Pygame 中文文档

pygame-最顶层的 Pygame 模块

pygame.init—初始化所有导入的 pygame 模块
pygame.quit—卸载掉导入的 pygame 模块
pygame.error—标准 pygame 异常
pygame.get_error—得到当前的错误信息
pygame.set_error—设置当前的错误信息
pygame.get_sdl_version—得到 SDL 的版本号
pygame.get_sdl_byteorder—获取 SDL 的字节顺序
pygame.register_quit—在 pygame 退出时注册一个函数
pygame.encode_string—编码一个 Unicode 或字节对象
pygame.encode_file_path—将 Unicode 或字节对象编码为文件系统路径

pygame 包代表了其他人使用的顶层包。Pygame 本身被分解成许多子模块，但这并不影响使用 Pygame 的程序。

为了方便起见，pygame 中的大多数顶级变量都被放置在名为“pygame.localas”的模块中。这是用来和 pygame 一起使用的。本地间谍游戏的常数导入，除了“导入 pygame”之外。
当你导入 pygame 的时候，所有可用的 pygame 子模块都是自动导入的。请注意，一些 pygame 模块被认为是“可选的”，并且可能无法使用。在这种情况下，pygame 将提供一个占位符对象，而不是模块，后者可用于测试可用性。

pygame.init() 初始化所有导入的 pygame 模块

init() -> (numpass, numfail)

初始化所有导入的 pygame 模块。如果一个模块失败，将不会出现异常，但是如果成功且失败了，那么它的总数量将作为一个元组返回。您总是可以手动初始化各个模块，但是 pygame.init（）初始化所有导入的 pygame 模块，这是一种方便的方法来启动一切。个别模块的 init（）函数将在失败时引发异常。

您可能想要分别初始化不同的模块以加速您的程序，或者不使用您的游戏所没有的东西。可以安全地调用 init（）不止一次：重复调用将没有效果。即使你有 pygame.quit（）所有的模块，这也是正确的。

pygame.quit() 卸载所有 pygame 模块

quit() -> None

取消先前已初始化的所有 pygame 模块。当 Python 解释器关闭时，不管怎样，这个方法都被调用，所以您的程序不需要它，除非它想要终止它的 pygame 资源并继续。调用这个函数是安全的：重复调用没有效果。注意，pygame.quit（）取消初始化所有 pygame 模块不会退出您的程序。考虑让您的程序以与普通 python 程序结束相同的方式结束。

pygame.get_error()获取当前的错误消息

get_error() -> errorstr

SDL 维护一个内部错误消息。当 pygame.error（）标准 pygame 异常被提出时，通常会给您这个消息。你很少需要调用这个函数。

pygame.set_error()设置当前错误消息

set_error(error_msg) -> None

pygame.get_sdl_version()获得 SDL 的版本号

get_sdl_version() -> major, minor, patch
返回 SDL 库的三个版本号。这个版本是在编译时构建的。它可以用来检测哪些功能可能无法通过 pygame 获得。

pygame.get_sdl_byteorder()获取 SDL 的字节顺序

get_sdl_byteorder() -> int
返回 SDL 库的字节顺序。它返回 lilendian，为小的 endian 字节顺序和 bigendian，为大的 endian 字节顺序。

pygame.register_quit()在 pygame 退出时运行一个函数

register_quit(callable) -> None
当 pygame.quit（）取消所有 pygame 模块的初始化时，所有注册的退出函数都被调用。Pygame 模块在初始化时自动完成。对于普通的 pygame 用户来说，这个功能是不需要的。

pygame.encode_string()编码一个 Unicode 或字节对象

encode_string([obj [, encoding [, errors [, etype]]]]) -> bytes or None
obj:如果 Unicode 编码;如果字节,返回一成不变的;如果有其他的，不要返回;如果没有给出，就会提高语法错误。
encoding （字符串）：如果存在，编码要使用。默认值是“unicode_escape”。
errors （字符串）：如果给定，如何处理不可编码字符。默认值是“backslashreplace”。
etype（例外类型）：如果给定，则为编码错误增加异常类型。默认的是 UnicodeEncodeError，由 pyunicodeasencodedstring（）返回。对于默认的编码和错误值，应该没有编码错误。
这个函数用于编码文件路径。支持关键字参数。

pygame.encode_file_path()将 Unicode 或字节对象编码为文件系统路径 encode_file_path([obj [, etype]]) -> bytes or None

obj:如果 Unicode 编码;如果字节,返回一成不变的;如果有其他的，不要返回;如果没有给出，就会提高语法错误。
etype（例外类型）：如果给定，则为编码错误增加异常类型。默认的是 UnicodeEncodeError，由 pyunicodeasencodedstring（）返回。
这个函数用于在 pygame 中编码文件路径。编码是由 sys.getfilesystemencoding（）返回的编解码器。支持关键字参数。

pygame.version.ver 返回的版本号作为字符串

pygame.version.vernum 返回版本的整型元组

pygame.version.rev 构建库的版本修订

pygame.time 监测时间的 pygame 模块

pygame.time.get_ticks—得到以毫秒为间隔的时间
pygame.time.wait—暂停程序一段时间
pygame.time.delay—暂停程序一段时间
pygame.time.set_timer—在事件队列上重复创建事件
pygame.time.Clock—创建一个对象来帮助跟踪时间

pygame 中的时间以毫秒为单位（1/1000 秒）。大多数平台的时间分辨率都在 10 毫秒左右。这个分辨率在毫秒内，是在时间分辨率常数中给出的。

pygame.time.get_ticks() 以毫秒为间隔

get_ticks() -> milliseconds
返回自 pygame.init（）调用以来的毫秒数。在 pygame 初始化之前，这个总是为 0。

pygame.time.wait() 暂停程序一段时间

wait(milliseconds) -> time
会暂停一个给定的毫秒数。这个函数可以休眠进程，以便与其他程序共享处理器。一个等待数毫秒的程序将消耗非常少的处理器时间。它比 pygame.time.延迟（）函数稍微不那么准确。这将返回实际使用的毫秒数。

pygame.time.delay() 暂停程序一段时间

delay(milliseconds) -> time
会暂停一个给定的毫秒数。这个函数将使用处理器（而不是休眠）来使延迟比 pygame.time.wait（）更准确。这将返回实际使用的毫秒数。

pygame.time.set_timer() 在事件队列上重复创建事件

set_timer(eventid, milliseconds) -> None
设置事件类型，在每给定的毫秒数上出现在事件队列上。第一个事件要到时间的流逝才会出现。每个事件类型都可以有一个单独的计时器连接到它。最好是使用 pygame.USEREVENT 和 pygame.NUMEVENTS 之间的值。为了使事件的计时器失效，将毫秒参数设置为 0。

pygame.time.Clock 创建一个对象来帮助跟踪时间

Clock() -> Clock
pygame.time.Clock.tick—更新时钟
pygame.time.Clock.tick_busy_loop—更新时钟
pygame.time.Clock.get_time—在前一个滴答声中使用的时间
pygame.time.Clock.get_rawtime—在前一个滴答声中使用的实际时间
pygame.time.Clock.get_fps—计算时钟帧速率

创建一个新的时钟对象，它可以用来跟踪大量的时间。时钟还提供了一些功能来帮助控制游戏的 framerate。

tick()更新时钟

tick(framerate=0) -> milliseconds

这个方法应该在每个帧中调用一次。它将计算自上次调用以来已经传递了多少毫秒。如果你通过了可选的 framerate 参数，这个函数将会延迟，以使游戏运行速度低于给定的每秒滴答数。这可以用来帮助限制游戏的运行时速度。通过在每帧中调用时钟。滴答（40）一次，程序将永远不会以每秒 40 帧的速度运行。注意，这个函数使用 sdl 延时函数，它在每个平台上都不准确，但是不使用太多的 CPU。如果你想要一个精确的计时器，请使用 tickbusyloop，并且不介意咀嚼 CPU。

tick_busy_loop()更新时钟

tick_busy_loop(framerate=0) -> milliseconds

这个方法应该在每个帧中调用一次。它将计算自上次调用以来已经传递了多少毫秒。如果你通过了可选的 framerate 参数，这个函数将会延迟，以使游戏运行速度低于给定的每秒滴答数。这可以用来帮助限制游戏的运行时速度。通过在每帧中调用时钟。滴答滴答（40）一次，这个程序永远不会超过 40 帧每秒。注意，这个函数使用 pygame.time.delay()暂停程序的时间，它在一个繁忙的循环中使用大量的 CPU 来确保时间更准确。

get_time()在前一个滴答声中使用的时间

get_time() -> milliseconds

在前两个调用之间传递的毫秒数到 Clock.tick()。

get_rawtime()在前一个滴答声中使用的实际时间

get_rawtime() -> milliseconds

类似于 Clock.get_time()，但不包括 Clock.tick()延迟限制 framerate 的时间。

get_fps()计算时钟帧速率

get_fps() -> float

计算你的游戏的 framerate（每秒帧数）。它是通过对 Clock.tick()的最后十个调用来计算的。

pygame 中的 mouse 模块

1、pygame.mouse 函数功能介绍

鼠标函数可以用来获取鼠标设备的当前状态。这些函数还可以修改鼠标的系统指针。

pygame.mouse.get_pressed	得到鼠标按钮的状态信息
pygame.mouse.get_pos	得到鼠标箭头的位置坐标
pygame.mouse.get_rel	获取鼠标移动的数量
pygame.mouse.set_pos	设置鼠标箭头的位置坐标
pygame.mouse.set_visible	隐藏或者显示鼠标箭头
pygame.mouse.get_focused	检查程序是否正在接收来自鼠标的数据
pygame.mouse.set_cursor	为系统鼠标光标设置图像
pygame.mouse.get_cursor	获取系统鼠标光标的图像

当设置显示模式时，事件队列将开始接收鼠标事件。鼠标按钮被按下和释放时会产生 pygame.MOUSEBUTTONDOWN 和 pygame.MOUSEBUTTONUP 事件。这些事件包含一个按钮属性，表示按下了哪个按钮。鼠标滚轮会产生 pygame。MOUSEBUTTONDOWN pygame。鼠标滚轮在滚动时会产生 pygame.MOUSEBUTTONUP 和 pygame.MOUSEBUTTONDOWN 事件。当轮子向上滚动时，这个按钮将被设置为 4，当轮子向下滚动时，按钮被设置成 5。当鼠标移动时，它会生成一个 pygame.MOUSEMOTION 事件。鼠标移动被分解成小而精确的运动事件。当鼠标移动时，许多动作事件将被放置在队列上。没有正确清理事件队列的鼠标移动事件是事件队列填满的主要原因。

如果鼠标光标被隐藏，并且输入被抓取到当前的显示，鼠标将进入一个虚拟输入模式，鼠标的相对移动将永远不会被屏幕的边界停止。查看函数 pygame.mouse.set_visible() 和 pygame.event.set_grab()来得到这个配置。

2、pygame.mouse 函数详解

pygame.mouse.get_pressed()

获取鼠标按钮的状态

get_pressed() -> (button1, button2, button3)

返回一个表示所有鼠标按钮状态的布尔序列。值为 1 或者 True 意味着在调用的时候鼠标正在被按下。

注意，要获得所有鼠标事件，最好使用它看看他们是否是 MOUSEBUTTONDOWN、MOUSEBUTTONUP 或者 MOUSEMOTION。

注意，在 X11 中，一些 X 服务器使用中间按钮模拟。当您同时单击两个按钮 1 和 3 时，可以发出一个 2 按钮事件。

注意，记住在这个函数之前调用 pygame.event.get()。否则它就行不通了。

pygame.mouse.get_pos()

获取鼠标光标位置

get_pos() -> (x, y)

返回鼠标光标的 X 和 Y 位置。这个位置相对于显示器左上角的位置。光标位置可以位于显示窗口的外部，但总是被限制在屏幕上。

pygame.mouse.get_rel()

获取鼠标移动的数量
get_rel() -> (x, y)
返回 X 和 Y 的移动量,这是之前对该函数的调用。鼠标光标的相对移动被限制在屏幕的边缘，但是可以看到虚拟输入鼠标模式。在页面顶部描述了虚拟输入模式。

pygame.mouse.set_pos()

设置鼠标光标位置
set_pos([x, y]) -> None
将当前鼠标位置设置为给定的参数。如果鼠标指针是可见的，它将跳转到新的坐标。移动鼠标将会产生一个新的 pygame.MOUSEMOTION 事件。

pygame.mouse.set_visible()

隐藏或显示鼠标光标
set_visible(bool) -> bool
如果 bool 参数是 True，那么鼠标光标将是可见的。这将返回游标的前一个可见状态。

pygame.mouse.get_focused()

检查显示是否接收了鼠标输入
get_focused() -> bool
当 pygame 接收到鼠标输入事件时，返回 true。
当在窗口中工作时，这种方法非常有用。相比之下，在全屏模式下，这种方法总是返回 true。
注意:在 MS 窗口下，有鼠标焦点的窗口也有键盘焦点。但在 x-windows 下，一个窗口可以接收鼠标事件和另一个接收键盘事件。pygame.mouse.get_focused()指示 pygame 窗口是否接收到鼠标事件。

pygame.mouse.set_cursor()

为系统鼠标光标设置图像
set_cursor(size, hotspot, xormasks, andmasks) -> None
当鼠标光标可见时，它将被显示为一个黑白的位图，使用给定的位掩码阵列。大小是一个包含游标宽度和高度的序列。Hotspot 是一个包含游标热点位置的序列。xormasks 是包含游标数据掩码的字节序列。最后是 andmasks，这是一个包含了指针位掩码数据的字节序列。宽度必须是 8 的倍数，并且蒙版必须是给定宽度和高度的正确大小。否则就会出现异常。参考 pygame.cursor 模块用于帮助创建系统游标的默认设置和自定义掩码。

pygame.mouse.get_cursor()

