Pygame 1.8

Referencia para programadores en español.

Última actualización: 05/02/09

Resumen

Este documento contiene una traducción no oficial de la documentación de referencia de la biblioteca pygame. Un biblioteca que permite desarrollar aplicaciones multimedia, como juegos, usando el lenguaje de programación python.

La traducción ha sido desarrollada por integrantes del grupo Loserjsuegos. Puede visitar el sitio www.losersjuegos.com.ar para obtener una versión mas reciente de este documento.

Nota

Tenga en cuenta que este documento es una duplicación inexacta de la documentación en formato web. Para obtener una versión fiable del texto visite la versión web: www.losersjuegos.com.ar/traducciones/pygame

Contenido

pygame

El paquete pygame de nivel superior.

- init
- quit
- error
- get_error
- get_sdl_version
- get_sdl_byteorder
- register_quit

El paquete pygame representa el paquete de nivel superior para los demás. Pygame en sí está dividida en varios submódulos, aunque esto no afecta a los programas que utilizan la biblioteca.

Por convención, la mayoría de las variables de nivel superior en pygame se han colocado dentro de un módulo llamado locals. De forma que se pueda combinar `from pygame.locals import *` además de `import pygame`.

Cuando incorpora pygame mediante `import pygame`, todos los módulos de pygame disponibles se cargan automáticamente. Tenga presente que algunos submódulos se consideran *opcionales*, por lo tanto podrían no estar disponibles. En ese caso, pygame proveerá un objeto marcador de posición en lugar del módulo, que se puede usar para verificar la disponibilidad del módulo [1)].

Editar

init

Inicializa todos los módulos de pygame incorporados.

```
pygame.init(): return (numpass, numfail)
```

Inicializa todos los módulos de pygame incorporados. No se notificarán excepciones si un módulo falla, aunque el número total de módulos inicializados correctamente o fallidos se retornarán como una tupla cuando llame a esta función. De todas formas, siempre puede inicializar módulos de forma individual, solo que `pygame.init` es una forma útil de inicializar todos los módulos a la vez. Las funciones `init` para módulos individuales sí notificarán excepciones cuando fallen.

Tal vez usted quiera inicializar los diferentes módulos de manera separada para incrementar la velocidad de su programa, o no solicitar cosas que su programa no utilizará.

Es seguro llamar a esta función más de una vez, llamadas sucesivas no tendrán efecto. Esto es así incluso cuando halla llamado a quit para descargar todos los módulos.

- buscar código donde se use esta función.

Editar

quit

Deshabilita todos los módulos de pygame.

```
pygame.quit(): return None
```

Deshabilita todos los módulos de pygame que anteriormente se han inicializado. Cuando el intérprete de python se cierra, esté método se llamará automáticamente, por lo tanto sus programas podrían no necesitar llamarlo, excepto si usted quiere liberar sus recursos de pygame y continuar.

Es seguro llamar a esta función mas de una vez, sucesivas llamadas no tendrán efecto.

Note que `pygame.quit` no terminará su programa. Considere cerrar o terminar su programa de la misma forma que lo hace en un programa de python habitual.

- buscar código donde se use esta función.

Editar

error

Excepción de pygame por defecto.

```
raise pygame.error, message
```

Esta excepción se notifica siempre que una operación de pygame o SDL falla. Puede capturar problemas anticipados y tratar con el error. La excepción se notifica siempre con un mensaje descriptivo acerca del problema.

Como deriva de la excepción `RuntimeError`, también se puede usar para capturar estos errores.

- buscar código donde se use esta función.

Editar

get_error

Obtiene el mensaje de error actual.

```
pygame.get_error(): return errorstr
```

SDL mantiene un mensaje de error interno. Este mensaje generalmente se le informará cuando se notifique una excepción error. Por lo tanto seguramente no necesitará llamar a esta función.

- buscar código donde se use esta función.

Editar

get_sdl_version

Obtiene el número de versión de SDL.

```
pygame.get_sdl_version(): return major, minor, patch
```

Retorna los tres números de versión de la biblioteca SDL. Esta versión se construye al momento de la compilación de pygame (instalación) no en tiempo de ejecución. Puede usarse para detectar que características pueden o no estar disponibles a través de pygame.

`get_sdl_version` es una nueva característica en pygame 1.7.0.

- buscar código donde se use esta función.

Editar

get_sdl_byteorder

Obtiene el orden de bytes de SDL.

```
pygame.get_sdl_byteorder(): return int
```

Obtiene el orden de bytes de la biblioteca SDL. Retorna `LIL_ENDIAN` para el orden *little endian* o `BIG_ENDIAN` para el orden *big endian*.

`get_sdl_byteorder` es una característica nueva en pygame 1.8.

- buscar código donde se use esta función.

Editar

register_quit

Registra una función para se llame cuando pygame finaliza.

```
register_quit(callable): return None
```

Cuando se llama a la función `pygame.quit`, todas las funciones de salida se llamarán. Los módulos de pygame hacen esto automáticamente cuando son inicializadas. Esta función no es necesaria para usuarios de pygame en general.

- buscar código donde se use esta función.

cdrom

Módulo de pygame para controlar CDROM de audio.

- init
- quit
- get_init
- get_count

Otras páginas:

- CD

El módulo cdrom gestiona dispositivos de CD y DVD en una computadora. Puede incluso controlar la reproducción de CDs de audio. Antes de hacer algo tiene que iniciar el módulo. Cada objeto CD que usted genera representa un dispositivo de CD que además necesita inicializarse individualmente antes de hacer la mayoría de las cosas.

Editar

init

Inicializa el módulo de CDROM.

```
pygame.cdrom.init(): return None
```

Inicializa el módulo cd CDROM. Esta función explorará el sistema buscando todos los dispositivos de CD. El módulo se debe inicializar antes de utilizar cualquier otra función. Esta inicialización se realiza automáticamente cuando llama a la función init.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el módulo de CDROM.

```
pygame.cdrom.quit(): return None
```

Deshabilita el módulo de CDROM. Después de llamar a esta función cualquier objecto CD existente dejará de funcionar.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

get_init

Informa `true` si el módulo cdrom está inicializado.

```
pygame.cdrom.get_init(): return bool
```

Verifica si el módulo cdrom está inicializado o no. Esto se diferente a CD.init() ya que cada dispositivo también se debe inicializar de forma individual.

- buscar código donde se use esta función.

Editar

get_count

Informa el número de dispositivos de CD en el sistema.

```
pygame.cdrom.get_count(): return count
```

Devuelve el número de unidades de CD en el sistema. Cuando usted genera objetos CD necesita enviar un número identificador que sea menor que este contador. El contador será 0 si no hay dispositivos de CD en el sistema.

- buscar código donde se use esta función.

CD

Clase para gestionar un dispositivo de cdrom.

Editar

CD

`pygame.cdrom.CD(id): return CD`

Puede crear un objeto CD por cada dispositivo de cdrom en el sistema. Use `pygame.cdrom.get_count()` para determinar cuantos dispositivos existen actualmente. El argumento `id` es un número entero que representa al dispositivo, comenzando en 0.

El objeto CD no está inicializado, solo puede llamar a `CD.get_id()` y `CD.get_name()` en un dispositivo no inicializado.

Es seguro crear múltiples objetos CD para el mismo dispositivo, actuarán cooperativamente de forma normal.

Editar

init

Inicializa un dispositivo de cdrom para utilizar.

`CD.init(): return None`

Inicializa un dispositivo de cdrom para utilizar. El dispositivo se debe inicializar para que funcionen la mayoría de los métodos. Incluso si el resto de pygame ha sido inicializado.

Puede haber una pausa mientras el dispositivo se inicializa. Evite usar `CD.init()` si el programa

parece no detenerse por uno o dos segundos.

Editar

quit

Deshabilita un dispositivo de cdrom.

```
CD.quit(): return None
```

Deshabilita un dispositivo de cdrom. Llame a este método cuando su programa no valla a acceder al dispositivo por un tiempo.

Editar

get_init

Devuelve true si el dispositivo de cd está inicializado.

```
CD.get_init(): return bool
```

Comprueba si esta unidad de cdrom está inicializada. Es difierente de pygame.cdrom.init, ya que cada unidad debe ser inicializada individualmente.

Editar

play

Comienza a reproducir audio.

```
CD.play(track, start=None, end=None): return None
```

Reproduce audio de un cdrom en la unidad. Además del argumento del número de pista, también puedes pasarle el tiempo de comienzo y final de la reproducción. El tiempo de inicio y fin están en segundos, y puedes limitar la sección de una pista de audio reproducida.

Si le pasa el tiempo de inicio pero no el final, se reproducirá audio hasta el final de la pista. Si le pasas un tiempo de inicio y 'None' para el final, el audio se reproducirá hasta el final del disco.

Ver `CD.get_numtracks` y `CD.get_track_audio` para buscar pistas a reproducir.

Nota: track 0 es track 1 en el CD. Los números de pista empiezan en cero.

Editar

stop

Detiene la reproducción de audio.

```
CD.stop(): return None
```

Detiene la reproducción de audio desde el cdrom. Esto también hará perder la posición actual de reproducción. Este método no hace nada si la unidad no está reproduciendo en ese momento.

Editar

pause

Detiene temporalmente la reproducción de audio.

```
CD.pause(): return None
```

Detiene temporalmente la reproducción de audio en el CD. La reproducción se puede resumir en la misma posición con el método `CD.resume()`. Este método no hace nada si la unidad no está reproduciendo en ese momento.

