# Pygame

## Pygame模块

|  |  |
| --- | --- |
| 模块名 | 功能 |
| pygame.cdrom | 访问光驱 |
| pygame.cursors | 加载光标 |
| pygame.display | 访问显示设备 |
| pygame.draw | 绘制形状、线和点 |
| pygame.event | 管理事件 |
| pygame.font | 使用字体 |
| pygame.image | 加载和存储图片 |
| pygame.joystick | 使用游戏手柄或者类似的东西 |
| pygame.key | 读取键盘按键 |
| pygame.mixer | 声音 |
| pygame.mouse | 鼠标 |
| pygame.movie | 播放视频 |
| pygame.music | 播放音频 |
| pygame.overlay | 访问高级视频叠加 |
| Pygame | 就是我们在学的这个东西了…… |
| pygame.rect | 管理矩形区域 |
| pygame.sndarray | 操作声音数据 |
| pygame.sprite | 操作移动图像 |
| pygame.surface | 管理图像和屏幕 |
| pygame.surfarray | 管理点阵图像数据 |
| pygame.time | 管理时间和帧信息 |
| pygame.transform | 缩放和移动图像 |

Convert方法是将图像数据都转化为Surface对象，每次加载完图像以后就应该做这件事件（事实上因为 它太常用了，如果你不写pygame也会帮你做）；convert\_alpha相比convert，保留了Alpha 通道信息（可以简单理解为透明的部分），这样我们的光标才可以是不规则的形状。

获知本计算机上的屏幕模式

>>> import pygame

>>> pygame.init()

(6, 0)

>>> pygame.display.list\_modes()

[(1366, 768), (1360, 768), (1280, 768), (1280, 720), (1024, 768), (800, 600), (640, 480), (640, 400), (512, 384), (400, 300), (320, 240), (320, 200)]

控制速度：

在定义时实例化clock() = pygame.Clock()

在主循环中clock.tick(200)

screen.blit(font.render(str(event), True, (0,255,0)), (0, position))

把字符串渲染成Surface对象（一个图层），抗锯齿=True， 颜色代码，(0, position)表示x轴为0，y轴为position

## 显示界面display

这个模块提供控制 Pygame 显示界面（display）的各种函数。Pygame 的 Surface 对象即可显示为一个窗口，也可以全屏模式显示。当你创建并显示一个常规的 Surface 对象后，在该对象上的改变并不会立刻反映到可见屏幕上，你必须使用两个更新函数之一来显示改动后的画面。

显示的原点是 (x=0, y=0) 的位置，及屏幕的左上角，坐标轴向右下角增长。

Pygame 的 display 事实上可以有几种初始化的方式。默认情况下，display 作为一个软件驱动的帧缓冲区。除此之外，你可以使用硬件加速和 OpenGL 支持的特殊模块。这些是通过给 pygame.display.set\_mode() 传入 flags 参数来控制的。

Pygame 在任何时间内都只允许有一个显示界面。使用 pygame.display.set\_mode() 创建的新显示界面会自动替换掉旧的。如果需要精确控制像素格式或显示分辨率，使用 pygame.display.mode\_ok()，pygame.display.list\_modes()，和 pygame.display.Info() 来查询显示界面相关的信息。

一旦 Surface 对象的显示界面被创建出来，这个模块的函数就只影响当前的显示界面。如果该模块未初始化，Surface 对象也会变为“非法”。如果新的显示模式被设置，当前的 Surface 对象将会自动切换到新的显示界面。

当一个新的显示模式被设置，会在 Pygame 的事件队列中放入几个相关事件。当用于希望关闭程序时，pygame.QUIT 事件会被发送；当显示界面获得和失去焦点时，窗口会得到 pygame.ACTIVEEVENT 事件；如果显示界面设置了 pygame.RESIZABLE 标志，那么当用户调整窗口尺寸时，pygame.VIDEORESIZE 事件会被发送；硬件显示指当接收到 pygame.VIDEOEXPOSE 事件时，将部分需要被重绘的窗口直接绘制到屏幕上。

一些显示环境拥有自动拉伸所有窗口的选项。当该选项被启动时，自动拉伸会扭曲 Pygame 窗口的外观。在 Pygame 的例子目录中，有一个演示代码（prevent\_display\_stretching.py）展示如何在微软系统（Vista 以上系统）中关闭 Pygame 显示的自动拉伸属性。

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 作用 | 参数 | 返回值 |
| pygame.display.init() | 初始化 display 模块 | 无 | None |
| pygame.display.quit() | 结束 display 模块 | 无 | None |
| pygame.display.get\_init() | 如果 display 模块已经初始化，返回 True | 无 | True if display 模块已经初始化 |
| pygame.display.set\_mode() | 初始化一个准备显示的窗口或屏幕 | 无 | Surface界面 |
| pygame.display.get\_surface() | 获取当前显示的 Surface 对象 | (resolution=(0,0), flags=0, depth=0) | Surface界面 |
| pygame.display.flip() | 更新整个待显示的 Surface 对象到屏幕上 | 无 | None |
| pygame.display.update() | 更新部分软件界面显示 | update(rectangle=None)  update(rectangle\_list) | None  None |
| pygame.display.get\_driver() | 获取 Pygame 显示后端的名字 | 无 | name |
| pygame.display.Info() | 创建有关显示界面的信息对象 | 无 | VideoInfo |
| pygame.display.get\_wm\_info() | 获取关于当前窗口系统的信息 | 无 | dict |
| pygame.display.list\_modes() | 获取全屏模式下可使用的分辨率 | (depth=0, flags=pygame.FULLSCREEN) | list  -1 if 支持所有分辨率（窗口模式） |
| pygame.display.mode\_ok() | 为显示模式选择最合适的颜色深度 | (size, flags=0, depth=0) | depth  0 if 不支持当前显示模式 |
| pygame.display.gl\_get\_attribute() | 获取当前显示界面 OpenGL 的属性值 | (flag) | value |
| pygame.display.gl\_set\_attribute() | 设置当前显示模式的 OpenGL 属性值 | (flag, value) | None |
| pygame.display.get\_active() | 当前显示界面显示在屏幕上时返回 True | 无 | True if当前显示界面显示在屏幕上  False if显示界面最小化 |
| pygame.display.iconify() | 最小化显示的 Surface 对象 | 无 | True if该函数调用成功 |
| pygame.display.toggle\_fullscreen() | 切换全屏模式和窗口模式，这个函数只在 unix x11 显示驱动下工作。在大多数情况下，建议调用 pygame.display.set\_mode() 创建一个新的显示模式进行切换。 | 无 | True if该函数调用成功 |
| pygame.display.set\_gamma() | 修改硬件显示的 gama 坡道 | (red, green=None, blue=None) | True if该函数调用成功 |
| pygame.display.set\_gamma\_ramp() | 自定义修改硬件显示的 gama 坡道，使用自定义表设置硬件驱动显示的红色、绿色和蓝色伽马坡道。每个参数必须是 256 位整数的列表。每位整数应该在 0 和 0xffff 之间。不是所有的操作系统和硬件都支持伽马坡道。 | (red, green, blue) | True if函数修改成功 |
| pygame.display.set\_icon() | 修改显示窗口的图标 | (Surface) | None |
| pygame.display.set\_caption() | Set the current window caption | (title字符串, icontitle字符串=None) | None |
| pygame.display.get\_caption() | 设置当前的窗口标题 | 无 | (title, icontitle) |
| pygame.display.set\_palette() | 设置显示Surface的调色板 | (palette=None) | None |

函数详解

### pygame.display.init()

初始化 display 模块。

初始化后，Pygame 将自动从几个内部的显示后端中选择一个。显示模式由平台和当前用户权限决定。在 display 模块被初始化之前，可以通过环境变量 SDL\_VIDEODRIVER 设置哪一个显示后端将被使用。具有多种显示后端的系统如下：

Windows : windib, directx

Unix : x11, dga, fbcon, directfb, ggi, vgl, svgalib, aalib

在一些平台上，可以将 Pygame 的 display 嵌入到已经存在的窗口中。如果这么做，环境变量 SDL\_WINDOWID 必须被设置为一个包含窗口 ID 或句柄的字符串。当 Pygame 的 display 被初始化的时候，将检测环境变量。注意，在一个运行的窗口嵌入 display 会产生许多奇怪的副作用。

多次调用该函数并没有任何问题，但也不会有什么效果。

### pygame.display.quit()

结束 display 模块。

这个函数会关闭整个 display 模块。这将意味着任何一个活跃的显示界面都将被关闭。当主程序退出时，该函数也会被自动调用。多次调用该函数并没有任何问题，但也不会有什么效果。

### pygame.display.get\_init()

如果 display 模块已经初始化，返回 True。

### pygame.display.set\_mode()

初始化一个准备显示的窗口或屏幕。

这个函数将创建一个 Surface 对象的显示界面。传入的参数用于指定显示类型。最终创建出来的显示界面将是最大可能地匹配当前操作系统。

resolution 参数是一个二元组，表示宽和高。flags 参数是附件选项的集合。depth 参数表示使用的颜色深度。

返回的 Surface 对象可以像常规的 Surface 对象那样去绘制，但发生的改变最终会显示到屏幕上。

如果没有传入 resolution 参数，或者使用默认设置 (0, 0)，且 Pygame 使用 SDL1.2.10 以上版本，那么创建出来的 Surface 对象将与当前屏幕用户一样的分辨率。如果只有宽或高其中一项被设置为 0，那么 Surface 对象将使用屏幕分辨率的宽或高代替它。如果 SDL 版本低于 1.2.10，那么将抛出异常。

通常来说，最好的做法是不要传递 depth 参数。因为默认 Pygame 会根据当前操作系统选择最好和最快的颜色深度。如果你的游戏确实需要一个特殊的颜色格式，那么你可以通过控制 depth 参数来实现。Pygame 将为模拟一个非现成的颜色深度而耗费更多的时间。

当使用全屏显示模式的时候，有时候无法完全匹配到需要的分辨率。在这种情况下，Pygame 将自动选择最匹配的分辨率使用，而返回的 Surface 对象将保持与需求的分辨率一致。

flags 参数指定你想要的显示类型。提供几个选择给你，你可以通过位操作同时使用多个类型（管道操作符 "|"）。如果你传入 0 或没有传入 flags 参数，默认会使用软件驱动窗口。这儿是 flags 参数提供的几个可选项：

**pygame.set\_mode**会返回一个Surface对象，代表了在桌面上出现的那个窗口，三个参数第一个为元祖，代表分辨率（必须）；第二个是一个标志位，具体意思见下表，如果不用什么特性，就指定0；第三个为色深。

set\_mode(resolution = (0,0,), flags=0, depth=0)

|  |  |
| --- | --- |
| 标志位 flags | 功能 |
| FULLSCREEN | 创建一个全屏窗口 |
| DOUBLEBUF | 创建一个“双缓冲”窗口，建议在HWSURFACE或者OPENGL时使用 |
| HWSURFACE | 创建一个硬件加速的窗口，必须和FULLSCREEN同时使用 |
| OPENGL | 创建一个OPENGL渲染的窗口 |
| RESIZABLE | 创建一个可以改变大小的窗口 |
| NOFRAME | 创建一个没有边框的窗口 |

举个栗子： # 在屏幕中创建一个 700 \* 400 的窗口 screen\_width=700 screen\_height=400 screen=pygame.display.set\_mode([screen\_width, screen\_height])复制代码

### pygame.display.get\_surface()

获取当前显示的 Surface 对象。

返回当前显示的 Surface 对象。如果没有设置任何显示模式，那么返回 None。

### pygame.display.flip()

更新整个待显示的 Surface 对象到屏幕上。 这个函数将更新整个显示界面的内容。如果你的显示模式使用了 pygame.HWSURFACE（硬件加速）和 pygame.DOUBLEBUF（双缓冲）标志，那么将等待垂直会扫并切换显示界面。如果你使用不同类型的显示模式，那么它将简单的更新整个显示界面的内容。

当使用 pygame.OPENGL（使用 OPENGL 渲染）显示模式时，将创建一个 gl 缓冲切换区。

小甲鱼温馨提示：垂直回扫是与视频显示相关的时间测量，它代表了一个帧的结束和下一帧的开始时间之间的时间间隔。

### pygame.display.update()

更新部分软件界面显示。

update(rectangle=None) -> None

update(rectangle\_list) -> None

这个函数可以看作是 pygame.display.flip() 函数在软件界面显示的优化版。它允许更新屏幕的部分内容，而不必完全更新。如果没有传入任何参数，那么该函数就像 pygame.display.flip() 那样更新整个界面。

你可以传递一个或多个矩形区域给该函数。一次性传递多个矩形区域比多次传递更有效率。如果传入的是一个空列表或者 None，那么将忽略参数。

该函数不能在 pygame.OPENGL 显示模式下调用，否则会抛出异常。

### pygame.display.get\_driver()

获取 Pygame 显示后端的名字。

初始化的时候，Pygame 会从多个可用的显示后端中选择一个。这个函数返回显示后端内部使用的名字。可以用来提供有关显示性能加速的一些信息。可以参考 pygame.display.set\_mode() 的 SDL\_VIDEODRIVER 环境变量。

### pygame.display.Info()

创建有关显示界面的信息对象。

创建一个对象，包含对当前图形环境一些属性的描述。在一些平台上，如果这个函数在 pygame.display.set\_mode() 前被调用，可以提供一些关于默认显示模式的信息。也可以在设置完显示模式后调用该函数，以确认显示选项是否如愿以偿。