获取系统鼠标光标的图像
get_cursor() -> (size, hotspot, xormasks, andmasks)
获取关于鼠标系统光标的信息。返回值与传递给 pygame.mouse.set_cursor()的参数相同。

pygame 中的 Color 模块

1、pygame.Color 函数功能介绍

用于颜色表示的 pygame 对象

pygame.Color.r	得到或设置颜色的红色值
pygame.Color.g	得到或设置颜色的绿色值
pygame.Color.b	得到或设置颜色的蓝色值
pygame.Color.a	得到或设置颜色的透明度
pygame.Color.cmy	获取或设置 cmy 表示颜色
pygame.Color.hsva	获取或设置 hsva 表示的颜色
pygame.Color.hsla	获取或设置 hsla 表示颜色
pygame.Color.i1i2i3	获取或设置 i1i2i3 表示颜色
pygame.Color.normalize	返回规范化的 RGBA 值
pygame.Color.correct_gamma	对颜色应用一个特定的伽马值
pygame.Color.set_length	将颜色的元素设置为 1,2,3 或 4

Color(name) -> Color
Color(r, g, b, a) -> Color
Color(rgbvalue) -> Color

颜色类表示 RGBA 颜色值，使用 0-255 的值范围。它允许基本的算术运算+、-、//、%和一元操作来创建新的颜色，支持转换到其他颜色空间，例如 HSV 或 HSL，让你调整单一的颜色通道。Alpha 默认值为 255，而没有给出。correct_gamma()方法保存子类。对于二进制操作符，返回的颜色的类是操作符的左手颜色对象。

rgbvalue 可以是一个颜色名称、一个 HTML 颜色格式字符串、一个十六进制数字字符串，或者一个整数像素值。HTML 格式是#rrggbbaa，rr、gg、bb 和 aa 是 2 个字符的十六进制数字。阿尔法 aa 是可选的。十六进制数字字符串有形式 0xrrggbbaa，其中 aa 是可选的。

颜色对象支持与其他颜色对象和 3 或 4 个整数元组(新的 1.9.0)进行相等的比较。在 pygame 1.8.1 中有一个 bug，默认的 alpha 值为 0，而不是之前的 255。

颜色对象导出 C 级数组接口。该接口导出一个只读的一维无符号字节数组，其长度与颜色相同。对于 CPython 2.6 和以后，新的缓冲区接口也被导出，具有与数组接口相同的特征。在 pygame 1.9.2 新。

浮点除法，//，模数，%，运算符不会在 0 处引起一个异常。相反，如果一个颜色，或者 alpha 通道，在右边的颜色是 0，那么结果就是 0。例如：

```
# These expressions are True
Color(255, 255, 255, 255)//Color(0, 64, 64, 64) == Color(0, 3, 3, 3)
Color(255, 255, 255, 255) % Color(64, 64, 64, 0) == Color(63, 63, 63, 0)
```

颜色的新实现是在 pygame 1.8.1 中完成的。

2、pygame.color 函数详解

r

得到或设置颜色的红色值。

r -> int

g

得到或设置颜色的绿色值。

g -> int

b

得到或设置颜色的蓝色值。

b -> int

a

获取或设置颜色的 alpha 值。

a -> int

cmy

获取或设置 CMY 表示颜色。

cmy -> tuple

颜色的 CMY 表示。CMY 分量范围在 C=0~1，M=0~1，Y=0~1。请注意，在所有情况下，这将不会返回完全正确的 CMY 值。由于 0-255 的 RGB 映射和 CMY 映射从 0-1 的舍入错误可能会导致 CMY 值与您所期望的略有不同。

hsva

获取或设置 HSVA 的颜色表示。

hsva -> tuple

HSVA 的颜色表示。HSVA 的分量在 H=0~360，S=0~100，V=0~100，A=0~100。请注意，在所有情况下，这将不会返回完全精确的 HSV 值。由于 0-255 的 RGB 映射和 HSV 映射从 0-100 和 0-360 的舍入错误可能导致 HSV 值与您期望的略有不同。

hsla

获取或设置颜色的 HSLA 表示。

hsla -> tuple

颜色的 HSLA 表示。HSLA 组件在 H=0~360，S=0~100，V=0~100（官网是 V 我怀疑这是 L），A=0~100。请注意，在所有情况下，这将不会返回完全精确的 HSL 值。由于 0-255 的 RGB 映射以及从 0-100 和 0-360 的舍入误差的 HSL 映射，可能导致 HSL 值与您所期望的略有不同。

i1i2i3

获取或设置颜色的 I1I2I3 表示。

i1i2i3 -> tuple

颜色的 I1I2I3 表示。I1I2I3 组件在 I1=0~1、I2=-0.5~0.5、I3=-0.5~0.5。请注意，在所有情况下，这将不会返回完全正确的 I1I2I3 值。由于 0-255 的 RGB 映射和 I1I2I3 映射的 0-1 舍入错误可能导致 I1I2I3 的值与您预期的略有不同。

normalize()

返回颜色的规范化的 RGBA 值。

normalize() -> tuple

将颜色的规范化的 RGBA 值作为浮点值返回。

correct_gamma()

对颜色应用一个特定的伽马值。

correct_gamma (gamma) -> Color

对颜色应用一个特定的伽马值，并使用调整后的 RGBA 值返回一个新的颜色。

set_length()

将颜色的元素设置为 1,2,3 或 4。

set_length(len) -> None

默认的颜色长度是 4。颜色可以有长度 1,2,3 或 4。如果你想把它解到 r g b 而不是 r g b a，这很有用。如果你想要得到一个颜色的长度，请使用 len(acolor)。在 pygame 1.9.0 中更新。

pygame 中的 key 模块

pygame 中的使用键盘的模块

pygame.key.get_focused	是否显示正在接收来自系统的键盘输入
pygame.key.get_pressed	获得所有键盘按钮的状态
pygame.key.get_mods	确定哪些修饰符被持有
pygame.key.set_mods	临时设置哪些修饰符键被按下
pygame.key.set_repeat	控件如何重复控制键
pygame.key.get_repeat	查看持有的键是如何重复的
pygame.key.name	得到一个键标识符的名称

这个模块包含了处理键盘的函数。

当键盘按钮按下并释放时，事件队列得到了 pygame.KEYDOWN 和 pygame.KEYUP 事件。这两个事件都有一个键属性，它是一个代表键盘上所有键的整数 ID。

pygame.KEYDOWN 事件有额外的属性 unicode 和 scancode。unicode 表示一个字符字符串，它是输入的完全转换的字符。这将考虑到移位和组合键。scancode 表示特定于平台的密钥代码。这可能与键盘和键盘不同，但对于键选择像多媒体键这样的怪异键是很有用的。

有很多键盘常量，它们被用来表示键盘上的键。下面是所有键盘常量的列表:

```
KeyASCII      ASCII   Common Name
K_BACKSPACE   \b      backspace
K_TAB         \t      tab
K_CLEAR               clear
K_RETURN      \r      return
K_PAUSE               pause
K_ESCAPE      ^[      escape
K_SPACE               space
K_EXCLAIM     !       exclaim
K_QUOTEDBL    "       quotedbl
K_HASH        #       hash
K_DOLLAR      $       dollar
K_AMPERSAND   &       ampersand
K_QUOTE               quote
K_LEFTPAREN   (       left parenthesis
K_RIGHTPAREN  )       right parenthesis
K_ASTERISK    *       asterisk
K_PLUS        +       plus sign
K_COMMA       ,       comma
K_MINUS       -       minus sign
K_PERIOD      .       period
K_SLASH       /       forward slash
K_0           0       0
K_1           1       1
K_2           2       2
```

K_3	3	3
K_4	4	4
K_5	5	5
K_6	6	6
K_7	7	7
K_8	8	8
K_9	9	9
K_COLON	:	colon
K_SEMICOLON	;	semicolon
K_LESS	<	less-than sign
K_EQUALS	=	equals sign
K_GREATER	>	greater-than sign
K_QUESTION	?	question mark
K_AT	@	at
K_LEFTBRACKET	[	left bracket
K_BACKSLASH	\	backslash
K_RIGHTBRACKET	]	right bracket
K_CARET	^	caret
K_UNDERSCORE	_	underscore
K_BACKQUOTE	`	grave
K_a	a	a
K_b	b	b
K_c	c	c
K_d	d	d
K_e	e	e
K_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_r	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            delete
K_KP0               keypad 0
K_KP1               keypad 1
K_KP2               keypad 2
K_KP3               keypad 3
K_KP4               keypad 4
K_KP5               keypad 5
K_KP6               keypad 6
K_KP7               keypad 7
K_KP8               keypad 8
K_KP9               keypad 9
K_KP_PERIOD   .      keypad period
K_KP_DIVIDE   /      keypad divide
K_KP_MULTIPLY *      keypad multiply
K_KP_MINUS    -      keypad minus
K_KP_PLUS     +      keypad plus
K_KP_ENTER    \r     keypad enter
K_KP_EQUALS   =      keypad equals
K_UP                up arrow
K_DOWN              down arrow
K_RIGHT             right arrow
K_LEFT              left arrow
K_INSERT            insert
K_HOME              home
K_END               end
K_PAGEUP            page up
K_PAGEDOWN          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            numlock
K_CAPSLOCK           capslock
K_SCROLLOCK          scrollock
K_RSHIFT             right shift
K_LSHIFT             left shift
K_RCTRL              right control
K_LCTRL              left control
K_RALT               right alt
K_LALT               left alt
K_RMETA              right meta
K_LMETA              left meta
K_LSUPER             left Windows key
K_RSUPER             right Windows key
K_MODE               mode shift
K_HELP               help
K_PRINT              print screen
K_SYSREQ             sysrq
K_BREAK              break
K_MENU               menu
K_POWER              power
K_EURO               Euro
```

键盘也有一个修饰符列表，可以通过对它们进行逐位的方式来组装

```
KMOD_NONE, KMOD_LSHIFT, KMOD_RSHIFT, KMOD_SHIFT, KMOD_CAPS,KMOD_LCTRL,
KMOD_RCTRL, KMOD_CTRL, KMOD_LALT, KMOD_RALT,KMOD_ALT, KMOD_LMETA,
KMOD_RMETA, KMOD_META, KMOD_NUM, KMOD_MODE
```

pygame.key.get_focused()

如果显示正在接收来自系统的键盘输入返回 True

```
get_focused() -> bool
```

当显示窗口有来自系统的键盘焦点时，这是正确的。如果显示需要确保它不会丢失键盘焦点，它可以使用 pygame.event.set_grab() 来获取所有的输入。

pygame.key.get_pressed()

获得所有键盘按钮的状态

```
get_pressed() -> bools
```

返回一个布尔值的序列，表示键盘上的每个键的状态。使用键常量值来索引数组。一个真正的值意味着那个按钮被按下。

使用这个函数获取按钮的列表并不是处理用户的文本条目的正确方法。您无法知道按下键的顺序，可以调用 pygame.key.get_pressed()，可以完全忽略键的顺序。也没有办法将这些被推的键转换成一个完全翻译的字符值。参考 pygame.KEYDOWN 这个功能的事件队列上的事件。

pygame.key.get_mods()

确定正在进行哪些修饰符键

```
get_mods() -> int
```

返回一个整数，表示被持有的所有修饰符的位掩码。使用位操作符可以测试是否按下了特定的 shift 键、capslock 按钮的状态等等。

pygame.key.set_mods()

临时设置修改器的按键

```
set_mods(int) -> None
```

创建一个你想要对你的程序施加的修改器常量的位掩码。

pygame.key.set_repeat()

控制持有的键是如何重复的

```
set_repeat() -> None
set_repeat(delay, interval) -> None
```

当键盘重复被激活时，被压制的键会产生多个 pygame.KEYDOWN 事件。delay 是第一次 pygame.KEYDOWN 事件发出后重复发出延迟的毫秒数。之后,另一 pygame.KEYDOWN 事件每隔一段 delay 时间就会发送。如果没有参数被传递，那么关键的重复就会被禁用。当初始化 pygame 时，将禁用密钥重复。

pygame.key.get_repeat()

查看持有键是如何重复的

```
get_repeat() -> (delay, interval)
```

当键盘重复被激活时，被压制的键会产生多个 pygame.KEYDOWN 事件。delay 是第一次重复的 pygame 之前的毫秒数。KEYDOWN 将被发送。之后,另一个 pygame.KEYDOWN 事件每隔一段时间就会发送。当初始化 pygame 时，将禁用密钥重复。

在 pygame 1.8 中更新。

pygame.key.name()

获取关键标识符的名称