Nota: track 0 es la primer pista del CD. Los números de pista comienzan en 0.

Editar

resume

Reanuda la reproducción de audio.

```
CD.resume(): return None
```

Reanuda un CD en pausa. Este método no hace nada si el CD no está en pausa o se encuentra reproduciendo.

Editar

eject

Expulsa o abre la unidad de cdrom.

```
CD.eject(): return None
```

Abrirá la unidad de cdrom y expulsará la bandeja. Si el dispositivo está reproduciendo o en pausa se interrumpirá.

Editar

get_id

Obtiene el índice de la unidad de cdrom.

```
CD.get_id(): return id
```

Retorna el identificador entero `id` que se utilizó para crear la instancia de CD. Este método puede operar en un CD no inicializado.

Editar

get_name

Obtiene el nombre de sistema de la unidad de cdrom.

`CD.get_name(): return name`

Retorna el nombre de un dispositivo. Este es el nombre de sistema usado para representar la unidad. Puede ser el nombre del dispositivo o la letra de la unidad. Este método puede funcionar en un dispositivo de CD sin inicializar.

Editar

get_busy

Retorna `True` si el dispositivo está reproduciendo audio.

`CD.get_busy(): return bool`

Retorna `True` si el dispositivo de cd está ocupado reproduciendo audio.

Editar

get_paused

Devuelve `True` si el dispositivo está en pausa.

`CD.get_paused(): return bool`

Devuelve `True` si el dispositivo está en pausa.

Editar

get_current

Obtiene la posición de la reproducción actual.

`CD.get_current(): return track, seconds`

Retorna la pista actual y el tiempo de reproducción de esa pista. Este método funciona cuando el dispositivo está en pausa o reproduciendo.

Nota: track 0 es la primer pista del CD. Los números de pista comienzan en cero.

Editar

get_empty

`False` si un cdrom está dentro de la unidad.

`CD.get_empty(): return bool`

Retorna `False` si actualmente hay un cdrom en la unida. Si la unidad está vacía retornará `True`.

Editar

get_numtracks

Obtiene el número de pistas del cdrom.

```
CD.get_numtracks(): return count
```

Retorna el número de pistas del cdrom en la unidad. Retornará 0 si la unidad está vacía o no hay pistas.

Editar

get_track_audio

`True` si la pista del cdrom tienen datos de audio.

```
CD.get_track_audio(track): return bool
```

Determina si una pista del cdrom contiene datos de audio. También puede llamar a `CD.num_tracks()` y `CD.get_all()` para obtener mas información acerca del cdrom.

Nota: track 0 es la primer pista del CD. Los números de pista comienzan en cero.

Editar

get_all

Obtiene toda la información de pistas.

```
CD.get_all(): return [(audio, start, end, lenth), ...]
```

Retorna una lista con información de cada pista del cdrom. La información consiste en una tupla con cuatro valores. El valor `audio` será `True` si la pista contiene datos de audio. Los valores `start`, `end` y `length` son números reales en segundos. Tanto `start` como `end` representan tiempos absolutos del disco entero.

Editar

get_track_start

Obtiene el tiempo de inicio de una pista de cdrom.

```
CD.get_track_start(track): return seconds
```

Retorna el tiempo absoluto en segundos donde está el inicio de la pista de cdrom.

Nota: track 0 es la primer pista del CD. Los números de pista comienzan en cero.

Editar

get_track_length

Obtiene la duración de una pista.

```
CD.get_track_length(track): return seconds
```

Retorna un valor en número real que representa la duración en segundos de una pista del cdrom.
Nota: track 0 es la primer pista del CD. Los números de pista comienzan en cero.

Color

Objeto de pygame para representaciones de color.

Editar

Color

```
pygame.Color(name): Return Color
pygame.Color(r, g, b, a): Return Color
pygame.Color(rgbvalue): Return Color
```

La clase Color representa valores de color RGBA usando un rango de valores de 0 a 255. Permite operaciones aritméticas básica para crear colores nuevos, soporta conversiones a otros formato de color como HSV o HLS y le permite ajuntar canales de color de manera individual.

Esta clase es nueva en pygame 1.8

Editar

r

Obtiene o define el componente rojo del Color.

```
Color.r: Return int
```

El valor rojo de un Color.

Editar

g

Obtiene o define el componente green del Color.

```
Color.g: Return int
```

El valor verde del Color.

Editar

b

Obtiene o define el componente azul del Color.

```
Color.b: Return int
```

El valor azul del Color.

Editar

a

Obtiene o define el componente alpha (transparencia) del Color.

```
Color.a: Return int
```

El valor alpha (transparencia) del Color.

Editar

cmy

Obtiene o define la representación CMY del Color.

```
Color.cmy: Return tuple
```

Es la representación CMY del Color. Los componentes CMY están en el rango C = [0, 1], M = [0, 1], Y = [0, 1]. Note que no se devolverá la representación absolutamente exacta para los valores RGB en todos los casos. Dado que los errores de aproximación de la representación RGB (de 0 a 255) y la representación CMY (de 0 a 1) pueden causar que los valores CMY difieran ligeramente de los que usted espera.

Editar

hsva

Obtiene o define la representación HSVA del Color.

```
Color.hsva: Return tuple
```

Es la representación HSVA del Color. Los componentes HSVA están en el rango H = [0, 320], S = [0, 100], V = [0, 100], A = [0, 100]. Note que no se devolverá la representación absolutamente exacta para los valores RGB en todos los casos. Dado que los errores de aproximación de la representación RGB (de 0 a 255) y la representación HSVA (de 0 a 100 y 0 a 360) pueden causar que los valores HSV difieran ligeramente de los que usted espera.

Editar

hsla

Obtiene o define la representación HSLA del Color.

```
Color.hsla: Return tuple
```

Es la representación HSLA del Color. Los componentes HSLA están en los rangos H = [0, 360], S = [0, 100], V = [0, 100], A = [0, 100]. Note que no se devolverá la representación absolutamente exacta para los valores RGB en todos los casos. Dado que los errores de aproximación de la representación RGB (de 0 a 255) y la representación HSLA (de 0 a 100 y 0 a 360) pueden causar que los valores HSL difieran ligeramente de los que usted espera.

Editar

i1i2i3

Obtiene o define la representación `I1I2I3` del Color.

```
Color.i1i2i3: Return tuple
```

Es la representación I1I2I3 del color. Los componentes I1I2I3 están en el rango I1 = [0, 1], I2 = [-0.5, 0.5], I3 = [-0.5, 0.5]. Note que no se devolverá la representación absolutamente exacta para los valores RGB en todos los casos. Dado que los errores de aproximación de la representación RGB (de 0 a 255) y la representación I1I2I3 (de 0 a 1) pueden causar que los valores I1I2I3 difieran ligeramente de los que usted espera.

Editar

normalize

Retorna los valores RGBA normalizados de un Color.

```
Color.normalize(): Return tuple
```

Retorna los valores RGBA normalizados de un Color como valores de números reales.

Editar

correct_gamma

Aplica un cierto valor gamma a el Color.

```
Color.correct_gamma (gamma): Return Color
```

Aplica un cierto valor gamma a el Color y retorna un nuevo color con los valores RGBA ajustados.

cursors

Módulo de pygame para cursores de mouse.

Pygame ofrece control sobre el cursor de hardware del sistema. Pygame solo soporta cursores blancos y negros para el sistema. Usted controla el cursor del mouse con funciones del módulo `pygame.mouse`.

Esta módulo contiene funciones para cargar y decodificar varios formatos gráficos de cursor. Esto le permite fácilmente almacenar sus cursores en archivos externos o directamente en cadenas de python con formato.

El módulo incluye varios cursores estándar. La función `pygame.mouse.set_cursor` toma varios argumentos, todos estos argumentos se pueden agrupar en una tupla para que usted puede llamarlo así:

```
pygame.mouse.set_cursor(*pygame.cursors.arrow)
```

Esta módulo también contiene unos pocos cursos en cadenas con formato. Necesitará pasar estas cadenas a la función `pygame.cursors.compile` antes de usarlos. Una llamada de ejemplo podría verse así:

```
cursor = pygame.cursors.compile(pygame.cursors.textmarker_strings)
pygame.mouse.set_cursor(*cursor)
```

Las siguientes variables son imágenes que se pueden usar como cursor:

- pygame.cursors.arrow
- pygame.cursors.diamond
- pygame.cursors.broken_x
- pygame.cursors.tri_left
- pygame.cursors.tri_right

Las siguientes cadenas se pueden convertir en imágenes de cursor con la función `pygame.cursor.compile`.

- pygame.cursors.thickarrow_strings
- pygame.cursors.sizer_x_strings
- pygame.cursors.sizer_y_strings
- pygame.cursors.sizer_xy_strings

Editar

compile

Genera datos de cursor binario desde una simple cadena.

```
pygame.cursor.compile(strings, black='X', white='.', xor='o'): return data, mask
```

Se puede usar una secuencia de cadenas para crear datos de cursor binario para el cursor de sistema. Los valores de retorno tiene el mismo formato necesario por `pygame.mouse.set_cursor`.

Si está creando su propias cadenas de cursor, puede usar cualquier valor representando los pixeles blancos y negros. Algunos sistemas le permiten definir un color especial de contraste (o inversión),

generalmente llamado el color xor. Si el sistema no soporta cursores xor, este color simplemente se verá como negro.

