏加上“一颗”子弹。

上次的背景图和飞机图，我自己稍微处理了下，包括这一课要用到的子弹图片，都放在论坛上，需要的自行下载。

大体的思路是这样的：

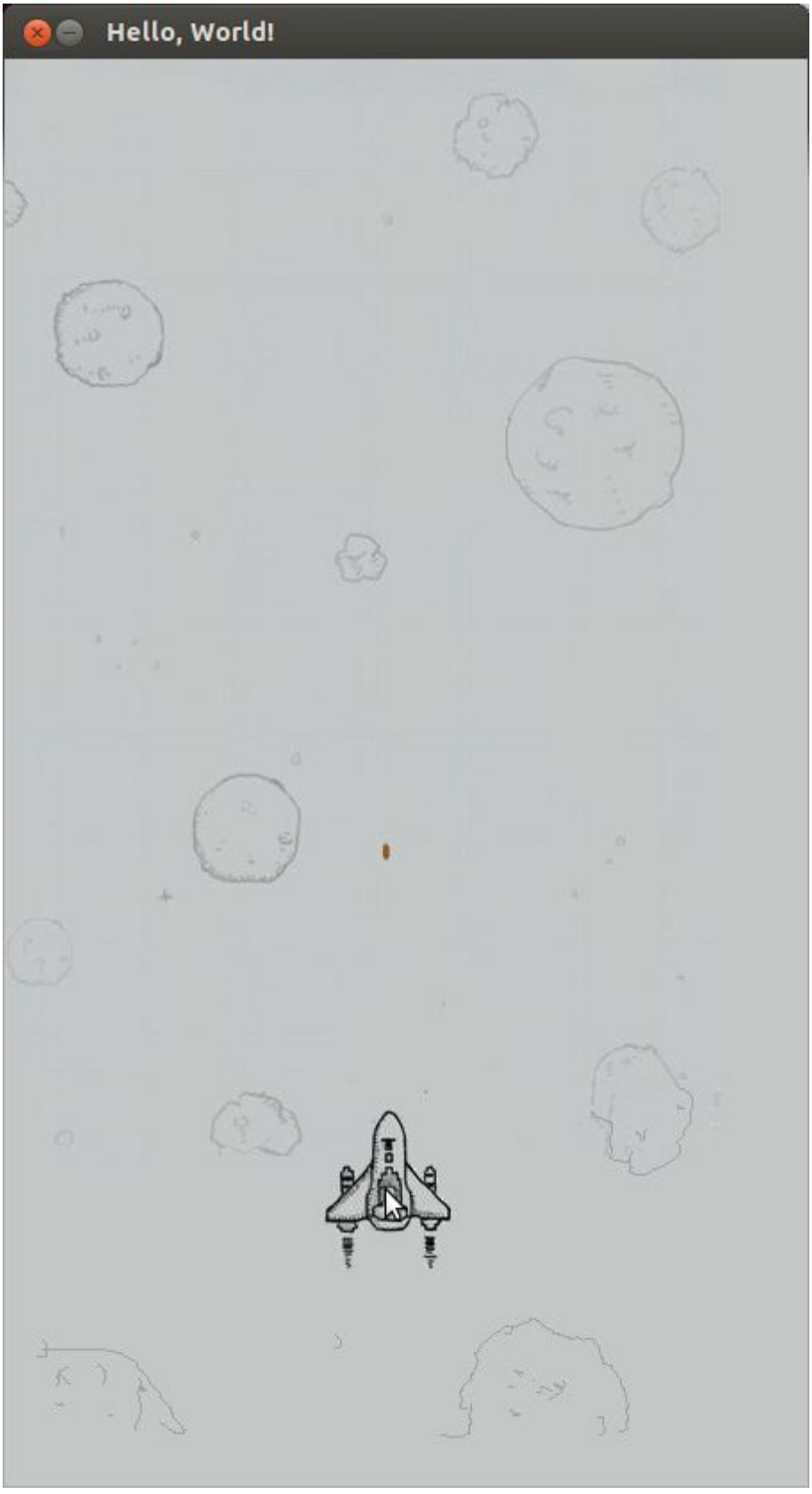
1. 用之前在屏幕上绘制飞机的方法，再绘制一张很小的子弹图片。补充一下：当你需要绘制一张带透明部分的图片时，要用 convert_alpha()替代之前的 convert()，具体用法参见代码中。
2. 子弹被发射的位置是飞机的位置，也就是鼠标的位置。注意，要让它们的中心点对齐，而不是左上角对齐，处理方法我们已经说过。
3. 让这个子弹往上运动。还记得我在第 2 课《游戏的本质》里面说的吗：在游戏主循环中，要处理物理运动。所以在程序中要做的就是，每次循环里，把子弹图片的 y 坐标减少一个量（因为屏幕左上角的坐标是(0,0)）。为了能记住子弹上一次循环中的位置，要有变量专门来记录子弹的坐标值。
4. 当子弹移动到屏幕上方外部之后（y 坐标小于 0），再把它的位置重置回发射的位置。这样看上去就是又一颗子弹被发射出来了，尽管我们一直是在操作同一张图片。游戏中经常会使用到诸如此类的小技巧，来欺骗你的视觉，这也是我觉得开发游戏很有意思的一个地方，好像是在变魔术。
5. 为了看起来更符合常理，你得把子弹的图片放在飞机的图片下面，这样看上去才会是从飞机上发射出去，而不是凭空冒出来的。在程序中，就是先绘制子弹，再绘制飞机，像是画油画，后画的会覆盖掉先画的。
6. 我在一开始就将子弹的位置设到屏幕上方之外，这样它就会自动被循环内的条件判断给重置位置，而不需要我再额外手动去初始化它的位置。

