[Pygame] display | Pygame中文文档 [复制链接]

display | Pygame中文文档

pygame.display

Pygame 中用于控制窗口和屏幕显示的模块。

小甲鱼注：为了适应语境，display 在该文档中有时翻译为“显示”，有时翻译为“显示界面”。

函数

- pygame.display.init() — 初始化 display 模块
- pygame.display.quit() — 结束 display 模块
- 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
- pygame.display.get_surface() — 获取当前显示的 Surface 对象
- pygame.display.flip() — 更新整个待显示的 Surface 对象到屏幕上
- pygame.display.update() — 更新部分软件界面显示
- pygame.display.get_driver() — 获取 Pygame 显示后端的名字
- pygame.display.Info() — 创建有关显示界面的信息对象
- pygame.display.get_wm_info() — 获取关于当前窗口系统的信息
- pygame.display.list_modes() — 获取全屏模式下可使用的分辨率
- pygame.display.mode_ok() — 为显示模式选择最合适的颜色深度
- pygame.display.gl_get_attribute() — 获取当前显示界面 OpenGL 的属性值
- pygame.display.gl_set_attribute() — 设置当前显示模式的 OpenGL 属性值
- pygame.display.get_active() — 当前显示界面显示在屏幕上时返回 True
- pygame.display.iconify() — 最小化显示的 Surface 对象
- pygame.display.toggle_fullscreen() — 切换全屏模式和窗口模式
- pygame.display.set_gamma() — 修改硬件显示的 gama 坡道
- pygame.display.set_gamma_ramp() — 自定义修改硬件显示的 gama 坡道
- pygame.display.set_icon() — 修改显示窗口的图标
- pygame.display.set_caption() — Set the current window caption
- pygame.display.get_caption() — Get the current window caption
- pygame.display.set_palette() — Set the display color palette for indexed displays

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，你必须选择一个翻转函数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。

函数详解

pygame.display.init()

初始化 display 模块。

init() -> None

初始化 Pygame 的 display 模块。在初始化之前，display 模块无法做任何事情。但当你调用更高级别的 pygame.init()，变会自动调用 pygame.display.init() 进行初始化。

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。

quit() -> None

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。

get_init() -> bool

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。

set_mode(resolution=(0,0), flags=0, depth=0) -> Surface

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：

选项	含义
pygame.FULLSCREEN	创建一个全屏显示
pygame.DOUBLEBUF	1. 双缓冲模式 2. 推荐和 HWSURFACE 或 OPENGL 一起使用
pygame.HWSURFACE	硬件加速，只有在 FULLSCREEN 下可以使用
pygame.OPENGL	创建一个 OPENGL 渲染的显示
pygame.RESIZABLE	创建一个可调整尺寸的窗口
pygame.NOFRAME	创建一个没有边框和控制按钮的窗口

举个栗子：

```
01.  # 在屏幕中创建一个 700 * 400 的窗口
02.  screen_width=700
03.  screen_height=400
04.  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。

get_surface() -> Surface

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。

flip() -> None

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。

get_driver() -> name

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。

Info() -> VideoInfo

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。

get_wm_info() -> dict

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 1.7.1 新增加的。

pygame.display.list_modes()

获取全屏模式下可使用的分辨率。

list_modes(depth=0, flags=pygame.FULLSCREEN) -> list

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。

mode_ok(size, flags=0, depth=0) -> depth

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。

gl_get_attribute(flag) -> value

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。

gl_set_attribute(flag, value) -> None

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。

get_active() -> bool

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。

iconify() -> bool

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。

set_gamma(red, green=None, blue=None) -> bool

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量。

pygame.display.set_gamma_ramp()

自定义修改硬件显示的 gama 坡道

set_gamma_ramp(red, green, blue) -> bool

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。

set_icon(Surface) -> None

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。

set_caption(title, icontitle=None) -> None

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。

get_caption() -> (title, icontitle)

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。

set_palette(palette=None) -> None

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。