La longitud de las cadenas debe ser igual para todas y debe ser divisible por 8. Un ejemplo de cadenas de cursor se ven así:

```
thickarrow_strings = (               #sized 24x24
  "XX                      ",
  "XXX                     ",
  "XXXX                    ",
  "XX.XX                   ",
  "XX..XX                  ",
  "XX...XX                 ",
  "XX....XX                ",
  "XX.....XX               ",
  "XX......XX              ",
  "XX.......XX             ",
  "XX........XX            ",
  "XX........XXX           ",
  "XX......XXXXX           ",
  "XX.XXX..XX              ",
  "XXXX XX..XX             ",
  "XX   XX..XX             ",
  "     XX..XX             ",
  "      XX..XX            ",
  "      XX..XX            ",
  "       XXXX             ",
  "       XX               ",
  "                        ",
  "                        ",
  "                        ")
```

- buscar código donde se use esta función.

Editar

load_xbm

Carga datos de cursor desde un archivo xbm.

```
pygame.cursors.load_xbm(cursorfile, maskfile=None): return cursor_args
```

Esta función cargar cursor desde archivos XBM. Los archivos XBM se usan tradicionalmente para almacenar cursor en sistemas UNIX, estos están en formato ascii para representar imágenes simples.

A veces los valores de color blanco y negro se pueden dividir en dos archivos XBM separados. Puede pasar un segundo argumento `maskfile` para cargar las dos imágenes en un solo cursor.

Los argumentos `maskfile` y `cursorfile` pueden ser tanto nombres de archivo como objetos similares a `file`, es decir, con el método `readlines`.

El valor de retorno `cursor_args` se puede pasar directamente a la función `pygame.mouse.set_cursor`.

- buscar código donde se use esta función.

display

Módulo de pygame para controlar la ventana y pantalla de visualización.

- init
- quit
- get_init
- set_mode
- get_surface
- flip
- update
- get_driver
- get_wm_info
- list_modes
- mode_ok
- gl_get_attribute
- gl_set_attribute
- get_active
- iconify
- toggle_fullscreen
- set_gamma
- set_gamma_ramp
- set_icon
- set_caption
- get_caption
- set_palette

Otras páginas: :

- Info

Este módulo ofrece control sobre la visualización de pygame. Pygame contiene una sola superficie de visualización que puede estar contenida en una ventana o bien correr en pantalla completa. Una vez que cree la pantalla de visualización podrá tratarla como una superficie normal. Los cambios no se verán inmediatamente en pantalla, deberá elegir una de las dos funciones para actualizar la pantalla.

El punto de origen de la pantalla es `x=0` e `y=0`, este punto representa la esquina superior izquierda de la pantalla. Ambos ejes aumentan en dirección a la esquina inferior derecha de la pantalla.

La pantalla de pygame se puede inicializar en uno de varios modos de video. Por defecto la pantalla es una porción de memoria gestionada por software. Puede solicitar modos especiales como aceleración de video y soporte OpenGL enviando opciones a la función pygame.display.set_mode().

Pygame solo puede tener una pantalla activa a la vez. Crear una nueva usando `pygame.display.set_mode()` cerrará la anterior pantalla. Si se necesita tener control preciso sobre el formato de *pixels* o las resoluciones de video, utilice las funciones pygame.display.mode_ok(), pygame.display.list_modes() y pygame.display.Info() para consultar información acerca de la pantalla.

Una vez creada la superficie de visualización, las funciones de este módulo afectan a la pantalla existente. Si el módulo se deshabilita la superficie se convierte en inválida. Si se define un nuevo modo de video, la superficie existente automáticamente se cambiará para operar en la nueva pantalla.

Cuando se define el nuevo modo de video, se habilitan varios eventos de pygame para gestionar el manejo de la ventana. El evento `pygame.QUIT` se genera cuando el usuario solicita cerrar el programa. La ventana recibirá los eventos `pygame.ACTIVEEVENT` cuando obtenga o pierda el foco del sistema de ventanas. Si la ventana se construye con las opciones `pygame.RESIZABLE`, se generarán los eventos `pygame.VIDEORESIZE` cuando el usuario ajuste las dimensiones de la ventana. Las superficies de video en Hardware que imprimen directamente en la pantalla obtendrán eventos `pygame.VIDEOEXPOSE` cuando se deba redibujar porciones de la ventana.

Editar

init

Inicializa el módulo display.

```
pygame.display.init(): return None
```

Inicializa el módulo display de pygame. El módulo display no puede hacer nada a menos que sea inicializado. Esto generalmente se hace por usted de forma automática cuando llama a la función de mayor nivel `pygame.init()`.

Pygame seleccionará una de varias extensiones internas de visualización cuando se inicialice. La extensión interna se escogerá dependiendo de la plataforma y los permisos de usuario. Antes de inicializar el módulo de video se puede alterar la variable de entorno `SDL_VIDEODRIVER` para controlar la extensión interna de visualización. Los sistemas que tienen varias de estas extensiones se muestran aquí:

- Windows: windib, directx
- Unix, GNU/Linux: x11, dga, fbcon, directfb, ggi, vgl, svgalib, aalib

En algunas plataformas es posible incrustar la pantalla de pygame en una ventana existente. Para hacer esto, se debe definir la variable de entorno `SDL_WINDOWID` con una cadena que contenga el identificador o manejador de la ventana. La variable de entorno será verificada cuando se inicie el modo de video. Tenga en cuenta que pueden ocurrir varios efectos secundarios cuando utilice este modo de video incrustado.

Es inofensivo llamar a esta función mas de una vez, las sucesivas llamadas no tendrán efecto.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el módulo de video.

```
pygame.display.quit(): return None
```

Apagará el módulo display entero. Esto significa que cualquier pantalla activa se cerrará. Esto también se puede manejar de forma automática cuando el programa se cierra.

Es inofensivo llamar a esta función mas de una vez, las sucesivas llamadas no tendrán efecto.

- buscar código donde se use esta función.

Editar

get_init

Devuelve `True` si el módulo display se ha inicializado.

```
pygame.display.get_init(): return bool
```

Retorna `True` si el módulo `pygame.display` está inicializado actualmente.

- buscar código donde se use esta función.

Editar

set_mode

Inicializa una ventana o la pantalla para visualizar.

```
pygame.display.set_mode(resolution=(0,0), flags=0, depth=0): return Surface
```

Esta función construirá una superficie de pantalla. Los argumentos que se le envían solicitan el tipo de pantalla deseado. Si no especifica ningún argumento, la pantalla generada será tan grande como el mejor modo de video soportado por el sistema.

El argumento `resolution` es una tupla de números que representan el ancho y alto. El argumento `flags` es una colección de opciones adicionales. El argumento `depth` representa el número de bits utilizados por color.

El objeto Surface retornado se puede manipular como un objeto Surface normal, solo que los cambios eventualmente se podrán observar en el monitor.

Si no se especifica una resolución, o esta se define en (0, 0) y pygame utiliza la versión 1.2.10 de SDL o superior, la superficie creada tendrá el mismo tamaño que la resolución de pantalla actual. Si solo se define el ancho o alto a 0, la superficie tendrá el mismo alto o ancho de la resolución de pantalla. Se lanzará una excepción si está utilizando una versión menor a 1.2.10 de SDL.

Generalmente es mejor no especificar el argumento `depth`. Por defecto este coincidirá con la mejor y mas rápida profundidad de color para el sistema. Si su juego requiere un formato de color específico, el argumento `depth` puede definirlo. Pygame puede emular una profundidad de color no disponible aunque esto puede ser muy lento.

Cuando solicita modos de video en pantalla completa, a veces no se puede definir un modo de video exactamente igual al solicitado. En esas situaciones pygame seleccionará el modo de video mas cercano. Aún así, la superficie que se retorna siempre coincidirá exactamente con la resolución solicitada.

El argumento *flags* define que tipo de pantalla desea. Hay varias opciones para elegir, e incluso usted puede combinar varias opciones usando operaciones de bits, usando el operador "|" *pipe*.

Opción	Significado
pygame.FULLSCREEN	Genera una visualización de pantalla completa
pygame.DOUBLEBUF	Recomendado para combinar con HWSURFACE u OPENGL
pygame.HWSURFACE	Aceleración por hardware, solo funciona conjuntamente con FULLSCREEN
pygame.OPENGL	Genera una pantalla que se puede dibujar con opengl
pygame.RESIZABLE	La ventana se debe poder cambiar de tamaño
pygame.NOFRAME	La ventana no deberá tener bordes, titulo o controles

- buscar código donde se use esta función.

Editar

get_surface

Obtiene una referencia a la superficie de pantalla actual.

```
pygame.display.get_surface(): return Surface
```

Retorna una referencia a la superficie de pantalla actual. Si no se ha definido un modo de video esta función retornará `None`.

- buscar código donde se use esta función.

Editar

flip

Actualizar la superficie de visualización por completo sobre la pantalla.

```
pygame.display.flip(): return None
```

Actualizará el contenido de la pantalla entera. Si su modo de video usa las opciones `pygame.HWSURFACE` y `pygame.DOUBLEBUF`, esta operación esperará el retraso vertical e intercambiará las superficies. Si está usando un modo de video de diferente tipo, esta función simplemente actualizará el contenido completo de la superficie.

Cuando se usa un modo de video de tipo `pygame.OPENGL` esta función realizará un intercambio de buffer de OpenGL.

- buscar código donde se use esta función.

Editar

update

Actualiza porciones de la pantalla para modos de video de software.

```
pygame.display.update(rectangle=None): return None
pygame.display.update(rectangle_list): return None
```

Esta función es como una versión optimizada de pygame.display.flip para pantallas de software. Permite que solo una porción de la pantalla se actualice, en lugar de toda el área de pantalla. Si no se pasan argumentos, se actualizará la superficie completa como en pygame.display.flip.