返回的 VideoInfo 对象包含以下这些属性：

|  |  |
| --- | --- |
| 属性 | 含义 |
|  |  |
| hw | 如果是 True，则表示启用硬件加速 |
| wm | 如果是 True，则表示显示窗口模式 video\_mem 表示显存是多少兆字节（mb），0 表示不清楚 |
| bitsize | 表示每个像素存放多少位 |
| bytesize | 表示每个像素存放多少字节 |
| masks 4 | 个值用于打包像素的 RGBA 值 |
| shifts 4 | 个值用于打包像素的 RGBA 值 |
| losses | 4 个值用于打包像素的 RGBA 值 |
| blit\_hw | 如果是 True，则表示加速硬件驱动的 Surface 对象绘制 |
| blit\_hw\_CC | 如果是 True，则表示加速硬件驱动的 Surface 对象 colorkey 绘制 |
| blit\_hw\_A | 如果是 True，则表示加速硬件驱动的 Surface 对象 pixel alpha 绘制 |
| blit\_sw | 如果是 True，则表示加速软件驱动的 Surface 对象绘制 |
| blit\_sw\_CC | 如果是 True，则表示加速软件驱动的 Surface 对象 colorkey 绘制 |
| blit\_sw\_A | 如果是 True，则表示加速软件驱动的 Surface 对象 pixel alpha 绘制 |
| current\_w, current\_h | 1.表示当前显示模式的宽和高（如果在 display.set\_mode() 前被调用，则表示当前桌面的宽和高）  2. current\_w, current\_h 在 Pygame 1.8.0 以后，SDL 1.2.10 以后才支持  3. -1 表示错误，或者 SDL 版本太旧 |

### pygame.display.get\_wm\_info()

获取关于当前窗口系统的信息。

创建一个由操作系统填充数据的字典。一些操作系统可能不会往里边填充信息，则返回一个空字典。大多数平台将返回一个 "window" 键，对应的值是当前显示界面的系统 ID。

### pygame.display.list\_modes()

获取全屏模式下可使用的分辨率。

这个函数返回一个列表，包含指定颜色深度所支持的所有分辨率。如果显示模式非全屏，则返回一个空列表。如果返回 -1 表示支持任何分辨率（类似于窗口模式）。返回的列表由大到小排列。

如果颜色深度是 0，SDL 将选择当前/最合适的颜色深度显示。flags 参数默认值是 pygame.FULLSCREEN，但你可能需要添加额外的全屏模式标志。

### pygame.display.mode\_ok()

为显示模式选择最合适的颜色深度。这个函数使用与 pygame.display.set\_mode() 函数一样的参数。一般用于判断一个显示模式是否可用。如果显示模式无法设置，则返回 0。正常情况下将会返回显示需求的像素深度。

通常不用理会 depth 参数，除非一些支持多个显示深度的平台，它会提示哪个颜色深度是更合适的。

最有用的 flags 参数是 pygame.HWSURFACE，pygame.DOUBLEBUF 和 pygame.FULLSCREEN。如果这些标志不支持，那么该函数会返回 0。

### pygame.display.gl\_get\_attribute()

获取当前显示界面 OpenGL 的属性值。

在调用设置了 pygame.OPENGL 标志的 pygame.display.set\_mode() 函数之后，检查 OpenGL 的属性值不失为一个好的习惯。参考 pygame.display.gl\_set\_attribute() 关于合法标志的列表。

### pygame.display.gl\_set\_attribute()

设置当前显示模式的 OpenGL 属性值。

当调用设置了 pygame.OPENGL 标志的 pygame.display.set\_mode() 函数时，Pygame 会自动设置 OpenGL 的一些属性值，例如颜色和双缓冲区。OpenGL 其实还提供了其他一些属性值供你控制。在 flag 参数中传入属性名，并将其值设置在 value 参数中。这个函数必须在 pygame.display.set\_mode() 前设置。

这些 OPENGL 标志是：

GL\_ALPHA\_SIZE, GL\_DEPTH\_SIZE, GL\_STENCIL\_SIZE, GL\_ACCUM\_RED\_SIZE, GL\_ACCUM\_GREEN\_SIZE, GL\_ACCUM\_BLUE\_SIZE, GL\_ACCUM\_ALPHA\_SIZE, GL\_MULTISAMPLEBUFFERS, GL\_MULTISAMPLESAMPLES, GL\_STEREO

### pygame.display.get\_active()

当前显示界面显示在屏幕上时返回 True。

pygame.display.set\_mode() 函数被调用之后，Surface 对象将被显示在屏幕上。大多数窗口都支持隐藏，如果显示的 Surface 对象被隐藏和最小化，那么该函数将返回 False。

### pygame.display.iconify()

最小化显示的 Surface 对象。

将显示的 Surface 对象最小化或隐藏。并不是所有的操作系统都支持最小化显示界面。如果该函数调用成功，返回 True。

当显示界面最小化时，pygame.display.get\_active() 返回 False。事件队列将接收到 ACTIVEEVENT 事件。

### pygame.display.toggle\_fullscreen()

切换全屏模式和窗口模式。

toggle\_fullscreen() -> bool

切换全屏模式和窗口模式。这个函数只在 unix x11 显示驱动下工作。在大多数情况下，建议调用 pygame.display.set\_mode() 创建一个新的显示模式进行切换。

### pygame.display.set\_gamma()

修改硬件显示的 gama 坡道。

设置硬件驱动显示的红色、绿色和蓝色伽马值。如果没有传递 green 和 blue 参数，它们将与 red 值相等。不是所有的操作系统和硬件都支持伽马坡道。如果函数修改成功，则返回 True。

伽马值为 1.0 创建一个线性颜色表，较低的值会使屏幕变暗，较高的值会使屏幕变亮。

### pygame.display.set\_gamma\_ramp()

自定义修改硬件显示的 gama 坡道使用自定义表设置硬件驱动显示的红色、绿色和蓝色伽马坡道。每个参数必须是 256 位整数的列表。每位整数应该在 0 和 0xffff 之间。不是所有的操作系统和硬件都支持伽马坡道。如果函数修改成功，则返回 True。

### pygame.display.set\_icon()

修改显示窗口的图标。

设置显示窗口执行时的图标。所有的操作系统默认都是以简单的 Pygame LOGO 作为图标。

你可以传入任何 Surface 对象作为图标，但大多数操作系统要求图标的大小是 32 \* 32。图标可以设置 colorkey 透明度。

一些操作系统不允许修改显示中的窗口图标。对于这类操作系统，该函数需要再调用 pygame.display.set\_mode() 前先创建并设置图标。

### pygame.display.set\_caption()

设置当前窗口的标题栏。

如果显示窗口拥有一个标题栏，这个函数将修改窗口标题栏的文本。一些操作系统支持最小化窗口时切换标题栏，通过设置 icontitle 参数实现。

### pygame.display.get\_caption()

获取当前窗口的标题栏。

返回当前窗口的标题栏和最小化标题栏，通常这两个值是一样的。

### pygame.display.set\_palette()

设置显示界面的调色板。

这个函数将修改显示界面的 8 位调色板。这不会改变 Surface 对象实际的调色板，仅用于 Surface 对象的显示。如果没有传入参数，将恢复系统默认调色板。调色板是一组 RGB 三元组序列。

## 矩形对象Rect

Pygame通过Rect对象存储和操作矩形区域。一个Rect对象可以由left, top, width, height几个值创建。Rect也可以是由Pygame的对象所创建，它们拥有一个属性rect

任何需要一个Rect对象作为参数的pygame函数都可以使用以上值构造一个Rect，这使得作为参数传递的同时创建Rect成为可能。

Rect对象的大部分方法在修改矩形的位置、尺寸后，会返回一个Rect拷贝，但有的方法比较特殊，它们会“原地”修改Rect对象，以ip作为后缀。

x,y

top, left, bottom, right (单值)

topleft, bottomleft, topright, bottomright (二维坐标点)

midtop, midleft, mid bottom, midright (二维坐标点)

center (二维坐标点), centerx, centery

size(宽和高), width, height给这三个属性赋值将改变矩形的尺寸

坐标值为整数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 作用 | 参数 | 返回值 |
| pygame.Rect.copy() | 拷贝Rect对象 | 无参数 | 新的Rect对象 |
| pygame.Rect.move() | 移动Rect对象 | (x, y) | 新的Rect对象 |
| pygame.Rect.move\_ip() | 原地移动Rect对象 | (x, y) | None |
| pygame.Rect.inflate | 放大和缩小Rect对象的尺寸 | (x, y)表示x和y放大或缩小多少个像素 | 新的Rect对象 |
| pygame.Rect.inflate\_ip() | 原地放大和缩小Rect对象的尺寸 | (x, y)表示x和y放大或缩小多少个像素 | None |
| pygame.Rect.clamp() | 将一个Rect对象移动到另一个对象的中心，返回一个新的Rect对象，范围是以Rect参数指定的对象为中心，保持原始Rect对象的尺寸不变，如果原始Rect对象的尺寸比Rect参数要大，那么保持中心重叠，尺寸不变 | Rect对象 | 新的Rect对象 |
| pygame.Rect.clamp\_ip() | 原地将一个Rect对象移动到另一个对象的中心 | Rect对象 | None |
| pygame.Rect.clip() | 获取两个Rect对象互相重叠的部分，如果没有任何重叠，则返回一个(0,0,0,0)的  对象 | Rect对象 | 新的Rect对象 |
| pygame.Rect.union() | 将两个Rect对象合并 | Rect对象 | 新的Rect对象 |
| pygame.Rect.union\_ip() | 原地将两个Rect对象合并，会产生多余空间（合并图层） | Rect对象 | None |
| pygame.Rect.unionall() | 将多个Rect对象合并 | Rect对象序列 | 新的Rect对象 |
| pygame.Rect.unionall\_ip() | 原地将将多个Rect对象合并 | Rect对象序列 | None |
| pygame.Rect.fit() | 按照一定的宽高比调整Rect对象，参数中的Rect对象按照原始Rect对象宽高比调整 | Rect对象 | 新的Rect对象 |
| pygame.Rect.normalize() | 翻转Rect的（如果尺寸是负数） | Rect对象 | None |
| pygame.Rect.contains() | 检测一个Rect对象是否完全包含在该Rect对象内 | Rect对象 | True or False |
| pygame.Rect.collidepoint() | 检测一个点是否完全包含在该Rect对象内，边上right bottom并不认为是包含 | (x,y)或x,y | True or False |
| pygame.Rect.colliderect() | 检测两个Rect对象是否重叠 | Rect | True or False |
| pygame.Rect.collidelist() | 检测该Rect对象是否与列表中的任何一个Rect对象有交集 | Rect list | 索引if True ，-1 if False |
| pygame.Rect.collidelistall() | 检测该Rect对象是否与字典中的任何一个Rect对象有交集 | Rect list | 元素列表True，空列表 False |
| pygame.Rect.collidedict() | 检测该Rect对象是否与字典中的任何一个Rect对象有交集 | Rect dictionary | (key,value)第一个有相交的矩形所在字典中的键和值if True，None if False |
| pygame.Rect.collidedictall() | 检测该Rect对象是否与字典中的每个Rect对象有交集 | Rect dictionary | [(key,value],] if True, 空字典if False |

## 图像变换的方法

pygame.transform.rotate(对象,逆时针的度数)

|  |  |
| --- | --- |
| flip | 上下、左右翻转图像 |
| scale | 缩放图像（快速） |
| rotate | 旋转图像 |
| rotozome | 缩放并旋转图像 |
| scale2x | 快速放大一倍图像 |
| Smoothscale | 平滑缩放图像（精准） |
| Chop | 裁剪图像 |

## 事件event

用于处理事件与事件队列的 Pygame 模块。

函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 作用 | 参数 | 返回值 |
| pygame.event.pump() | 让pygame内部自动处理事件 | 无 | None |
| pygame.event.get() | 从队列中获取事件 | 无  (type)  (typelist) | event 列表  event 列表  event 列表 |
| pygame.event.poll() | 从队列中获取一个事件 | 无 | EventType instance  pygame.NOEVENT if 事件队列为空 |
| pygame.event.wait() | 等待并从队列中获取一个事件 | 无 | EventType instance |
| pygame.event.peek() | 检测某类型事件是否在队列中 | (type)  (typelist) | True if参数指定的类型的事件存在于队列中 |
| pygame.event.clear() | 从队列中删除所有的事件 | 无  (type)  (typelist) | None |
| pygame.event.event\_name() | 通过 id 获得该事件的字符串名字 | type | 字符串 |
| pygame.event.set\_blocked() | 控制哪些事件禁止进入队列 | (type)  (typelist)  None | None |
| pygame.event.set\_allowed() | 控制哪些事件允许进入队列 | (type)  (typelist)  None | None |
| pygame.event.get\_blocked() | 检测某一类型的事件是否被禁止进入队列 | (type) | True if被禁止 |
| pygame.event.set\_grab() | 控制输入设备与其他应用程序的共享 | (Trur or False) | None |
| pygame.event.get\_grab() | 检测程序是否共享输入设备 | 无 | None |
| pygame.event.post() | 放置一个新的事件到队列中 | (Event) |  |
| pygame.event.Event() | 创建一个新的事件对象 | (type, dict)  (type, \*\*attributes) | EventType instance  EventType instance |
| pygame.event.EventType | 代表 SDL 事件的 Pygame 对象 |  |  |

