

[鱼C论坛](#) [论坛](#) [技术交流区](#) [Python交流](#) [《零基础入门学习Python》](#) [mask | Pygame中文文档](#)

发帖


回复

[返回列表](#)

查看: 814 | 回复: 0

[Pygame] mask | Pygame中文文档 [\[复制链接\]](#)



 发表于 2015-6-19 00:00:59 | 只看该作者

1# [电梯直达](#) 

小甲鱼  



累计签到: 2113 天  
连续签到: 11 天

最佳答案 78

2613 主题 | 1万 帖子 | 1万 荣誉



管理员

鱼C - 小甲鱼



技术值 184

[发消息](#)

## mask | Pygame中文文档

注: 本文档由 [wei\\_Y](#) 翻译, 小甲鱼校对。

### pygame.mask

Pygame 中处理图形遮罩的模块。

#### 函数

- pygame.mask.from\_surface() —— 从指定 Surface 对象中返回一个 Mask
- pygame.mask.from\_threshold() —— 从给定阈值的 Surface 对象中创建一个 Mask

#### 类

- pygame.mask.Mask —— 用于表示 2d 位掩码 ( 遮罩 ) 的 Pygame 对象

用于快速实现完美的碰撞检测, Mask 可以精确到 1 个像素级别的判断。

Pygame 1.8 新增加的。

#### 函数详解

##### pygame.mask.from\_surface()

从指定 Surface 对象中返回一个 Mask。

from\_surface(Surface, threshold = 127) -> Mask

Surface 对象中透明的部分设置为 1, 不透明部分设置为 0。

检查每个像素的 alpha 值是否大于 threshold 参数指定的值。( alpha 通道使用 0 ~ 255 描述像素的透明度 )

如果 Surface 对象是基于 colorkeys 实现的透明 ( 而不是基于 pixel alphas ), 则忽略 threshold 参数。

注: 关于 Surface 对象的透明实现, 可参考: [Surface | Pygame中文文档](#)

##### pygame.mask.from\_threshold()

从给定阈值的 Surface 对象中创建一个 Mask。

from\_threshold(Surface, color, threshold = (0,0,0,255), othersurface = None, palette\_colors = 1) -> Mask

从 Surface 对象中获取 Mask, 这个方法更有特色。如果只提供一个 Surface 对象, 那么 Surface 对象中所有与 threshold 参数提供颜色匹配的像素均被选入 Mask 中。如果指定了 othersurface 可选参数, 那么被选入 Mask 中的像素不仅需要与 threshold 参数提供颜色匹配, 还需要包含在 othersurface 指定的 Surface 对象中。

## class pygame.mask.Mask

用于表示 2d 位掩码 ( 遮罩 ) 的 Pygame 对象。

Mask((width, height)) -> Mask

### 方法

- pygame.mask.Mask.get\_size() —— 返回 Mask 的大小
- pygame.mask.Mask.get\_at() —— 如果像素 (x, y) 被设置, 返回值是非 0
- pygame.mask.Mask.set\_at() —— 设置 Mask 中给定位置的值
- pygame.mask.Mask.overlap() —— 返回两个 Mask 在指定偏移处的重叠坐标 ( 如果没有返回 None )
- pygame.mask.Mask.overlap\_area() —— 返回两个 Mask 重叠的像素数量
- pygame.mask.Mask.overlap\_mask() —— 将两个 Mask 重叠的部分创建一个新的 Mask
- pygame.mask.Mask.fill() —— 将所有的位置设置为 1
- pygame.mask.Mask.clear() —— 将所有的位置设置为 0
- pygame.mask.Mask.invert() —— 翻转 Mask 中所有的位 ( 0 变 1, 1 变 0 )
- pygame.mask.Mask.scale() —— 缩放 Mask 的尺寸
- pygame.mask.Mask.draw() —— 将 Mask 绘制到另一个 Mask 上边
- pygame.mask.Mask.erase() —— 用另一个 Mask 擦除 Mask
- pygame.mask.Mask.count() —— 返回 Mask 被设置 ( 为 1 ) 的像素的数量
- pygame.mask.Mask.centroid() —— 返回 Mask 的重心点
- pygame.mask.Mask.angle() —— 返回像素的方向
- pygame.mask.Mask.outline() —— 用列表的形式返回组成对象轮廓的点
- pygame.mask.Mask.convolve() —— 返回其它 Mask 的卷积
- pygame.mask.Mask.connected\_component() —— 返回与某像素区域的连接的 Mask
- pygame.mask.Mask.connected\_components() —— 返回一组连接某像素区域的 Mask 的列表
- pygame.mask.Mask.get\_bounding\_rects() —— 返回一组像素边界矩形的列表

### 方法详解

#### pygame.mask.Mask.get\_size()

返回 Mask 的大小。

get\_size() -> width,height

#### pygame.mask.Mask.get\_at()

如果像素 (x, y) 被设置, 返回值是非 0。

get\_at((x,y)) -> int

跟 Surface 对象一样, (0, 0) 表示左上角坐标。

#### pygame.mask.Mask.set\_at()

设置 Mask 中给定位置的值。

set\_at((x,y),value) -> None

#### pygame.mask.Mask.overlap()

返回两个 Mask 在指定偏移处的重叠坐标 ( 如果没有返回 None )。

overlap(othermask, offset) -> x,y

重叠检测的偏移原理如下 ( 偏移可以为负数 ) :

```

01.  +----+-----..
02.  |A  | yoffset
03.  | +-+-----..
04.  +--|B
05.  |xoffset
06.  | |
07.  : :