Puede pasar a la función un rectángulo, o una secuencia de rectángulos. Es mas eficiente pasar varios rectángulos de una sola vez en lugar de llamar muchas veces a `update` con un solo rectángulo o una lista parcial de rectángulos. Es seguro pasar una secuencia de rectángulos con elementos `None` en ella, estos elementos serán ignorados.

Esta función no se debe usar en pantallas `pygame.OPENGL`, dado que generarán una excepción.

- buscar código donde se use esta función.

Editar

get_driver

Obtiene el nombre del controlador interno de visualización.

`pygame.display.get_driver(): return name`

Pygame elige uno de varios controladores de video internos cuando se inicializa. Esta función retorna el nombre interno del controlador utilizado. Esto puede utilizarse para proveer información limitada acerca de qué capacidades de video se pueden acelerar. Vea la opción `SDL_VIDEODRIVER` in la función pygame.display.set_mode() para ver algunas opciones adicionales.

- buscar código donde se use esta función.

Editar

get_wm_info

Obtiene información acerca sistema de ventana actual.

`pygame.display.get_wm_info(): return dict`

Genera un diccionario lleno de claves. Las cadenas y los valores son creados arbitrariamente por el sistema. Algunos sistemas podrían no brindar información, en cuyo caso se retorna un diccionario vacío. La mayoría de las plataformas retornan una clave *window* con el identificador de la ventana asignado por el sistema.

Esta función es nueva en pygame 1.7.1

- buscar código donde se use esta función.

Editar

list_modes

Obtiene una lista de los modos de pantalla completa disponibles

`pygame.display.list_modes(depth=0, flags=pygame.FULLSCREEN): return list`

Esta función retorna una lista de las posibles resoluciones para una profundidad de color definida. El valor de retorno será una lista vacía si no hay modos de video disponibles para los argumentos dados. Un valor de retorno -1 significa que cualquier resolución debería funcionar (este es generalmente el caso de los modos en ventana). Los modos de video se ordenan de mayor (primeros) a menor (últimos).

Si el argumento `depth` es 0, SDL seleccionará la actual/mejor profundidad de color para la pantalla. El argumento `flags` está por defecto en el valor `pygame.FULLSCREEN`, pero usted puede agregar mas opciones para modos de video en pantalla completa específicos.

- buscar código donde se use esta función.

Editar

mode_ok

Elige la mejor profundidad de color para el modo de video.

```
pygame.display.mode_ok(size, flags=0, depth=0): return depth
```

Esta función utiliza los mismos argumentos que la función pygame.display.set_mode. Se usa para determinar si está disponible el modo de video solicitado. En caso de no estar disponible retorna la profundidad de color que mejor coincida con la pantalla solicitada.

Generalmente no se especifica el argumento `depth`, dado que algunas plataformas suportan varios modos de video. Si especifica este argumento dará a entender cual es el mejor modo de video.

Las opciones `flags` mas útiles son `pygame.HWSURFACE`, `pygame.DOUBLEBUF` y tal vez `pygame.FULLSCREEN`. La función retornará 0 si no se puede definir este tipo de pantalla.

- buscar código donde se use esta función.

Editar

gl_get_attribute

Obtiene el valor de los atributos de opengl para la pantalla actual.

```
pygame.display.gl_get_attribute(flag): return value
```

Después de llamar a `pygame.display.set_mode()` con la opción `pygame.OPENGL`, es una buena idea consultar el valor de cualquier atributo de opengl solicitado. Vea las función pygame.display.gl_set_attribute() para obtener una lista de opciones válida.

- buscar código donde se use esta función.

Editar

gl_set_attribute

Solicita un atributo de OpenGL para el modo de video.

```
pygame.display.gl_set_attribute(flag, value): return None
```

Cuando llama a pygame.display.set_mode() con la opción `pygame.OPENGL`, pygame automáticamente maneja la configuración de los atributos de OpenGL como el color y el *double buffer*. OpenGL ofrece muchos otros atributos que tal vez quiera controlar por su cuenta. Indique uno de estos atributos como el parámetro `flag` y su valor apropiado como `value`. Esto debe llamarse antes de ejecutar pygame.display.set_mode().

Las opciones de OpenGL, para el argumento `flag` son:

- 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
- buscar código donde se use esta función.

Editar

get_active

Devuelve True cuando la ventana se está mostrando en la pantalla.

```
pygame.display.get_active(): return bool
```

Luego de llamar a `pygame.display.set_mode` la superficie de visualización se hará visible en la pantalla. La mayoría de los sistemas de ventanas permiten que el usuario pueda ocultar las ventanas. Esta función retornará False si la pantalla de visualización está oculta o minimizada.

- buscar código donde se use esta función.

Editar

iconify

Minimiza la aplicación.

```
pygame.display.iconify(): return bool
```

Solicita al sistema de ventanas que oculte o minimice la pantalla. No todos los sistemas y pantallas soportan esta opción. La función retornará True en caso de éxito.

Cuando la pantalla está minimizada la función `pygame.display.get_active()` retornará False. La cola de eventos debería recibir el evento `ACTIVEEVENT` cuando la ventana se minimiza.

- buscar código donde se use esta función.

Editar

toggle_fullscreen

Alterna entre la visualización de ventana o pantalla completa.

```
pygame.display.toggle_fullscreen(): return bool
```

Alterna entre los modos de video pantalla completa y ventana. Esta operación solo funciona bajo el controlador x11 de unix. Para la mayoría de las situaciones es mejor llamar a `pygame.display.set_mode()` con las nuevas opciones de pantalla.

- buscar código donde se use esta función.

Editar

set_gamma

Cambia los niveles de color del hardware.

```
pygame.display.set_gamma(red, green=None, blue=None): return bool
```

Define los niveles de rojo, verde y azul en el hardware de video. Si no se especifican los valores verde y azul, ambos se evaluarán como si tuvieran el mismo valor que el componente rojo. No todos los sistemas y hardware soportan paletas gama, si la función retorna True si opera correctamente.

Un valor gamma de 1.0 genera una tabla de color lineal. Los valores inferiores harán mas oscura la pantalla y los valores superiores aumentarán el brillo.

- buscar código donde se use esta función.

Editar

set_gamma_ramp

Cambia la paleta de colores de hardware con una búsqueda personalizada.

```
pygame.display.set_gamma_ramp(red, green, blue): return bool
```

Define los niveles de rojo, verde y azul con una tabla de búsqueda explícita. Cada argumento debería ser una secuencia de 256 números enteros. Los enteros deberían estar entre `0` y `0xffff`. No todos los sistemas y hardware soportan paletas gama, si la función retorna True si opera correctamente.

- buscar código donde se use esta función.

Editar

set_icon

Cambia la imagen de sistema para la ventana.

```
pygame.display.set_icon(Surface): return None
```

Define el icono de ejecución que el sistema usará para representar la ventana. Por defecto todas las ventanas muestran el logo de pygame como icono de la ventana.

Puede especificar cualquier superficie, aunque la mayoría de los sistemas esperan una imagen pequeña estilo 32×32. La imagen puede tener una transparencia por color clave que será enviada al sistema.

Algunos sistemas no permite que se cambie el icono luego de que la ventana se ha mostrado. Esta función se puede llamar antes de `pygame.display.set_mode()` para crear el icono antes de especificar el modo de video.

- buscar código donde se use esta función.

Editar

set_caption

Define el título de la ventana.

```
pygame.display.set_caption(title, icontitle=None): return None
```

Si la pantalla tiene un título de ventana, esta función cambiará ese título. Algunos sistemas soportan un titulo alternativo mas corto para utilizarse en ventanas minimizadas.

- buscar código donde se use esta función.

Editar

get_caption

Obtiene el título de la ventana.

`pygame.display.get_caption(): return (title, icontitle)`

Retorna el título principal y el titulo alternativo (para ventanas minimizadas) de la ventana principal. Estos generalmente tienen el mismo valor.

- buscar código donde se use esta función.

Editar

set_palette

Define la paleta de colores para modos de video indexado.

`pygame.display.set_palette(palette=None): return None`

Cambiará la paleta de colores para pantallas de 8bit. No cambia la paleta de la superficie de visualización, solo cambia la palera que se usa para mostrar la superficie. Si no se especifica un argumento, se restaurará la paleta por defecto. La palera es una secuencia de ternas RGB.

- buscar código donde se use esta función.

Info

Genera un objeto de información sobre el dispositivo de video.

`pygame.display.Info(): return VideoInfo`

Genera un objeto simple que contiene varios atributos para describir el entorno gráfico actual. Si es generado antes de llamar a `pygame.displat.set_mode()` algunas plataformas pueden otorgar información acerca del modo de video por defecto. También se puede llamar después de definir el modo de video para verificar que opciones de video específicas se han cumplido. El objeto VideoInfo tiene varios atributos.