Pygame 通过事件队列控制所有的时间消息。该模块中的程序将帮你管理事件队列。输入队列很大程度依赖于 pygame 的 display 模块。如果 display 没有被初始化，显示模式没有被设置，那么事件队列就还没有开始真正工作。

常规的队列是由 pygame.event.EventType 定义的事件对象的组成，有多种方法来访问里边的事件对象：从简单的检测事件是否存在，到直接从栈中获取它们。

所有事件都有一个类型标识符，这个标识符对应的值定义在 NOEVENT 到 NUMEVENTS 之间（小甲鱼温馨提示：类似于 C 语言的宏定义，明白？）。用户可以自行定义事件，但类型标识符的值应该高于或等于 USEREVENT。

获取各种输入设备的状态，推荐你直接通过它们相应的模块（mouse，key 和 joystick）提供的函数访问，而不是通过事件队列； 如果你使用此函数，请记住，Pygame 需要通过一些方式与系统的窗口管理器和平台的其他部分进行通信。为了保持 Pygame 和系统同步，你需要调用 pygame.event.pump() 确保实时更新，你将在游戏的每次循环中调用这个函数。

事件队列提供了一些简单的过滤。通过阻止某些事件进入事件队列，可以略微提高游戏的性能（小甲鱼温馨提示：因为这样事件队列的尺寸就会小一些，所以说可以略微提升性能）。使用 pygame.event.set\_allowed() 和 pygame.event.set\_blocked() 来控制某些事件是否允许进入事件队列。默认所有事件都会进入事件队列。

事件子系统应该在主线程被调用。如果你希望从其他线程中投递事件消息进入事件队列，请使用 fastevent 包。

Joysticks（游戏手柄）只有在设备初始化后才会发送事件。

一个 EventType 事件对象包含一个事件类型标识符和一组成员数据（事件对象不包含方法，只有数据）。EventType 对象从 Python 的事件队列中获得，你也可以使用 pygame.event.Event() 函数创建自定义的新事件。

由于 SDL 的事件队列限制了事件数量的上限（标准的 SDL 1.2 限制为 128），所以当队列已满时，新的事件将会被扔掉。为了防止丢失事件消息，尤其是代表退出的输入事件（因为当用户点击退出按钮没有反应，往往会被认为“死机”了），你的程序必须定期检测事件，并对其进行处理。

为了加快事件队列的处理速度，可以使用 pygame.event.set\_blocked() 函数阻止一些我们不关注的事件进入队列中。

所有的 EventType 实例对象都拥有一个事件类型标识符，属性名是 type。你也可以通过事件对象的 \_\_dict\_\_ 属性来完全访问其他属性。所有其他成员属性的值都是通过事件对象的字典来传递。

在做调试和实验时，你可以打印事件对象以及相应的类型和成员。来自系统的事件都有一个事件类型和对应的成员属性，下边是每个事件类型以及对应的成员属性列表：

事件类型成员属性

|  |  |  |
| --- | --- | --- |
| 事件 | 产生途径 | 参数 |
| QUIT | 用户按下关闭按钮 | none |
| ATIVEEVENT | Pygame被激活或者隐藏 | gain, state |
| KEYDOWN | 键盘被按下 | unicode, key, mod |
| KEYUP | 键盘被放开 | key, mod |
| MOUSEMOTION | 鼠标移动 | pos, rel, buttons |
| MOUSEBUTTONDOWN | 鼠标按下 | pos, button |
| MOUSEBUTTONUP | 鼠标放开 | pos, button |
| JOYAXISMOTION | 游戏手柄(Joystick or pad)移动 | joy, axis, value |
| JOYBALLMOTION | 游戏球(Joy ball)?移动 | joy, axis, value |
| JOYHATMOTION | 游戏手柄(Joystick)?移动 | joy, axis, value |
| JOYBUTTONDOWN | 游戏手柄按下 | joy, button |
| JOYBUTTONUP | 游戏手柄放开 | joy, button |
| VIDEORESIZE | Pygame窗口缩放 | size, w, h |
| VIDEOEXPOSE | Pygame窗口部分公开(expose)? | None |
| USEREVENT | 触发了一个用户事件 | Code |

事件支持等值比较。如果两个事件具有相同的类型和属性值，那么认为两个事件是相等的。（Pygame 1.9.2 新增加的）

函数详解

### pygame.event.pump()

让 Pygame 内部自动处理事件。

对于游戏中的每一帧，你都需要通过某种形式去调用事件队列，这将确保你的程序在内部可以与操作系统的其他部分进行交互。如果你不打算使用其他事件函数，那么你应该调用 pygame.event.pump()，这将允许 Pygame 内部自动处理事件。

如果你的程序始终通过其他 event 模块的函数处理队列中的事件，那么该函数是没必要的。

事件队列中的内部处理是非常重要的事情。主窗口可能需要重新绘制或对系统做出响应。如果你太长时间没有调用事件队列，系统可能会认定你的程序已锁定（假死）。

### pygame.event.get()

从队列中获取事件。

这将获取并从队列中删除事件。如果指定一个或多个 type 参数，那么只获取并删除指定类型的事件。

请注意，如果你只从队列中获取和删除指定的事件，那么久而久之，队列可能被你不关注的事件所填满。

### pygame.event.poll()

从队列中获取一个事件。

从队列中返回并删除一个事件。

如果事件队列为空，那么会立刻返回类型为 pygame.NOEVENT 的事件。

### pygame.event.wait()

等待并从队列中获取一个事件。

从队列中返回并删除一个事件。如果队列为空，那么该函数将持续等待直至队列中有一个事件。当程序在等待时，它将保持睡眠状态。这对于希望与其他应用程序共享系统来说，是非常重要的。

### pygame.event.peek()

检测某类型事件是否在队列中。

如果参数指定的类型的事件存在于队列中，返回 True。

如果参数指定多个类型的事件，则只需队列中拥有其中的任何一个事件便返回 True。

### pygame.event.clear()

从队列中删除所有的事件。

从队列中删除所有的事件，如果通过参数指定事件的类型，则删除该类型的所有事件。该函数的效果跟 pygame.event.get() 相同，只是没有返回任何东西。当处理完关注的事件后，清空整个队列可以提高一些效率。

### pygame.event.event\_name()

通过 id 获得该事件的字符串名字。

Pygame 通过整数 id 代表事件类型。如果你需要将这些类型的事件展示给用户看，那么你需要将它们转换成字符串（一堆数字谁知道你想表示啥？）。该函数将返回事件类型对应的字符串名字。返回值是以单词大写的样式（小甲鱼温馨提示：DanCiDaXieDe）。

### pygame.event.set\_blocked()

控制哪些事件禁止进入队列。

参数指定的类型的事件均不允许出现在事件队列中。默认是允许所有事件进入队列。多次禁止同一类型的事件并不会引发什么问题。

如果传入 None，则表示允许所有的事件进入队列。

### pygame.event.set\_allowed()

控制哪些事件允许进入队列。

参数指定的类型的事件均允许出现在事件队列中。默认是允许所有事件进入队列。多次允许同一类型的事件并不会引发什么问题。

如果传入 None，则表示禁止所有的事件进入队列。

### pygame.event.get\_blocked()

检测某一类型的事件是否被禁止进入队列。

如果参数指定类型的事件被禁止进入队列，则返回 True。

### pygame.event.set\_grab()

控制输入设备与其他应用程序的共享。

当你的程序运行在窗口环境中，它将与其他拥有焦点的应用程序分享鼠标和键盘设备的输入。如果你的程序设置事件独占为 True，那么你的程序将锁定所有的输入（小甲鱼温馨提示：不共享给其他程序了）。

最好不要经常独占输入，因为这将阻止用户在操作系统上的其他操作。

### pygame.event.get\_grab()

检测程序是否共享输入设备。

当程序独占输入事件时，返回 True。使用 pygame.event.set\_grab() 函数控制这一状态。

### pygame.event.post()

放置一个新的事件到队列中。

该函数将放置一个新的事件到事件队列的末端。这些事件将最迟被其他队列函数获取。

该函数通常用于放置 pygame.USEREVENT（用户自定义事件）事件到队列中。尽管你可以放置所有类型的事件，但你需要确保为系统事件类型相应的属性传递合适的值。

如果 SDL 事件队列已满，将抛出 pygame.error 异常。

### pygame.event.Event()

创建一个新的事件对象。

根据参数给定的类型创建一个新的事件。dict 参数指定事件的属性以及相应的值。

### class pygame.event.EventType

代表 SDL 事件的 Pygame 对象。

### pygame.event.EventType.type

SDL event type identifier.

### pygame.event.EventType.\_\_dict\_\_

vent object attribute dictionary

用于代表 SDL 事件的 Pygame 对象。通过 pygame.event.Event() 创建用户自定义事件。EventType 类型并不是直接可以被调用的。EventType 实例对象支持属性赋值和删除。

### type

SDL 事件类型标识符。

type -> int

只读。预定义事件标识符是 QUIT 和 MOUSEMOTION 等。对于用于创建的事件对象，这是传递给 pygame.event.Event() 的 type 参数。

### \_\_dict\_\_

事件对象的属性字典。

\_\_dict\_\_ -> dict只读。事件类型指定的属性。例如，KEYDOWN 事件包含 unicode，key 和 mod 属性。

可变属性是 Pygame 1.9.2 新增加的。

## 图像image

用于图像传输的 Pygame 模块。

函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 作用 | 参数 | 返回值 |
| pygame.image.load() | 从文件加载新图片 | (filename)  (fileobj, namehint=””) | Surface  Surface |
| pygame.image.save() | 将图像保存到磁盘上 | (Surface, filename) | None |
| pygame.image.get\_extended() | 检测是否支持载入扩展的图像格式 | 无 | True if 支持 |
| pygame.image.tostring() | 将图像转换为字符串描述 | (Surface, format, flipped=False) | string |
| pygame.image.fromstring() | 将字符串描述转换为图像 | (string, size, format, flipped=False) | Surface |
| pygame.image.frombuffer() | 创建一个与字符串描述共享数据的 Surface 对象 | (string, size, format) -> Surface | Surface |
| pygame.image.load() | 从文件加载新图片 |  |  |

image 模块包含了加载和保存图像的函数，同时转换为 Surface 对象支持的格式。

注意：没有 Image 类；当一个图像被成功载入后，将转换为 Surface 对象。Surface 对象允许你在上边画线、设置像素、捕获区域等。

Image 是 Pygame 相当依赖的一个模块，支持载入的图像格式如下：

JPG,PNG,GIF（无动画）BMP,PCX,TGA（无压缩）TIF,LBM（和 PBM）PBM（和 PGM，PPM）XPM

支持保存为以下格式：

BMP, TGA, PNG, JPEG

其中，保存为 PNG 和 JPEG 格式是 Pygame 1.8 新增加的。

函数详解

### pygame.image.load()

从文件加载新图片。

从文件加载一张图片，你可以传递一个文件路径或一个 Python 的文件对象。

Pygame 将自动判断图像的格式（比如 GIF 或位图）并创建一个新的 Surface 对象。有时它可能需要知道文件的后缀名（比如 GIF 图像应该以 ".gif" 为后缀）。如果你传入原始文件对象，你需要传入它对应的文件名到 namehint 参数中。

返回的 Surface 对象将包含与源文件相同的颜色格式，colorkey 和 alpha 透明度通道。你通常需要调用 Surface.convert() 函数进行转换，这样可以使得在屏幕上绘制的速度更快。

对于含有 alpha 通道的图片（支持部分位置透明，像 PNG 图像），需要使用 surface.convert\_alpha() 函数进行转换。

在某些环境下，Pygame 可能无法支持上述所有的图像格式，但至少无压缩的 BMP 格式是支持的。你可以调用 pygame.image.get\_extended() 函数，如果返回 True，说明可以加载上述的格式（包含 PNG，JPG 和 GIF）

你应该使用 os.path.join() 提高代码的兼容性： asurf = pygame.image.load(os.path.join('data', 'FishC.png'))复制代码

### pygame.image.save()

将图像保存到磁盘上。

该函数将保存 Surface 对象到磁盘上，支持存储为 BMP，TGA，PNG 或 JPEG 格式的图像。如果 filename 没有指定后缀名，那么默认是保存为 TGA 格式。TGA 和 BMP 格式是无压缩的文件。

保存为 PNG 和 JPEG 格式是 Pygame 1.8 新增的。

### pygame.image.get\_extended()

检测是否支持载入扩展的图像格式。

如果 Pygame 支持上述所有的扩展图像格式，则返回 True。

### pygame.image.tostring()

将图像转换为字符串描述。

将图像转换为一个字符串描述，可以被 Python 的其他图像模块通过 "fromstring" 转换回图像。一些 Python 图像模块喜欢“自下而上”的存储格式（例如 PyOpenGL）。如果 flipped 参数为 True，那么字符串将会垂直翻转以适用这类图像模块。

format 参数可以是下表中任何一个字符串。注意：只有 8 位的 Surface 对象可以使用 "P" 格式。其他格式可以用于任何 Surface 对象上。

|  |  |
| --- | --- |
| 字符串 | 含义 |
| P | 8 位调色板的 Surface 对象 |
| RGB | 24 位图像 |
| RGBX | 32 位图像，不留空白 |
| RGBA | 32 位图像，带 alpha 通道 ARGB 32 位图像，带 alpha 通道，并将 alpha 放在前边 |
| RGBA\_PREMULT | 32 位图像，通过 alpha 通道缩放 |
| ARGB\_PREMULT 32 位图像，通过 alpha 通道缩放，并将 alpha 放在前边 |  |