```
name(key) -> string
```

从键盘按钮 id 常量中获取按钮的描述性名称。

pygame 中的 music 模块

控制流音频的 pygame 模块

pygame.mixer.music.load	加载一个用于播放的音乐文件
pygame.mixer.music.play	开始播放音乐流
pygame.mixer.music.rewind	重新启动音乐
pygame.mixer.music.stop	停止播放音乐
pygame.mixer.music.pause	暂停音乐播放
pygame.mixer.music.unpause	恢复暂停的音乐
pygame.mixer.music.fadeout	在淡出后停止播放音乐
pygame.mixer.music.set_volume	设置音量
pygame.mixer.music.get_volume	获取音乐音量
pygame.mixer.music.get_busy	检查音乐流是否在播放
pygame.mixer.music.set_pos	设置的位置
pygame.mixer.music.get_pos	获得音乐播放时间
pygame.mixer.music.queue	队列一个音乐文件以跟随当前
pygame.mixer.music.set_endevent	当播放停止时，音乐会发送一个事件
pygame.mixer.music.get_endevent	当播放停止时，获取一个通道发送的事件

音乐模块与 pygame.mixer 紧密相连。用于加载和播放声音的 pygame.mixer 模块。使用音乐模块控制在混音器中播放音乐。音乐播放和常规的声音回放之间的区别在于，音乐是流媒体播放的，而且从来没有真正加载过。混音器系统只支持一次单一的音乐流。请注意，MP3 支持是有限的。在某些系统上，不支持的格式可能会破坏程序，例如 Debian Linux。考虑使用 OGG。

pygame.mixer.music.load()

加载一个播放音乐的文件

load(filename) -> None

load(object) -> None

这将加载一个音乐文件名/文件对象并准备播放。如果一个音乐流已经播放，它就会被停止。这并不是音乐的开始。

pygame.mixer.music.play()

开始播放音乐流

play(loops=0, start=0.0) -> None

这将播放载入的音乐流。如果音乐已经播放，它就会重新启动。

循环参数控制音乐播放的次数。播放(5)将使音乐播放一次，然后重复 5 次，总共是 6 次。如果循环是-1，那么音乐就会无限重复。

起始位置的参数控制着歌曲开始播放的地方。起始位置取决于音乐演奏的格式。MP3 和 OGG 以时间为单位(以秒为单位)。MOD 音乐是模式的序号。如果不能设置起始位置，通过一个 startpos 将会抛出一个 NotImplementedError。

pygame.mixer.music.rewind()

重新启动音乐

rewind() -> None

将当前音乐的播放重新设置为一开始

pygame.mixer.music.stop()

停止音乐播放

stop() -> None

如果当前播放音乐，停止播放音乐。

pygame.mixer.music.pause()

暂时停止音乐播放

pause() -> None

暂时停止播放音乐流。它可以用 pygame.mixer.music.unpause()函数恢复。

pygame.mixer.music.unpause()

恢复暂停音乐

unpause() -> None

这将在暂停之后重新播放音乐流。

pygame.mixer.music.fadeout()

在淡出后停止播放音乐

fadeout(time) -> None

这将在指定的时间(以毫秒为单位)消失后停止播放音乐。

请注意，此函数将阻塞，直到音乐淡出。

pygame.mixer.music.set_volume()

调节音乐音量

set_volume(value) -> None

设置音乐播放的音量。值参数在 0.0 和 1.0 之间。当加载新音乐时，音量就会重置。

pygame.mixer.music.get_volume()

得到音乐音量

get_volume() -> value

返回混合器的当前音量。值将在 0.0 和 1.0 之间。

pygame.mixer.music.get_busy()

检查音乐流是否在播放

get_busy() -> bool

当音乐流在积极播放时，就会返回 True。当音乐空闲时，返回 False

pygame.mixer.music.set_pos()

设定播放位置

set_pos(pos) -> None

这将在播放播放的音乐文件中设置位置。“pos”的含义，一个浮点数(或一个可以转换为浮点数的数字)，取决于音乐的格式。对于 MOD 文件，它是模块中的整数模式号。从声音的开始，在几秒钟内，就会得到绝对的位置。对于 MP3 文件，它是相对位置，在几秒内，从当前位置。对于 MP3 文件中的绝对定位，首先调用 rewind()。其他文件格式是不支持的。更新版本的 sdl 混音版本比以前更有定位支持。如果某个特定格式不支持定位，则会提高一个 SDLError。这将在播放播放的音乐文件中设置位置。“pos”的含义，一个浮点数(或一个可以转换为浮点数的数字)，取决于音乐的格式。对于 MOD 文件，它是模块中的整数模式号。从声音的开始，在几秒钟内，就会得到绝对的位置。对于 MP3 文件，它是相对位置，在几秒内，从当前位置。对于 MP3 文件中的绝对定位，首先调用后退()。其他文件格式是不支持的。更新版本的 sdl 混音版本比以前更有定位支持。如果某个特定格式不支持定位，则会提高一个 SDLError。

函数 setpos()调用下划线的 sdl 混音函数 mixsetmusic 睡姿。

在 pygame 1.9.2 中更新。

pygame.mixer.music.get_pos()

获得音乐播放时间

get_pos() -> time

这就得到了音乐一直在播放的毫秒数。返回的时间只代表了音乐播放的时间;它没有考虑任何起始位置偏移量。

pygame.mixer.music.queue()

按当前的格式排队一个音乐文件，下一首播放

queue(filename) -> None

这将加载一个音乐文件并对其进行排队。当当前的音乐自然结束时，一个排队的音乐文件就会开始。如果当前的音乐停止或改变，排队的歌就会消失。

下面的例子将演奏巴赫的音乐六次，然后演奏莫扎特的音乐:

```
pygame.mixer.music.load('bach.ogg')
pygame.mixer.music.play(5)
pygame.mixer.music.queue('mozart.ogg')
```

pygame.mixer.music.set_endevent()

当播放停止时，音乐会发送一个事件

set_endevent() -> None

set_endevent(type) -> None

这使得 pygame 在播放音乐时发出信号(通过事件队列的方式)。参数确定将要排队的事件的类型。

每次音乐结束时，这个事件都会被排队，而不仅仅是第一次。为了防止事件被排队，请调用这个方法，没有参数。

pygame.mixer.music.get_endevent()

当播放停止时，获取一个通道发送的事件
get_endevent() -> type
返回每次音乐结束播放时发送的事件类型。如果没有 endevent，函数将返回 pygame.NOEVENT。

pygame.draw

用于绘制形状的 pygame 模块
pygame.draw.rect - 画一个矩形的形状
pygame.draw.polygon - 绘制具有任意数量边的形状
pygame.draw.circle - 围绕一个点画一个圆圈
pygame.draw.ellipse - 在矩形内绘制圆形
pygame.draw.arc - 绘制椭圆的局部剖面
pygame.draw.line - 绘制一条直线段
pygame.draw.lines - 绘制多个连续的线段
pygame.draw.aaline - 绘制精细的抗锯齿线
pygame.draw.aalines - 绘制连接的抗锯齿线序列
在 Surface 上绘制几个简单的形状。这些函数可用于渲染任何格式的 Surface。渲染到硬件 Surfaces 将比常规软件 Surfaces 慢。
大多数函数使用 width 参数来表示形状边缘周围的笔触大小。如果宽度为 0，则函数实际上将实心填充整个形状。
所有绘图功能都遵循 Surface 的剪辑区域，并将限制在该区域。这些函数返回一个矩形，表示已更改像素的边界区域。
大多数参数都接受一个 RGB 三元组的颜色参数。这些也可以接受 RGBA 四胞胎。如果 alpha 值包含像素 alpha，则 alpha 值将直接写入 Surface，但绘制函数不会透明绘制。color 参数也可以是已映射到 Surface 的像素格式的整数像素值。
这些功能必须暂时锁定它们正在操作的 Surface。通过在绘制调用周围锁定和解锁 Surface 对象，可以加快许多顺序绘图调用。

pygame.draw.rect（）
画一个矩形的形状
rect（Surface，color，Rect，width = 0） - > Rect
在 Surface 上绘制一个矩形形状。给定的 Rect 是矩形的区域。width 参数是绘制外边缘的粗细。如果 width 为零，则填充矩形。
请记住，该 Surface.fill()方法也适用于绘制填充矩形。事实上，Surface.fill()在某些平台上可以通过软件和硬件显示模式进行硬件加速。

pygame.draw.polygon（）
绘制具有任意数量边的形状
polygon（Surface，color，pointlist，width = 0） - > Rect
在 Surface 上绘制多边形。pointlist 参数是多边形的顶点。width 参数是绘制外边缘的粗细。如果 width 为零，则填充多边形。

对于 aapolygon，使用带有'closed'参数的 aalines。

pygame.draw.circle（）
围绕一个点画一个圆圈
circle（Surface，color，pos，radius，width = 0） - > Rect
在 Surface 上绘制圆形。pos 参数是圆的中心，radius 是大小。width 参数是绘制外边缘的粗细。如果宽度为零，则圆圈将被填充。

pygame.draw.ellipse（）
在矩形内绘制圆形
椭圆（Surface，color，Rect，width = 0） - > Rect
在 Surface 上绘制椭圆形状。给定的矩形是圆圈将填充的区域。width 参数是绘制外边缘的粗细。如果 width 为零，则将填充椭圆。

pygame.draw.arc（）
绘制椭圆的局部剖面
arc（Surface，color，Rect，start_angle，stop_angle，width = 1） - > Rect
在 Surface 上绘制一个椭圆弧。rect 参数是椭圆将填充的区域。两个角度参数是以弧度表示的初始和最终角度，右侧为零。width 参数是绘制外边缘的粗细。

pygame.draw.line（）
绘制一条直线段
line（Surface，color，start_pos，end_pos，width = 1） - > Rect
在 Surface 上绘制直线段。没有端盖，端部是粗线的方形。

pygame.draw.lines（）
绘制多个连续的线段
线条（Surface，color，closed，pointlist，width = 1） - > Rect
在 Surface 上绘制一系列线条。pointlist 参数是一系列由一条线连接的点。如果 closed 参数为 true，则在第一个和最后一个点之间绘制一个额外的线段。

这不会绘制任何端盖或斜接接头。具有尖角和宽线宽的线条可能具有不正确的视角。

pygame.draw.aaline（）
绘制精细的抗锯齿线
aaline（Surface，color，startpos，endpos，blend = 1） - > Rect
在曲面上绘制抗锯齿线。这将遵循剪裁矩形。受影响区域的边界框将作为矩形返回。如果 blend 为 true，则阴影将与现有像素阴影混合而不是覆盖它们。此函数接受端点的浮点值。

pygame.draw.aalines（）
绘制连接的抗锯齿线序列
aalines（Surface，color，closed，pointlist，blend = 1） - > Rect

在表面上绘制序列。您必须在点序列中至少传递两个点。closed 参数是一个简单的布尔值，如果为 true，则在第一个和最后一个点之间绘制一条线。布尔混合参数设置为 true 将将阴影与现有阴影混合而不是覆盖它们。此函数接受端点的浮点值。

pygame.image

用于图像传输的 pygame 模块
pygame.image.load - 从文件加载新图像
pygame.image.save - 将图像保存到磁盘
pygame.image.get_extended - 测试是否可以加载扩展图像格式
pygame.image.tostring - 将图像传输到字符串缓冲区
pygame.image.fromstring - 从字符串缓冲区创建新的 Surface
pygame.image.frombuffer - 创建一个在字符串缓冲区内共享数据的新 Surface
图像模块包含用于加载和保存图片的功能，以及将 Surface 转换为其他包可用的格式。。

请注意，没有 Image 类; 图像作为 Surface 对象加载。Surface 类允许操作（绘制线条，设置像素，捕获区域等）。

图像模块是 pygame 的必需依赖项，但它只能选择性地支持任何扩展文件格式。默认情况下，它只能加载未压缩的 BMP 图像。使用完整图像支持构建时，该 pygame.image.load() 功能可以支持以下格式。

JPG
PNG
GIF （非动画）
BMP
PCX
TGA （未压缩）
TIF
LBM（和 PBM）
PBM（和 PGM，PPM）
XPM
保存图像仅支持一组有限的格式。您可以保存为以下格式。
BMP
TGA
PNG
JPEG
PNG，JPEG 在 pygame 1.8 中保存新功能。

pygame.image.load（）
从文件加载新图像
load（filename） -> Surface

load（fileobj，namehint =“”） - > Surface
从文件源加载图像。您可以传递文件名或类似 Python 文件的对象。

Pygame 将自动确定图像类型（例如，GIF 或位图），并从数据中创建一个新的 Surface 对象。在某些情况下,它需要知道文件扩展名(例如,GIF 图像应以“.gif”结尾）。如果传递原始文件类对象，则可能还希望将原始文件名作为 namehint 参数传递。

返回的 Surface 将包含与其来源相同的颜色格式，colorkey 和 alpha 透明度。您通常希望 Surface.convert()不带参数调用,以创建一个可以在屏幕上更快地绘制的副本。

对于 Alpha 透明度，例如.png 图像，请 convert_alpha() 在加载后使用该方法，以使图像具有每像素透明度。

可能并不总是构建 Pygame 来支持所有图像格式。至少它将支持未压缩 BMP。如果 pygame.image.get_extended() 返回“True”，您应该能够加载大多数图像（包括 PNG，JPG 和 GIF）。

您应该使用 os.path.join()兼容性。

eg. asurf = pygame.image.load(os.path.join('data', 'bla.png'))

pygame.image.save（）
将图像保存到磁盘
save（Surface，filename） - >无
这将你的面保存无论是作为 BMP，TGA，PNG，或 JPEG 图像。如果文件扩展名无法识别，则默认为 TGA。两者 TGA 和 BMP 文件格式都会创建未压缩的文件。

PNG，JPEG 在 pygame 1.8 中保存新功能。

pygame.image.get_extended（）
测试是否可以加载扩展图像格式
get_extended（） - > bool
如果 pygame 是使用扩展图像格式构建的，则此函数将返回 True。仍然无法确定哪些格式可用，但通常您可以将它们全部加载。

pygame.image.tostring（）
将图像传输到字符串缓冲区
tostring（Surface，format，flipped = False） - > string
创建一个可以使用其他 Python 映像包中的“fromstring”方法传输的字符串。一些 Python 图像包更喜欢它们的图像，从底部到顶部格式（例如 PyOpenGL）。如果为翻转的参数传递 True，则字符串缓冲区将垂直翻转。

format 参数是以下值之一的字符串。请注意，只有 8 位 Surface 可以使用“P”格式。其他格式适用于任何 Surface。另请注意，其他 Python 映像包支持的格式比 pygame 更多。

P，8 位调色表面
RGB，24 位图像
RGBX，32 位图像，未使用空间
RGBA，带有 alpha 通道的 32 位图像
ARGB，首先是 alpha 通道的 32 位图像
RGBA_PREMULT，32 位图像，颜色由 alpha 通道缩放
ARGB_PREMULT，32 位图像，颜色由 alpha 通道缩放，alpha 通道优先
评论 3
pygame.image.fromstring（）
从字符串缓冲区创建新的 Surface
fromstring（string，size，format，flipped = False） - > Surface
此函数采用类似的参数 pygame.image.tostring()。size 参数是一对表示宽度和高度的数字。创建新 Surface 后，您可以销毁字符串缓冲区。

大小和格式图像必须计算与传递的字符串缓冲区完全相同的大小。否则将引发异常。

请参阅该 pygame.image.frombuffer()方法，以便将图像传输到 pygame 中。

pygame.image.frombuffer（）
创建一个在字符串缓冲区内共享数据的新 Surface
frombuffer（字符串，大小，格式） - > Surface
创建一个直接从字符串缓冲区共享像素数据的新 Surface。此方法采用相同的参数 pygame.image.fromstring()，但无法垂直翻转源数据。

这比 pygame.image.fromstring()从字符串缓冲区创建新 Surface 要快得多，因为不必分配和复制像素数据。

pygame.font

用于加载和渲染字体的 pygame 模块
pygame.font.init - 初始化字体模块
pygame.font.quit - 取消初始化字体模块
pygame.font.get_init - 如果字体模块已初始化，则为 true
pygame.font.get_default_font - 获取默认字体的文件名
pygame.font.get_fonts - 获取所有可用的字体
pygame.font.match_font - 在系统上找到特定的字体
pygame.font.SysFont - 从系统字体创建一个 Font 对象
pygame.font.Font - 从文件创建一个新的 Font 对象
字体模块允许将 TrueType 字体呈现为新的 Surface 对象。它接受任何 UCS-2 字符（'u0001'到'uFFFF'）。此模块是可选的，需要 SDL_ttf 作为依赖项。 在尝

试使用模块之前，您应该测试 pygame.font 用于加载和渲染字体的 pygame 模块是否可用并初始化。

使用字体完成的大部分工作都是使用实际的 Font 对象完成的。模块本身只有初始化模块和创建 Font 对象的例程 pygame.font.Font()。

您可以使用该 pygame.font.SysFont() 功能从系统加载字体。还有一些其他功能可以帮助查找系统字体。

Pygame 附带内置默认字体。可以通过传递 None 作为字体名称来访问它。

要使用 pygame.freetype 加载和渲染计算机字体增强 Pygame 的模块基于 pygame.ftfont 作为 pygame.fontpygame 的模块加载和渲染字体的首次进口之前定义环境变量 PYGAME_FREETYPE pygame 顶级 pygame 的包。Module pygame.ftfont 是一个 pygame.fontpygame 模块，用于加载和渲染字体 兼容模块，它通过除了一个字体模块单元测试之外的所有模块：它没有基于 SDL_ttf 的字体模块的 UCS-2 限制，因此无法引发代码点的异常大于'uFFFF'。如果 pygame.freetype 用于加载和呈现计算机字体的增强型 pygame 模块不可用，则将加载 SDL_ttf 字体模块。

pygame.font.init（）
初始化字体模块
init（） - >无
此方法由 pygame.init()。自动调用。它初始化字体模块。在任何其他功能起作用之前，必须初始化模块。

不止一次调用此函数是安全的。

评论 54
pygame.font.quit（）
取消初始化字体模块
退出（） - >无
手动取消初始化 SDL_ttf 的字体系统。这是由自动调用的 pygame.quit()。

即使当前未初始化字体，也可以安全地调用此函数。

pygame.font.get_init（）
如果字体模块已初始化，则为 true
get_init（） - > bool
测试字体模块是否已初始化。

pygame.font.get_default_font（）
获取默认字体的文件名
get_default_font（） - > string

返回系统字体的文件名。这不是文件的完整路径。此文件通常可以在与字体模块相同的目录中找到，但也可以捆绑在单独的存档中。

pygame.font.get_fonts（）
获取所有可用的字体
get_fonts（） - >字符串列表
返回系统上可用的所有字体的列表。字体的名称将设置为小写，并删除所有空格和标点符号。这适用于大多数系统，但有些系统会在找不到字体时返回空列表。

评论 7
pygame.font.match_font（）
在系统上找到特定的字体
match_font（name，bold = False，italic = False） - > path
返回系统上字体文件的完整路径。如果将粗体或斜体设置为 true，则会尝试查找正确的字体系列。

字体名称实际上可以是逗号分隔的字体名称列表。如果找不到任何给定名称，则返回 None。

例：

