Pygame中文文档

1. 安装
2. API
   1. 地址

http://www.pygame.org/docs/

* 1. pygame.mouse
     1. get\_pressed()

获得鼠标按钮的状态

get\_press()->(button1, button2, button3)

返回一个序列代表鼠标按钮的状态（布尔值）

True：该鼠标正在被调用

注意：要获得所有的鼠标事件，最好使用pygame.event.wait()或pygame.event.get()，并检查所有这些事件是MOUSEBUTTONDOWN,MOUSEBUTTONUP还是MOUSEMOTION

注意：记得在这个函数前先调用pygame.event.get()，否则它不会工作。

* + 1. get\_pos()

获得鼠标光标位置

get\_pos() -> (x, y)

返回鼠标相对于显示的左上角的位置，光标位置可以位于显示窗口的外部，但始终被限制在屏幕上

* + 1. get\_rel()

获得鼠标的移动量

get\_rel() -> (x, y)

返回前一次调用该函数至今的(x,y)的移动距离。鼠标的相对运动被限制在屏幕边缘。

* + 1. 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位置。如果鼠标光标是可见的，它会跳转到新的坐标系。移动鼠标将会生成一个新的pygame.MOUSEMOTION事件

* + 1. set\_visible()

隐藏或显示鼠标光标

set\_visible(bool) -> bool

返回的bool值若为True：显示光标

* + 1. get\_focused()

检查是否接受鼠标输入

get\_focused() -> bool

当pygame接受鼠标事件的时候，返回True（或’active’,’focus’）

在窗口工作时，这种方法总是最有效的。相比之下，在全屏模式下，这种方法总是返回True。

注：在微软窗口下，有鼠标焦点的窗口就也有键盘焦点；但在X-Windows，一个窗口接收鼠标事件而另一个接收键盘事件。

pygame.mouse.get\_focused()表面pygame窗口接收鼠标事件。

* + 1. set\_cursor()

设置系统鼠标光标的图像

set\_cursor(size, hotspot, xormasks, andmasks) -> None

当鼠标光标是可见的，他将显示给定的黑白位图。

* size是一个序列（光标的宽度，高度）
* hotspot是一个包含光标热点位置的序列
* xormasks是一个包含光标异或数据掩码的序列
* andmasks是一个包含光标位掩码数据的序列

宽度必须是8的倍数，掩码阵列的大小必须按照给定的宽高，否则将引发异常。

* + 1. get\_cursor()

获取系统鼠标光标的图像

get\_cursor() -> (size, hotspot, xormasks, andmasks)

参数含义同pygame.mouse.set\_cursor()

* 1. pygame.event
     1. pump()

内部处理pygame事件的操作

pump() -> None

对于你游戏的每个框架，你需要对事件队列进行某种形式的调用。这确保你的程序可以与其他操作系统的交互。如果你不在游戏中使用其他事件函数，你需要调用pygame.event.pump()来让pygame处理内部操作。

如果你的程序是通过其他pygame.event模块持续处理队列中的事件来与事件和队列里的函数进行交互，那么这个函数就不是必须的。

在事件队列中有必须要分配的重要事件。主窗口也许要重新绘制或响应系统。如果你调用事件队列太久以至于失败的话，系统可能会决定你的程序已经被锁定。

* + 1. get()

从队列中获取事件

get() -> Eventlist

get(type) -> Eventlist

get(typelist) -> Eventlist

这将获得所有的信息，并从队列中删除它们。如果给定类型或类型的序列，那么将只会删除这些信息。

如果你只是从队列中提取特定的事件，队列中最后将会被你不感兴趣的事件填满。

* + 1. poll()

从队列中获取一个单独的事件

poll() -> EventType instance（事件类型实例）

返回队列中单个事件。如果队列是空的，将立即返回pygame.NOEVENT类型的事件。返回的事件将从队列中删除。

* + 1. wait()

从队列中等待一个单独的事件

wait() -> EventType instance

返回队列中的单个事件。如果队列是空的，这个函数将等待直到该事件被创建。一旦返回该事件，该事件将从队列中删除。当程序在等待时，它将在空闲状态休眠。这对程序是非常重要的，如果想要和其他应用程序共享系统的话。

* + 1. peek()

测试事件是否在队列中等待

peek(type) -> bool

peek(typelist) -> bool

如果给定的事件在队列中等待，那么返回True。如果传递的是一个事件序列，那么只要其中有一个在队列中，就会返回True

* + 1. clear()

从队列中删除所有事件

clear() -> None

clear(type) -> None

clear(typelist) -> None

除了没有返回值，这与pygame.event.get()具有相同效果。当删除一个满了的事件队列时，这个方法更加效率。

* + 1. event\_name()

从事件id获取名字（字符串）

event\_name(type) -> string

pygame使用int型的id来代表事件类型。如果你想将这些类型报告给用户，它们应该被转为字符串。这将返回一个事件类型的简单名称。字符串是全大写形式。

* + 1. set\_blocked()