理清了如上的思路之后，能不能搞定代码了？如果能的话，就先别往下看，试着在程序里写写看。

以下是我的实现代码：

```
1. # -*- coding: utf-8 -*-
2. import pygame
3. from sys import exit
4. pygame.init()
5. screen = pygame.display.set_mode((450, 800), 0, 32)
6. pygame.display.set_caption("Hello, World!")
7. background = pygame.image.load('back.jpg').convert()
8. plane = pygame.image.load('plane.png').convert_alpha()
9. bullet = pygame.image.load('bullet.png').convert_alpha()
10. #加载子弹图像
11. bullet_x = 0
```

```
bullet_y = -1
#初始化子弹位置
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    x, y = pygame.mouse.get_pos()
    if bullet_y < 0:
        #如果子弹位置超出了屏幕上端
        bullet_x = x - bullet.get_width() / 2
        bullet_y = y - bullet.get_height() / 2
        #把子弹的中心位置设为鼠标坐标
    else:
        bullet_y -= 5
        #子弹的位置往上移
    screen.blit(bullet, (bullet_x, bullet_y))
    #把子弹画到屏幕上
    x-= plane.get_width() / 2
    y-= plane.get_height() / 2
    screen.blit(plane, (x, y))
    pygame.display.update()
```


Hello, World!

【Pygame 第 6 课】 面向对象的游戏设计

上节课中，我们的飞机已经可以发射子弹了，尽管只有一颗。为什么我只加了一颗？试着多加几颗你就会发现，你得用好几个变量去分别记录它们的 xy 坐标，在主循环中判断每一颗子弹的状态。你可

以用 list 把程序写得稍稍不那么复杂，但这还没完。别忘了你打飞机的对手--敌机还没有加入到游戏。到时候你又需要更多的变量去记录它们的坐标，去判断它们的状态，去处理敌机、子弹、玩家飞机之间的关系。想想都觉得头大。

于是乎，我之前煞费苦心讲解的面向对象就该派上用场了。我要把子弹相关的东西都封装在一起。

先看看目前子弹相关的有哪些东西：x、y 坐标，一张图片，好像就这么多。然后，还有一段处理子弹运动状态的代码。来建一个 Bullet 类，把 x、y、image 作为成员变量，再提供一个叫做 move 的成员函数，处理子弹的运动。

#定义一个 Bullet 类，封装子弹相关的数据和方法

```
class Bullet:
    def __init__(self):
        #初始化成员变量，x，y，image
        self.x = 0
        self.y = -1
        self.image = pygame.image.load('bullet.png').convert_alpha()

    def move(self):
        #处理子弹的运动
        if self.y < 0:
            mouseX, mouseY = pygame.mouse.get_pos()
            self.x = mouseX - self.image.get_width() / 2
            self.y = mouseY - self.image.get_height() / 2
        else:
            self.y -= 5
```

代码的内容基本和之前一样，只是改为了面向对象的写法。如果你对__init__，self 这些字眼感到陌生的话，请发送数字 47 到 50，回顾一下关于 python 面向对象的课程。

接下来，程序主体就可以瘦身了。在原本加载子弹图片、初始化位置的地方，直接创建一个 Bullet 的实例。

```
bullet = Bullet()
```

在主循环中处理子弹运动的地方，调用 Bullet 的 move 方法。

```
bullet.move()
```

绘制子弹的时候，从 bullet 实例中取数据。

```
screen.blit(bullet.image, (bullet.x, bullet.y))
```