Atributo	Significado
hw	True si el modo de video soporta aceleración por hardware.
wm	True se se puede utilizar un modo de video en ventana.
video_mem	Los megabytes de la memoria de video en pantalla. 0 será el valor si se desconoce.
bitsize	Número de bits utilizados para almacenar cada pixel.
bytesize	Número de bytes utilizados para almacenar cada pixel.
masks	Cuatro valores que se usan para agrupar componentes RGB en *pixeles*.
shifts	Cuatro valores que se usan para agrupar componentes RGB en *pixeles*.
losses	Cuatro valores que se usan para agrupar componentes RGB en *pixeles*.
blit_hw	Vale True si la impresión de superficies de hardware soporta aceleración.
blit_hw_CC	Vale True si la impresión mediante color clave (*colorkey*) soporta aceleración.
blit_hw_A	Vale True si la impresión de superficies con */pixeles* transparentes soporta aceleración.
blit_sw	Vale True si la impresión de superficies soporta aceleración.
blit_sw_CC	Vale True si la impresión mediante color clave (*colorkey*) en superficies de software soporta aceleración.
blit_sw_A	Vale True si la impresión de superficies de software con *pixeles* alpha soporta aceleración.
current_h, current_h	Ancho y alto del modo de video actual, o el tamaño del escritorio si es llamado antes de ejecutar `pygame.display.set_mode`. (current_h, current_w están disponibles desde la versión 1.2.10 de SDL, y pygame 1.8.0). Valdrá -1 en caso de error o si se ha utilizado una versión antigua de SDL.

draw

Módulo de pygame para dibujar figuras.

Dibuja varias figuras simples a una superficie. Estas funciones servirán para pintar en cualquier formato de superficie. Aunque dibujar en superficies almacenadas en Hardware será mas lento que en superficies de Software.

La mayoría de las funciones admiten un argumento `width` para representar el tamaño de la linea alrededor del borde de la figura. Si se pasa el parámetro `width` con valor 0, la función pintará la figura entera como sólida.

Todas las funciones de dibujo respetan el área de recorte para la superficies, y se limitará a esa área. Las funciones retornan un rectángulo representando el área límite de los *pixels* modificados.

La mayoría de las funciones aceptan un argumento `color`, que es una terna o tupla RGB. Incluso aceptan una tupla de 4 elementos para RGBA. El valor `alpha` (o componente de transparencia) se dibujará directamente en la superficie si ésta contiene la propiedad de transparencia de *pixels*, aunque la función de dibujo no dibujará de forma transparente. El argumento `color` también puede ser un valor de *pixel* entero que ya esté convertido al formato de *pixel* de la imagen.

Estas funciones bloquearan temporalmente la superficie donde están operante. Así que muchas llamadas de dibujo se pueden acelerar bloqueando y desbloqueando la superficie donde se trabaja antes y después de llamar a las funciones de dibujo.

Editar

rect

Dibuja una figura rectangular.

```
pygame.draw.rect(Surface, color, Rect, width=0): return Rect
```

Dibuja una figura rectangular sobre una superficie. El parámetro `Rect` dado es el área del rectángulo. El argumento `width` es el espesor para dibujar el borde exterior de la figura. Si `width` se define con valor 0 entonces el rectángulo se pintará completo.

Recuerde que el método Surface.fill funciona igual de bien para dibujar rectángulos completos. De hecho, Surface.fill se puede acelerar por medio de Hardware en algunas plataformas tanto con modos de video por Software como Hardware.

- buscar código donde se use esta función.

Editar

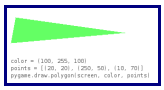
polygon

Dibuja una figura con cualquier número de lados.

`pygame.draw.polygon(Surface, color, pointlist, width=0): return Rect`

Dibuja una figura poligonal en una superficie. El argumento `pointlist` es una lista de los vértices del polígono. El argumento `width` es el espesor para dibujar el borde de la figura. Si `width` es 0 entonces el polígono se pinta por completo.

Para lograr un polígono con bordes suaves utilice aalines con el parámetro `closed`.



- buscar código donde se use esta función.

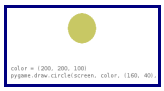
Editar

circle

Dibujar un círculo alrededor de un punto.

`pygame.draw.circle(Surface, color, pos, radius, width=0): return Rect`

Dibuja una figura circular sobre una superficie. El argumento `pos` es el centro del círculo, y `radius` es el tamaño. El argumento `with` es el espesor del borde de la figura. Si `with` es 0 entonces el círculo se pintará por completo.



- buscar código donde se use esta función.

Editar

ellipse

Dibuja una elipse dentro del área indicada por un rectángulo.

`pygame.draw.ellipse(Surface, color, Rect, width=0): return Rect`

Dibuja una figura elíptica en una superficie. La figura se dibujará en el área delimitada por el rectángulo dado.

- buscar código donde se use esta función.

Editar

arc

Dibuja una sección parcial de una elípse.

`pygame.draw.arc(Surface, color, Rect, start_angle, stop_angle, width=1): return Rect`

Dibuja un arco elíptico en una superficie. La figura se dibujará en el área delimitada por el rectángulo. Los dos argumentos `angle` son los ángulos inicial y final en radianes, donde 0 representa la izquierda. El argumento `width` es el espesor de la linea de dibujo.

- buscar código donde se use esta función.

Editar

line

Dibuja un segmento de linea recto.

`pygame.draw.line(Surface, color, start_pos, end_pos, width=1): return Rect`

Dibuja un segmento de linea recto en una superficie. No se dibujan puntas en los extremos de la linea, las terminaciones serán cuadradas para lineas muy gruesas.

- buscar código donde se use esta función.

Editar

lines

Dibuja múltiples segmentos de linea continuos.

`pygame.draw.lines(Surface, color, closed, pointlist, width=1): return Rect`

Dibuja una secuencia de lineas en una superficies. El argumento `pointlist` es una serie de puntos que se conectarán por una linea. Si el argumento `closed` es true se dibujará un segmento de linea adicional entre el primer y último punto.

Esta función no dibuja ninguna punta o nodo. Las lineas con esquinas afiladas y grosor de linea muy grande podrían mostrar las intersecciones de segmentos de manera inapropiada.

- buscar código donde se use esta función.

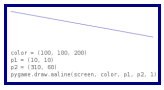
Editar

aaline

Dibuja lineas suavizadas.

`pygame.draw.aaline(Surface, color, startpos, endpos, blend=1): return Rect`

Dibuja una linea suavizada en una superficie. Esta función estima el rectángulo afectado. Se retorna el rectángulo que representa el área afectada. Si el parámetro `blend` está en valor `True`, las figuras se mezclarán con la tonalidad de los *pixels* existentes en lugar de sobre-escribirlos. Esta función acepta valores en números realizar para los extremos de los segmentos.



- buscar código donde se use esta función.

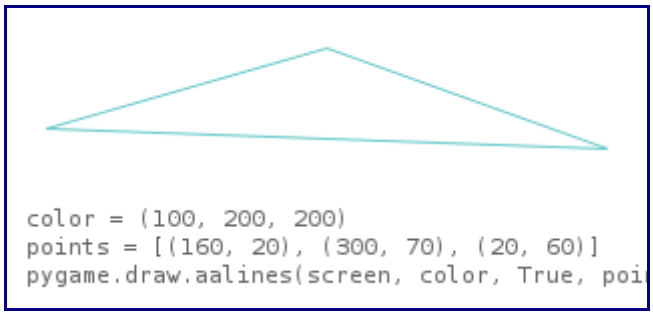
Editar

aalines

Dibuja múltiples segmentos de linea continuos.

`pygame.draw.aalines(Surface, color, closed, pointlist, blend=1): return Rect`

Dibuja una secuencia de lineas en una superficie. Debe indicar al menos dos puntos en la secuencia de puntos (parámetro `pointlist`). El argumento `closed` es un valor booleano, si es `True` se dibujará un segmento de linea adicional entre el primer y último punto. Si el argumento `blend` se define a `True` la figura se dibujará con las tonalidades existentes en lugar de reemplazarlas. Esta función acepta valores en números reales para especificar los puntos.



```
color = (100, 200, 200)
points = [(160, 20), (300, 70), (20, 60)]
pygame.draw.aalines(screen, color, True, poi
```

- buscar código donde se use esta función.

event

Módulo de pygame para interactuar con eventos y pedidos.

- pump
- get
- poll
- wait
- peek
- clear
- event_name
- set_blocked
- set_allowed
- get_blocked
- set_grab
- get_grab
- post

Otras páginas:

- Event

Pygame maneja todos sus mensajes de eventos a través de una cola de eventos. Las rutinas de este módulo le ayudarán a manejar esta cola de eventos. Los eventos de entrada son extremadamente dependientes del módulo `display` de pygame. La cola de eventos no funcionará a menos que se halla inicializado el módulo display y el modo de video.

La cola de eventos es una lista de objetos Event, hay diversas maneras de acceder a los eventos que esta contiene. Desde consultar por la existencia de eventos, a extraerlos directamente de la pila.

Todos los eventos tienen un identificador `type`. Este identificador está entre los valores NOEVENT y NUMEVENTS. Todos los eventos definidos por el usuario pueden tener el valor de USEREVENT o superior. Se recomienda asegurarse de que sus identificadores de evento sigan este sistema.

Para obtener el estado de varios dispositivos de entrada puede olvidar la cola de eventos y acceder a los dispositivos directamente desde sus módulos asignados: `mouse`, `key` y `joystick`. Si usa este método, recuerde que pygame requiere alguna forma de comunicación con el sistema de ventanas y otras partes del sistema. Para mantener a pygame en coherencia con el sistema necesita llamar a `pygame.event.pump` periodicamente, usualmente una vez por ciclo del bucle de juego.

La cola de eventos ofrece una forma de filtro simple. Esto puede ayudar a mejorar el rendimiento bloqueando ciertos tipos de eventos de la cola, use las funciones `pygame.event.set_allowed()` y `pygame.event.set_blocked()` para trabajar con este filtrado. Por defecto todos los eventos están permitidos.

Los controles de Joystick no emitirán ningún evento a menos que se inicialice el dispositivo.