```
print pygame.font.match_font('bitstreamverasans')
# output is: /usr/share/fonts/truetype/ttf-bitstream-vera/Vera.ttf
# (but only if you have Vera on your system)
```

评论 2
pygame.font.SysFont（）
从系统字体创建一个 Font 对象
SysFont（名称，大小，粗体=假，斜体=假） - >字体
返回从系统字体加载的新 Font 对象。该字体将匹配请求的粗体和斜体标志。如果找不到合适的系统字体，则会在加载默认的 pygame 字体时返回。字体名称可以是逗号分隔的要查找的字体名称列表。

评论 5
pygame.font.Font
从文件创建一个新的 Font 对象
字体（文件名，大小） - >字体
字体（对象，大小） - >字体
pygame.font.Font.render - 在新 Surface 上绘制文本
pygame.font.Font.size - 确定渲染文本所需的空间量
pygame.font.Font.set_underline - 控制文本是否使用下划线呈现
pygame.font.Font.get_underline - 检查文本是否将使用下划线呈现
pygame.font.Font.set_bold - 启用粗体文本的伪渲染
pygame.font.Font.get_bold - 检查文本是否将呈现为粗体
pygame.font.Font.set_italic - 启用斜体文本的虚假渲染

pygame.font.Font.metrics - 获取传递的字符串中每个字符的指标
pygame.font.Font.get_italic - 检查文本是否将呈斜体
pygame.font.Font.get_linesize - 获取字体文本的行间距
pygame.font.Font.get_height - 获取字体的高度
pygame.font.Font.get_ascent - 得到字体的上升
pygame.font.Font.get_descent - 得到字体的下降
从给定文件名或 python 文件对象加载新字体。大小是字体的高度（以像素为单位）。如果文件名为 None，则将加载 pygame 默认字体。如果无法从参数中加载字体，则会引发异常。创建字体后，无法更改大小。

字体对象主要用于将文本渲染到新的 Surface 对象中。渲染可以模拟粗体或斜体特征，但最好从具有实际斜体或粗体字形的字体加载。渲染文本可以是常规字符串或 unicode。

render（）
在新 Surface 上绘制文本
渲染（文本，抗锯齿，颜色，背景=无） - >表面
这将创建一个新的 Surface，并在其上呈现指定的文本。pygame 无法直接在现有 Surface 上绘制文本：您必须使用它 Font.render()来创建文本的图像（Surface），然后将此图像 blit 到另一个 Surface 上。

文本只能是一行：不呈现换行符。空字符（'x00'）引发 TypeError。Unicode 和 char（字节）字符串都被接受。对于 Unicode 字符串，只能识别 UCS-2 字符（'u0001'到'uFFFF'）。任何更大的东西都会引发 UnicodeError。对于 char 字符串，LATIN1 假定编码。antialias 参数是一个布尔值：如果为 true，则字符将具有平滑边。颜色参数是文本的颜色[例如：蓝色的（0,0,255）]。可选的背景参数是用于文本背景的颜色。如果没有传递背景，则文本外部的区域将是透明的。

返回的 Surface 将具有保存文本所需的尺寸。（与 Font.size（）返回的相同）。如果为文本传递空字符串，则将返回一个像素宽的空白表面和字体的高度。

根据所使用的背景和抗锯齿类型，返回不同类型的 Surface。出于性能原因，最好知道将使用何种类型的图像。如果未使用抗锯齿，则返回图像将始终为具有双色调色板的 8 位图像。如果背景是透明的，则将设置颜色键。抗锯齿图像渲染为 24 位 RGB 图像。如果背景是透明的，则将包括像素 alpha。

优化：如果您知道文本的最终目的地（在屏幕上）将始终具有纯色背景，并且文本是抗锯齿的，则可以通过指定背景颜色来提高性能。这将导致生成的图像通过 colorkey 维护透明度信息，而不是（效率低得多）alpha 值。

如果渲染'\ n'，将呈现未知的字符。通常是一个矩形。相反，你需要自己处理新的线条。

字体呈现不是线程安全的：只有一个线程可以随时呈现文本。

size（）
确定渲染文本所需的空间量
尺寸（文字） - >（宽度，高度）
返回呈现文本所需的尺寸。这可用于帮助确定文本在呈现之前所需的定位。它还可以用于文字包装和其他布局效果。

请注意，大多数字体都使用字距调整来调整特定字母对的宽度。例如，“ae”的宽度并不总是与“a”+“e”的宽度相匹配。

set_underline（）
控制文本是否使用下划线呈现
set_underline（bool） - >无
启用后，所有渲染字体都将包含下划线。无论字体大小如何，下划线始终为一个像素厚。这可以与粗体和斜体模式混合使用。

get_underline（）
检查文本是否将使用下划线呈现
get_underline（） - > bool
启用字体下划线时返回 True。

set_bold（）
启用粗体文本的伪渲染
set_bold（bool） - >无
启用粗体呈现文本。这是对许多字体类型看起来不太好的字体的假拉伸。如果可能，从真正的粗体字体文件加载字体。粗体时，字体的宽度与正常情况不同。这可以与斜体和下划线模式混合使用。

get_bold（）
检查文本是否将呈现为粗体
get_bold（） - > bool
启用字体粗体渲染模式时返回 True。

set_italic（）
启用斜体文本的虚假渲染
set_italic（bool） - >无
启用斜体文本的虚假渲染。这是对许多字体类型看起来不太好的字体的假歪斜。如果可能，从真正的斜体字体文件加载字体。斜体字体的宽度与正常情况下的宽度不同。这可以与粗体和下划线模式混合使用。

metrics（）
获取传递的字符串中每个字符的指标
metrics（文本） - > list

该列表包含每个字符的元组，其中包含字符的最小 X 偏移量，最大 X 偏移量，最小 Y 偏移量，最大 Y 偏移量和前进偏移量（方位加宽度）。[（minx，maxx，miny，maxy，advance），（minx，maxx，miny，maxy，advance），...]。列表中没有输入每个无法识别的字符。

get_italic（）
检查文本是否将呈斜体
get_italic（） - > bool
启用字体斜体渲染模式时返回 True。

get_linesize（）
获取字体文本的行间距
get_linesize（） - > int
返回带有字体的文本行的高度（以像素为单位）。渲染多行文本时，这是行之间建议的空间量。

get_height（）
获取字体的高度
get_height（） - > int
返回实际渲染文本的高度（以像素为单位）。这是字体中每个字形的平均大小。

get_ascent（）
得到字体的上升
get_ascent（） - > int
返回字体上升的高度（以像素为单位）。上升是从字体基线到字体顶部的像素数。

get_descent（）
得到字体的下降
get_descent（） - > int
返回字体下降的高度（以像素为单位）。下降是从字体基线到字体底部的像素数。

pygame.display

pygame 模块控制显示窗口和屏幕
pygame.display.init - 初始化显示模块
pygame.display.quit - 取消初始化显示模块
pygame.display.get_init - 如果已初始化显示模块，则返回 True
pygame.display.set_mode - 初始化窗口或屏幕以进行显示
pygame.display.get_surface - 获取当前设置的显示表面的参考
pygame.display.flip - 将完整显示 Surface 更新到屏幕
pygame.display.update - 更新屏幕的部分以显示软件
pygame.display.get_driver - 获取 pygame 显示后端的名称
pygame.display.Info - 创建视频显示信息对象
pygame.display.get_wm_info - 获取有关当前窗口系统的信息

pygame.display.list_modes - 获取可用的全屏模式列表
pygame.display.mode_ok - 为显示模式选择最佳颜色深度
pygame.display.gl_get_attribute - 获取当前显示的 OpenGL 标志的值
pygame.display.gl_set_attribute - 请求显示模式的 OpenGL 显示属性
pygame.display.get_active - 当显示器在显示器上处于活动状态时返回 True
pygame.display.iconify - 图标化显示表面
pygame.display.toggle_fullscreen - 在全屏和窗口显示之间切换
pygame.display.set_gamma - 更改硬件伽玛斜坡
pygame.display.set_gamma_ramp - 使用自定义查找更改硬件伽玛斜坡
pygame.display.set_icon - 更改显示窗口的系统图像
pygame.display.set_caption - 设置当前窗口标题
pygame.display.get_caption - 获取当前窗口标题
pygame.display.set_palette - 设置索引显示的显示调色板
该模块可控制 pygame 显示。Pygame 有一个显示 Surface，可以包含在窗口中，也可以全屏运行。创建显示后，您将其视为常规 Surface。屏幕上无法立即看到更改；您必须从两个翻转功能中选择一个来更新实际显示。

显示的原点（x = 0 和 y = 0）位于屏幕的左上角。两个轴都朝向屏幕的右下方正向增加。

pygame 显示实际上可以在几种模式之一中初始化。默认情况下，显示器是基本的软件驱动帧缓冲器。您可以请求硬件加速和 OpenGL 支持等特殊模块。这些由传递给的标志控制 pygame.display.set_mode()。

Pygame 在任何时候都只能激活一个显示器。创建一个新的 pygame.display.set_mode()将关闭以前的显示。如果需要在像素格式或显示分辨率精确的控制，使用函数 pygame.display.mode_ok()，pygame.display.list_modes()和 pygame.display.Info()查询有关的显示信息。

创建显示 Surface 后，此模块中的功能将影响单个现有显示。如果模块未初始化，则 Surface 将变为无效。如果设置了新的显示模式，现有的 Surface 将自动切换到新显示器上。

设置显示模式后，pygame 事件队列中会放置几个事件。pygame.QUIT 当用户请求程序关闭时发送。窗口将 pygame.ACTIVEEVENT 在显示增益和输入焦点丢失时接收事件。如果使用 pygame.RESIZABLE 标志设置显示，则 pygame.VIDEORESIZE 在用户调整窗口尺寸时将发送事件。直接绘制到屏幕的硬件显示将 pygame.VIDEOEXPOSE 在必须重绘窗口的某些部分时获得事件。

某些显示环境具有自动拉伸所有窗口的选项。启用此选项后，此自动拉伸会扭曲 pygame 窗口的外观。在 pygame examples 目录中，有一个示例代码

（prevent_display_stretching.py），它显示了如何在 Microsoft Windows（Vista 或更新版本）上禁用 pygame 显示的自动拉伸。

pygame.display.init（）
初始化显示模块
init（） - >无
初始化 pygame 显示模块。在初始化之前，显示模块无法执行任何操作。当您呼叫更高级别时，通常会自动为您处理 pygame.init()。

Pygame 会在初始化时从几个内部显示后端中选择一个。将根据当前用户的平台和权限选择显示模式。在初始化显示模块之前，SDL_VIDEODRIVER 可以设置环境变量以控制使用哪个后端。此处列出了具有多个选项的系统。

Windows : windib, directx
Unix : x11, dga, fbcon, directfb, ggi, vgl, svgalib, aalib
在某些平台上，可以将 pygame 显示嵌入到现有窗口中。为此，SDL_WINDOWID 必须将环境变量设置为包含窗口标识或句柄的字符串。初始化 pygame 显示时，将检查环境变量。请注意，在嵌入式显示器中运行时可能会出现许多奇怪的副作用。

多次调用它是无害的，重复调用没有效果。

评论 12
pygame.display.quit（）
取消初始化显示模块
退出（） - >无
这将关闭整个显示模块。这意味着将关闭所有活动显示。程序退出时也会自动处理。

多次调用它是无害的，重复调用没有效果。

pygame.display.get_init（）
如果已初始化显示模块，则返回 True
get_init（） - > bool
如果当前初始化 pygame.display 控制显示窗口和屏幕模块的 pygame 模块，则返回 True 。

pygame.display.set_mode（）
初始化窗口或屏幕以进行显示
set_mode（resolution =（0,0），flags = 0，depth = 0） - > Surface
此功能将创建一个显示 Surface。传入的参数是对显示类型的请求。实际创建的显示将是系统支持的最佳匹配。

resolution 参数是一对表示宽度和高度的数字。flags 参数是其他选项的集合。depth 参数表示用于颜色的位数。

返回的 Surface 可以像常规 Surface 一样绘制，但最终会在监视器上看到更改。

如果未传递分辨率或设置为（0,0）且 pygame 使用 SDL 1.2.10 或更高版本，则创建的 Surface 将具有与当前屏幕分辨率相同的大小。如果仅将宽度或高度设置为 0，则 Surface 将具有与屏幕分辨率相同的宽度或高度。使用 SDL1.2.10 之前的 版本将引发异常。

通常最好不要传递深度参数。它将默认为系统的最佳和最快颜色深度。如果您的游戏需要特定的颜色格式，您可以使用此参数控制深度。Pygame 将模拟不可用的颜色深度，这可能很慢。

请求全屏显示模式时，有时无法完全匹配所请求的分辨率。在这些情况下，pygame 将选择最接近的兼容匹配。返回的曲面仍将始终与请求的分辨率匹配。

flags 参数控制您想要的显示类型。有几种可供选择，您甚至可以使用按位或运算符（管道“|”字符）组合多种类型。如果传递 0 或没有 flags 参数，它将默认为软件驱动的窗口。以下是您要选择的显示标志：