就这么简单。

运行程序看看效果是否正常。相比昨天，游戏的功能没有任何进展，但在结构上清晰了许多。之后，可以放心地添加更多子弹和敌机，而不会导致代码变成一坨。

```
# -*- coding: utf-8 -*-
import pygame
from sys import exit

#定义一个Bullet类，封装子弹相关的数据和方法
class Bullet:
    def __init__(self):
        self.x = 0
        self.y = -1
        self.image = pygame.image.load('bullet.png').convert_alpha()
        #初始化成员变量，x，y，image

    def move(self):
        #处理子弹的运动
        if self.y < 0:
            mouseX, mouseY = pygame.mouse.get_pos()
            self.x = mouseX - self.image.get_width() / 2
            self.y = mouseY - self.image.get_height() / 2
        else:
            self.y -= 5

pygame.init()
screen = pygame.display.set_mode((450, 800), 0, 32)
pygame.display.set_caption("Hello, World!")
background = pygame.image.load('back.jpg').convert()
plane = pygame.image.load('plane.png').convert_alpha()
bullet = Bullet()
#创建一个Bullet的实例
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    bullet.move()
    #调用move方法，处理子弹的运动
    screen.blit(bullet.image, (bullet.x, bullet.y))
    #绘制子弹，数据来自其成员变量
    x, y = pygame.mouse.get_pos()
    x-= plane.get_width() / 2
    y-= plane.get_height() / 2
    screen.blit(plane, (x, y))
    pygame.display.update()
```

【Pygame 第 7 课】 多变的宿敌

在游戏中，一般都有个宿敌什么的。在我们这个打飞机小游戏中，宿敌就是不断从天而降的敌机。它与本机、子弹构成了这个游戏的三个要素：

- 本机会发射子弹，子弹向上运动
- 敌机会不停产生，向下运动
- 子弹碰到敌机，敌机和子弹都销毁，加分
- 本机碰到敌机，本机和敌机都销毁，游戏结束

这节课就来创造这个宿敌。

同样，为敌机创建一个类：Enemy，类的内容大致与 Bullet 相似。

```
1. class Enemy:
2.     def __init__(self):
3.         self.x = 200
4.         self.y = -50
5.         self.image = pygame.image.load('enemy.png').convert_alpha()
6.
7.     def move(self):
8.         if self.y < 800:
9.             self.y += 0.3
10.        else:
11.            self.y = -50
```

复制代码

让敌机在屏幕上方外部靠中间的位置产生，并且在每一次循环中都向下移动，当飞出屏幕下方后，就回到屏幕上方重新开始。

和 bullet 一样，我们创建一个 Bullet 对象，然后在循环中调用它的 move 方法，并且绘制在屏幕上。

```
1. enemy = Enemy()
2.
3. while True:
4.     ###
5.     enemy.move()
6.     screen.blit(enemy.image, (enemy.x, enemy.y))
```

