Pygame游戏设计

第06章 碰撞检测

Python编程课

碰撞检测

- **碰撞检测**指的是检测两个动画精灵接触或者重叠。
- pygame提供了一种方法对动画精灵分组。例如，在保龄球游戏中，所有的瓶在一组，球在另一组。组与碰撞检测密切相关，我们可以检测球何时击倒某个瓶子，因此需要寻找精灵（球）与球瓶组中所有精灵之间的碰撞；当然，还可以检测组内部的碰撞（比如球瓶相互碰倒）。

- 碰撞检测分为两种：
  - 某一动画精灵与自己所在组的动画精灵之间的碰撞；
  - 某一动画精灵与其他组的动画精灵之间的碰撞；
- 那么怎么分组呢？Pygame提供了pygame.sprite.Group类，用来管理分组过个sprite对象。比如我们经常使用pygame.sprite.Group.add()方法添加动画精灵到组当中，使用pygame.sprite.Group.remove()方法从组当中移除某一个动画精灵。
- pygame.sprite.spritecollide()方法用来检测某个动画精灵组当中的“精灵们”是否与其他的动画精灵碰撞。

  spritecollide(sprite, group, dokill, collided = None) -> Sprite_list
  - sprite：单个动画精灵
  - groupe：精灵组
  - sokill：是否要移除组中所有的精灵？True or False
  - colloded： 什么样的方式碰撞，默认为矩形碰撞

- 下面，我们完成一个小游戏，加载几个小飞机，互相碰撞以后，反向移动。
- 首先，定义PlaneClass，与上节课程序类似。

```
import sys
import pygame
from pygame.locals import *
from random import *

class PlaneClass(pygame.sprite.Sprite):
    def __init__(self, img_file, location, speed):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(img_file)
        self.rect = self.image.get_rect()
        self.rect.left, self.rect.top = location
        self.speed = speed
    def move(self):
        self.rect = self.rect.move(self.speed)
        if self.rect.left < 0 or self.rect.right > width:
            self.speed[0] = -self.speed[0]
        if self.rect.top < 0 or self.rect.bottom > height:
            self.speed[1] = -self.speed[1]
```

- 主程序中，创建精灵组，并且将所有的精灵添加到组中。

```
pygame.init()
screen_size = width, height = 480, 700
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption(“AirplaneSprite”)
background = pygame.image.load(“background.png”).convert()

imgs = [“me1.png”, “enemy1.png”, “enemy2.png”]
img_file = “me1.png”
#创建精灵组
group = pygame.sprite.Group()

for i in range(3):
    location_img = [i *120, 10]
    speed = [choice([-2, 2]), choice([-2, 2])]
    airplane = PlaneClass(imgs[i], location_img, speed)
    #将各个精灵添加到组中
    group.add(airplane)
clock = pygame.time.Clock()
```

- 定义一个函数collide_check()用来检测小飞机之间是否产生碰撞。

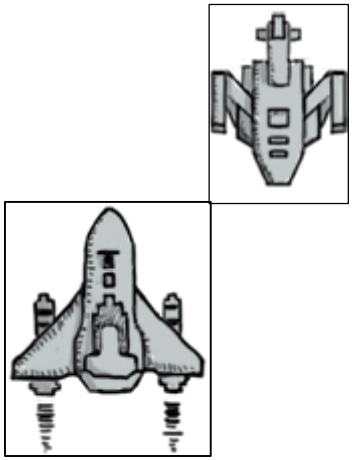
```
#传入参数group为小飞机动画精灵的组
def collide_check(group):
#先移动所有的小飞机
    for plane in group:
        plane.move()
    #分别检测每一个小飞机是否与其他小飞机碰撞
    for plane in group:
        #第一步，从组中删除该精灵（小飞机）
        group.remove(plane)
        #然后调用spritecollide()方法检测该精灵是否与组中其他精灵碰撞
        if pygame.sprite.spritecollide(plane, group, False):
            #如果产生碰撞，则该精灵的运动方向反向
            plane.speed[0] = -plane.speed[0]
            plane.speed[1] = -plane.speed[1]
        #第三步，再将该精灵加入组
        group.add(plane)
        screen.blit(plane.image, plane.rect)
```

- 主程序中调用动画精灵碰撞检测的函数。

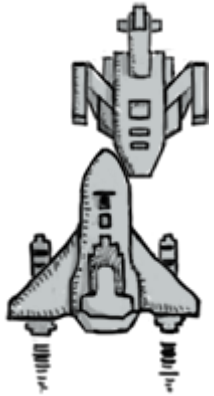
```
running = True
while running:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    screen.blit(background, (0, 0))
    #调用collide_check()函数，检测并更新精灵的位置
    collide_check(group)
    pygame.display.flip()
    clock.tick(30)
```

完美碰撞

- 程序完成，但是，有一个问题，小飞机并不是完全碰撞，而是碰撞到了小飞机图像的矩形边（如下图），就认为碰撞了。总觉得不太理想，该怎样让小飞机看上去真的碰撞呢？



矩形碰撞



完美碰撞

- pygame提供了一个图像蒙版的模块pygame.mask，用来实现完美检测，该模块中Pygame.mask.from_surface()方法可以通过制定的图像返回一个蒙版（图像轮廓）。
- 我们我们使用spritecollide(sprite, group, dokill, collided = None) 方法检测碰撞时，只传入了前三个参数，现在我们就要用到第四个参数了，我们在第四个参数传入pygame.sprite.collide_mask（实际是调用pygame.sprite.collide_mask ()方法），即实现蒙版检测碰撞，只有两个动画精灵的轮廓发生碰撞时，才被认为是碰撞，即完美碰撞。
- 除了使用蒙版外，还可以使用圆形范围检测。

- 修改部分代码，实现完美碰撞。

```
class PlaneClass(pygame.sprite.Sprite):
    def __init__(self, img_file, location, speed):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(img_file)
        self.rect = self.image.get_rect()
        self.rect.left, self.rect.top = location
        self.speed = speed
        #获取图像的蒙版
        self.mask = pygame.mask.from_surface(self.image)
def collide_check(group):
    for plane in group:
        plane.move()
    for plane in group:
        group.remove(plane)
            #使用图像的蒙版进行检测，从而实现完美碰撞检测
            if pygame.sprite.spritecollide(plane, group, False,
pygame.sprite.collide_mask):
                plane.speed[0] = -plane.speed[0]
                plane.speed[1] = -plane.speed[1]
            group.add(plane)
            screen.blit(plane.image, plane.rect)
```