```
pygame.FULLSCREEN    create a fullscreen display
pygame.DOUBLEBUF     recommended for HWSURFACE or OPENGL
pygame.HWSURFACE     hardware accelerated, only in FULLSCREEN
pygame.OPENGL        create an OpenGL-renderable display
pygame.RESIZABLE     display window should be sizeable
pygame.NOFRAME       display window will have no border or controls
```

例如：

```
# Open a window on the screen
screen_width=700
screen_height=400
screen=pygame.display.set_mode([screen_width,screen_height])
```

评论 12

pygame.display.get_surface（）

获取当前设置的显示表面的参考

get_surface（） - > Surface

返回对当前设置的显示 Surface 的引用。如果未设置显示模式，则返回 None。

评论 4

pygame.display.flip（）

将完整显示 Surface 更新到屏幕

flip（） - >无

这将更新整个显示的内容。如果您的显示模式，使用标志 pygame.HWSURFACE 和 pygame.DOUBLEBUF，这将等待垂直回扫和交换表面。如果您使用的是其他类型的显示模式，则只会更新曲面的全部内容。

使用 pygame.OPENGL 显示模式时，这将执行 gl 缓冲区交换。

pygame.display.update（）
更新屏幕的部分以显示软件
更新（矩形=无） - >无
更新（rectangle_list） - >无
此功能类似于 pygame.display.flip()软件显示的优化版本。它只允许更新屏幕的一部分，而不是整个区域。如果没有传递参数，它会更新整个 Surface 区域 pygame.display.flip()。

您可以将函数传递给单个矩形或一系列矩形。一次传递多个矩形比使用单个或部分矩形列表多次调用更新更有效。如果传递一系列矩形，则可以安全地在列表中包含 None 值，这将被跳过。

此调用无法在 pygame.OPENGL 显示器上使用，并将生成异常。

评论 3
pygame.display.get_driver（）
获取 pygame 显示后端的名称
get_driver（） - >名称
Pygame 在初始化时选择许多可用的显示后端之一。这将返回用于显示后端的内部名称。这可用于提供有关可能加速的显示功能的有限信息。请参阅 SDL_VIDEODRIVER 标志 pygame.display.set_mode()以查看一些常见选项。

pygame.display.Info（）
创建视频显示信息对象
Info（） - > VideoInfo
创建一个包含多个属性的简单对象来描述当前图形环境。如果在 pygame.display.set_mode()某些平台可以提供有关默认显示模式的信息之前调用此方法 。在设置显示模式以验证满足特定显示选项后，也可以调用此方法。VidInfo 对象有几个属性：