控制哪些事件被允许出现在队列中

set\_blocked(type) -> None

set\_blocked(typelist) -> None

set\_blocked(None) -> None

给定的事件类型是不允许出现在事件序列中的。默认情况下，所有事件都可以被放置在队列中。多次禁用事件类型是安全的。

如果没有传递任何参数，这将有相反的效果：所有的事件类型都被允许放置在队列中。

* + 1. set\_allowed()

控制哪些事件被允许出现在队列中

set\_allowed(type) -> None

set\_allowed(typelist) -> None

set\_allowed(None) -> None

给定的事件类型将被允许在队列中出现。默认情况下，所有事件都可以被放置在队列中。多次允许同样的事件类型是安全的。

如果没有传递任何参数，则可以将任何事件类型放入队列中。

* + 1. get\_blocked()

测试某个类型的事件是否被禁止出现在队列

get\_blocked(type) -> bool

如果被禁止，返回True

* + 1. set\_grab()

控制与其他应用程序的输入设备的共享

set\_grab(bool) -> None

当你的程序运行在一个窗口环境时，它将与其他被标记应用程序共享鼠标键盘。如果你的程序设置这个函数的bool=True，那么将在你的程序里锁定所有的输入。

最好不要总是抢霸占输入，因为这会妨碍用户在他们的系统上做其他事情。

* + 1. get\_grab()

测试程序是否共享了输入设备

get\_grab() -> bool

如果输入被这个应用程序霸占了，那么返回True

* + 1. post()

将一个新的事件放到队列的末尾

post(Event) -> None

这通常被用来防止pygame.USEREVENT事件。虽然任何事件都可以被放置，但如果使用系统事件类型，需要确保你的程序使用适当值创建了标准属性。

* + 1. Event()

创建一个新的事件对象

Event(type, dict) -> EventType instance

Event(type, \*\*attributes) -> EventType instance

使用给定的type来创建一个新的事件。创建时附带了给定的属性和值。这个属性可以是字典参数的字符串key，也可以是关键字参数

* + 1. class pygame.event.EventType()

是代表SDL事件的pygame对象

pygame.event.EventType.type → SDL事件类型的id

pygame.event.EventType.\_\_dict\_\_ → 事件对象的属性字典

一个表示SDL事件的python对象。用户事件实例是在事件函数调用的时候创建的。这个EventType不能直接调用。EventType的实例支持属性赋值和删除。

**type：**

SDL事件类型的id

type→int

只读。预定义的事件id是QUIT和MOUSEMOTION。举个例子，如果用户创建了事件对象，type参数将会被传递给pygame.event.Event()来创建一个新的事件对象。

**\_\_dict\_\_：**

事件对象的属性字典

\_\_dict\_\_ -> dict

只读。具有特定属性的事件类型。

* 1. pygame.font
     1. init()

初始化字体模块

init() -> None

pygame.init()会自动调用这个函数

在任何函数被调用前，该模块要先被初始化

多次调用该函数，是安全的

* + 1. quit()

取消初始化字体模块

quit() -> None

手动取消初始化SDL\_ttf的字体系统

这回被pygame.quit()自动调用

即使字体模块没有被初始化，就调用该函数，也是安全的

* + 1. get\_init()

如果字体模块被初始化，那么返回True

get\_init() -> bool

测试字体模块是否已经初始化

* + 1. 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。不返回完整路径，通常可以在字体模块的目录下找到文件。单页可以在捆绑在单独的文件中。

* + 1. get\_fonts()

获得所有可用的字体

get\_fonts() -> list of strings

返回系统上可用的所有字体的列表。字体名称会被设置为小写，所有空格和标点符号会被删除。大多数系统都使用，但有些时候如果找不到也会返回空列表。

* + 1. match\_font()

在系统中找到一个特定的字体

match\_font(name, bold=False, italic=False) -> path

返回系统上的字体文件的完整路径，如果要找加粗或斜体的字体，就设置相应参数为True。这将尝试找到正确的字体家族。

参数name实际上可以是一个用逗号分隔的字体名称列表。如果给出的字体名称没有被找到，会返回None

例如：

print pygame.font.match\_font('bitstreamverasans')

# 输出: /usr/share/fonts/truetype/ttf-bitstream-vera/Vera.ttf

# (只会在Vera字体存在在系统时，是这样的输出结果)

* + 1. SysFont()

从系统字体中创建一个字体对象

SysFont(name, size, bold=False, italic=False) -> Font

返回从系统字体加载的一个新的字体对象。字体将匹配设置的加粗和斜体参数。如果没有找到一个合适的系统字体，将会回退加载默认的pygame字体。

参数name可以是一个用逗号分隔的字体名称列表。

* + 1. class pygame.font.Font()