Un objeto Evento contiene un tipo de evento y un conjunto de miembros, o atributos, de solo lectura. El objeto Evento no contiene métodos, solo información. Estos objetos se obtienen desde la cola de eventos. E incluso puede crear sus propios eventos con una llamada a `pygame.event.Event()`.

Su programa debe seguir ciertos pasos para evitar que la cola de eventos se sobrepase del límite. Si el programa no limpia o elimina los eventos de la cola de eventos en intervalos regulares, esta podría desbordarse. Se lanzará una excepción si la cola desborda.

Todos los objetos `Event` contienen un identificador de tipo en el atributo `Event.type`. Usted

puede obtener acceso a todos los atributos del evento a través el método `Event.dict`. Todas las otras búsquedas de atributos pasarán a través de los valores de diccionario del evento.

Puede imprimir los objetos Event para ver rápidamente su tipo y atributos mientras depura o experimenta. Los eventos que vienen desde el sistema tendrán un grupo asegurado de atributos basados en cada tipo. Esta es una lista de los atributos de evento que se definen para cada tipo:

Tipo (atributo `type`)	Atributos
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

Editar

pump

Procesa internamente los manejadores de evento de pygame.

```
pygame.event.pump(): return None
```

Para cada cuadro de visualización de su juego necesitará hacer alguna clase de llamada a la cola de eventos. Esta función se asegura de que su programa puede interactuar internamente con el resto del sistema operativo. Debería utilizar esta función si no está usando otras funciones de eventos en su juego, esto permitirá que pygame pueda manejar acciones internas.

Esta función no se necesita si su programa procesa eventos de manera consistente a través de otras funciones de `pygame.event`.

Hay varias tareas importantes que se deben realizar internamente en la cola de eventos. La ventana principal podría necesitar responderle al sistema o ser redibujada. Si usted no consulta la cola de eventos por mucho tiempo, el sistema podría interpretar que su programa está inactivo o con un error.

- buscar código donde se use esta función.

Editar

get

Obtiene eventos de la cola.

```
pygame.event.get(): return Eventlist
pygame.event.get(type): return Eventlist
pygame.event.get(typelist): return Eventlist
```

Obtendrá todos los mensajes de eventos y los eliminará de la cola. Si se le especifica un tipo o secuencia de tipos de evento solo esos mensajes se eliminarán de la cola.

Si solo está interesado en eventos específicos de la cola, tenga cuidado que la cola podría llenarse eventualmente con eventos en los que no esté interesado.

- buscar código donde se use esta función.

Editar

poll

Obtiene un solo evento de la cola.

```
pygame.event.poll(): return Event
```

Retorna un evento individual de la cola. Si la cola de eventos está vacía se retornará un evento de tipo `pygame.NOEVENT` inmediatamente. El evento que se retorna se eliminará de la cola.

- buscar código donde se use esta función.

Editar

wait

Espera por la llegada de un evento en la cola.

```
pygame.event.wait(): return Event
```

Retorna un evento individual de la cola. Esta función espera hasta que un evento llegue si es que la cola está vacía. El evento se elimina de la cola una vez que ha sido retornado. Mientras el programa está esperando el proceso dormirá en un estado de espera. Esto es importante para programas que quieres compartir los recursos de sistema con otras aplicaciones.

- buscar código donde se use esta función.

Editar

peek

Consulta si un tipo de evento está esperando en la cola.

```
pygame.event.peek(type): return bool
pygame.event.peek(typelist): return bool
```

Retorna `True` si existe algunos de los eventos solicitados esperando en la cola de eventos. Si se pasa una secuencia de tipos de eventos, se retornará `True` si alguno de esos eventos está en la cola.

- buscar código donde se use esta función.

Editar

clear

Elimina todos los eventos de la cola.

```
pygame.event.clear(): return None
pygame.event.clear(type): return None
pygame.event.clear(typelist): return None
```

Elimina todos los eventos (o de un tipo específico) de la cola. Esta función tiene el mismo efecto que `pygame.event.get()` excepto que no retorna nada. Puede ser ligeramente mas eficiente cuando limpia toda la cola de eventos.

- buscar código donde se use esta función.

Editar

event_name

Obtiene el nombre de un identificador de evento.

```
pygame.event.event_name(type): return string
```

Pygame utiliza identificadores en números enteros para representar tipos de eventos. Si quiere reportar estos valores al usuario debería convertirlos a cadenas de texto. Esta función retornará el nombre para un tipo de evento. La cadena sigue el estilo PalabrasComoTitulo.

```
>>> pygame.event.event_name(1)
'ActiveEvent'
```

- buscar código donde se use esta función.

Editar

set_blocked

Controla que eventos se permiten en la cola.

```
pygame.event.set_blocked(type): return None
pygame.event.set_blocked(typelist): return None
pygame.event.set_blocked(None): return None
```

El tipo de evento especificado será excluido de poder aparecer en la cola de eventos. Por defecto todos los eventos podrá colocarse en la cola. Es seguro deshabilitar un tipo de evento muchas veces.

Si se pasa None como argumento, esto producirá el efecto contrario y se permitirán todos los tipos de eventos en la cola.

- buscar código donde se use esta función.

Editar

set_allowed

Controla que eventos se permitirán en la cola.

```
pygame.event.set_allowed(type): return None
pygame.event.set_allowed(typelist): return None
```

`pygame.event.set_allowed(None): return None`

El tipo indicado será incluido para aparecer en la cola de eventos. Por defecto todos los eventos pueden colocarse en la cola de eventos. Es seguro habilitar un tipo de evento varias veces.

Si se pasa `None` como argumento, ninguno de los tipos de eventos podrá estar en la cola de eventos.

- buscar código donde se use esta función.

Editar

get_blocked

Consulta si está bloqueado de la cola un tipo de evento.

`pygame.event.get_blocked(type): return bool`

Retorna `True` si el tipo de evento dado ha sido bloqueado de la cola de eventos.

- buscar código donde se use esta función.

Editar

set_grab

Controla la posibilidad de compartir los dispositivos de entrada con otras aplicaciones.

`pygame.event.set_grab(bool): return None`

Cuando su programa corre en un entorno de ventana, este compartirá los dispositivos de mouse y teclado con otras aplicaciones que estén seleccionadas (con foco). Si su programa llama a `set_grab(True)`, se bloquearán todas las entras de dispositivos a su programa.

Es mejor no usar esta función siempre, dado que imposibilita al usuario de hacer otras cosas en su sistema.

- buscar código donde se use esta función.

Editar

get_grab

Consulta si el programa está compartiendo los dispositivos en entrada.

`pygame.event.get_grab(): return bool`

Retorna `True` cuando los eventos de entrada son exclusivos de esta aplicación. Use `pygame.event.set_grab()` para controlar este estado.

- buscar código donde se use esta función.

Editar

post

Coloca un nuevo evento en la cola.

```
pygame.event.post(Event): return None
```

Esta función coloca un nuevo evento al final de la cola de eventos. Estos eventos se recibirán mas tarde que los otros eventos ya en la cola.

Se usa frecuentemente para colocar eventos pygame.USEREVENT en la cola. Aunque se puede colocar cualquier tipo de evento, si utiliza tipos de eventos de sistema se asegurará de crear los atributos por defecto con los valores adecuados.

- buscar código donde se use esta función.

Event

Genera un nuevo objeto de evento.

```
pygame.event.Event(type, dict): return Event
pygame.event.Event(type, * * attributes): return Event
```

Genera un nuevo evento a partir del tipo de evento dado. El evento se construye con los valores y atributos dados. Los atributos pueden venir desde un argumento diccionario, o como cadenas claves de un diccionario.

Los atributos dados serán atributos de solo lectura en el nuevo objeto de evento. Solo hay atributos en el objeto de evento, no hay métodos vinculados a los objetos de evento.

font

Módulo de pygame para cargar y dibujar fuentes.

Otras páginas:

El módulo font permite dibujar fuentes *TrueType* sobre objetos Surface nuevos. Este módulo es opcional y requiere *SDL_ttf* como una dependencia. Usted debería verificar si `pygame.font` está disponible e inicializado antes de intentar utilizarlo.

La mayor parte del trabajo con fuentes se realiza usando los objetos Font. El módulo en sí solo tiene rutinas para inicializar el módulo y crear objetos Font con `pygame.font.Font()`.

Puede cargar fuentes desde el sistema usando la función `pygame.font.SysFont()`. Hay otras funciones para ayudar a encontrar fuentes del sistema.

Pygame viene con una fuente incorporada por defecto. Se puede acceder a esta utilizando `None` como el nombre de fuente.

Editar

init

Inicializa el módulo font.

```
pygame.font.init(): return None
```

Este método se llama automáticamente por `pygame.init()`. Su función es inicializar el módulo font. El módulo se debe inicializar antes de utilizar cualquier otra función.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el módulo font.

```
pygame.font.quit(): return None
```

Deshabilita de forma manual el sistema de fuentes de SDL_ttf. Esta función se llama automáticamente por `pygame.quit()`.

Es seguro llamar a esta función incluso si el módulo no está inicializado.

- buscar código donde se use esta función.

Editar

get_init

Retorna True si el módulo font está inicializado.

`pygame.font.get_init(): return bool`

Verifica si el módulo font está inicializado o no.

- buscar código donde se use esta función.

Editar

get_default_font

Obtiene el nombre de archivo de la fuente por defecto.

`pygame.font.get_default_font(): return string`

Retorna el nombre de archivo de la fuente de sistema. El nombre no es la ruta completa al archivo. Este archivo se puede encontrar en el mismo directorio que el módulo font, aunque también puede estar en un paquete separado.