```
hw:         True if the display is hardware accelerated
wm:         True if windowed display modes can be used
video_mem:  The megabytes of video memory on the display. This is 0 if
unknown
bitsize:    Number of bits used to store each pixel
bytesize:   Number of bytes used to store each pixel
masks:      Four values used to pack RGBA values into pixels
shifts:     Four values used to pack RGBA values into pixels
losses:     Four values used to pack RGBA values into pixels
blit_hw:    True if hardware Surface blitting is accelerated
blit_hw_CC: True if hardware Surface colorkey blitting is accelerated
```

blit_hw_A: True if hardware Surface pixel alpha blitting is accelerated
blit_sw: True if software Surface blitting is accelerated
blit_sw_CC: True if software Surface colorkey blitting is accelerated
blit_sw_A: True if software Surface pixel alpha blitting is accelerated
current_h, current_w: Height and width of the current video mode, or of the
  desktop mode if called before the display.set_mode is called.
  (current_h, current_w are available since SDL 1.2.10, and pygame 1.8.0)
  They are -1 on error, or if an old SDL is being used.
评论 1
pygame.display.get_wm_info（）
获取有关当前窗口系统的信息
get_wm_info（） - > dict
创建一个填充字符串键的字典。字符串和值由系统任意创建。某些系统可能没有信息，将返回空字典。大多数平台将返回一个“窗口”键，其值设置为当前显示的系统 ID。

新的 pygame 1.7.1

pygame.display.list_modes（）
获取可用的全屏模式列表
list_modes（depth = 0，flags = pygame.FULLSCREEN） - > list
此函数返回指定颜色深度的可能尺寸列表。如果给定参数没有可用的显示模式，则返回值将为空列表。返回值为-1 表示任何请求的分辨率都应该有效（对于窗口模式可能就是这种情况）。模式大小从最大到最小排序。

如果深度为 0，SDL 将为显示选择当前/最佳颜色深度。标志默认为 pygame.FULLSCREEN，但您可能需要为特定的全屏模式添加其他标志。

评论 1
pygame.display.mode_ok（）
为显示模式选择最佳颜色深度
mode_ok（size，flags = 0，depth = 0） - > depth
此函数使用与...相同的参数 pygame.display.set_mode()。它用于确定所请求的显示模式是否可用。如果无法设置显示模式，它将返回 0。否则，它将返回与所要求的显示最匹配的像素深度。

通常深度参数不会传递，但某些平台可以支持多个显示深度。如果通过它将暗示哪个深度更好匹配。

要传递的最有用的标志是 pygame.HWSURFACE， pygame.DOUBLEBUF 也许 pygame.FULLSCREEN。如果无法设置这些显示标志，该函数将返回 0。

pygame.display.gl_get_attribute（）

获取当前显示的 OpenGL 标志的值
gl_get_attribute（flag） - > value
在 pygame.display.set_mode()使用 pygame.OPENGL 标志调用之后，最好检查任何请求的 OpenGL 属性的值。有关 pygame.display.gl_set_attribute()有效标志的列表，请参阅 。

pygame.display.gl_set_attribute（）
请求显示模式的 OpenGL 显示属性
gl_set_attribute（flag，value） - >无
pygame.display.set_mode()使用 pygame.OPENGL 标志调用时，Pygame 会自动处理设置 OpenGL 属性，如颜色和双缓冲。OpenGL 提供了一些您可能想要控制的其他属性。将其中一个属性作为标志及其适当的值传递。这必须在之前调用 pygame.display.set_mode()

该 OPENGL 标志;

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
评论 1
pygame.display.get_active（）
当显示器在显示器上处于活动状态时返回 True
get_active（） - > bool
之后 pygame.display.set_mode()被称为显示器表面会显示在屏幕上。大多数窗口显示都可以被用户隐藏。如果显示 Surface 隐藏或图标化，则返回 False。

pygame.display.iconify（）
图标化显示表面
iconify（） - > bool
请求显示表面的窗口被图标化或隐藏。并非所有系统和显示器都支持图标化显示。如果成功，该函数将返回 True。

当显示图标化时 pygame.display.get_active()将返回 False。ACTIVEEVENT 当窗口被图标化时，事件队列应该接收事件。

pygame.display.toggle_fullscreen（）
在全屏和窗口显示之间切换
toggle_fullscreen（） - > bool
在窗口模式和全屏模式之间切换显示窗口。此功能仅适用于 UNIX X11 视频驱动程序。对于大多数情况，最好 pygame.display.set_mode()使用新的显示标志进行调用。

评论 3
pygame.display.set_gamma（）

更改硬件伽玛斜坡
set_gamma（红色，绿色=无，蓝色=无） - >布尔
在显示硬件上设置红色，绿色和蓝色伽玛值。如果未传递绿色和蓝色参数，则它们将与红色相同。并非所有系统和硬件都支持 gamma 斜坡，如果功能成功，它将返回 True。

伽玛值为 1.0 会创建线性颜色表。较低的值会使显示屏变暗，较高的值会变亮。

pygame.display.set_gamma_ramp（）
使用自定义查找更改硬件伽玛斜坡
set_gamma_ramp（红色，绿色，蓝色） - >布尔
使用显式查找表设置红色，绿色和蓝色伽马斜坡。每个参数应该是 256 个整数的序列。整数应介于 0 和 0xffff 之间。并非所有系统和硬件都支持 gamma 斜坡，如果功能成功，它将返回 True。

pygame.display.set_icon（）
更改显示窗口的系统图像
set_icon（Surface） - >无
设置系统将用于表示显示窗口的运行时图标。所有窗口默认为窗口图标的简单 pygame 徽标。

您可以传递任何曲面，但大多数系统需要 32x32 左右的较小图像。图像可以具有 colorkey 透明度，该透明度将传递给系统。

某些系统在显示后不允许更改窗口图标。pygame.display.set_mode()在设置显示模式之前，可以在创建图标之前调用此功能。

评论 5
pygame.display.set_caption（）
设置当前窗口标题
set_caption（title，icontitle = None） - >无
如果显示器有窗口标题，则此功能将更改窗口上的名称。某些系统支持用于最小化显示的备用较短标题。

评论 4
pygame.display.get_caption（）
获取当前窗口标题
get_caption（） - >（title，icontitle）
返回显示 Surface 的标题和 icontitle。这些通常是相同的值。

pygame.display.set_palette（）
设置索引显示的显示调色板
set_palette（palette = None） - >无

这将改变8位显示器的视频显示调色板。这不会更改实际显示Surface的调色板，只会更改用于显示 Surface 的调色板。如果未传递调色板参数，则将还原系统默认调色板。调色板是 RGB 三连音的序列 。

pygame.Surface

用于表示图像的 pygame 对象
Surface((width, height), flags=0, depth=0, masks=None) -> Surface
Surface((width, height), flags=0, Surface) -> Surface

pygame.Surface.blit - 将一个图像绘制到另一个
pygame.Surface.blits - 将许多图像绘制到另一个
pygame.Surface.convert - 更改图像的像素格式
pygame.Surface.convert_alpha - 改变包括每像素 α 的图像的像素格式
pygame.Surface.copy - 创建 Surface 的新副本
pygame.Surface.fill - 用纯色填充 Surface
pygame.Surface.scroll - 将表面图像移动到位
pygame.Surface.set_colorkey - 设置透明颜色键
pygame.Surface.get_colorkey - 获取当前透明的 colorkey
pygame.Surface.set_alpha - 设置完整 Surface 图像的 Alpha 值
pygame.Surface.get_alpha - 获取当前的 Surface 透明度值
pygame.Surface.lock - 锁定 Surface 内存以进行像素访问
pygame.Surface.unlock - 从像素访问中解锁 Surface 存储器
pygame.Surface.mustlock - 测试 Surface 是否需要锁定
pygame.Surface.get_locked - 测试 Surface 是否被当前锁定
pygame.Surface.get_locks - 获取 Surface 的锁
pygame.Surface.get_at - 获取单个像素的颜色值
pygame.Surface.set_at - 设置单个像素的颜色值
pygame.Surface.get_at_mapped - 获取单个像素的映射颜色值
pygame.Surface.get_palette - 获取 8 位 Surface 的颜色索引调色板
pygame.Surface.get_palette_at - 获取调色板中单个条目的颜色
pygame.Surface.set_palette - 设置 8 位 Surface 的调色板
pygame.Surface.set_palette_at - 在 8 位 Surface 调色板中设置单个索引的颜色
pygame.Surface.map_rgb - 将颜色转换为映射的颜色值
pygame.Surface.unmap_rgb - 将映射的整数颜色值转换为颜色
pygame.Surface.set_clip - 设置 Surface 的当前剪切区域
pygame.Surface.get_clip - 获取 Surface 的当前剪切区域
pygame.Surface.subsurface - 创建一个引用其父级的新表面
pygame.Surface.get_parent - 找到地下的父母
pygame.Surface.get_abs_parent - 找到地下的顶级父级
pygame.Surface.get_offset - 在父母中找到子地下的位置

pygame.Surface.get_abs_offset - 在其顶级父级中查找子级子表面的绝对位置
pygame.Surface.get_size - 获取 Surface 的尺寸
pygame.Surface.get_width - 获取 Surface 的宽度
pygame.Surface.get_height - 获得 Surface 的高度
pygame.Surface.get_rect - 得到 Surface 的矩形区域
pygame.Surface.get_bitsize - 获取 Surface 像素格式的位深度
pygame.Surface.get_bytesize - 获取每个 Surface 像素使用的字节数
pygame.Surface.get_flags - 获取用于 Surface 的其他标志
pygame.Surface.get_pitch - 获取每个 Surface 行使用的字节数
pygame.Surface.get_masks - 位掩码需要在颜色和映射的整数之间进行转换
pygame.Surface.set_masks - 设置在颜色和映射整数之间转换所需的位掩码
pygame.Surface.get_shifts - 在颜色和映射的整数之间转换所需的位移
pygame.Surface.set_shifts - 设置在颜色和映射整数之间转换所需的位移
pygame.Surface.get_losses - 用于在颜色和映射整数之间进行转换的有效位
pygame.Surface.get_bounding_rect - 找到包含数据的最小 rect
pygame.Surface.get_view - 返回 Surface 像素的缓冲区视图。
pygame.Surface.get_buffer - 获取 Surface 的像素的缓冲对象。
pygame.Surface._pixels_address - 像素缓冲地址
pygame Surface 用于表示任何图像。Surface 具有固定的分辨率和像素格式。具有 8 位像素的表面使用调色板映射到 24 位颜色。

调用 pygame.Surface()以创建新的图像对象。Surface 将被清除为全黑。唯一需要的参数是尺寸。如果没有其他参数，Surface 将以与显示 Surface 最匹配的格式创建。

可以通过传递位深度或现有 Surface 来控制像素格式。flags 参数是表面附加功能的位掩码。您可以传递这些标志的任意组合：

HWSURFACE, creates the image in video memory
SRCALPHA, the pixel format will include a per-pixel alpha
这两个标志只是一个请求，可能不适用于所有显示和格式。

高级用户可以将一组位掩码与深度值组合在一起。掩码是一组 4 个整数，表示像素中的哪些位代表每种颜色。Normal Surfaces 不应该需要 mask 参数。

曲面可以有许多额外的属性，如 alpha 平面，颜色键，源矩形剪裁。这些函数主要影响 Surface 如何与其他 Surface 进行 blit。blit 例程将尽可能尝试使用硬件加速，否则他们将使用高度优化的软件 blitting 方法。

pygame 支持三种类型的透明度：colorkeys，surface alphas 和 pixel alphas。表面 alphas 可以与 colorkeys 混合使用，但是每像素 alphas 的图像不能使用其

他模式。Colorkey 透明度使单个颜色值透明。不会绘制与 colorkey 匹配的任何像素。曲面 alpha 值是单个值，用于更改整个图像的透明度。表面 alpha 为 255 是不透明的，值为 0 是完全透明的。

每个像素的 alpha 值不同，因为它们为每个像素存储透明度值。这允许最精确的透明效果，但它也是最慢的。每像素 alphas 不能与表面 alpha 和 colorkeys 混合使用。

支持 Surfaces 的像素访问。硬件表面上的像素访问速度很慢，不推荐使用。可以使用 get_at()和 set_at()函数访问像素 。这些方法适用于简单访问，但在使用它们进行像素处理时会相当慢。如果您计划进行大量的像素级工作，建议使用 pygame.PixelArraypygame 对象进行曲面的直接像素访问，这样就可以得到类似于曲面视图的数组。对于所涉及的数学操作，尝试 pygame.surfarray 使用阵列接口 模块访问表面像素数据的 pygame 模块（它非常快，但需要 NumPy。）

任何直接访问曲面像素数据的函数都需要将该曲面锁定（）。这些功能可以 lock()和 unlock()表面本身无需帮助。但是，如果一个函数被多次调用，那么表面的多次锁定和解锁会有很多开销。最好在多次调用函数之前手动锁定曲面，然后在完成后解锁。所有需要锁定表面的函数都会在他们的文档中说明。请记住仅在必要时锁定 Surface。

表面像素在内部存储为单个数字，其中包含编码的所有颜色。使用 Surface.map_rgb()和 Surface.unmap_rgb()将单个红色，绿色和蓝色值之间的转换为该 Surface 的压缩整数。

曲面还可以引用其他曲面的部分。这些是使用该 Surface.subsurface()方法创建的。对 Surface 的任何更改都会影响另一个。

每个 Surface 都包含一个剪切区域。默认情况下，剪辑区域覆盖整个 Surface。如果更改，则所有绘图操作仅影响较小的区域。

blit（）
将一个图像绘制到另一个
blit（source，dest，area = None，special_flags = 0） - > Rect
在此 Surface 上绘制源 Surface。可以使用 dest 参数定位绘图。Dest 可以是表示源左上角的坐标对。Rect 也可以作为目标传递，矩形的 topleft 角将用作 blit 的位置。目标矩形的大小不会影响 blit。

也可以传递可选的区域矩形。这表示要绘制的源 Surface 的较小部分。