### pygame.image.fromstring()

将字符串描述转换为图像。

该函数的使用跟 pygame.image.tostring() 相似。size 参数是一对表示宽度和高度的数字。一旦新的 Surface 对象创建成功，你就可以删除字符串描述。

size 和 format 参数指定的数据需要跟字符串描述相符，否则将抛出异常。

更快地将图片转换到 Pygame，请参考 pygame.image.frombuffer() 函数。

pygame.image.frombuffer()

创建一个与字符串描述共享数据的 Surface 对象。

创建一个新的 Surface 对象，与字符串描述直接共享像素数据。该函数的使用跟 pygame.image.fromstring() 类似，但没法垂直翻转原始数据。

该函数的速度会比 pygame.image.fromstring() 快很多，因为该函数不需要申请和拷贝任何像素数据。

## 光标cursors

Pygame 中使用光标资源的模块。

函数

|  |  |
| --- | --- |
| pygame.cursors.compile() | 由纯字符串创建二进制光标数据 |
| pygame.cursors.load\_xbm() | 由一个xbm 文件载入光标数据 |

Pygame 提供对系统硬件光标的控制，并且只支持白色和黑色光标格式。你可以通过使用 pygame.mouse 内的方法控制光标。

cursors 模块包含载入和解码各种光标格式的方法。这些方法允许你简便地将你的光标存储成扩展文件，或者直接作为编码后的 python 字符串存在。

这个模块包含若干个标准光标。pygame.mouse.set\_cursor() 方法能够接收若干个参数。所有的参数以单个元组的形式存储，你可以用如下方式调用此方法： >>> pygame.mouse.set\_cursor(\*pygame.cursors.arrow)复制代码

这个模块也包含了一些格式化字符串形式的光标。在你使用到这些光标之前，你需要把相应字符串传递给 pygame.cursors.compile() 方法。你可以参照如下示例来调用：

>>> cursor = pygame.cursors.compile(pygame.cursors.textmarker\_strings)

>>> pygame.mouse.set\_cursor(\*cursor)复制代码

以下变量是可以被用作光标的位图：

|  |
| --- |
| pygame.cursors.arrow |
| pygame.cursors.diamond |
| pygame.cursors.broken\_x |
| pygame.cursors.tri\_left |
| pygame.cursors.tri\_right |

以下字符串可以通过 pygame.cursors.compile() 函数转换成光标位图：

pygame.cursors.thickarrow\_strings

pygame.cursors.sizer\_x\_strings

pygame.cursors.sizer\_y\_strings

pygame.cursors.sizer\_xy\_strings

函数详解

### pygame.cursors.compile()

由纯字符串创建二进制光标数据。

compile(strings, black=’X’, white=’.’, xor=’o’) -> data, mask

一串连续的字符串可以被用于创建对应系统光标的二进制光标数据。返回值要和 pygame.mouse.set\_cursor() 所需要的参数格式相同。

如果你正在创建自己的光标字符串，你可使用任何值来代表白色和黑色像素。一些系统允许你根据系统颜色自己设置一种特殊的切换色，也被称为 xor 色。如果系统不支持 xor 光标，则光标颜色将会变为纯黑色。

字符串的长度必须全部相等，而且可以被 8 整除。一个光标字符串设定示例，如下所示：

### pygame.cursors.load\_xbm()

由一个xbm 文件载入光标数据。

load\_xbm(cursorfile) -> cursor\_args

load\_xbm(cursorfile, maskfile) -> cursor\_args

该方法将根据 XBM 文件的某一个简单子集载入光标。XBM 文件从传统上是被用于保存 UNIX 系统内光标，它们是被用于代表一些简单图像的 ASCII 码。

一些时候，白色和黑色值将会分开在两个独立的 XBM 文件中。你可以通过传递第二个 maskfile 参数将两个图像载入到同一个光标中。

Cursorfile 和 maskfile 参数可以是带有 readlines 方法的 filenames 或者 filelike 对象。

返回值 cursor\_args 可以被直接传递给 pygame.mouse.set\_cursor() 方法。

## 键盘pygame.key

与键盘相关的 Pygame 模块。

函数

|  |  |
| --- | --- |
| pygame.key.get\_focused() | 当窗口获得键盘的输入焦点时返回 True |
| pygame.key.get\_pressed() | 获取键盘上所有按键的状态 |
| pygame.key.get\_mods() | 检测是否有组合键被按下 |
| pygame.key.set\_mods() | 临时设置某些组合键为被按下状态 |
| pygame.key.set\_repeat() | 控制重复响应持续按下按键的时间 |
| pygame.key.get\_repeat() | 获取重复响应按键的参数 |
| pygame.key.name() | 获取按键标识符对应的名字 |

该模块包含处理与键盘操作相关的函数。当键盘按键被按下和释放时，事件队列将获得 pygame.KEYDOWN 和 pygame.KEYUP 事件消息。这两个消息均包含 key 属性，是一个整数的 id，代表键盘上具体的某个按键。

pygame.KYEDOWN 事件还有个额外的属性 unicode 和 scancode。unicode 代表一个按键翻译后的 Unicode 编码，这包含 shift 按键和组合键。scancode 是扫描码，不同键盘间该值可能不同。不过这对于特殊按键像多媒体键的选择是有用的。

小甲鱼温馨提示：当键盘按下的时候，键盘会发送一个扫描码给系统。扫描码是键盘反馈哪一个按键被按下的方式，不同类型的键盘扫描码不同。再由系统调用相应的函数将其转换为统一的 Unicode 编码。

key 属性的值是一个数字，为了方便使用，Pygame 将这些数字定义为以下这些常量：

|  |  |  |
| --- | --- | --- |
| KeyASCII | ASCII | 描述 |
| K\_BACKSPACE \b | 退格键（Backspace） |  |
| K\_TAB | \t | 制表键（Tab） |
| K\_CLEAR |  | 清楚键（Clear） |
| K\_RETURN | \r | 回车键（Enter） |
| K\_PAUS |  | 暂停键（Pause） |
| K\_ESCAPE | ^[ | 退出键（Escape） |
| K\_SPACE |  | 空格键（Space） |
| K\_EXCLAIM | ! | 感叹号（exclaim） |
| K\_QUOTEDBL | " | 双引号（quotedbl） |
| K\_HASH | # | 井号（hash） |
| K\_DOLLAR | $ | 美元符号（dollar） |
| K\_AMPERSAND | & | and 符号（ampersand） |
| K\_QUOTE | ' | 单引号（quote） |
| K\_LEFTPAREN | ( | 左小括号（left parenthesis） |
| K\_RIGHTPAREN | ) | 右小括号（right parenthesis） |
| K\_ASTERISK | \* | 星号（asterisk） |
| K\_PLUS | + | 加号（plus sign） |
| K\_COMMA | , | 逗号（comma） |
| K\_MINUS | - | 减号（minus sign） |
| K\_PERIOD | . | 句号（period） |
| K\_SLASH | / | 正斜杠（forward slash） |
|  |  |  |
| K\_0 | 0 | 0 |
| K\_1 | 1 | 1 |
| K\_2 | 2 | 2 |
| K\_3 | 3 | 3 |
| K\_4 | 4 | 4 |
| K\_55 | 5 |  |
| K\_66 | 6 |  |
| K\_77 | 7 |  |
| K\_88 | 8 |  |
| K\_99 | 9 |  |
| K\_COLON |  | :冒号（colon） |
|  |  |  |
| K\_SEMICOLON | ; | 分号（semicolon） |
| K\_LESS | < | 小于号（less-than sign） |
| K\_EQUALS | = | 等于号（equals sign） |
| K\_GREATER | > | 大于号（greater-than sign） |
| K\_QUESTION | ? | 问号（question mark） |
| K\_AT | @ | at符号（at） |
| K\_LEFTBRACKET | [ | 左中括号（left bracket） |
| K\_BACKSLASH | \ | 反斜杠（backslash） |
| K\_RIGHTBRACKET | ] | 右中括号（right bracket） |
| K\_CARET | ^ | 脱字符（caret） |
| K\_UNDERSCORE | \_ | 下划线（underscore） |
| K\_BACKQUOTE | ` | 重音符（grave） |
| K\_a | a | a |
| K\_b | b | b |
| K\_c | c | c |
| K\_d | d | d |
| K\_e | e | e |
| WK\_f | f | f |
| K\_g | g | g |
| K\_h | h | h |
| K\_i | i | i |
| K\_j | j | j |
| K\_k | k | k |
| K\_l | l | l |
| K\_m | m | m |
| K\_n | n | n |
| K\_o | o | o |
| K\_p | p | p |
| K\_q | q | q |
| K\_rr | r | r |
| K\_s | s | s |
| K\_t | t | t |
| K\_u | u | u |
| K\_v | v | v |
| K\_w | w | w |
| K\_x | x | x |
| K\_y | y | y |
| K\_z | z | z |
| K\_DELETE |  | 删除键（delete） |
| K\_KP0 |  | 0（小键盘） |
| K\_KP1 |  | 1（小键盘） |
| K\_KP2 |  | 2（小键盘） |
| K\_KP3 |  | 3（小键盘） |
| K\_KP4 |  | 4（小键盘） |
| K\_KP5 |  | 5（小键盘） |
| K\_KP6 |  | 6（小键盘） |
| K\_KP7 |  | 7（小键盘） |
| K\_KP8 |  | 8（小键盘） |
| K\_KP9 |  | 9（小键盘） |
| K\_KP\_PERIOD | . | 句号（小键盘） |
| K\_KP\_DIVIDE | / | 除号（小键盘） |
| K\_KP\_MULTIPLY | \* | 乘号（小键盘） |
| K\_KP\_MINUS | - | 减号（小键盘） |
| K\_KP\_PLUS | + | 加号（小键盘） |
|  |  |  |
| K\_KP\_ENTER | \r | 回车键（小键盘） |
| K\_KP\_EQUALS | = | 等于号（小键盘） |
| K\_UP |  | 向上箭头（up arrow） |
| K\_DOWN |  | 向下箭头（down arrow） |
| K\_RIGHT |  | 向右箭头（right arrow） |
| K\_LEFT |  | 向左箭头（left arrow） |
| K\_INSERT |  | 插入符（insert） |
| K\_HOME |  | Home键（home） |
| K\_END |  | End键（end） |
| K\_PAGEUP |  | 上一页（page up） |
| K\_PAGEDOWN |  | 下一页（page down） |
| K\_F1 |  | F1 |
| K\_F2 |  | F2 |
| K\_F3 |  | F3 |
| K\_F4 |  | F4 |
| K\_F5 |  | F5 |
| K\_F6 |  | F6 |
| K\_F7 |  | F7 |
| K\_F8 |  | F8 |
| K\_F9 |  | F9 |
| K\_F10 |  | F10 |
| K\_F11 |  | F11 |
| K\_F12 |  | F12 |
| K\_F13 |  | F13 |
| K\_F14 |  | F14 |
| K\_F15 |  | F15 |
| K\_NUMLOCK |  | 数字键盘锁定键（numlock） |
| K\_CAPSLOCK |  | 大写字母锁定键（capslock） |
| K\_SCROLLOCK |  | 滚动锁定键（scrollock） |
| K\_RSHIFT |  | 右边的shift键（right shift） |
| K\_LSHIFT |  | 左边的shift键（left shift） |
| K\_RCTRL |  | 右边的ctrl键（right ctrl） |
| K\_LCTRL |  | 左边的ctrl 键（left ctrl） |
| K\_RALT |  | 右边的alt 键（right alt） |
| K\_LALT |  | 左边的alt 键（left alt） |
| K\_RMETA |  | 右边的元键（right meta） |
| K\_LMETA |  | 左边的元键（left meta） |
| K\_LSUPER |  | 左边的Window键（left windows key） |
| K\_RSUPER |  | 右边的Window键（right windows key） |
| K\_MODE |  | 模式转换键（mode shift） |
| K\_HELP |  | 帮助键（help） |
| K\_PRINT |  | 打印屏幕键（print screen） |
| K\_SYSREQ |  | 魔术键（sysrq） |
| K\_BREAK |  | 中断键（break） |
| K\_MENU |  | 菜单键（menu） |
| K\_POWER |  | 电源键（power） |
| K\_EURO |  | 欧元符号（euro） |

还有一个 mod 属性，用于描述组合键状态。

以下是组合键的常量定义：

KeyASCII描述

|  |  |
| --- | --- |
| KMOD\_NONE | 木有同时按下组合键 |
| KMOD\_LSHIFT | 同时按下左边的shift键 |
| KMOD\_RSHIFT | 同时按下右边的shift键 |
| KMOD\_SHIFT | 同时按下shift键 |
| KMOD\_CAPS | 同时按下大写字母锁定键 |
| KMOD\_LCTRL | 同时按下左边的ctrl键 |
| KMOD\_RCTRL | 同时按下右边的ctrl键 |
| KMOD\_CTRL | 同时按下ctrl键 |
| KMOD\_LALT | 同时按下左边的alt键 |
| KMOD\_RALT | 同时按下右边的alt键 |
| KMOD\_ALT | 同时按下alt键 |
| KMOD\_LMETA | 同时按下左边的元键 |
| KMOD\_RMETA | 同时按下右边的元键 |
| KMOD\_META | 同时按下元键 |
| KMOD\_NUM | 同时按下数字键盘锁定键 |
| KMOD\_MODE | 同时按下模式转换键 |