复制代码

运行程序。敌机开始在屏幕中部周而复始地自上向下运动。

这宿敌也太呆了吧！
这样的游戏谁要玩！

所以我们要加点随机性。

让敌机的出现位置有变化，让它的速度有变化。

给 Enemy 增加一个 restart 方法：

```
1. def restart(self):
```

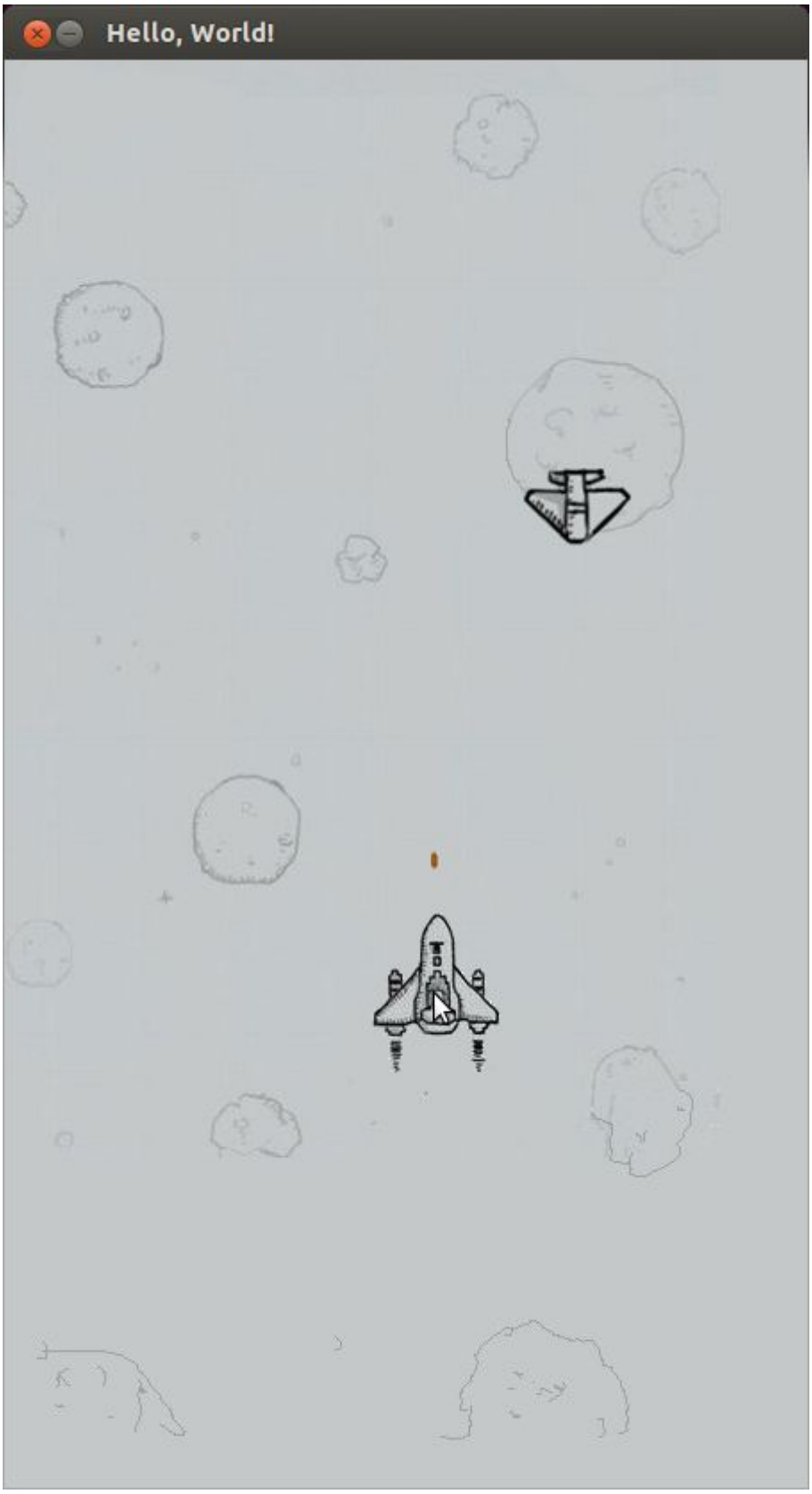
```
2.      self.x = random.randint(50, 400)
3.      self.y = random.randint(-200, -50)
4.      self.speed = random.random() + 0.1
```

复制代码

它的作用是，给敌机在屏幕上方的一定范围内随机一个初始位置，然后再给它增加一个叫做 speed 的随机量，作为它的速度，在 move 函数中被使用。这样，它的行为开始有了变化。这里用到了 random 模块，记得在程序开头 import 它。

restart 在__init__函数以及飞出屏幕下方时调用。后面，在敌机被击中的时候，也会要调用它。

再次运行程序，看上去有那么点意思了。如果敌机再多一点，子弹再多一点，就更好了。这个，留在下节课中说。


Hello, World!

```
class Enemy:
    def restart(self):
        #重置敌机位置和速度
        self.x = random.randint(50, 400)
        self.y = random.randint(-200, -50)
        self.speed = random.random() + 0.1

    def __init__(self):
        #初始化
        self.restart()
        self.image = pygame.image.load('enemy.png').convert_alpha()

    def move(self):
        if self.y < 800:
            #向下移动
            self.y += self.speed
        else:
            #重置
            self.restart()
```

```
pygame.init()
screen = pygame.display.set_mode((450, 800), 0, 32)
pygame.display.set_caption("Hello, World!")
background = pygame.image.load('back.jpg').convert()
plane = pygame.image.load('plane.png').convert_alpha()
bullet = Bullet()
#创建敌机
enemy = Enemy()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    bullet.move()
    screen.blit(bullet.image, (bullet.x, bullet.y))
    #更新敌机位置
    enemy.move()
    screen.blit(enemy.image, (enemy.x, enemy.y))
    x, y = pygame.mouse.get_pos()
    x-= plane.get_width() / 2
    y-= plane.get_height() / 2
    screen.blit(plane, (x, y))
    pygame.display.update()
```

【Pygame 第 8 课】 火力全开

游戏中的几个主要角色我们都有了，接下来就是去完善它们，用它们来组成一个完整的游戏。

首先我们要处理的是子弹。只有一发子弹显然是不够的，群众表示要火力全开！

所以，我们要有一个 list，这里 list 里面存放着一些 Bullet 的对象。但一个 list 的 Bullet 都按之前的方法创建是不行的，那样所有的子弹都会在同一时间发射出去，同时到达屏幕上方，又同时再次发射，这样的视觉效果和一发子弹没什么区别。所以我们要让它们按照一定的时间间隔，一个一个地发射。

