碰撞检测

首先要保证要检测的两个对象都是`pygame.sprite.Sprite`类。

对于自己实现的类，我们可以让其继承`pygame.sprite.Sprite`类：

```
class ClassName(pygame.sprite.Sprite):#这样就继承了这个类
    def __init__(self[...,...]):
        pygame.sprite.Sprite.__init__(self)#初始化这个精灵对象
        #yourowncode
```

对于没有再单独实现的类的对象，我们可以直接使用

```
Name = pygame.sprite.Sprite()
```

创建一个原生的pygame.sprite.Sprite对象。

我们之前提到，如果需要操作多个对象，可以选择使用列表。但同时，pygame提供了原生的组：

```
groupName = pygame.sprite.Group()
```

这样就得到了一个名字叫groupName的sprite组。

关于sprite组的详细知识，参见 pygame精灵类。

pygame的碰撞检测有很多种，下面介绍其中较常用的。

- pygame.sprite.spritecollide(sprite, group, dokill, collided=None) -> Sprite_list

  > 该函数的作用是检测所有与该sprite发生碰撞的group中所有的sprite，并把它们返回。dokill参数代表是否在发生碰撞时从group中移除它们， collided参数表示使用哪种方式检测，默认使用collide_rect，可以选择collide_rect_ratio, collide_circle, collide_circle_ratio, collide_mask

  - 我们使用第二节课上写的壁球小游戏(temp.py)举例。

    > 首先创建一个球组，并向其加入若干个球。这里只加一个便于举例
    >
    > ```
    > ballGroup = pygame.sprite.Group()
    > ballGroup.add(Ball())#加进去一个球
    > ```
    >
    > 然后再创建另一个单独的球
    >
    > ```
    > myBall = Ball()
    > ```
    >
    > 随后，我们在游戏主循环的最后加上一句

```
print(pygame.sprite.spritecollide(myBall, ballGroup, False,
collide_mask))
```

表示检测myball和ballGroup中的对象是否发生碰撞，并且将发生碰撞的所有对象返回，组成一个列表。

演示视频放在群内。

下面的举例都和本例相同。

- pygame.sprite.collide_rect(left, right) -> bool

  该函数检测两个sprite(left和right)是否发生碰撞，返回值是一个布尔量(True或False)。

  例如

  ```
  print(pygame.sprite.collide_rect(myBall, ballGroup.sprites()[0]))
  ```

- pygame.sprite.collide_mask(sprite1, sprite2) -> None or (int, int)

  该函数和上一函数作用相同，只是如果未发生碰撞，则返回None, 否则返回相互碰撞距离sprite1的mask左上角最近的像素的相对坐标(x,y)

  - 注意， 使用本函数时，如果调用的这两个对象都没有mask属性，那么会自动为他们创建mask属性。也就是说，建议使用该函数之前为每个需要判断的对象都加上自己的mask属性，这样就不用在每次调用该函数时都创建一个新属性，拖慢程序。

  比如在Ball中加上

  ```
  class Ball(pygame.sprite.Sprite):#Ball类的定义
      def __init__(self):#初始化方法的定义
          pygame.sprite.Sprite.__init__(self)#初始化Sprite
          self.image =
  pygame.transform.scale(pygame.image.load("./images/ball.png","The
  ball").convert(),(120,120))
          self.rect = self.image.get_rect()
          self.image.set_colorkey(BG_COLOR)
          self.windowsInfo = pygame.display.Info()
          self.rect.center =
  (self.windowsInfo.current_w/2,self.windowsInfo.current_h/2)
          self.speedX = 0
          self.speedY = 0#对象的初始属性

          self.mask = pygame.mask.from_surface(self.image)#在这里加上mask这
  个属性，可以加快判断碰撞的速度。
  ```

- pygame.sprite.groupcollide(group1, group2, dokill1, dokill2, collided=None) -> Sprite_dict

  该函数的作用是group1和group2之间的元素互相检测碰撞，dokill和collieded参数和上面相同。返回值是一个字典，键(key)是group1中的sprite，值(value)为group2中的sprite。

- pygame.sprite.spritecollideany(sprite, group, collided=None) -> Sprite or None

  该函数的作用是检测该sprite是否与group中的任意sprite发生碰撞，如果是，则返回group中与sprite碰撞的对象，否则返回None