小甲鱼温馨提示：如果 mod & KMOD\_CTRL 是真的话，表示用户同时按下了 Ctrl 键。

函数详解

### pygame.key.get\_focused()

当窗口获得键盘的输入焦点时返回 True。

get\_focused() -> bool

当窗口获得键盘的输入焦点时返回 True，如果窗口需要确保不失去键盘焦点，可以使用 pygame.event.set\_grab(True) 独占所有的输入接口。

小甲鱼温馨提示：注意，这样做你就无法将鼠标移出窗口客户区了，但你仍然可以通过 Ctrl - Alt - Delete 热键“解围”。

### pygame.key.get\_pressed()

获取键盘上所有按键的状态。

get\_pressed() -> bools

返回一个由布尔类型值组成的序列，表示键盘上所有按键的当前状态。使用 key 常量作为索引，如果该元素是 True，表示该按键被按下。使用该函数获取一系列按钮被按下的状态，并不能正确的获取用户输入的文本。因为你无法知道用户按键的被按下的顺序，并且快速的连续按下键盘可能无法完全被捕获（在两次调用 pygame.key.get\_pressed() 的过程中被忽略），也无法将这些按下的按键完全转化为字符值。实现此功能可以通过捕获 pygame.KEYDOWN 事件消息来实现。

### pygame.key.get\_mods()

检测是否有组合键被按下。

get\_mods() -> int

返回一个包含所有组合键位掩码的整数。使用位操作符 & 你可以检测某个组合键是否被按下。

小甲鱼温馨提示：假如 pygame.key.get\_mods() 返回值存放在 mods 变量中，如果 mods & KMOD\_CTRL 为 True，表示 ctrl 键正被按下。

### pygame.key.set\_mods()

临时设置某些组合键为被按下状态。

set\_mods(int) -> None

创建一个位掩码整数，包含你需要设置为被按下状态的组合键小甲鱼温馨提示：比如我们需要设置 ctrl 和 alt 组合键为按下状态，则可以 mods = KMOD\_CTRL | KMOD\_ALT，然后调用 pygame.key.set\_mods(mods)，这样尽管用户没有按下 ctrl 和 alt 组合键，它们依然是显示被按下状态。

### pygame.key.set\_repeat()

控制重复响应持续按下按键的时间。set\_repeat() -> None

set\_repeat(delay, interval) -> None

当开启重复响应按键，那么用户持续按下某一按键，就会不断产生同一 pygame.KEYDOWN 事件。delay 参数设置多久后（单位是毫秒）开始发送第一个 pygame.KEYDOWN 事件。interval 参数设置发送两个事件之间的间隔。如果不传入任何参数，表示取消重复响应按键。

pygame.key.get\_repeat()获取重复响应按键的参数。

get\_repeat() -> (delay, interval)

当开启重复响应按键，那么用户持续按下某一按键，就会不断产生同一 pygame.KEYDOWN 事件。返回值是一个二元组，第一个元素 delay 表示多久后（单位是毫秒）开始发送第一个 pygame.KEYDOWN 事件。第二个元素 interval 表示发送两个事件之间的间隔。

默认情况下重复响应按键是没有开启的。Pygame 1.8 新增加的。

### pygame.key.name()

获取按键标识符对应的名字。name(key) -> string

获取一个按键标识符对应的字符串描述。

## 绘制图形draw

函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称 | 作用 | 参数 | 返回值 |
| pygame.draw.rect() | 绘制矩形，Rect 参数指定矩形的位置和尺寸。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。  (screen, BLACK, (1, 1, 50, 50), 1)  表示绘制在screen上，黑色，起始位置为(1,1)  长宽各为50，不填充 | rect(Surface, color, Rect, width=0) | Rect |
| pygame.draw.polygon() | 绘制多边形，在 Surface 对象上绘制一个多边形。pointlist 参数指定多边形的各个顶点。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。  绘制一个抗锯齿的多边形，只需要将 aalines() 的 closed 参数设置为 True 即可。 | polygon(Surface, color, pointlist, width=0) | Rect |
| pygame.draw.circle() | 根据圆心和半径绘制圆形，pos 参数指定圆心的位置，radius 参数指定圆的半径。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。 | circle(Surface, color, pos, radius, width=0) | Rect |
| pygame.draw.ellipse() | 根据限定矩形绘制一个椭圆形，Rect 参数指定椭圆外围的限定矩形。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。 | ellipse(Surface, color, Rect, width=0) | Rect |
| pygame.draw.arc() | 绘制弧线，在 Surface 对象上绘制一条弧线。Rect 参数指定弧线所在的椭圆外围的限定矩形。两个 angle 参数指定弧线的开始和结束位置。width 参数指定边框的宽度。 | (Surface, color, Rect, start\_angle, stop\_angle, width=1) |  |
| pygame.draw.line() | 绘制线段 | (Surface, color, start\_pos, end\_pos, width=1) | Rect |
| pygame.draw.lines() | 绘制多条连续的线段。pointlist 参数是一系列短点。如果 closed 参数设置为 True，则绘制首尾相连。 | lines(Surface, color, closed, pointlist, width=1) | Rect |
| pygame.draw.aaline() | 绘制抗锯齿的线段, blend 参数指定是否通过绘制混合背景的阴影来实现抗锯齿功能。该函数的结束位置允许使用浮点数。 | (Surface, color, startpos, endpos, blend=1) | Rect |
| pygame.draw.aalines() | 绘制多条连续的线段（抗锯齿） | aalines(Surface, color, closed, pointlist, blend=1) | Rect |

该模块用于在 Surface 对象上绘制一些简单的形状。这些函数将渲染到任何格式的 Surface 对象上。硬件渲染会比普通的软件渲染更耗时。

**大部分函数用 width 参数指定图形边框的大小，如果 width = 0 则表示填充整个图形**。

所有的绘图函数仅能在 Surface 对象的剪切区域生效。这些函数返回一个 Rect，表示包含实际绘制图形的矩形区域。

大部分函数都有一个 color 参数，传入一个表示 RGB 颜色值的三元组，当然也支持 RGBA 四元组。其中的 A 是 Alpha 的意思，用于控制透明度。不过该模块的函数并不会绘制透明度，而是直接传入到对应 Surface 对象的 pixel alphas 中。color 参数也可以是已经映射到 Surface 对象的像素格式中的整型像素值。

当这些函数在绘制时，必须暂时锁定 Surface 对象。许多连续绘制的函数可以通过一次性锁定直到画完再解锁来提高效率。

函数详解

### pygame.draw.rect()

绘制矩形。

在 Surface 对象上绘制一个矩形。Rect 参数指定矩形的位置和尺寸。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。

### pygame.draw.polygon()

绘制多边形。

在 Surface 对象上绘制一个多边形。pointlist 参数指定多边形的各个顶点。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。

绘制一个抗锯齿的多边形，只需要将 aalines() 的 closed 参数设置为 True 即可。

### pygame.draw.circle()

根据圆心和半径绘制圆形。

在 Surface 对象上绘制一个圆形。pos 参数指定圆心的位置，radius 参数指定圆的半径。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。

### pygame.draw.ellipse()

根据限定矩形绘制一个椭圆形。

在 Surface 对象上绘制一个椭圆形。Rect 参数指定椭圆外围的限定矩形。width 参数指定边框的宽度，如果设置为 0 则表示填充该矩形。

### pygame.draw.arc()

绘制弧线。

在 Surface 对象上绘制一条弧线。Rect 参数指定弧线所在的椭圆外围的限定矩形。两个 angle 参数指定弧线的开始和结束位置。width 参数指定边框的宽度。

### pygame.draw.line()

绘制线段。

在 Surface 对象上绘制一条线段。两端以方形结束。

### pygame.draw.lines()

绘制多条连续的线段。

在 Surface 对象上绘制一系列连续的线段。pointlist 参数是一系列短点。如果 closed 参数设置为 True，则绘制首尾相连。

### pygame.draw.aaline()

绘制抗锯齿的线段。

在 Surface 对象上绘制一条抗锯齿的线段。blend 参数指定是否通过绘制混合背景的阴影来实现抗锯齿功能。该函数的结束位置允许使用浮点数。

### pygame.draw.aalines()

绘制多条连续的线段（抗锯齿）。

在 Surface 对象上绘制一系列连续的线段（抗锯齿）。如果 closed 参数为 True，则首尾相连。blend 参数指定是否通过绘制混合背景的阴影来实现抗锯齿功能。该函数的结束位置允许使用浮点数。

## pygame.sprite

*pygame module with basic game object classes*

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.Sprite](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite) | — | Simple base class for visible game objects. |
| [pygame.sprite.DirtySprite](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.DirtySprite) | — | A subclass of Sprite with more attributes and features. |
| [pygame.sprite.Group](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group) | — | A container class to hold and manage multiple Sprite objects. |
| [pygame.sprite.RenderPlain](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.RenderPlain) | — | Same as pygame.sprite.Group |
| [pygame.sprite.RenderClear](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.RenderClear) | — | Same as pygame.sprite.Group |
| [pygame.sprite.RenderUpdates](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.RenderUpdates) | — | Group sub-class that tracks dirty updates. |
| [pygame.sprite.OrderedUpdates](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.OrderedUpdates) | — | RenderUpdates sub-class that draws Sprites in order of addition. |
| [pygame.sprite.LayeredUpdates](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates) | — | LayeredUpdates is a sprite group that handles layers and draws like OrderedUpdates. |
| [pygame.sprite.LayeredDirty](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty) | — | LayeredDirty group is for DirtySprite objects. Subclasses LayeredUpdates. |
| [pygame.sprite.GroupSingle](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.GroupSingle) | — | Group container that holds a single sprite. |
| [pygame.sprite.spritecollide](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.spritecollide) | — | Find sprites in a group that intersect another sprite. |
| [pygame.sprite.collide\_rect](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_rect) | — | Collision detection between two sprites, using rects. |
| [pygame.sprite.collide\_rect\_ratio](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_rect_ratio) | — | Collision detection between two sprites, using rects scaled to a ratio. |
| [pygame.sprite.collide\_circle](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_circle) | — | Collision detection between two sprites, using circles. |
| [pygame.sprite.collide\_circle\_ratio](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_circle_ratio) | — | Collision detection between two sprites, using circles scaled to a ratio. |
| [pygame.sprite.collide\_mask](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.collide_mask) | — | Collision detection between two sprites, using masks. |
| [pygame.sprite.groupcollide](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.groupcollide) | — | Find all sprites that collide between two groups. |
| [pygame.sprite.spritecollideany](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.spritecollideany) | — | Simple test if a sprite intersects anything in a group. |

This module contains several simple classes to be used within games. There is the main Sprite class and several Group classes that contain Sprites. The use of these classes is entirely optional when using Pygame. The classes are fairly lightweight and only provide a starting place for the code that is common to most games.

The Sprite class is intended to be used as a base class for the different types of objects in the game. There is also a base Group class that simply stores sprites. A game could create new types of Group classes that operate on specially customized Sprite instances they contain.

The basic Sprite class can draw the Sprites it contains to a Surface. The Group.draw() method requires that each Sprite have a Surface.image attribute and aSurface.rect. The Group.clear() method requires these same attributes, and can be used to erase all the Sprites with background. There are also more advanced Groups: pygame.sprite.RenderUpdates() and pygame.sprite.OrderedUpdates().

Lastly, this module contains several collision functions. These help find sprites inside multiple groups that have intersecting bounding rectangles. To find the collisions, the Sprites are required to have a Surface.rect attribute assigned.

The groups are designed for high efficiency in removing and adding Sprites to them. They also allow cheap testing to see if a Sprite already exists in a Group. A given Sprite can exist in any number of groups. A game could use some groups to control object rendering, and a completely separate set of groups to control interaction or player movement. Instead of adding type attributes or bools to a derived Sprite class, consider keeping the Sprites inside organized Groups. This will allow for easier lookup later in the game.

Sprites and Groups manage their relationships with the add() and remove() methods. These methods can accept a single or multiple targets for membership. The default initializers for these classes also takes a single or list of targets for initial membership. It is safe to repeatedly add and remove the same Sprite from a Group.

While it is possible to design sprite and group classes that don’t derive from the Sprite and AbstractGroup classes below, it is strongly recommended that you extend those when you add a Sprite or Group class.

Sprites are not thread safe. So lock them yourself if using threads.

pygame.sprite.Sprite

*Simple base class for visible game objects.*

Sprite(\*groups) -> Sprite

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.Sprite.update](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.update) | — | method to control sprite behavior |
| [pygame.sprite.Sprite.add](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.add) | — | add the sprite to groups |
| [pygame.sprite.Sprite.remove](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.remove) | — | remove the sprite from groups |
| [pygame.sprite.Sprite.kill](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.kill) | — | remove the Sprite from all Groups |
| [pygame.sprite.Sprite.alive](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.alive) | — | does the sprite belong to any groups |
| [pygame.sprite.Sprite.groups](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite.groups) | — | list of Groups that contain this Sprite |

The base class for visible game objects. Derived classes will want to override the Sprite.update() and assign a Sprite.image and Sprite.rect attributes. The initializer can accept any number of Group instances to be added to.