另外,之前到了屏幕顶端就回头的方法也会带来问题,重新发射的子弹会和按时发射的子弹混在一起，打乱发射的节奏。所以，子弹“回收”的方法也要改。有种最简单的方法，就是不回收，每次发射都是创建一个新的 Bullet 对象，飞出屏幕之后就抛弃它。这当然是可以的，但每次都要创建对象，读取图片，并在 list 上做添加和删除的操作。这样会比较消耗资源，在游戏开发中一般都尽量避免。即使现在这个小游戏中它还不至于影响到体验，也应该养成节约的良好习惯。

我们今天要解决的就是两个问题：定时和回收。

python 中有定时运行的方法。但这里，我不打算用它。我们有现成的循环在这儿，只要设定好隔多少次循环运行，就简单地实现了定时的效果。尽管在游戏中，每次循环的时间并不相同，这么做会有潜在的问题：间隔的真实时间会受电脑运行速度的影响。但暂时你可以忽略这个细节（它是有解决办法的）。

我们设定一个变量 interval_b，作为 Bullet 的发射间隔。在每次循环中，让 interval_b 递减，当减到 0 以下时，便运行，并重置 interval_b 的间隔。

```
1. interval_b = 0
2. while True:
3.     interval_b -= 1
4.     #当间隔小于 0 时，激活一发子弹
5.     if interval_b < 0:
6.       ###（发射子弹代码）
7.       interval_b = 100
```

关于子弹的重复利用，我们增加一个变量 active，只有 active 为 True 的子弹，我们才去处理它的运动。另外，为了依次使用 list 中有限的子弹，还需要一个变量 index_b 来记录下一颗子弹是第几号。每激活发射一颗子弹，就把 index_b 指向它的下一号，最后一号之后再回头使用第 0 号。

修改一下 Bullet 类，增加 active，并根据 active 的状态处理运动。增加一个 restart 方法，用来重新发射子弹。

```
1. class Bullet:
2.     def __init__(self):
3.       self.x = 0
4.       self.y = -1
5.       self.image = pygame.image.load('bullet.png').convert_alpha()
6.       #默认不激活
7.       self.active = False
8.
9.     def move(self):
10.       #激活状态下，向上移动
11.       if self.active:
```

```
            self.y -= 3
        #当飞出屏幕，就设为不激活
        if self.y < 0:
            self.active = False

    def restart(self):
        #重置子弹位置
        mouseX, mouseY = pygame.mouse.get_pos()
        self.x = mouseX - self.image.get_width() / 2
        self.y = mouseY - self.image.get_height() / 2
        #激活子弹
        self.active = True
```

在游戏中创建5发子弹的list(5发足够了,只要保证你的子弹数足够在打完一轮之前到达屏幕顶端)。

```
#创建子弹的 list
bullets = []
#向 list 中添加 5 发子弹
for i in range(5):
    bullets.append(Bullet())
#子弹总数
count_b = len(bullets)
#即将激活的子弹序号
index_b = 0
#发射子弹的间隔
interval_b = 0
```

时间间隔到达时，restart 一颗子弹，并将序号递增。

```
while True:
    #发射间隔递减
    interval_b -= 1
    #当间隔小于 0 时，激活一发子弹
    if interval_b < 0:
        bullets[index_b].restart()
        #重置间隔时间
        interval_b = 100
        #子弹序号周期性递增
        index_b = (index_b + 1) % count_b
    #判断每个子弹的状态
    for b in bullets:
        #处于激活状态的子弹，移动位置并绘制
        if b.active:
            b.move()
            screen.blit(b.image, (b.x, b.y))
```