```

复制代码

**pygame.mask.Mask.overlap\_area()**

返回两个 Mask 重叠的像素数量。

overlap\_area(othermask, offset) -> numpixels

返回两个 Mask 重叠的像素数量，这可以用于查看在某方向上发生碰撞的部分，或者查看两个 Mask 有多少部分发生碰撞。相似的碰撞一般会通过计算重叠部分的梯度差分被发现。

```
01. dx = Mask.overlap_area(othermask, (x+1,y)) - Mask.overlap_area(othermask, (x-1,y))
02. dy = Mask.overlap_area(othermask, (x,y+1)) - Mask.overlap_area(othermask, (x,y-1))
```

[复制代码](#)

**pygame.mask.Mask.overlap\_mask()**

将两个 Mask 重叠的部分创建一个新的 Mask。

overlap\_mask(othermask, offset) -> Mask

返回的 Mask 尺寸是原始 Mask 和 othermask 参数指定的 Mask 重叠部分。

**pygame.mask.Mask.fill()**

将所有的位置设置为 1。

fill() -> None

将 Mask 中所有的位置设置为 1。

**pygame.mask.Mask.clear()**

将所有的位置设置为 0。

clear() -> None

将 Mask 中所有的位置设置为 0。

**pygame.mask.Mask.invert()**

翻转 Mask 中所有的位置 ( 0 变 1 , 1 变 0 )。

invert() -> None

翻转 Mask 中所有的位置 ( 0 变 1 , 1 变 0 )。

**pygame.mask.Mask.scale()**

缩放 Mask 的尺寸。

scale((x, y)) -> Mask

Mask 根据指定尺寸缩放后返回一个新的 Mask。

**pygame.mask.Mask.draw**

将 Mask 绘制到另一个 Mask 上边。

draw(othermask, offset) -> None

将 Mask 绘制到另一个 Mask 上边，执行的是按位 or 操作。

**pygame.mask.Mask.erase()**

用另一个 Mask 擦除 Mask。

`erase(othermask, offset) -> None`

从 Mask 上擦除 othermask 指定的像素。

#### **pygame.mask.Mask.count()**

返回 Mask 被设置 ( 为 1 ) 的像素的数量。

`count() -> pixels`

返回 Mask 被设置 ( 为 1 ) 的像素的数量。

#### **pygame.mask.Mask.centroid()**

返回 Mask 的重心点。

`centroid() -> (x, y)`

找到 Mask 的重心点。如果 Mask 是空的，那么返回值是 (0, 0)。

#### **pygame.mask.Mask.angle()**

返回像素的方向。

`angle() -> theta`

找到图像中像素的大致方向 ( -90 度 ~ 90 度 )，这对于实现像素对接很有用。如果 Mask 是空的，那么返回值是 0.0。

#### **pygame.mask.Mask.outline()**

用列表的形式返回组成对象轮廓的点。

`outline(every = 1) -> [(x,y), (x,y) ...]`

返回值是一个由点组成的列表，用于描绘穿过 Mask 的第一个对象的轮廓线。

every 可选参数用于设置点的跨度，默认是每 1 个像素。

#### **pygame.mask.Mask.convolve()**

返回其它 Mask 的卷积。

`convolve(othermask, outputmask=None, offset=(0,0)) -> Mask`

返回一个由位组 (i-offset[0], j-offset[1]) 组成的 Mask，如果转换的 othermask 参数在右下角 (i, j) 处，那么会与自身重叠。

如果 outputmask 参数被指定，那么就会在 outputmask 参数上进行绘制，outputmask 参数会被返回。否则这个 Mask 的大小是 `self.get_size() + othermask.get_size() - (1, 1)`。

#### **pygame.mask.Mask.connected\_component()**

返回与某像素区域的连接的 Mask。

`connected_component((x,y) = None) -> Mask`

它会使用 SAUF 算法进行连接需要连接的 Mask。它会连通 8 个点。默认情况下，它会返回在连接图像中最大的 Mask。可选项：一对指定的坐标，与其相连的组件会被返回。如果像素位置没有被设置，那么返回的这个 Mask 就是空的。这个 Mask 的大小会与原始 Mask 一样。

#### **pygame.mask.Mask.connected\_components()**

返回一组连接某像素区域的 Mask 的列表。

`connected_components(min = 0) -> [Masks]`

返回一组连接某像素区的 Mask 的列表。min 可选参数用于对每个连接区的指定部分过滤噪点。

pygame.mask.Mask.get\_bounding\_rects()

返回一组像素边界矩形的列表。

get\_bounding\_rects() -> Rects

它会获取连接着像素块的边界矩形。每个边界矩形都是一个连接每个像素的内矩形。

分享到: QQ好友和群 QQ空间 腾讯微博 腾讯朋友

收藏 评分 分享 淘帖 顶 踩



如果您的【问题求助】得到满意的解答，请自行将分类修改为【已经解决】

回复

举报

回复

发帖

返回列表



高级模式

发表回复

☐ 回帖后跳转到最后一页

本版积分规则