When subclassing the Sprite, be sure to call the base initializer before adding the Sprite to Groups. For example:

**class** **Block**(pygame.sprite.Sprite):

*# Constructor. Pass in the color of the block,*

*# and its x and y position*

**def** \_\_init\_\_(self, color, width, height):

*# Call the parent class (Sprite) constructor*

pygame.sprite.Sprite.\_\_init\_\_(self)

*# Create an image of the block, and fill it with a color.*

*# This could also be an image loaded from the disk.*

self.image = pygame.Surface([width, height])

self.image.fill(color)

*# Fetch the rectangle object that has the dimensions of the image*

*# Update the position of this object by setting the values of rect.x and rect.y*

self.rect = self.image.get\_rect()

update()

*method to control sprite behavior*

update(\*args) -> None

The default implementation of this method does nothing; it’s just a convenient “hook” that you can override. This method is called by Group.update()with whatever arguments you give it.

There is no need to use this method if not using the convenience method by the same name in the Group class.

窗体顶端



窗体底端

窗体顶端



窗体底端

add()

*add the sprite to groups*

add(\*groups) -> None

Any number of Group instances can be passed as arguments. The Sprite will be added to the Groups it is not already a member of.

窗体顶端



窗体底端

窗体顶端



窗体底端

remove()

*remove the sprite from groups*

remove(\*groups) -> None

Any number of Group instances can be passed as arguments. The Sprite will be removed from the Groups it is currently a member of.

窗体顶端



窗体底端

窗体顶端



窗体底端

kill()

*remove the Sprite from all Groups*

kill() -> None

The Sprite is removed from all the Groups that contain it. This won’t change anything about the state of the Sprite. It is possible to continue to use the Sprite after this method has been called, including adding it to Groups.

窗体顶端



窗体底端

窗体顶端



窗体底端

alive()

*does the sprite belong to any groups*

alive() -> bool

Returns True when the Sprite belongs to one or more Groups.

窗体顶端



窗体底端

窗体顶端



窗体底端

groups()

*list of Groups that contain this Sprite*

groups() -> group\_list

Return a list of all the Groups that contain this Sprite.

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.DirtySprite

*A subclass of Sprite with more attributes and features.*

DirtySprite(\*groups) -> DirtySprite

Extra DirtySprite attributes with their default values:

dirty = 1

if set to 1, it is repainted and then set to 0 again

if set to 2 then it is always dirty ( repainted each frame,

flag is not reset)

0 means that it is not dirty and therefor not repainted again

blendmode = 0

its the special\_flags argument of blit, blendmodes

source\_rect = None

source rect to use, remember that it is relative to

topleft (0,0) of self.image

visible = 1

normally 1, if set to 0 it will not be repainted

(you must set it dirty too to be erased from screen)

layer = 0

(READONLY value, it is read when adding it to the

LayeredDirty, for details see doc of LayeredDirty)

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.Group

*A container class to hold and manage multiple Sprite objects.*

Group(\*sprites) -> Group

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.Group.sprites](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.sprites) | — | list of the Sprites this Group contains |
| [pygame.sprite.Group.copy](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.copy) | — | duplicate the Group |
| [pygame.sprite.Group.add](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.add) | — | add Sprites to this Group |
| [pygame.sprite.Group.remove](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.remove) | — | remove Sprites from the Group |
| [pygame.sprite.Group.has](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.has) | — | test if a Group contains Sprites |
| [pygame.sprite.Group.update](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.update) | — | call the update method on contained Sprites |
| [pygame.sprite.Group.draw](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.draw) | — | blit the Sprite images |
| [pygame.sprite.Group.clear](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.clear) | — | draw a background over the Sprites |
| [pygame.sprite.Group.empty](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group.empty) | — | remove all Sprites |

A simple container for Sprite objects. This class can be inherited to create containers with more specific behaviors. The constructor takes any number of Sprite arguments to add to the Group. The group supports the following standard Python operations:

in test if a Sprite is contained

len the number of Sprites contained

bool test if any Sprites are contained

iter iterate through all the Sprites

The Sprites in the Group are not ordered, so drawing and iterating the Sprites is in no particular order.

sprites()

*list of the Sprites this Group contains*

sprites() -> sprite\_list

Return a list of all the Sprites this group contains. You can also get an iterator from the group, but you cannot iterator over a Group while modifying it.

窗体顶端



窗体底端

窗体顶端



窗体底端

copy()

*duplicate the Group*

copy() -> Group

Creates a new Group with all the same Sprites as the original. If you have subclassed Group, the new object will have the same (sub-)class as the original. This only works if the derived class’s constructor takes the same arguments as the Group class’s.

窗体顶端



窗体底端

窗体顶端



窗体底端

add()

*add Sprites to this Group*

add(\*sprites) -> None

Add any number of Sprites to this Group. This will only add Sprites that are not already members of the Group.

Each sprite argument can also be a iterator containing Sprites.

窗体顶端



窗体底端

窗体顶端



窗体底端

remove()

*remove Sprites from the Group*

remove(\*sprites) -> None

Remove any number of Sprites from the Group. This will only remove Sprites that are already members of the Group.

Each sprite argument can also be a iterator containing Sprites.

窗体顶端



窗体底端

窗体顶端



窗体底端

has()

*test if a Group contains Sprites*

has(\*sprites) -> None

Return True if the Group contains all of the given sprites. This is similar to using the “in” operator on the Group (“if sprite in group: ...”), which tests if a single Sprite belongs to a Group.

Each sprite argument can also be a iterator containing Sprites.

窗体顶端



窗体底端

窗体顶端



窗体底端

update()

*call the update method on contained Sprites*

update(\*args) -> None

Calls the update() method on all Sprites in the Group. The base Sprite class has an update method that takes any number of arguments and does nothing. The arguments passed to Group.update() will be passed to each Sprite.

There is no way to get the return value from the Sprite.update() methods.

窗体顶端



窗体底端

窗体顶端



窗体底端

draw()

*blit the Sprite images*

draw(Surface) -> None

Draws the contained Sprites to the Surface argument. This uses the Sprite.image attribute for the source surface, and Sprite.rect for the position.

The Group does not keep sprites in any order, so the draw order is arbitrary.

窗体顶端



窗体底端

窗体顶端



窗体底端

clear()

*draw a background over the Sprites*

clear(Surface\_dest, background) -> None

Erases the Sprites used in the last Group.draw() call. The destination Surface is cleared by filling the drawn Sprite positions with the background.

The background is usually a Surface image the same dimensions as the destination Surface. However, it can also be a callback function that takes two arguments; the destination Surface and an area to clear. The background callback function will be called several times each clear.

Here is an example callback that will clear the Sprites with solid red:

**def** clear\_callback(surf, rect):

color = 255, 0, 0

surf.fill(color, rect)

窗体顶端



窗体底端

窗体顶端



窗体底端

empty()

*remove all Sprites*

empty() -> None

Removes all Sprites from this Group.

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.RenderPlain

*Same as pygame.sprite.Group*

This class is an alias to pygame.sprite.Group(). It has no additional functionality.

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.RenderClear

*Same as pygame.sprite.Group*

This class is an alias to pygame.sprite.Group(). It has no additional functionality.

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.RenderUpdates

*Group sub-class that tracks dirty updates.*

RenderUpdates(\*sprites) -> RenderUpdates

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.RenderUpdates.draw](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.RenderUpdates.draw) | — | blit the Sprite images and track changed areas |

This class is derived from pygame.sprite.Group(). It has an extended draw() method that tracks the changed areas of the screen.

draw()

*blit the Sprite images and track changed areas*

draw(surface) -> Rect\_list

Draws all the Sprites to the surface, the same as Group.draw(). This method also returns a list of Rectangular areas on the screen that have been changed. The returned changes include areas of the screen that have been affected by previous Group.clear() calls.

The returned Rect list should be passed to pygame.display.update(). This will help performance on software driven display modes. This type of updating is usually only helpful on destinations with non-animating backgrounds.

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

[Comments 2](http://www.pygame.org/docs/ref/sprite.html#comment_pygame_sprite_RenderUpdates)

pygame.sprite.OrderedUpdates()

*RenderUpdates sub-class that draws Sprites in order of addition.*

OrderedUpdates(\*spites) -> OrderedUpdates

This class derives from pygame.sprite.RenderUpdates(). It maintains the order in which the Sprites were added to the Group for rendering. This makes adding and removing Sprites from the Group a little slower than regular Groups.

窗体顶端



窗体底端

窗体顶端



窗体底端

[Comments 1](http://www.pygame.org/docs/ref/sprite.html#comment_pygame_sprite_OrderedUpdates)

pygame.sprite.LayeredUpdates

*LayeredUpdates is a sprite group that handles layers and draws like OrderedUpdates.*

LayeredUpdates(\*spites, \*\*kwargs) -> LayeredUpdates

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.LayeredUpdates.add](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.add) | — | add a sprite or sequence of sprites to a group |
| [pygame.sprite.LayeredUpdates.sprites](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.sprites) | — | returns a ordered list of sprites (first back, last top). |
| [pygame.sprite.LayeredUpdates.draw](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.draw) | — | draw all sprites in the right order onto the passed surface. |
| [pygame.sprite.LayeredUpdates.get\_sprites\_at](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_sprites_at) | — | returns a list with all sprites at that position. |
| [pygame.sprite.LayeredUpdates.get\_sprite](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_sprite) | — | returns the sprite at the index idx from the groups sprites |
| [pygame.sprite.LayeredUpdates.remove\_sprites\_of\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.remove_sprites_of_layer) | — | removes all sprites from a layer and returns them as a list. |
| [pygame.sprite.LayeredUpdates.layers](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.layers) | — | returns a list of layers defined (unique), sorted from botton up. |
| [pygame.sprite.LayeredUpdates.change\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.change_layer) | — | changes the layer of the sprite |
| [pygame.sprite.LayeredUpdates.get\_layer\_of\_sprite](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_layer_of_sprite) | — | returns the layer that sprite is currently in. |
| [pygame.sprite.LayeredUpdates.get\_top\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_top_layer) | — | returns the top layer |
| [pygame.sprite.LayeredUpdates.get\_bottom\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_bottom_layer) | — | returns the bottom layer |
| [pygame.sprite.LayeredUpdates.move\_to\_front](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.move_to_front) | — | brings the sprite to front layer |
| [pygame.sprite.LayeredUpdates.move\_to\_back](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.move_to_back) | — | moves the sprite to the bottom layer |
| [pygame.sprite.LayeredUpdates.get\_top\_sprite](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_top_sprite) | — | returns the topmost sprite |
| [pygame.sprite.LayeredUpdates.get\_sprites\_from\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.get_sprites_from_layer) | — | returns all sprites from a layer, ordered by how they where added |
| [pygame.sprite.LayeredUpdates.switch\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredUpdates.switch_layer) | — | switches the sprites from layer1 to layer2 |

This group is fully compatible with [pygame.sprite.SpriteSimple base class for visible game objects.](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Sprite).

You can set the default layer through kwargs using ‘default\_layer’ and an integer for the layer. The default layer is 0.

If the sprite you add has an attribute layer then that layer will be used. If the \*\*kwarg contains ‘layer’ then the sprites passed will be added to that layer (overriding the sprite.layer attribute). If neither sprite has attribute layer nor \*\*kwarg then the default layer is used to add the sprites.

New in pygame 1.8.0

add()

*add a sprite or sequence of sprites to a group*

add(\*sprites, \*\*kwargs) -> None

If the sprite(s) have an attribute layer then that is used for the layer. If \*\*kwargs contains ‘layer’ then the sprite(s) will be added to that argument (overriding the sprite layer attribute). If neither is passed then the sprite(s) will be added to the default layer.

窗体顶端



窗体底端

窗体顶端



窗体底端

sprites()

*returns a ordered list of sprites (first back, last top).*

sprites() -> sprites

窗体顶端



窗体底端

窗体顶端



窗体底端

draw()

*draw all sprites in the right order onto the passed surface.*

draw(surface) -> Rect\_list

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_sprites\_at()

*returns a list with all sprites at that position.*

get\_sprites\_at(pos) -> colliding\_sprites

Bottom sprites first, top last.

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_sprite()

*returns the sprite at the index idx from the groups sprites*

get\_sprite(idx) -> sprite

Raises IndexOutOfBounds if the idx is not within range.

窗体顶端



窗体底端

窗体顶端



窗体底端

remove\_sprites\_of\_layer()

*removes all sprites from a layer and returns them as a list.*

remove\_sprites\_of\_layer(layer\_nr) -> sprites

窗体顶端



窗体底端

窗体顶端



窗体底端

layers()

*returns a list of layers defined (unique), sorted from botton up.*

layers() -> layers

窗体顶端



窗体底端

窗体顶端



窗体底端

change\_layer()

*changes the layer of the sprite*

change\_layer(sprite, new\_layer) -> None

sprite must have been added to the renderer. It is not checked.

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_layer\_of\_sprite()

*returns the layer that sprite is currently in.*

get\_layer\_of\_sprite(sprite) -> layer

If the sprite is not found then it will return the default layer.