- buscar código donde se use esta función.

Editar

get_fonts

Obtiene todas las fuentes disponibles.

`pygame.font.get_fonts(): return list of strings`

Retorna una lista de todos las fuentes disponibles en el sistema. Los nombres de las fuentes se mostrarán en minúsculas sin espacios o puntuación. Esta rutina funciona en la mayoría de los sistema, pero en algunos retorna una lista vacía si no puede entrar las fuentes.

- buscar código donde se use esta función.

Editar

match_font

Busca una fuentes específica del sistema.

`pygame.font.match_font(name, bold=False, italic=False): return path`

Retorna la ruta completa a un archivo de fuente en el sistema. Si se usa True en los argumentos `bold` o `italic`, se intentará encontrar la familia o fuente correcta.

El nombre de la fuente puede ser una lista de nombres para probar. Se retorna None si ninguno de los nombres dados se encuentra.

Editar

Por ejemplo

```
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)
```

- buscar código donde se use esta función.

Font

Genera un nuevo objeto Font a partir de un archivo.

- Font
- render
- size
- set_underline
- get_underline
- set_bold
- get_bold
- set_italic
- metrics
- get_italic
- get_linesize
- get_height
- get_ascent
- get_descent

Editar

Font

```
pygame.font.Font(filename, size): return Font
pygame.font.Font(object, size): return Font
```

Carga una fuente nueva a partir de un nombre de archivo u objeto archivo de python dado. El argumento `size` es la altura de la fuente en pixeles. Si el argumento `filename` vale `None`, pygame cargará la tipografía por defecto. Se lanzará una excepción si la fuente indicada por los argumentos no se puede cargar. No se puede cambiar el tamaño de una fuente una vez creado el objeto.

Los objetos Font se usan principalmente para imprimir texto en superficies nuevas. La impresión puede simular características como **negrita** y *cursiva*, aunque es mejor cargar fuentes que ya tengan estos atributos. El texto a imprimir puede ser unicode o texto normal.

- buscar código donde se use esta función.

Editar

render

Imprime texto en una nueva superficie.

```
Font.render(text, antialias, color, background=None): return Surface
```

Genera una nueva superficie con el texto especificado dentro de ella. Pygame no provee una manera directa de imprimir texto en una superficie existente: en lugar de eso debe usar esta función para crear una imagen del texto y luego volcar esta imagen en la otra superficie.

El texto solamente puede ser de una linea: Los caracteres que indican salto de linea no se imprimen. El argumento `antialias` es un valor booleano: si vale `True` los caracteres tendrán bordes suaves. El argumento `color` es el color del texto, por ejemplo puede usar (0, 0, 255) para definir el

color azul. El argumento opcional `background` es el color para utilizar de fondo. Si no se especifica un fondo, el área fuera del texto será transparente.

La superficie retornada será del tamaño necesario para alojar el texto (el tamaño lo retorna la función `Font.size()`). Si se envía una cadena vacía en lugar de texto, se retornará una superficie negra que tendrá un *pixel* de ancho y la altura de la fuente.

Retornará distintos tipos de superficie dependiendo del tipo de fondo y suavidad (`antialias`) solicitado. Por razones de rendimiento es bueno conocer el tipo de imagen que se usará. Si no usa `antialias` se retornará una superficie de 8 bits con una paleta de dos colores. Si el fondo es transparente se usará una transparencia por color (colorkey). Las imágenes con `antialias` se generan en superficies RGB de 24 bit, y se incluirá un pixel *alpha* si el fondo es transparente.

Sugerencia de optimización: Si sabe que el destino de impresión para el texto siempre tendrá un color uniforme, y utiliza `antialias`, podrá mejorar el rendimiento especificando el color de fondo al crear la superficie. Esto produce una imagen que tiene información de transparencia por color en lugar de transparencia alpha (mucho menos eficiente).

Si usa '\n' en el texto, se pintará una letra desconocida, generalmente un rectángulo. En lugar de ello tendrá que manejar los saltos de linea usted mismo.

La operación de imprimir texto no se puede realizar simultáneamente desde diferentes hilos (concepto http://es.wikipedia.org/wiki/Thread-Safety:Thread safe), por lo tanto solo un hilo puede imprimir texto a la vez.

- buscar código donde se use esta función.

Editar

size

Determina la cantidad de tamaño necesario para dibujar texto.

```
Font.size(text): return (width, height)
```

Retorna la dimensión necesaria para dibujar el texto. Se puede usar para determinar la posición necesaria para el texto antes de imprimirlo. Y también se puede usar para separar palabras u otros efectos de posicionado.

Tengan en cuenta que la mayoría de las fuentes utiliza el procedimiento Kerning que ajusta el ancho de ciertos pares de letras. Por ejemplo, el ancho de “ae” no siempre coincidirá con el ancho de “a” + “b”.

- buscar código donde se use esta función.

Editar

set_underline

Controla si el texto se dibujará con un subrayado.

```
Font.set_underline(bool): return None
```

Cuando se habilita, todas las fuentes que se dibujen incluirán un subrayado. El subrayado siempre tendrá un pixel de grosor, independiente del tamaño de la fuente. Esto se puede combinar con los modos **negrita** y *cursiva*.

- buscar código donde se use esta función.

Editar

get_underline

Consulta si el texto se dibujará con un subrayado.

```
Font.get_underline(): return bool
```

Retorna `True` cuando está habilitado el subrayado de texto.

- buscar código donde se use esta función.

Editar

set_bold

Habilita el dibujado de texto en negrita falso.

```
Font.set_bold(bool): return None
```

Habilita el dibujado de texto en negrita. Este es un estiramiento falso de la fuente que no se verá muy bien en varios tipos de fuente. Si puede cargar la fuente de un archivo de fuente negrita real, la fuente sí tendrá diferentes grosores que la fuente normal. Este modo se puede mezclar con los modos subrayado y *cursiva*.

- buscar código donde se use esta función.

Editar

get_bold

Consulta si el texto se dibujará en negrita.

```
Font.get_bold(): return bool
```

Retorna `True` cuando el modo de dibujado en negrita está habilitado.

- buscar código donde se use esta función.

Editar

set_italic

Habilita el dibujado de texto en cursiva falso.

```
Font.set_bold(bool): return None
```

Habilita el dibujado de texto en cursiva. Esto es una deformación falsa de la fuente que no se verá muy bien en varios tipos de fuente. Si puede cargar la fuente de un archivo de fuente cursiva real, la fuente sí tendrá diferente aspecto que la fuente normal. Este modo se puede mezclar con los modos negrita y subrayado.

- buscar código donde se use esta función.

Editar

metrics

Retorna las medidas de cada letra de la cadena indicada.

```
Font.metrics(text): return list
```

La lista retornada contiene tuplas para cada caracter, que contienen el desplazamiento mínimo en X, el desplazamiento máximo en X, el desplazamiento mínimo en Y, el desplazamiento mínimo en Y el desplazamiento anticipado de el otro caracter. `[(minx, maxx, miny, maxy, advance), (minx, maxx, miny, maxy, advance), …]`.

- buscar código donde se use esta función.

Editar

get_italic

Consulta si el texto se dibujará en cursiva.

```
Font.get_italic(): return bool
```

Retorna `True` cuando el dibujado de fuente en cursiva está habilitado.

- buscar código donde se use esta función.

Editar

get_linesize

Obtiene el espacio de linea de un texto.

```
Font.get_linesize(): return int
```

Retorna la altura en *pixeles* de una linea de texto. Se recomienda esta cantidad de espacio entre lineas cuando dibuje varias lineas de texto.

- buscar código donde se use esta función.

Editar

get_height

Obtiene la altura de una fuente.

```
Font.get_height(): return int
```

Retorna la altura en *pixels* del texto que se dibuja. Este es el tamaño aproximado de cada figura en la fuente.

- buscar código donde se use esta función.

Editar

get_ascent

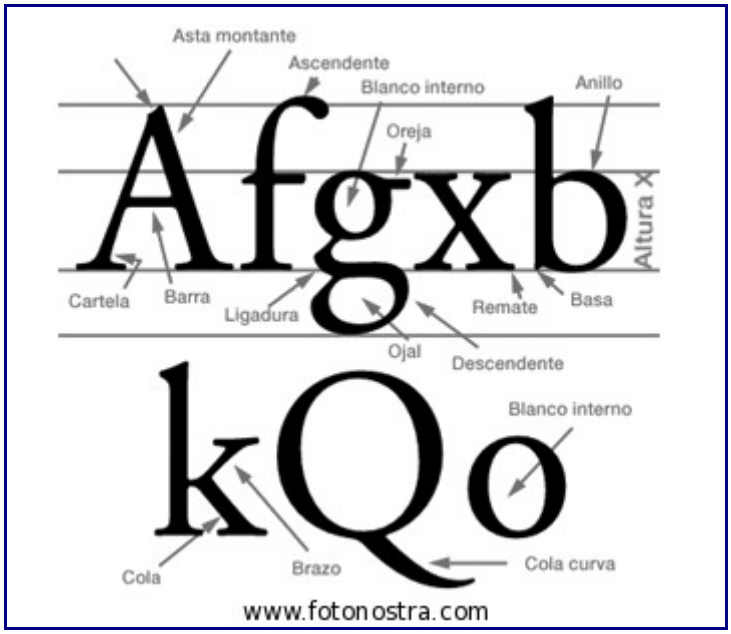
get the ascent of the font

`Font.get_ascent(): return int`

Retorna la altura en *pixeles* del ascendente de la fuente. El ascendente es el numero de *pixeles* desde el anillo (o linea superior) de la parte superior de la fuente.