可选的特殊标志是传递新的 1.8.0： ，BLEND_ADD， BLEND_SUB，，BLEND_MULT 新的 1.8.1： ， ，，， ，， ，， 随着或许在将来添加其他特殊的 blitting 标志。
BLEND_MINBLEND_MAXBLEND_RGBA_ADDBLEND_RGBA_SUBBLEND_RGBA_MULTBLEND_RG

BA_MINBLEND_RGBA_MAX
BLEND_RGB_ADDBLEND_RGB_SUBBLEND_RGB_MULTBLEND_RGB_MINBLEND_RGB_MAX

返回矩形是受影响像素的区域，不包括目标 Surface 外部或剪切区域外的任何像素。

当 blitting 到 8 位 Surface 时，将忽略像素 alpha。

pygame 1.8 中的 special_flags 新增功能。

对于具有 colorkey 或毯子 alpha 的表面，对于 self 的 blit 可能会给出与非 self-blit 稍微不同的颜色。

blits（）
将许多图像绘制到另一个
blits(blit_sequence=(source, dest), ...), doreturn=1) ->(Rect,...)
blits（（source，dest，area），...）） - >（Rect，...）
blits（（source，dest，area，special_flags），...）） - >（Rect，...）
在此 Surface 上绘制许多曲面。它需要一个序列作为输入，每个元素对应于 Surface.blit()。它至少需要一个（source，dest）序列。

参数：
blit_sequence - 一系列曲面，以及 blit 它们的参数。它们对应于 Surface.blit()参数。
doreturn - 如果为 true，我们返回否则返回 None。
返回：
更改区域的列表。如果 doreturn 为 false，则返回 None。

pygame 1.9.4 中的新功能。

convert（）
更改图像的像素格式
转换（曲面） - >曲面
convert（depth，flags = 0） - > Surface
convert（mask，flags = 0） - > Surface
convert（） - > Surface
使用更改的像素格式创建 Surface 的新副本。可以从另一个现有 Surface 确定新的像素格式。否则，可以使用 depth，flags 和 mask 参数，类似于 pygame.Surface() 调用。

如果没有传递参数，则新 Surface 将具有与显示 Surface 相同的像素格式。这总是最快的 blitting 格式。在将所有 Surface 多次 blit 之前转换它们是一个好主意。

转换后的 Surface 将没有像素 alpha。如果原件有它们，它们将被剥离。请参阅 Surface.convert_alpha()保留或创建每像素 alpha。

新副本将与复制的表面具有相同的类。这使得 Surface 子类继承此方法而无需覆盖，除非子类特定的实例属性也需要复制。

convert_alpha（）
改变包括每像素 α 的图像的像素格式
convert_alpha（Surface） - > Surface
convert_alpha（） - > Surface
使用所需的像素格式创建曲面的新副本。新表面将采用适合快速 blitting 到给定格式的格式，每像素 alpha。如果没有给出表面，则新表面将针对当前显示的 blitting 进行优化。

与该 Surface.convert()方法不同，新图像的像素格式与请求的源不完全相同，但它将针对到目的地的快速 alpha blitting 进行优化。

与 Surface.convert()返回的曲面具有与转换曲面相同的类。

copy（）
创建 Surface 的新副本
copy（） - > Surface
制作 Surface 的副本。新曲面将具有与原始曲面相同的像素格式，调色板，透明度设置和类。如果 Surface 子类还需要复制任何特定于实例的属性，那么它应该覆盖 copy()。

fill（）
用纯色填充 Surface
fill（color，rect = None，special_flags = 0） - > Rect
用纯色填充表面。如果没有给出 rect 参数，则将填充整个 Surface。rect 参数将填充限制为特定区域。填充也将包含在 Surface 剪辑区域中。

color 参数可以是 RGB 序列，RGBA 序列或映射的颜色索引。如果使用 RGBA，RGBA 则忽略 Alpha(A 的一部分),除非表面使用每像素 alpha(Surface 具有 SRCALPHA 标记）。

可选 special_flags 是通过在新的 1.8.0： ，BLEND_ADD， BLEND_SUB，，BLEND_MULT 新的 1.8.1： ， ，，， ，， ，， 与其他特别的 blitting 标志也许在将来添加。
BLEND_MINBLEND_MAXBLEND_RGBA_ADDBLEND_RGBA_SUBBLEND_RGBA_MULTBLEND_RGBA_MINBLEND_RGBA_MAX
BLEND_RGB_ADDBLEND_RGB_SUBBLEND_RGB_MULTBLEND_RGB_MINBLEND_RGB_MAX

这将返回受影响的 Surface 区域。

scroll（）
将表面图像移动到位
滚动（dx = 0，dy = 0） - >无
将图像向右移动 dx 像素，向下移动 dy 像素。对于左侧和上侧滚动，dx 和 dy 可能分别为负。未被覆盖的表面区域保留其原始像素值。滚动包含在 Surface 剪辑区域中。使 dx 和 dy 值超过表面大小是安全的。

pygame 1.9 中的新功能

set_colorkey（）
设置透明颜色键
set_colorkey（Color，flags = 0） - >无
set_colorkey（无） - >无
设置 Surface 的当前颜色键。将此 Surface blitting 到目标上时，与 colorkey 颜色相同的任何像素都将是透明的。颜色可以是 RGB 颜色或映射的颜色整数。如果未传递，则将取消设置颜色键。

如果将 Surface 格式化为使用每像素 alpha 值，则将忽略 colorkey。colorkey 可以与完整的 Surface alpha 值混合使用。

可以将可选的 flags 参数设置为 pygame.RLEACCEL 在非加速显示上提供更好的性能。的 RLEACCEL 表面会更慢进行修改，但更快的 blit 作为源。

get_colorkey（）
获取当前透明的 colorkey
get_colorkey（） - > RGB 或 None
返回 Surface 的当前 colorkey 值。如果未设置 colorkey，则返回 None。

set_alpha（）
设置完整 Surface 图像的 Alpha 值
set_alpha（value，flags = 0） - >无
set_alpha（无） - >无
设置 Surface 的当前 alpha 值。将此 Surface 渲染到目标上时，像素将被绘制为略微透明。alpha 值是 0 到 255 之间的整数，0 表示完全透明，255 表示完全不透明。如果为 Alpha 值传递 None，则将禁用 Surface alpha。

此值与每像素 Surface alpha 不同。对于具有每像素 alpha 的曲面，将忽略覆盖 alpha 并 None 返回。

可以将可选的 flags 参数设置为 pygame.RLEACCEL 在非加速显示上提供更好的性能。的 RLEACCEL 表面会更慢进行修改，但更快的 blit 作为源。

get_alpha（）

获取当前的 Surface 透明度值
get_alpha（） - > int_value
返回 Surface 的当前 Alpha 值。

lock（）
锁定 Surface 内存以进行像素访问
lock（） - >无
锁定 Surface 的像素数据以进行访问。在加速表面上，像素数据可以存储在易失性视频存储器或非线性压缩形式中。当 Surface 锁定时，像素存储器可供常规软件访问。读取或写入像素值的代码需要锁定 Surface。

表面不应保持锁定超过必要的程度。通常不能通过 pygame 显示或管理锁定的 Surface。

并非所有 Surfaces 都需要锁定。该 Surface.mustlock()方法可以确定它是否实际需要。锁定和解锁不需要它的 Surface 没有性能损失。

所有 pygame 函数都会根据需要自动锁定和解锁 Surface 数据。如果一段代码要进行多次重复锁定和解锁 Surface 的调用，将块包装在锁定和解锁对中会很有帮助。

嵌套锁定和解锁呼叫是安全的。只有在释放最终锁定后才能解锁表面。

unlock（）
从像素访问中解锁 Surface 存储器
解锁（） - >无
锁定后解锁 Surface 像素数据。解锁的 Surface 可以再次由 pygame 绘制和管理。有关 Surface.lock()详细信息，请参阅 文档。

所有 pygame 函数都会根据需要自动锁定和解锁 Surface 数据。如果一段代码要进行多次重复锁定和解锁 Surface 的调用，将块包装在锁定和解锁对中会很有帮助。

嵌套锁定和解锁呼叫是安全的。只有在释放最终锁定后才能解锁表面。

mustlock（）
测试 Surface 是否需要锁定
mustlock（） - > bool
如果需要锁定 Surface 以访问像素数据，则返回 True。通常纯软件 Surfaces 不需要锁定。很少需要这种方法，因为根据需要锁定所有 Surface 是安全且最快的。

所有 pygame 函数都会根据需要自动锁定和解锁 Surface 数据。如果一段代码要进行多次重复锁定和解锁 Surface 的调用，将块包装在锁定和解锁对中会很有帮助。

get_locked（）
测试 Surface 是否被当前锁定
get_locked（） - > bool
Surface 锁定时返回 True。Surface 被锁定的次数无关紧要。

get_locks（）
获取 Surface 的锁
get_locks（） - >元组
返回 Surface 的当前现有锁。

get_at（）
获取单个像素的颜色值
get_at（（x，y））- >颜色
返回 RGBA 给定像素的 Color 值的副本。如果 Surface 没有每像素 alpha，那么 alpha 值将始终为 255（不透明）。如果像素位置在 Surface 区域之外，则会引发 IndexError 异常。

一次获取和设置一个像素通常太慢，无法在游戏或实时情况下使用。最好使用一次操作多个像素的方法，比如 blit，fill 和 draw 方法 - 或者使用 surfarray / PixelArray。

此功能将根据需要临时锁定和解锁 Surface。

返回颜色而不是元组，在 pygame 1.9.0 中新增。使用 tuple(surf.get_at((x,y)))，如果你想有一个元组，而不是颜色。这应该只在您想要将颜色用作字典中的键时才有意义。

set_at（）
设置单个像素的颜色值
set_at（（x，y），Color）- >无
RGBA 为单个像素设置或映射整数颜色值。如果 Surface 没有每像素 alphas，则忽略 alpha 值。在“曲面”区域外部或“曲面”剪裁外部设置像素将不起作用。

一次获取和设置一个像素通常太慢，无法在游戏或实时情况下使用。

此功能将根据需要临时锁定和解锁 Surface。

get_at_mapped（）
获取单个像素的映射颜色值
get_at_mapped（（x，y））- >颜色
返回给定像素的整数值。如果像素位置在 Surface 区域之外，则会引发 IndexError 异常。

此方法适用于 pygame 单元测试。它不太可能在应用程序中使用。

此功能将根据需要临时锁定和解锁 Surface。

pygame 中的新功能。1.9.2。

get_palette（）
获取 8 位 Surface 的颜色索引调色板
get_palette（） - > [RGB，RGB，RGB，...]
返回最多 256 个颜色元素的列表，这些颜色元素表示 8 位 Surface 中使用的索引颜色。返回的列表是调色板的副本，更改将对 Surface 无效。

在 pygame 1.9.0 中返回实例列表而不是元组，NewColor(with length 3)

get_palette_at（）
获取调色板中单个条目的颜色
get_palette_at（index） - > RGB
返回“曲面”调板中单个索引的红色，绿色和蓝色值。索引应该是 0 到 255 之间的值。

在 pygame 1.9.0 中返回实例而不是元组，NewColor(with length 3)

set_palette（）
设置 8 位 Surface 的调色板
set_palette（[RGB，RGB，RGB，...]） - >无
设置 8 位 Surface 的完整调色板。这将替换现有调色板中的颜色。可以传递部分调色板，仅更改原始调色板中的第一种颜色。

此功能对每像素超过 8 位的 Surface 没有影响。

set_palette_at（）
在 8 位 Surface 调色板中设置单个索引的颜色
set_palette_at（index，RGB） - >无
在“曲面”调板中为单个条目设置调色板值。索引应该是 0 到 255 之间的值。

此功能对每像素超过 8 位的 Surface 没有影响。

map_rgb（）
将颜色转换为映射的颜色值
map_rgb（Color） - > mapped_int
将 RGBA 颜色转换为此 Surface 的映射整数值。返回的整数将不包含比 Surface 的位深度更多的位。映射的颜色值通常不在 pygame 中使用，但可以传递给大多数需要 Surface 和颜色的函数。

有关颜色和像素格式的详细信息，请参阅 Surface 对象文档。

unmap_rgb（）
将映射的整数颜色值转换为颜色
unmap_rgb（mapped_int） - >颜色
将映射的整数颜色转换 RGB 为此 Surface 的颜色分量。映射的颜色值通常不在 pygame 中使用，但可以传递给大多数需要 Surface 和颜色的函数。

有关颜色和像素格式的详细信息，请参阅 Surface 对象文档。

set_clip（）
设置 Surface 的当前剪切区域
set_clip（rect） - >无
set_clip（无） - >无
每个 Surface 都有一个活动的剪切区域。这是一个矩形，表示 Surface 上可以修改的唯一像素。如果为矩形传递无，则完整的 Surface 将可用于更改。

剪切区域始终限制在 Surface 本身的区域。如果剪辑矩形太大，它将缩小以适合 Surface 内部。

get_clip（）
获取 Surface 的当前剪切区域
get_clip（） - > Rect
返回当前剪切区域的矩形。Surface 将始终返回一个永远不会超出图像边界的有效矩形。如果 Surface 为剪切区域设置了 None，则 Surface 将返回一个带有 Surface 整个区域的矩形。

subsurface（）
创建一个引用其父级的新表面
subsurface（Rect） - > Surface
返回一个与其新父级共享像素的新 Surface。新的 Surface 被认为是原始的孩子。对 Surface 像素的修改将相互影响。剪切区域和颜色键等曲面信息对于每个曲面都是唯一的。

新的 Surface 将从其父级继承调色板，颜色键和 alpha 设置。

在父级上可以有任意数量的子表面和子表面。如果显示模式不是硬件加速，也可以在显示器表面下表面。

请参阅 Surface.get_offset()，Surface.get_parent()以了解有关地下状态的更多信息。

地下将与父表面具有相同的类。

get_parent（）
找到地下的父母
get_parent（） - > Surface
返回地下的父 Surface。如果这不是地下，那么将返回 None。

get_abs_parent（）
找到地下的顶级父级
get_abs_parent（） - > Surface
返回地下的父 Surface。如果这不是地下，那么将返回此表面。

get_offset（）
在父母中找到子地下的位置
get_offset（） - >（x，y）
获取父项内子项下的偏移位置。如果 Surface 不是地下，则返回（0,0）。

get_abs_offset（）
在其顶级父级中查找子级子表面的绝对位置
get_abs_offset（） - >（x，y）
获取子级子表面在其顶级父级 Surface 内的偏移位置。如果 Surface 不是地下，则返回（0,0）。

get_size（）
获取 Surface 的尺寸
get_size（） - >（宽度，高度）
返回 Surface 的宽度和高度（以像素为单位）。

get_width（）
获取 Surface 的宽度
get_width（） - > width
返回 Surface 的宽度（以像素为单位）。

get_height（）
获得 Surface 的高度
get_height（） - >高度
返回 Surface 的高度（以像素为单位）。

get_rect（）
得到 Surface 的矩形区域
get_rect（** kwargs） - > Rect
返回覆盖整个曲面的新矩形。此矩形始终从 0 开始，宽度为 0。和高度与图像大小相同。

您可以将关键字参数值传递给此函数。这些命名值将在返回之前应用于 Rect 的属性。一个例子是'mysurf.get_rect（center =（100,100））'来创建一个以给定位置为中心的 Surface 矩形。

get_bitsize（）
获取 Surface 像素格式的位深度
get_bitsize（） - > int
返回用于表示每个像素的位数。此值可能无法准确填充每个像素使用的字节数。例如，15 位 Surface 仍需要完整的 2 个字节。

get_bytesize（）
获取每个 Surface 像素使用的字节数
get_bytesize（） - > int
返回每个像素使用的字节数。

get_flags（）
获取用于 Surface 的其他标志
get_flags（） - > int
返回一组当前 Surface 要素。每个功能都在标志位掩码中。典型的标志是 HWSURFACE，RLEACCEL， SRCALPHA，和 SRCCOLORKEY。

这是一个更完整的标志列表。完整列表可以在中找到 SDL_video.h