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_top\_layer()

*returns the top layer*

get\_top\_layer() -> layer

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_bottom\_layer()

*returns the bottom layer*

get\_bottom\_layer() -> layer

窗体顶端



窗体底端

窗体顶端



窗体底端

move\_to\_front()

*brings the sprite to front layer*

move\_to\_front(sprite) -> None

Brings the sprite to front, changing sprite layer to topmost layer (added at the end of that layer).

窗体顶端



窗体底端

窗体顶端



窗体底端

move\_to\_back()

*moves the sprite to the bottom layer*

move\_to\_back(sprite) -> None

Moves the sprite to the bottom layer, moving it behind all other layers and adding one additional layer.

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_top\_sprite()

*returns the topmost sprite*

get\_top\_sprite() -> Sprite

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_sprites\_from\_layer()

*returns all sprites from a layer, ordered by how they where added*

get\_sprites\_from\_layer(layer) -> sprites

Returns all sprites from a layer, ordered by how they where added. It uses linear search and the sprites are not removed from layer.

窗体顶端



窗体底端

窗体顶端



窗体底端

switch\_layer()

*switches the sprites from layer1 to layer2*

switch\_layer(layer1\_nr, layer2\_nr) -> None

The layers number must exist, it is not checked.

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

[Comments 1](http://www.pygame.org/docs/ref/sprite.html#comment_pygame_sprite_LayeredUpdates)

pygame.sprite.LayeredDirty

*LayeredDirty group is for DirtySprite objects. Subclasses LayeredUpdates.*

LayeredDirty(\*spites, \*\*kwargs) -> LayeredDirty

|  |  |  |
| --- | --- | --- |
| [pygame.sprite.LayeredDirty.draw](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.draw) | — | draw all sprites in the right order onto the passed surface. |
| [pygame.sprite.LayeredDirty.clear](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.clear) | — | used to set background |
| [pygame.sprite.LayeredDirty.repaint\_rect](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.repaint_rect) | — | repaints the given area |
| [pygame.sprite.LayeredDirty.set\_clip](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.set_clip) | — | clip the area where to draw. Just pass None (default) to reset the clip |
| [pygame.sprite.LayeredDirty.get\_clip](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.get_clip) | — | clip the area where to draw. Just pass None (default) to reset the clip |
| [pygame.sprite.LayeredDirty.change\_layer](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.change_layer) | — | changes the layer of the sprite |
| [pygame.sprite.LayeredDirty.set\_timing\_treshold](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.LayeredDirty.set_timing_treshold) | — | sets the treshold in milliseconds |

This group requires [pygame.sprite.DirtySpriteA subclass of Sprite with more attributes and features.](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.DirtySprite) or any sprite that has the following attributes:

image, rect, dirty, visible, blendmode (see doc of DirtySprite).

It uses the dirty flag technique and is therefore faster than the [pygame.sprite.RenderUpdatesGroup sub-class that tracks dirty updates.](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.RenderUpdates) if you have many static sprites. It also switches automatically between dirty rect update and full screen drawing, so you do no have to worry what would be faster.

Same as for the [pygame.sprite.GroupA container class to hold and manage multiple Sprite objects.](http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group). You can specify some additional attributes through kwargs:

\_use\_update: True/False default is False

\_default\_layer: default layer where sprites without a layer are added.

\_time\_threshold: treshold time for switching between dirty rect mode

and fullscreen mode, defaults to 1000./80 == 1000./fps

New in pygame 1.8.0

draw()

*draw all sprites in the right order onto the passed surface.*

draw(surface, bgd=None) -> Rect\_list

You can pass the background too. If a background is already set, then the bgd argument has no effect.

窗体顶端



窗体底端

窗体顶端



窗体底端

clear()

*used to set background*

clear(surface, bgd) -> None

窗体顶端



窗体底端

窗体顶端



窗体底端

repaint\_rect()

*repaints the given area*

repaint\_rect(screen\_rect) -> None

screen\_rect is in screencoordinates.

窗体顶端



窗体底端

窗体顶端



窗体底端

set\_clip()

*clip the area where to draw. Just pass None (default) to reset the clip*

set\_clip(screen\_rect=None) -> None

窗体顶端



窗体底端

窗体顶端



窗体底端

get\_clip()

*clip the area where to draw. Just pass None (default) to reset the clip*

get\_clip() -> Rect

窗体顶端



窗体底端

窗体顶端



窗体底端

change\_layer()

*changes the layer of the sprite*

change\_layer(sprite, new\_layer) -> None

sprite must have been added to the renderer. It is not checked.

窗体顶端



窗体底端

窗体顶端



窗体底端

set\_timing\_treshold()

*sets the treshold in milliseconds*

set\_timing\_treshold(time\_ms) -> None

Default is 1000./80 where 80 is the fps I want to switch to full screen mode.

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.GroupSingle()

*Group container that holds a single sprite.*

GroupSingle(sprite=None) -> GroupSingle

The GroupSingle container only holds a single Sprite. When a new Sprite is added, the old one is removed.

There is a special property, GroupSingle.sprite, that accesses the Sprite that this Group contains. It can be None when the Group is empty. The property can also be assigned to add a Sprite into the GroupSingle container.

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.spritecollide()

*Find sprites in a group that intersect another sprite.*

spritecollide(sprite, group, dokill, collided = None) -> Sprite\_list

Return a list containing all Sprites in a Group that intersect with another Sprite. Intersection is determined by comparing the Sprite.rect attribute of each Sprite.

The dokill argument is a bool. If set to True, all Sprites that collide will be removed from the Group.

The collided argument is a callback function used to calculate if two sprites are colliding. it should take two sprites as values, and return a bool value indicating if they are colliding. If collided is not passed, all sprites must have a “rect” value, which is a rectangle of the sprite area, which will be used to calculate the collision.

collided callables:

collide\_rect, collide\_rect\_ratio, collide\_circle,

collide\_circle\_ratio, collide\_mask

Example:

*# See if the Sprite block has collided with anything in the Group block\_list*

*# The True flag will remove the sprite in block\_list*

blocks\_hit\_list = pygame.sprite.spritecollide(player, block\_list, True)

*# Check the list of colliding sprites, and add one to the score for each one*

**for** block **in** blocks\_hit\_list:

score +=1

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.collide\_rect()

*Collision detection between two sprites, using rects.*

collide\_rect(left, right) -> bool

Tests for collision between two sprites. Uses the pygame rect colliderect function to calculate the collision. Intended to be passed as a collided callback function to the \*collide functions. Sprites must have a “rect” attributes.

New in pygame 1.8.0

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.collide\_rect\_ratio()

*Collision detection between two sprites, using rects scaled to a ratio.*

collide\_rect\_ratio(ratio) -> collided\_callable

A callable class that checks for collisions between two sprites, using a scaled version of the sprites rects.

Is created with a ratio, the instance is then intended to be passed as a collided callback function to the \*collide functions.

A ratio is a floating point number - 1.0 is the same size, 2.0 is twice as big, and 0.5 is half the size.

New in pygame 1.8.1

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.collide\_circle()

*Collision detection between two sprites, using circles.*

collide\_circle(left, right) -> bool

Tests for collision between two sprites, by testing to see if two circles centered on the sprites overlap. If the sprites have a “radius” attribute, that is used to create the circle, otherwise a circle is created that is big enough to completely enclose the sprites rect as given by the “rect” attribute. Intended to be passed as a collided callback function to the \*collide functions. Sprites must have a “rect” and an optional “radius” attribute.

New in pygame 1.8.1

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.collide\_circle\_ratio()

*Collision detection between two sprites, using circles scaled to a ratio.*

collide\_circle\_ratio(ratio) -> collided\_callable

A callable class that checks for collisions between two sprites, using a scaled version of the sprites radius.

Is created with a floating point ratio, the instance is then intended to be passed as a collided callback function to the \*collide functions.

A ratio is a floating point number - 1.0 is the same size, 2.0 is twice as big, and 0.5 is half the size.

The created callable tests for collision between two sprites, by testing to see if two circles centered on the sprites overlap, after scaling the circles radius by the stored ratio. If the sprites have a “radius” attribute, that is used to create the circle, otherwise a circle is created that is big enough to completely enclose the sprites rect as given by the “rect” attribute. Intended to be passed as a collided callback function to the \*collide functions. Sprites must have a “rect” and an optional “radius” attribute.

New in pygame 1.8.1

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.collide\_mask()

*Collision detection between two sprites, using masks.*

collide\_mask(SpriteLeft, SpriteRight) -> point

Returns first point on the mask where the masks collided, or None if there was no collision.

Tests for collision between two sprites, by testing if thier bitmasks overlap. If the sprites have a “mask” attribute, that is used as the mask, otherwise a mask is created from the sprite image. Intended to be passed as a collided callback function to the \*collide functions. Sprites must have a “rect” and an optional “mask” attribute.

You should consider creating a mask for your sprite at load time if you are going to check collisions many times. This will increase the performance, otherwise this can be an expensive function because it will create the masks each time you check for collisions.

sprite.mask = pygame.mask.from\_surface(sprite.image)

New in pygame 1.8.0

窗体顶端



窗体底端

窗体顶端



窗体底端

[Comments 1](http://www.pygame.org/docs/ref/sprite.html#comment_pygame_sprite_collide_mask)

pygame.sprite.groupcollide()

*Find all sprites that collide between two groups.*

groupcollide(group1, group2, dokill1, dokill2, collided = None) -> Sprite\_dict

This will find collisions between all the Sprites in two groups. Collision is determined by comparing the Sprite.rect attribute of each Sprite or by using the collided function if it is not None.

Every Sprite inside group1 is added to the return dictionary. The value for each item is the list of Sprites in group2 that intersect.

If either dokill argument is True, the colliding Sprites will be removed from their respective Group.

The collided argument is a callback function used to calculate if two sprites are colliding. It should take two sprites as values and return a bool value indicating if they are colliding. If collided is not passed, then all sprites must have a “rect” value, which is a rectangle of the sprite area, which will be used to calculate the collision.

窗体顶端



窗体底端

窗体顶端



窗体底端

pygame.sprite.spritecollideany()

*Simple test if a sprite intersects anything in a group.*

spritecollideany(sprite, group, collided = None) -> Sprite Collision with the returned sprite.

spritecollideany(sprite, group, collided = None) -> None No collision

If the sprite collides with any single sprite in the group, a single sprite from the group is returned. On no collision None is returned.

If you don’t need all the features of the pygame.sprite.spritecollide() function, this function will be a bit quicker.

The collided argument is a callback function used to calculate if two sprites are colliding. It should take two sprites as values and return a bool value indicating if they are colliding. If collided is not passed, then all sprites must have a “rect” value, which is a rectangle of the sprite area, which will be used to calculate the collision.

pygame.mixer

pygame module for loading and playing sounds

pygame.mixer.init — initialize the mixer module

pygame.mixer.pre\_init — preset the mixer init arguments

pygame.mixer.quit — uninitialize the mixer

pygame.mixer.get\_init — test if the mixer is initialized

pygame.mixer.stop — stop playback of all sound channels

pygame.mixer.pause — temporarily stop playback of all sound channels

pygame.mixer.unpause — resume paused playback of sound channels

pygame.mixer.fadeout — fade out the volume on all sounds before stopping

pygame.mixer.set\_num\_channels — set the total number of playback channels

pygame.mixer.get\_num\_channels — get the total number of playback channels

pygame.mixer.set\_reserved — reserve channels from being automatically used

pygame.mixer.find\_channel — find an unused channel

pygame.mixer.get\_busy — test if any sound is being mixed

pygame.mixer.Sound — Create a new Sound object from a file or buffer object

pygame.mixer.Channel — Create a Channel object for controlling playback

This module contains classes for loading Sound objects and controlling playback. The mixer module is optional and depends on SDL\_mixer. Your program should test that pygame.mixerpygame module for loading and playing sounds is available and intialized before using it.

The mixer module has a limited number of channels for playback of sounds. Usually programs tell pygame to start playing audio and it selects an available channel automatically. The default is 8 simultaneous channels, but complex programs can get more precise control over the number of channels and their use.

All sound playback is mixed in background threads. When you begin to play a Sound object, it will return immediately while the sound continues to play. A single Sound object can also be actively played back multiple times.

The mixer also has a special streaming channel. This is for music playback and is accessed through the pygame.mixer.musicpygame module for controlling streamed audio module.

The mixer module must be initialized like other pygame modules, but it has some extra conditions. The pygame.mixer.init() function takes several optional arguments to control the playback rate and sample size. Pygame will default to reasonable values, but pygame cannot perform Sound resampling, so the mixer should be initialized to match the values of your audio resources.

NOTE: Not to get less laggy sound, use a smaller buffer size. The default is set to reduce the chance of scratchy sounds on some computers. You can change the default buffer by calling pygame.mixer.pre\_init()preset the mixer init arguments before pygame.mixer.init()initialize the mixer module or pygame.init()initialize all imported pygame modules is called. For example: pygame.mixer.pre\_init(44100,-16,2, 1024) The default size was changed from 1024 to 3072 in pygame 1.8.

pygame.mixer.init()

initialize the mixer module

init(frequency=22050, size=-16, channels=2, buffer=4096) -> None

Initialize the mixer module for Sound loading and playback. The default arguments can be overridden to provide specific audio mixing. Keyword arguments are accepted. For backward compatibility where an argument is set zero the default value is used (possible changed by a pre\_init call).