只处理 active 的子弹，绘制它们。

如此一来，你可以不停地向敌机开火了。根据你电脑的运行状况，适当调整一下子弹的移动速度和发

射间隔，让它看起来更自然。

至于敌机的行为，比子弹要简单一些，因为不需要定时出现，所以之前用的回收方法可以继续使用，我们下次再说。

```
class Bullet:
    def __init__(self):
        self.x = 0
        self.y = -1
        self.image = pygame.image.load('bullet.png').convert_alpha()
        #默认不激活
        self.active = False

    def move(self):
        #激活状态下，向上移动
        if self.active:
            self.y -= 3
        #当飞出屏幕，就设为不激活
        if self.y < 0:
            self.active = False

    def restart(self):
        #重置子弹位置
        mouseX, mouseY = pygame.mouse.get_pos()
        self.x = mouseX - self.image.get_width() / 2
        self.y = mouseY - self.image.get_height() / 2
        #激活子弹
        self.active = True
```

```
pygame.init()
screen = pygame.display.set_mode((450, 800), 0, 32)
pygame.display.set_caption("Hello, World!")
background = pygame.image.load('back.jpg').convert()
plane = pygame.image.load('plane.png').convert_alpha()
#创建子弹的list
bullets = []
#向list中添加5发子弹
for i in range(5):
    bullets.append(Bullet())
#子弹总数
count_b = len(bullets)
#即将激活的子弹序号
index_b = 0
#发射子弹的间隔
interval_b = 0
enemy = Enemy()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
    screen.blit(background, (0,0))
    #发射间隔递减
    interval_b -= 1
    #当间隔小于0时，激活一发子弹
    if interval_b < 0:
        bullets[index_b].restart()
        #重置间隔时间
        interval_b = 100
        #子弹序号周期性递增
        index_b = (index_b + 1) % count_b
    #判断每个子弹的状态
    for b in bullets:
        #处于激活状态的子弹，移动位置并绘制
        if b.active:
            b.move()
            screen.blit(b.image, (b.x, b.y))
    enemy.move()
    screen.blit(enemy.image, (enemy.x, enemy.y))
    x, y = pygame.mouse.get_pos()
    x-= plane.get_width() / 2
    y-= plane.get_height() / 2
    screen.blit(plane, (x, y))
    pygame.display.update()
```

【Pygame 第 9 课】 一大波飞机

又到周一了，小伙伴们周末过得可好？是不是有人刚刚结束了无聊的暑假，又开始丰富多彩的校园生活了？

上周最后的课里说了，这次我们要来加入一大波飞机正在接近的效果。这个要比之前的子弹容易实现多了。因为只要让飞机不停地从屏幕上方出现就好了，不用管它具体的间隔怎样，看上去像那么回事就可以。

之前我们单个飞机已经完成了在屏幕上方区域内随机出现，并且到底底部后重新回上方的功能。现在要做的，仅仅是把一架敌机换成一组敌机：

```
1. enemies = []
2. for i in range(5):
3.     enemies.append(Enemy())
```

创建 5 个 Enemy 的对象，把它们添加到一个叫做 enemies 的 list 中。
然后，在主循环里，去处理每一架敌机的运动：

```
1. for e in enemies:
2.     e.move()
3.     screen.blit(e.image, (e.x, e.y))
```

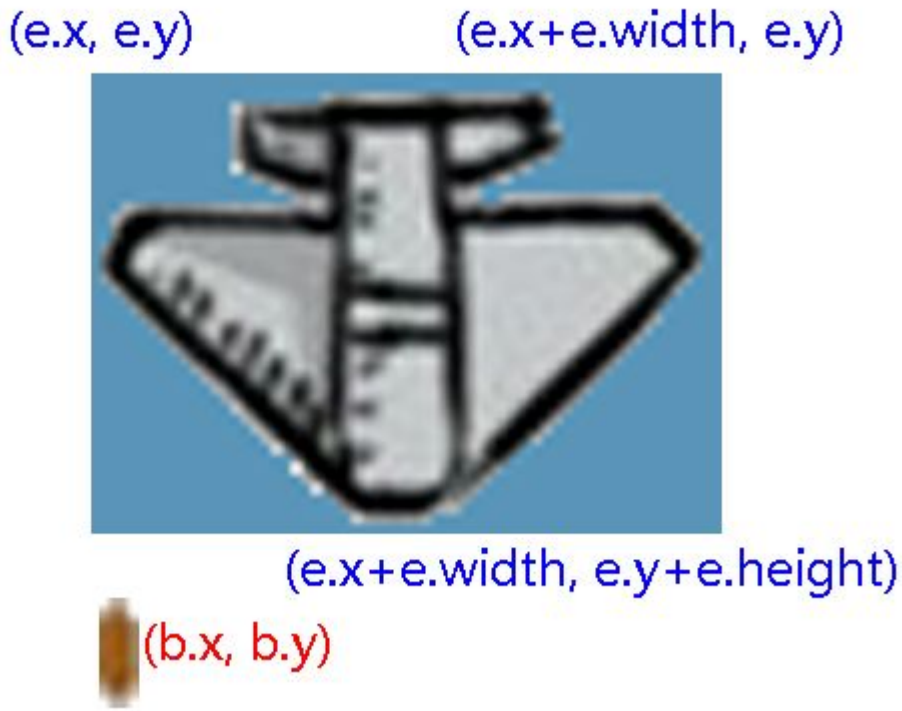
如果你觉得几架敌机的运动状态还是比较接近，试着调节它们速度随机范围，以及出现的位置范围。它们在屏幕上方出现的范围选择越大，在游戏中新增敌机的间隔随机性就越大。