```
SWSURFACE       0x00000000      # Surface is in system memory
HWSURFACE       0x00000001      # Surface is in video memory
ASYNCBLIT       0x00000004      # Use asynchronous blits if possible
```
可以用来 pygame.display.set_mode()

```
ANYFORMAT       0x10000000      # Allow any video depth/pixel-format
HWPALETTE       0x20000000      # Surface has exclusive palette
DOUBLEBUF       0x40000000      # Set up double-buffered video mode
FULLSCREEN      0x80000000      # Surface is a full screen display
OPENGL        0x00000002      # Create an OpenGL rendering context
OPENGLBLIT      0x0000000A      # Create an OpenGL rendering context
                                #   and use it for blitting.  Obsolete.
RESIZABLE       0x00000010      # This video mode may be resized
NOFRAME       0x00000020        # No window caption or edge frame
```
内部使用（只读）

```
HWACCEL       0x00000100        # Blit uses hardware acceleration
SRCCOLORKEY     0x00001000      # Blit uses a source color key
RLEACCELOK      0x00002000      # Private flag
RLEACCEL        0x00004000      # Surface is RLE encoded
SRCALPHA        0x00010000      # Blit uses source alpha blending
```

PREALLOC 0x01000000 # Surface uses preallocated memory
get_pitch（）
获取每个 Surface 行使用的字节数
get_pitch（） - > int
返回分隔 Surface 中每行的字节数。视频内存中的表面并不总是线性打包。次表面也将具有比其实际宽度更大的间距。

正常的 pygame 使用不需要此值。

get_masks（）
位掩码需要在颜色和映射的整数之间进行转换
get_masks（） - >（R，G，B，A）
返回用于隔离映射整数中每种颜色的位掩码。

正常的 pygame 使用不需要此值。

set_masks（）
设置在颜色和映射整数之间转换所需的位掩码
set_masks（（r，g，b，a））- >无
正常的 pygame 使用不需要这个。pygame 1.8.1 中的新功能

get_shifts（）
在颜色和映射的整数之间转换所需的位移
get_shifts（） - >（R，G，B，A）
返回在每种颜色和映射的整数之间转换所需的像素移位。

正常的 pygame 使用不需要此值。

set_shifts（）
设置在颜色和映射整数之间转换所需的位移
set_shifts（（r，g，b，a））- >无
正常的 pygame 使用不需要这个。pygame 1.8.1 中的新功能

get_losses（）
用于在颜色和映射整数之间进行转换的有效位
get_losses（） - >（R，G，B，A）
返回从映射整数中的每种颜色剥离的最低有效位数。

正常的 pygame 使用不需要此值。

get_bounding_rect（）
找到包含数据的最小 rect
get_bounding_rect（min_alpha = 1） - > Rect

返回包含曲面中具有大于或等于最小alpha值的alpha值的所有像素的最小矩形区域。

此功能将根据需要临时锁定和解锁Surface。

新的pygame 1.8。

get_view（）
返回Surface像素的缓冲区视图。
get_view（<kind>='2'） - > BufferProxy
返回一个对象，该对象将表面的内部像素缓冲区导出为C级数组结构，Python级别数组接口或C级缓冲区接口。像素缓冲区是可写的。在Python 2.6及更高版本的CPython中支持新的缓冲区协议。Python 2.x也支持旧的缓冲区协议。旧缓冲区数据在一个段中用于类型'0'，多段用于其他缓冲区视图类型。

类型参数是长度为1的字符串'0'，'1'，'2'，'3'，'r'，'g'，'b'或'a'。这些字母不区分大小写; 'A'也会起作用。参数可以是Unicode或字节（char）字符串。默认值为“2”。

'0'返回连续的非结构化字节视图。没有给出表面形状信息。如果曲面的像素不连续，则会引发ValueError。

'1'返回连续像素的（表面宽度*表面高度）数组。如果表面像素不连续，则会引发ValueError。

'2'返回原始像素的（表面宽度，表面高度）数组。像素是surface-bytesize-d无符号整数。像素格式是表面特定的。除了其他pygame函数之外，其他任何东西都不太可能接受24位表面的3字节无符号整数。

'3'返回（表面宽度，表面高度，3）RGB颜色分量数组。红色，绿色和蓝色组件中的每一个都是无符号字节。仅支持24位和32位曲面。颜色分量必须在像素内RGB或BGR在像素内。

红色为'r'，绿色为'g'，蓝色为'b'，alpha为'a'，返回表面内单个颜色分量的（表面宽度，表面高度）视图：颜色平面。颜色分量是无符号字节。24位和32位表面都支持'r'，'g'和'b'。只SRCALPHA支持'a'的32位曲面 。

只有在访问公开的接口时才会锁定表面。对于新的缓冲区接口访问，一旦释放最后一个缓冲区视图，就会解锁表面。对于阵列接口和旧缓冲区接口访问，表面将保持锁定状态，直到释放BufferProxy对象。

pygame 1.9.2中的新功能。

get_buffer（）

获取 Surface 的像素的缓冲对象。
get_buffer（） - > BufferProxy
返回 Surface 的像素的缓冲区对象。缓冲区可用于直接像素访问和操作。表面像素数据表示为非结构化的内存块，其起始地址和长度以字节为单位。数据不必是连续的。任何间隙都包含在长度中，但否则会被忽略。

此方法隐式锁定 Surface。当返回的 BufferProxy 对象被垃圾回收时，将释放锁。

新的 pygame 1.8。

_pixels_address
像素缓冲地址
_pixels_address - > int
曲面的原始像素字节的起始地址。

pygame 1.9.2 中的新功能

pygame.event

pygame 模块，用于与事件和队列进行交互
pygame.event.pump - 内部处理 pygame 事件处理程序
pygame.event.get - 从队列中获取事件
pygame.event.poll - 从队列中获取单个事件
pygame.event.wait - 等待队列中的单个事件
pygame.event.peek - 测试事件类型是否在队列中等待
pygame.event.clear - 从队列中删除所有事件
pygame.event.event_name - 从中获取字符串名称和事件 ID
pygame.event.set_blocked - 控制队列中允许哪些事件
pygame.event.set_allowed - 控制队列中允许哪些事件
pygame.event.get_blocked - 测试是否从队列中阻止了某种类型的事件
pygame.event.set_grab - 控制与其他应用程序共享输入设备
pygame.event.get_grab - 测试程序是否共享输入设备
pygame.event.post - 在队列上放置一个新事件
pygame.event.Event - 创建一个新的事件对象
pygame.event.EventType - 用于表示 SDL 事件的 pygame 对象
Pygame 通过事件队列处理所有事件消息。此模块中的例程可帮助您管理该事件队列。输入队列严重依赖于 pygame 显示模块。如果显示尚未初始化且未设置视频模式，则事件队列将无法正常工作。

该队列是 pygame.event.EventTypepygame 对象的常规队列，用于表示 SDL 事件事件对象，有多种方法可以访问它包含的事件。从简单地检查事件的存在，直接从堆栈中抓取它们。

所有事件都有一个类型标识符。此事件类型是在的值之间 NOEVENT 和 NUMEVENTS。所有用户定义的事件都可以具有 USEREVENT 或更高的值 。建议确保您的活动 ID 遵循此系统。

要获得各种输入设备的状态，您可以放弃事件队列并直接使用适当的模块访问输入设备; 鼠标，键和操纵杆。如果您使用此方法，请记住 pygame 需要与系统窗口管理器和平台的其他部分进行某种形式的通信。为了使 pygame 与系统保持同步，您需要调用 pygame.event.pump()以保持最新状态。你通常会在每个游戏循环中调用此函数一次。

事件队列提供了一些简单的过滤。这可以通过阻止队列中的某些事件类型来略微提高性能，使用 pygame.event.set_allowed()和 pygame.event.set_blocked()来处理此过滤。所有事件都默认为允许。

应该从主线程调用事件子系统。如果要将事件从其他线程发布到队列中，请使用紧固包。

在设备初始化之前，操纵杆不会发送任何事件。

一个 EventType 事件对象包含的事件类型标识符和一组构件的数据。事件对象不包含方法函数，只包含成员数据。从 pygame 事件队列中检索 EventType 对象。您可以使用该 pygame.event.Event()功能创建自己的新事件。

SDL 事件队列对其可以保留的事件数有一个上限（标准 SDL 1.2 为 128）。当队列变满时，新事件将被悄然丢弃。为了防止丢失事件，特别是发出退出命令信号的输入事件，您的程序必须定期检查事件并处理它们。要加速队列处理，请使用 pygame.event.set_blocked()控制队列上允许哪些事件来限制哪些事件排队。

所有 EventType 实例都有一个事件类型标识符，可作为 EventType.type 属性访问 。您还可以通过该 EventType.__dict__属性完全访问事件对象的属性。所有其他成员查找将传递到对象的字典值。

在调试和试验时，您可以打印事件对象以快速显示其类型和成员。来自系统的事件将根据类型具有一组有保证的成员项。以下是使用每种事件类型定义的事件属性的列表。

QUIT	none
ACTIVEEVENT	gain, state
KEYDOWN	unicode, key, mod
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value

JOYBALLMOTION joy, ball, rel
JOYHATMOTION joy, hat, value
JOYBUTTONUP joy, button
JOYBUTTONDOWN joy, button
VIDEORESIZE size, w, h
VIDEOEXPOSE none
USEREVENT code
事件支持平等比较。如果它们是相同的类型且具有相同的属性值，则两个事件是相等的。不平等检查也有效。

1.9.2 版中的新功能：在 MacOSX 上，USEREVENT 可以拥有 code = pygame.USEREVENT_DROPFILE。这意味着用户正在尝试使用您的应用程序打开文件。文件名可以在 event.filename 中找到

pygame.event.pump（）
内部处理 pygame 事件处理程序
pump（） - >无
对于游戏的每个帧，您需要对事件队列进行某种调用。这可确保您的程序可以在内部与操作系统的其余部分进行交互。如果您未在游戏中使用其他事件功能，则应调用 pygame.event.pump()允许 pygame 处理内部操作。

如果程序通过另一个 pygame.eventpygame 模块一致地处理队列上的事件以与事件和队列函数进行交互，则不需要此函数。

必须在事件队列内部处理重要事项。主窗口可能需要重新绘制或响应系统。如果您未能长时间调用事件队列，系统可能会判断您的程序已被锁定。

评论 4
pygame.event.get（）
从队列中获取事件
get（） - > Eventlist
get（type） - > Eventlist
get（typelist） - > Eventlist
这将获取所有消息并将其从队列中删除。如果给出类型或类型序列，则只从队列中删除这些消息。

如果您只从队列中获取特定事件，请注意该队列最终可能会填满您不感兴 趣的事件。

评论 4
pygame.event.poll（）
从队列中获取单个事件
poll（） - > EventType 实例

从队列中返回单个事件。如果事件队列为空，pygame.NOEVENT 则将立即返回类型的事件。返回的事件将从队列中删除。

pygame.event.wait（）
等待队列中的单个事件
wait（） - > EventType 实例
从队列中返回单个事件。如果队列为空，则此函数将等待，直到创建一个。返回后，事件将从队列中删除。程序在等待时，它将处于空闲状态。这对于想要与其他应用程序共享系统的程序非常重要。

评论 2
pygame.event.peek（）
测试事件类型是否在队列中等待
偷看（类型） - >布尔
peek（typelist） - > bool
如果给定类型的任何事件在队列上等待，则返回 true。如果传递了一系列事件类型，则如果队列中有任何事件，则返回 True。

评论 1
pygame.event.clear（）
从队列中删除所有事件
clear（） - >无
清除（类型） - >无
清除（类型列表） - >无
从队列中删除特定类型的所有事件或事件。pygame.event.get()除了没有返回之外，这具有相同的效果。清除完整事件队列时，这可能会稍微提高效率。

pygame.event.event_name（）
从中获取字符串名称和事件 ID
event_name（type） - > string
Pygame 使用整数 id 来表示事件类型。如果要向用户报告这些类型，则应将它们转换为字符串。这将返回事件类型的简单名称。该字符串采用 WordCap 样式。

评论 1
pygame.event.set_blocked（）
控制队列中允许哪些事件
set_blocked（type） - >无
set_blocked（typelist） - >无
set_blocked（无） - >无
不允许在事件队列中显示给定的事件类型。默认情况下，所有事件都可以放在队列中。多次禁用事件类型是安全的。

如果将 None 作为参数传递，则会产生相反的效果，并且 ALL 允许将事件类型放在队列中。

评论 3
pygame.event.set_allowed（）
控制队列中允许哪些事件
set_allowed（type） - >无
set_allowed（typelist） - >无
set_allowed（无） - >无
允许给定的事件类型出现在事件队列中。默认情况下，所有事件都可以放在队列中。多次启用事件类型是安全的。

如果将 None 作为参数传递，NONE 则允许将事件类型放在队列中。

评论 4
pygame.event.get_blocked（）
测试是否从队列中阻止了某种类型的事件
get_blocked（type） - > bool
如果从队列中阻止给定的事件类型，则返回 true。

pygame.event.set_grab（）
控制与其他应用程序共享输入设备
set_grab（bool） - >无
当您的程序在窗口环境中运行时，它将与其他具有焦点的应用程序共享鼠标和键盘设备。如果您的程序将事件抓取设置为 True，它将锁定您的程序中的所有输入。

最好不要总是抓取输入，因为它阻止用户在他们的系统上做其他事情。

评论 1
pygame.event.get_grab（）
测试程序是否共享输入设备
get_grab（） - > bool
在为此应用程序获取输入事件时返回 true。使用 pygame.event.set_grab()来控制这种状态。

评论 2
pygame.event.post（）
在队列上放置一个新事件
发布（事件） - >无
这会在事件队列的末尾放置一个新事件。稍后将从其他队列函数中检索这些事件。

这通常用于 pygame.USEREVENT 在队列上放置事件。虽然可以放置任何类型的事件，但如果使用系统事件类型，则程序应确保使用适当的值创建标准属性。

如果 SDL 事件队列已满，则会引发 pygame.error 标准的 pygame 异常。

评论 4
pygame.event.Event（）
创建一个新的事件对象
事件（类型，字典） - > EventType 实例
事件（类型，**属性） - > EventType 实例
使用给定类型创建新事件。使用给定的属性和值创建事件。属性可以来自带字符串键的字典参数，也可以来自关键字参数。

评论 8
pygame.event.EventType
用于表示 SDL 事件的 pygame 对象
pygame.event.EventType.type - SDL 事件类型标识符。
pygame.event.EventType .__ dict__ - 事件对象属性字典
表示 SDL 事件的 Python 对象。使用 Event 函数调用创建用户事件实例。该事件类型类型不是直接调用。EventType 实例支持属性分配和删除。

type
SDL 事件类型标识符。
type - > int
只读。例如，预定义的事件标识符是 QUIT 和 MOUSEMOTION。对于用户创建的事件对象，这是为创建新事件对象而传递的类型参数。pygame.event.Event()

__dict__
事件对象属性字典
__dict__ - > dict
只读。事件的事件类型特定属性。例如，这将包含 KEYDOWN 事件的 unicode，key 和 mod 属性 。的字典属性是同义词，为了向后兼容。

pygame.Rect

用于存储直角坐标的 pygame 对象
Rect(left, top, width, height) -> Rect
Rect((left, top), (width, height)) -> Rect
Rect(object) -> Rect
pygame.Rect.copy - 复制矩形
pygame.Rect.move - 移动矩形
pygame.Rect.move_ip - 将矩形移动到位
pygame.Rect.inflate - 增大或缩小矩形大小
pygame.Rect.inflate_ip - 在适当的位置增大或缩小矩形大小
pygame.Rect.clamp - 将矩形移到另一个内部
pygame.Rect.clamp_ip - 将矩形移动到另一个内部
pygame.Rect.clip - 在另一个内部种植一个矩形
pygame.Rect.union - 将两个矩形连接成一个

pygame.Rect.union_ip - 将两个矩形连接成一个到位
pygame.Rect.unionall - 许多矩形的联合
pygame.Rect.unionall_ip - 许多矩形的结合，到位
pygame.Rect.fit - 调整大小并移动纵横比矩形
pygame.Rect.normalize - 正确的负尺寸
pygame.Rect.contains - 测试一个矩形是否在另一个矩形内
pygame.Rect.collidepoint - 测试一个点是否在矩形内
pygame.Rect.colliderect - 测试两个矩形是否重叠
pygame.Rect.collidelist - 测试列表中的一个矩形是否相交
pygame.Rect.collidelistall - 测试列表中的所有矩形是否相交
pygame.Rect.collidedict - 测试字典中的一个矩形是否相交
pygame.Rect.collidedictall - 测试字典中的所有矩形是否相交
Pygame 使用 Rect 对象来存储和操作矩形区域。可以从 left,top,width 和 height 值的组合创建 Rect。也可以从已经是 Rect 或具有名为“rect”的属性的 python 对象创建 Rect。

任何需要 Rect 参数的 pygame 函数也接受任何这些值来构造 Rect。这使得动态创建 Rects 更容易作为函数的参数。

更改 Rect 的位置或大小的 Rect 函数返回带有受影响的更改的 Rect 的新副本。原始的 Rect 未被修改。某些方法有一个备用的“就地”版本，它返回 None 但会影响原始的 Rect。这些“就地”方法用“ip”后缀表示。

Rect 对象有几个虚拟属性，可用于移动和对齐 Rect：

x,y
top, left, bottom, right
topleft, bottomleft, topright, bottomright
midtop, midleft, midbottom, midright
center, centerx, centery
size, width, height
w,h
所有这些属性都可以分配给：

rect1.right = 10
rect2.center = (20,30)
分配大小，宽度或高度会改变矩形的尺寸; 所有其他分配移动矩形而不调整其大小。请注意，某些属性是整数，其他属性是整数对。

如果 Rect 具有非零宽度或高度，则对于非零测试，它将返回 True。某些方法返回一个 0 大小的 Rect 来表示无效的矩形。

Rect 对象的坐标都是整数。可以将大小值编程为具有负值，但对于大多数操作，这些被认为是非法的 Rect。

其他矩形之间有几个碰撞测试。可以搜索大多数python容器与单个Rect的冲突。

Rect覆盖的区域不包括像素的最右边和最底边。如果一个Rect的底部边框是另一个Rect的顶部边框（即rect1.bottom = rect2.top），则两者完全在屏幕上相遇但不重叠，并rect1.colliderect(rect2)返回false。

Rect类可以是子类。诸如copy()和move() 之类的方法将识别这个并返回子类的实例。但是，__init__()不调用子类的方法，并__new__()假定它不带参数。因此，如果需要复制任何额外属性，则应覆盖这些方法。pygame 1.9.2中的新功能。

copy（）
复制矩形
copy（） - > Rect
返回与原始位置和大小相同的新矩形。

pygame 1.9中的新功能

move（）
移动矩形
move（x，y） - > Rect
返回由给定偏移量移动的新矩形。x和y参数可以是任何整数值，正数或负数。

move_ip（）
将矩形移动到位
move_ip（x，y） - >无
与Rect.move()方法相同，但在适当的位置操作。

inflate（）
增大或缩小矩形大小
膨胀（x，y） - > Rect
返回一个新的矩形，其大小由给定的偏移量改变。矩形保持以其当前中心为中心。负值会缩小矩形。注意，使用整数，如果给定的偏移量太小（<2>-2），则中心将关闭。

inflate_ip（）
在适当的位置增大或缩小矩形大小
inflate_ip（x，y） - >无
与Rect.inflate()方法相同，但在适当的位置操作。

clamp（）
将矩形移到另一个内部
clamp（Rect） - > Rect

返回一个新的矩形，该矩形完全移动到参数 Rect 中。如果矩形太大而无法放入内部，则它在参数 Rect 内居中，但其大小不会更改。

clamp_ip（）
将矩形移动到另一个内部
clamp_ip（Rect） - >无
与 Rect.clamp()方法相同，但在适当的位置操作。

clip（）
在另一个内部种植一个矩形
clip（Rect） - > Rect
返回一个新的矩形，该矩形被裁剪为完全位于参数 Rect 内。如果两个矩形不重叠，则返回一个 0 大小的 Rect。

union（）
将两个矩形连接成一个
union（Rect） - > Rect
返回一个完全覆盖两个提供的矩形区域的新矩形。新 Rect 中可能存在未被原件覆盖的区域。

union_ip（）
将两个矩形连接成一个到位
union_ip（Rect） - >无
与 Rect.union()方法相同，但在适当的位置操作。

unionall（）
许多矩形的联合
unionall（Rect_sequence） - > Rect
返回一个矩形与一系列矩形序列的并集。

unionall_ip（）
许多矩形的结合，到位
unionall_ip（Rect_sequence） - >无
与 Rect.unionall()方法相同，但操作到位。

fit（）
调整大小并移动纵横比矩形
fit（Rect） - > Rect
返回一个移动并调整大小以适合另一个矩形的新矩形。保留原始 Rect 的纵横比，因此新的矩形可以在宽度或高度上小于目标。

normalize（）
正确的负尺寸
normalize（） - >无

如果矩形的负大小，这将翻转矩形的宽度或高度。矩形将保持在同一位置，只交换侧面。

contains（）
测试一个矩形是否在另一个矩形内
contains（Rect） - > bool
当参数完全在 Rect 内部时返回 true。

collidepoint（）
测试一个点是否在矩形内
collidepoint（x，y） - > bool
collidepoint（（x，y）） - > bool
如果给定的点在矩形内，则返回 true。沿右边或底边的点不被视为在矩形内。

colliderect（）
测试两个矩形是否重叠
colliderect（Rect） - > bool
如果任一矩形的任何部分重叠（顶部+底部或左侧+右侧边缘除外），则返回 true。

collidelist（）
测试列表中的一个矩形是否相交
collidelist（list） - > index
测试矩形是否与矩形序列中的任何一个发生碰撞。返回找到的第一个碰撞的索引。如果未发现冲突，则返回-1 的索引。

collidelistall（）
测试列表中的所有矩形是否相交
collidelistall（list） - > indices
返回包含与 Rect 冲突的矩形的所有索引的列表。如果未找到相交的矩形，则返回空列表。

collidedict（）
测试字典中的一个矩形是否相交
collidedict（dict） - >（键，值）
返回与 Rect 冲突的第一个字典值的键和值。如果未找到任何冲突，则返回 None。

Rect 对象不可清除，不能用作字典中的键，只能用作值。

collidedictall（）
测试字典中的所有矩形是否相交
collidedictall（dict） - > [（键，值），...]
返回与 Rect 相交的所有键和值对的列表。如果未找到冲突，则返回空列表。

Rect 对象不可清除，不能用作字典中的键，只能用作值。