The size argument represents how many bits are used for each audio sample. If the value is negative then signed sample values will be used. Positive values mean unsigned audio samples will be used. An invalid value raises an exception.

The channels argument is used to specify whether to use mono or stereo. 1 for mono and 2 for stereo. No other values are supported (negative values are treated as 1, values greater than 2 as 2).

The buffer argument controls the number of internal samples used in the sound mixer. The default value should work for most cases. It can be lowered to reduce latency, but sound dropout may occur. It can be raised to larger values to ensure playback never skips, but it will impose latency on sound playback. The buffer size must be a power of two (if not it is rounded up to the next nearest power of 2).

Some platforms require the pygame.mixerpygame module for loading and playing sounds module to be initialized after the display modules have initialized. The top level pygame.init() takes care of this automatically, but cannot pass any arguments to the mixer init. To solve this, mixer has a function pygame.mixer.pre\_init() to set the proper defaults before the toplevel init is used.

It is safe to call this more than once, but after the mixer is initialized you cannot change the playback arguments without first calling pygame.mixer.quit().

Comments 2

pygame.mixer.pre\_init()

preset the mixer init arguments

pre\_init(frequency=22050, size=-16, channels=2, buffersize=4096) -> None

Call pre\_init to change the defaults used when the real pygame.mixer.init() is called. Keyword arguments are accepted. The best way to set custom mixer playback values is to call pygame.mixer.pre\_init() before calling the top level pygame.init(). For backward compatibility argument values of zero is replaced with the startup defaults.

Comments 1

pygame.mixer.quit()

uninitialize the mixer

quit() -> None

This will uninitialize pygame.mixerpygame module for loading and playing sounds. All playback will stop and any loaded Sound objects may not be compatible with the mixer if it is reinitialized later.

Comments 1

pygame.mixer.get\_init()

test if the mixer is initialized

get\_init() -> (frequency, format, channels)

If the mixer is initialized, this returns the playback arguments it is using. If the mixer has not been initialized this returns None

pygame.mixer.stop()

stop playback of all sound channels

stop() -> None

This will stop all playback of all active mixer channels.

pygame.mixer.pause()

temporarily stop playback of all sound channels

pause() -> None

This will temporarily stop all playback on the active mixer channels. The playback can later be resumed with pygame.mixer.unpause()

pygame.mixer.unpause()

resume paused playback of sound channels

unpause() -> None

This will resume all active sound channels after they have been paused.

pygame.mixer.fadeout()

fade out the volume on all sounds before stopping

fadeout(time) -> None

This will fade out the volume on all active channels over the time argument in milliseconds. After the sound is muted the playback will stop.

pygame.mixer.set\_num\_channels()

set the total number of playback channels

set\_num\_channels(count) -> None

Sets the number of available channels for the mixer. The default value is 8. The value can be increased or decreased. If the value is decreased, sounds playing on the truncated channels are stopped.

pygame.mixer.get\_num\_channels()

get the total number of playback channels

get\_num\_channels() -> count

Returns the number of currently active playback channels.

Comments 2

pygame.mixer.set\_reserved()

reserve channels from being automatically used

set\_reserved(count) -> None

The mixer can reserve any number of channels that will not be automatically selected for playback by Sounds. If sounds are currently playing on the reserved channels they will not be stopped.

This allows the application to reserve a specific number of channels for important sounds that must not be dropped or have a guaranteed channel to play on.

Comments 2

pygame.mixer.find\_channel()

find an unused channel

find\_channel(force=False) -> Channel

This will find and return an inactive Channel object. If there are no inactive Channels this function will return None. If there are no inactive channels and the force argument is True, this will find the Channel with the longest running Sound and return it.

If the mixer has reserved channels from pygame.mixer.set\_reserved() then those channels will not be returned here.

pygame.mixer.get\_busy()

test if any sound is being mixed

get\_busy() -> bool

Returns True if the mixer is busy mixing any channels. If the mixer is idle then this return False.

Comments 1

pygame.mixer.Sound

Create a new Sound object from a file or buffer object

Sound(filename) -> Sound

Sound(file=filename) -> Sound

Sound(buffer) -> Sound

Sound(buffer=buffer) -> Sound

Sound(object) -> Sound

Sound(file=object) -> Sound

Sound(array=object) -> Sound

pygame.mixer.Sound.play — begin sound playback

pygame.mixer.Sound.stop — stop sound playback

pygame.mixer.Sound.fadeout — stop sound playback after fading out

pygame.mixer.Sound.set\_volume — set the playback volume for this Sound

pygame.mixer.Sound.get\_volume — get the playback volume

pygame.mixer.Sound.get\_num\_channels — count how many times this Sound is playing

pygame.mixer.Sound.get\_length — get the length of the Sound

pygame.mixer.Sound.get\_raw — return a bytestring copy of the Sound samples.

Load a new sound buffer from a filename, a python file object or a readable buffer object. Limited resampling will be performed to help the sample match the initialize arguments for the mixer. A Unicode string can only be a file pathname. A Python 2.x string or a Python 3.x bytes object can be either a pathname or a buffer object. Use the ‘file’ or ‘buffer’ keywords to avoid ambiguity; otherwise Sound may guess wrong. If the array keyword is used, the object is expected to export a version 3, C level array interface or, for Python 2.6 or later, a new buffer interface (The object is checked for a buffer interface first.)

The Sound object represents actual sound sample data. Methods that change the state of the Sound object will the all instances of the Sound playback. A Sound object also exports an array interface, and, for Python 2.6 or later, a new buffer interface.

The Sound can be loaded from an OGG audio file or from an uncompressed WAV.

Note: The buffer will be copied internally, no data will be shared between it and the Sound object.

For now buffer and array support is consistent with sndarray.make\_sound for Numeric arrays, in that sample sign and byte order are ignored. This will change, either by correctly handling sign and byte order, or by raising an exception when different. Also, source samples are truncated to fit the audio sample size. This will not change.

pygame.mixer.Sound(buffer) is new in pygame 1.8 pygame.mixer.SoundCreate a new Sound object from a file or buffer object keyword arguments and array interface support new in pygame 1.9.2

play()

begin sound playback

play(loops=0, maxtime=0, fade\_ms=0) -> Channel

Begin playback of the Sound (i.e., on the computer’s speakers) on an available Channel. This will forcibly select a Channel, so playback may cut off a currently playing sound if necessary.

The loops argument controls how many times the sample will be repeated after being played the first time. A value of 5 means that the sound will be played once, then repeated five times, and so is played a total of six times. The default value (zero) means the Sound is not repeated, and so is only played once. If loops is set to -1 the Sound will loop indefinitely (though you can still call stop() to stop it).

The maxtime argument can be used to stop playback after a given number of milliseconds.

The fade\_ms argument will make the sound start playing at 0 volume and fade up to full volume over the time given. The sample may end before the fade-in is complete.

This returns the Channel object for the channel that was selected.

stop()

stop sound playback

stop() -> None

This will stop the playback of this Sound on any active Channels.

fadeout()

stop sound playback after fading out

fadeout(time) -> None

This will stop playback of the sound after fading it out over the time argument in milliseconds. The Sound will fade and stop on all actively playing channels.

set\_volume()

set the playback volume for this Sound

set\_volume(value) -> None

This will set the playback volume (loudness) for this Sound. This will immediately affect the Sound if it is playing. It will also affect any future playback of this Sound. The argument is a value from 0.0 to 1.0.

get\_volume()

get the playback volume

get\_volume() -> value

Return a value from 0.0 to 1.0 representing the volume for this Sound.

get\_num\_channels()

count how many times this Sound is playing

get\_num\_channels() -> count

Return the number of active channels this sound is playing on.

get\_length()

get the length of the Sound

get\_length() -> seconds

Return the length of this Sound in seconds.

get\_raw()

return a bytestring copy of the Sound samples.

get\_raw() -> bytes

Return a copy of the Sound object buffer as a bytes (for Python 3.x) or str (for Python 2.x) object.

New in pygame 1.9.2.

Comments 7

pygame.mixer.Channel

Create a Channel object for controlling playback

Channel(id) -> Channel

pygame.mixer.Channel.play — play a Sound on a specific Channel

pygame.mixer.Channel.stop — stop playback on a Channel

pygame.mixer.Channel.pause — temporarily stop playback of a channel

pygame.mixer.Channel.unpause — resume pause playback of a channel

pygame.mixer.Channel.fadeout — stop playback after fading channel out

pygame.mixer.Channel.set\_volume — set the volume of a playing channel

pygame.mixer.Channel.get\_volume — get the volume of the playing channel

pygame.mixer.Channel.get\_busy — check if the channel is active

pygame.mixer.Channel.get\_sound — get the currently playing Sound

pygame.mixer.Channel.queue — queue a Sound object to follow the current

pygame.mixer.Channel.get\_queue — return any Sound that is queued

pygame.mixer.Channel.set\_endevent — have the channel send an event when playback stops

pygame.mixer.Channel.get\_endevent — get the event a channel sends when playback stops

Return a Channel object for one of the current channels. The id must be a value from 0 to the value of pygame.mixer.get\_num\_channels().

The Channel object can be used to get fine control over the playback of Sounds. A channel can only playback a single Sound at time. Using channels is entirely optional since pygame can manage them by default.

play()

play a Sound on a specific Channel

play(Sound, loops=0, maxtime=0, fade\_ms=0) -> None

This will begin playback of a Sound on a specific Channel. If the Channel is currently playing any other Sound it will be stopped.

The loops argument has the same meaning as in Sound.play(): it is the number of times to repeat the sound after the first time. If it is 3, the sound will be played 4 times (the first time, then three more). If loops is -1 then the playback will repeat indefinitely.

As in Sound.play(), the maxtime argument can be used to stop playback of the Sound after a given number of milliseconds.

As in Sound.play(), the fade\_ms argument can be used fade in the sound.

stop()

stop playback on a Channel

stop() -> None

Stop sound playback on a channel. After playback is stopped the channel becomes available for new Sounds to play on it.

pause()

temporarily stop playback of a channel

pause() -> None

Temporarily stop the playback of sound on a channel. It can be resumed at a later time with Channel.unpause()

unpause()

resume pause playback of a channel

unpause() -> None

Resume the playback on a paused channel.

fadeout()

stop playback after fading channel out

fadeout(time) -> None

Stop playback of a channel after fading out the sound over the given time argument in milliseconds.

set\_volume()

set the volume of a playing channel

set\_volume(value) -> None

set\_volume(left, right) -> None

Set the volume (loudness) of a playing sound. When a channel starts to play its volume value is reset. This only affects the current sound. The value argument is between 0.0 and 1.0.

If one argument is passed, it will be the volume of both speakers. If two arguments are passed and the mixer is in stereo mode, the first argument will be the volume of the left speaker and the second will be the volume of the right speaker. (If the second argument is None, the first argument will be the volume of both speakers.)

If the channel is playing a Sound on which set\_volume() has also been called, both calls are taken into account. For example:

sound = pygame.mixer.Sound("s.wav")

channel = s.play() # Sound plays at full volume by default

sound.set\_volume(0.9) # Now plays at 90% of full volume.

sound.set\_volume(0.6) # Now plays at 60% (previous value replaced).

channel.set\_volume(0.5) # Now plays at 30% (0.6 \* 0.5).

get\_volume()

get the volume of the playing channel

get\_volume() -> value

Return the volume of the channel for the current playing sound. This does not take into account stereo separation used by Channel.set\_volume(). The Sound object also has its own volume which is mixed with the channel.

get\_busy()

check if the channel is active

get\_busy() -> bool

Returns true if the channel is activily mixing sound. If the channel is idle this returns False.

get\_sound()

get the currently playing Sound

get\_sound() -> Sound

Return the actual Sound object currently playing on this channel. If the channel is idle None is returned.

queue()

queue a Sound object to follow the current

queue(Sound) -> None

When a Sound is queued on a Channel, it will begin playing immediately after the current Sound is finished. Each channel can only have a single Sound queued at a time. The queued Sound will only play if the current playback finished automatically. It is cleared on any other call to Channel.stop() or Channel.play().

If there is no sound actively playing on the Channel then the Sound will begin playing immediately.

get\_queue()

return any Sound that is queued

get\_queue() -> Sound

If a Sound is already queued on this channel it will be returned. Once the queued sound begins playback it will no longer be on the queue.

set\_endevent()

have the channel send an event when playback stops

set\_endevent() -> None

set\_endevent(type) -> None

When an endevent is set for a channel, it will send an event to the pygame queue every time a sound finishes playing on that channel (not just the first time). Use pygame.event.get() to retrieve the endevent once it’s sent.

Note that if you called Sound.play(n) or Channel.play(sound,n), the end event is sent only once: after the sound has been played “n+1” times (see the documentation of Sound.play).

If Channel.stop() or Channel.play() is called while the sound was still playing, the event will be posted immediately.

The type argument will be the event id sent to the queue. This can be any valid event type, but a good choice would be a value between pygame.locals.USEREVENT and pygame.locals.NUMEVENTS. If no type argument is given then the Channel will stop sending endevents.

get\_endevent()

get the event a channel sends when playback stops

get\_endevent() -> type

Returns the event type to be sent every time the Channel finishes playback of a Sound. If there is no endevent the function returns pygame.NOEVENT.

Comments 1