That's all. 就这么多。感谢面向对象让我们省去了很多代码量。运行游戏看看有没有不断各种敌机前赴后继进攻的感觉？

【Pygame 第 10 课】 命中目标

现在，我们的“打飞机”游戏已经到了万事俱备只欠东风的阶段：有了子弹也有了敌机，但它们之间还没有办法擦出火花。这节课的内容就是让玩家可以真正的“打”飞机。

我们判断子弹命中飞机的依据很简单：就是子弹的位置在飞机图片的内部。这不需要很精确，因为在快速的游戏过程中，人眼也无法分辨细微的差别。来看下面这张示意图：



按照我们的设定（为了简化，假设子弹的长宽忽略不计），当子弹的坐标(b.x, b.y)在飞机的图片范围，也就是(e.x, e.y)到(e.x+e.width, e.y+e.height)所围成的矩形当中时，就可以认定是命中了。也就是满足：

e.x < b.x < e.x+e.width

e.y < b.y < e.y+e.height

当命中后，要做两件事：把敌机重置；把子弹重置。（之后还可以加上得分）

在代码中的实现：

```
1. def checkHit(enemy, bullet):
2.       if (bullet.x > enemy.x and bullet.x < enemy.x + enemy.image.get_width())
   and (bullet.y > enemy.y and bullet.y < enemy.y + enemy.image.get_height()):
3.           enemy.restart()
4.           bullet.active = False
```

我们把这段是否命中的检测代码写成一个函数 checkHit(enemy, bullet)，在主循环中，检测每一颗 active 的子弹是否命中任何一个 enemy：

```
1. for b in bullets:
2.     if b.active:
3.       for e in enemies:
4.           checkHit(e, b)
```

运行代码，你就可以痛击敌机了，虽然效果还很突兀。

在游戏中，我们把这种判断两样物体是否有重合关系的处理称为“碰撞检测”。其实在 pygame 中，已经为我们实现好了更方便更高效的碰撞检测方法。这里自己手动实现一个简单的碰撞检测，可以对其原理有更好的认识。

```
def checkHit(enemy, bullet):
    #如果子弹在敌机的图片范围之内
    if (bullet.x > enemy.x and bullet.x < enemy.x + enemy.image.get_width()) and (
        bullet.y > enemy.y and bullet.y < enemy.y + enemy.image.get_height()
    ):
        #重置敌机
        enemy.restart()
        #重置子弹
        bullet.active = False
```

```
    for b in bullets:
        if b.active:
            #检测每一颗active的子弹是否与enemy碰撞
            for e in enemies:
                checkHit(e, b)
            b.move()
            screen.blit(b.image, (b.x, b.y))
```

【Pygame 第 11 课】 GAME OVER

继续我们的打飞机游戏。完成了子弹和敌机之间的碰撞检测之后，自然还要来处理敌机与本体之间的碰撞检测，这决定了游戏是否结束。

之前我们没有把 plane 作为一个对象来处理，现在为了能更方便地做碰撞检测，我们还是要把它封装一下。这和我们之前对 bullet 和 enemy 所做的操作类似。

```
class Plane:
    def restart(self):
      self.x = 200
      self.y = 600

    def __init__(self):
      self.restart()
      self.image = pygame.image.load('plane.png').convert_alpha()

     def move(self):
       x, y = pygame.mouse.get_pos()
       x -= self.image.get_width() / 2
       y -= self.image.get_height() / 2
       self.x = x
       self.y = y

plane = Plane()
```

在 move 方法中，依旧根据鼠标的位置改变飞机的位置。
然后我们增加一个 checkCrash 的函数，和 checkHit 类似，它用来处理敌机和本体之间的碰撞。

```
def checkCrash(enemy, plane):
    if (plane.x + 0.7*plane.image.get_width() > enemy.x) and (plane.x +
   0.3*plane.image.get_width() < enemy.x + enemy.image.get_width()) and (plane.y +
   0.7*plane.image.get_height() > enemy.y) and (plane.y +
   0.3*plane.image.get_width() < enemy.y + enemy.image.get_height()):
      return True
    return False
```

这里的判断比之前要复杂一些，因为敌机和本体都有一定的面积，不能像子弹一样忽略长宽。但如果两张图片一旦有重合就算是碰撞，会让游戏看上去有些奇怪：有时候你觉得并没有撞上，而实际已经有了重合，游戏就失败了。所以为了避免这一现象，我们要给 plane 的长宽打上一点折扣。这也就是代码中判断条件里“0.3”“0.7”的意义所在。

checkCrash 把碰撞检测的结果用 True 或 False 返回。在游戏主循环里，我们增加一个记录游戏是否结束的变量 gameover。把之前的游戏逻辑放在 gameover 为 False 的情况下。而当 checkCrash 为 True 时，就把 gameover 设为 True。

```
gameover = False
while True:
    ###
    if not gameover:
      ###省略部分游戏逻辑
      for e in enemies:
         #如果撞上敌机，设 gameover 为 True
         if checkCrash(e, plane):
            gameover = True
         e.move()
         screen.blit(e.image, (e.x, e.y))
```

```
        #检测本体的运动
        plane.move()
        screen.blit(plane.image, (plane.x, plane.y))
    else:
        #待处理
        pass
```

运行代码，当你不幸被敌机撞上后，游戏陷入一片空白。然后，你只好关闭程序。下一课，我们来处理被撞后的善后工作。

【Pygame 第 12 课】 屡败屡战

打飞机游戏，我们已经做得差不多了。今天要再加上两个功能，让它看上去更完整：显示分数、重新开始。这样，玩家才能一次接一次地玩下去。

要显示分数，首先得有一个变量记录分数：

```
score = 0
```

当打中敌机的时候，把分数增加。为了达到这个目的，修改一下之前的 checkHit 函数，让它和 checkCrash 一样，返回一个 bool 值，表示是否发生了碰撞：

```
def checkHit(enemy, bullet):
    if (bullet.x > enemy.x and bullet.x < enemy.x + enemy.image.get_width()) and (
        bullet.y > enemy.y and bullet.y < enemy.y + enemy.image.get_height()
    ):
        enemy.restart()
        bullet.active = False
        #增加返回值
        return True
    return False
```

在主循环里，当 checkHit 为 True 时，就增加分数：

```
for b in bullets:
    if b.active:
        for e in enemies:
            #击中敌机后，分数加 100
            if checkHit(e, b):
                score += 100
        b.move()
        screen.blit(b.image, (b.x, b.y))
```

这样，就用 score 记录了游戏中的分数。

在 pygame 中要显示文字，不能直接 print，那样只会在命令行里输出，无法显示在屏幕上。需要先创建一个 font 对象：

```
1. font = pygame.font.Font(None, 32)
```

None 表示使用默认字体，32 是字号。

然后，用 font 渲染出字体，再绘制到 screen 上：

```
1. text = font.render("Socre: %d" % score, 1, (0, 0, 0))
2. screen.blit(text, (0, 0))
```

(0,0)是屏幕左上角的位置。
当游戏结束后，我们要把分数显示在屏幕中间，改变这个坐标就可以了。

为了让游戏结束后能方便地重新开始，我们再往事件响应的代码中增加一段处理：

```
1. #判断在 gameover 状态下点击了鼠标
2. if gameover and event.type == pygame.MOUSEBUTTONUP:
3.     #重置游戏
4.     plane.restart()
5.     for e in enemies:
6.         e.restart()
7.     for b in bullets:
8.         b.active = False
9.     score = 0
10.    gameover = False
```

当 gameover 状态下发生了鼠标按钮抬起的事件（即玩家点击了鼠标），我们就把本体和敌机都重置位置，子弹都设 active 为 False，分数清零，gameover 为 False，游戏重新开始。

好了，现在你可以一次又一次地去迎战敌机，再一次又一次地被撞毁了。不限次数，不用向好友索要飞机哦。至于记录最高分什么的，我想你应该也可以搞定吧。

```
57 def checkHit(enemy, bullet):
58     if (bullet.x > enemy.x and bullet.x < enemy.x + enemy.image.get_width()) and (
59         bullet.y > enemy.y and bullet.y < enemy.y + enemy.image.get_height()
60     ):
61         enemy.restart()
62         bullet.active = False
63         #增加返回值
64         return True
65     return False
```

```
gameover = False
#分数
score = 0
#用以显示文字的Font对象
font = pygame.font.Font(None, 32)
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        #判断在gameover状态下点击了鼠标
        if gameover and event.type == pygame.MOUSEBUTTONUP:
            #重置游戏
            plane.restart()
            for e in enemies:
                e.restart()
            for b in bullets:
                b.active = False
            score = 0
            gameover = False
    screen.blit(background, (0,0))
```

```
        for b in bullets:
            if b.active:
                for e in enemies:
                    #击中敌机后，分数加100
                    if checkHit(e, b):
                        score += 100
                b.move()
                screen.blit(b.image, (b.x, b.y))
```

```
        plane.move()
        screen.blit(plane.image, (plane.x, plane.y))
        #在屏幕左上角显示分数
        text = font.render("Socre: %d" % score, 1, (0, 0, 0))
        screen.blit(text, (0, 0))
    else:
        #在屏幕中央显示分数
        text = font.render("Socre: %d" % score, 1, (0, 0, 0))
        screen.blit(text, (190, 400))
    pygame.display.update()
```