- buscar código donde se use esta función.

Las siguiente imagen ilustra las medidas:



Editar

get_descent

Retorna el descendente de la fuente.

`Font.get_descent(): return int`

Retorna la altura en *pixeles* del descendente de la fuente. El descendente es el numero de *pixeles* desde la basa (o linea base) de la parte inferior de la fuente.

- buscar código donde se use esta función.

Las siguiente imagen ilustra las medidas:

SysFont

Genera un objeto Font a partir de las fuentes del sistema.

Editar

SysFont

`pygame.font.SysFont(name, size, bold=False, italic=False): return Font`

Retorna un nuevo objeto Font que se carga a partir de las fuentes del sistema. La fuente deberá coincidir con las opciones `bold` e `italic` solicitadas. Si no se encuentra una fuente de sistema adecuada esta función retornará la fuente por defecto de pygame. El parámetro `name` puede ser una lista de nombres para explorar.

- buscar código donde se use esta función.

pygame.image

Módulo de pygame para la transferencia de imagen.

El módulo `image` contiene funciones para leer y grabar imágenes, como así también para transferir superficies a formatos accesibles para otros paquetes.

Note que no hay una clase Image, una imagen se carga como un objeto `Surface`. La clase `Surface` permite operaciones como dibujar lineas, pintar *pixeles*, capturar regiones, etc.

El módulo `image` es una dependencia importante de pygame, aunque el soporte extendido a formatos es opcional. Por defecto solo puede cargar imágenes BMP sin compresión; pero cuando pygame se provee con soporte de imágenes completo, la función `pygame.image.load()` puede interpretar los siguientes formatos:

- JPG
- PNG
- GIF (sin animación)
- BMP
- PCX
- TGA (sin compresión)
- TIF
- LBM (y PBM)
- PBM (y PGM, PPM)
- XPM

La funcionalidad de guardar imágenes solo soporta un conjunto reducido de formatos. Puede grabar en los siguientes formatos:

- BMP
- TGA
- PNG
- JPEG

La grabación en formatos PNG y JPEG es una funcionalidad nueva de pygame 1.8.

Editar

load

Carga una nueva imagen desde un archivo.

```
pygame.image.load(filename): return Surface
pygame.image.load(fileobj, namehint=""): return Surface
```

Carga una imagen desde un archivo. Puede utilizar tanto un nombre de archivo como un objeto `file` de python.

Pygame determinará de forma automática el tipo de archivo (por ejemplo, gif o bmp) y generará un

nuevo objeto `Surface` con esa información. En algunos casos necesitará conocer la extensión del archivo (por ejemplo las imagenes GIF deberían terminar en ".gif"). Si utiliza un objeto archivo en formato crudo, seguramente necesitará enviar el nombre original del archivo como el argumento `namehint`.

La superficie retornada contendrá el mismo formato de color, colores clave o transparencia alpha que el fichero del que proviene. Generalmente querrá llamar a `Surface.convert()` sin argumentos para crear una copia que se pueda imprimir mas rápido en pantalla.

Para imágenes con transparencia alpha, como en las imágenes .png, use el método `convert_alpha()` luego de cargar la imágen, así la superficie resultante también tendrá transparencia.

Pygame no siempre tendrá soporte para todos los formatos. Como mínimo soportará el formato BMP sin compresión. Si `pygame.image.get_extended()` retorna `True`, usted podría ser capaz de cargar la mayoría de las imágenes, incluyendo png, jpg y gif.

Debería usar `os.path.join()` para otorgar mas compatibilidad.

```
surface = pygame.image.load(os.path.join('data', 'bla.png'))
```

- buscar código donde se use esta función.

Editar

save

Guarda una imagen en el disco.

```
pygame.image.save(Surface, filename): return None
```

Guardará la superficie como una imagen BMP, TGA, PNG o JPEG. Si la extensión del nombre de archivo no se reconoce, se utilizará por defecto .TGA. Tanto los formatos TGA como BMP generan archivos sin compresión.

Guardar archivos PNG y JPEG es una funcionalidad nueva de pygame 1.8.

- buscar código donde se use esta función.

Editar

get_extended

Consulta si los formatos de imagen extendidos se pueden cargar.

```
pygame.image.get_extended(): return bool
```

Si pygame fue construida con los formatos de imagen extendido esta función retornará `True`. Aún así no es posible determinar que formatos estarán disponibles, pero generalmente podrá leerlos todos.

- buscar código donde se use esta función.

Editar

tostring

Transfiere una imagen a una cadena de texto *string*.

```
pygame.image.tostring(Surface, format, flipped=False): return string
```

Genera una cadena que pude transferirse con el método `fromstring` en otros paquetes de imágenes de python. Algunos paquetes de imagen de python prefieren sus imágenes en el formato "de abajo hacia arriba", por ejemplo el paquete PyOpenGL). Se invertirá verticalmente la cadena retorno si envía `True` como valor para el argumento `flipped`.

El argumento `format` es una cadena con uno de los siguiente valores. Note que solo las superficies de 8 bits pueden usar el formato "P". Los otros formatos funcionarán con cualquier superficie. Ademas note que otros paquetes de imágenes de python suportan mas formatos que pygame.

Cadena format	Superficie
P	superficies de 8 bits con paleta.
RGB	imagen de 24 bits.
RGBX	imagen de 32 bits con un espacio sin utilizar.
RGBA	imagen de 32 bits con un canal alpha (transparencia).
ARGB	imagen de 32 bits con canal alpha en primer lugar.
RGBA_PREMULT	imagen de 32 bits bajo la escala del canal alpha.
ARGB_PREMULT	imagen de 32 bits bajo la escala del canal alpha en primer lugar.

- buscar código donde se use esta función.

Editar

fromstring

Genera una nueva superficie desde una cadena.

```
pygame.image.fromstring(string, size, format, flipped=False): return Surface
```

Esta función toma argumentos similares a pygame.image.tostring(). El argumento `size` es una tupla con números que representan el ancho y alto. Puede destruir la cadena original una vez que la nueva superficie se ha creado.

El formato y tamaño de la imagen debe coincidir exactamente con el mismo tamaño de la cadena. Se lanzará una excepción en otro caso.

Consulte el método pygame.image.frombuffer() para ver una forma posiblemente mas rápida de transferir imágenes en pygame.

- buscar código donde se use esta función.

Editar

frombuffer

Genera una nueva superficie que comparte los datos dentro de una cadena.

```
pygame.image.frombuffer(string, size, format): return Surface
```

Genera una nueva superficie que comparte los datos de los *pixeles* directamente desde la cadena.

Esta función toma los mismos argumentos que pygame.image.fromstring(), pero no puede invertir verticalmente los datos de origen.

Funcionará mucho mas rápido que pygame.image.fromstring dado que no se alojan o copian datos de *pixeles*.

- buscar código donde se use esta función.

joystick

Módulo de pygame para interactuar con dispositivos de joystick.

- init
- quit
- get_init
- get_count

En otra pagina:

- Joystick

El módulo joystick gestiona los dispositivos de joystick en una computadora (podría haber mas de uno). Los dispositivos de joystick incluye controles analógicos y tradicionales, y este módulo incluso permite usar múltiples botones y palancas.

Editar

init

Inicializa el módulo de joystick.

```
pygame.joystick.init(): return None
```

Esta función se llama automáticamente desde `pygame.init()`.

Explora el sistema para buscar todos los dispositivos de joystick. El módulo se debe inicializar antes de usar cualquier otra función.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el módulo de joystick.

```
pygame.joystick.quit(): return None
```

Deshabilita el módulo de joystick. Después de llamar a esta función cualquier objeto de joystick dejará de funcionar.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

get_init

Devuelve `True` si el módulo de joystick está inicializado.

```
pygame.joystick.get_init(): return bool
```

Consulta si se ha llamado a la función `pygame.joystick.init()`.

- buscar código donde se use esta función.

Editar

get_count

Cuenta la cantidad de joysticks en el sistema.

```
pygame.joystick.get_count(): return count
```

Retorna el número de dispositivos de joysticks en el sistema. La función retornará 0 si no hay joysticks en el sistema.

Debe usar un número inferior a este cuando genere objetos de joystick usando `Joystick(id)`.

- buscar código donde se use esta función.

Joystick

Genera un nuevo objeto Joystick.

Editar

Joystick

Genera un nuevo objeto Joystick.

```
pygame.joystick.Joystick(id): return Joystick
```

Genera un nuevo objeto Joystick para acceder al dispositivo físico. El argumento `id` debe ser un valor de 0 a `pygame.joystick.get_count() -1`.

Necesita llamar al método init() del Joystick para acceder a la mayoría de los métodos del Joystick. Este método está separado de ello para asegurarse que se inicializa el módulo joystick. El estado y valores para los objetos Joystick se puede compartir cuando se generan múltiples objetos Joystick a partir del mismo dispositivo (por ejemplo, si tienen el mismo identificador `ID`).

El objeto Joystick le permite obtener información acerca de los controles en el dispositivo de Joystick. La cola de eventos comenzará a recibir eventos de esta entrada una vez que el dispositivo esté inicializado.

Puede llamar a las funciones `Joystick.get_name()` y `Joystick.get_id()` sin inicializar el objeto Joystick.

Editar

init

Inicializa el Joystick.

```
Joystick.init(): return None
```

El Joystick se debe inicializar para obtener la mayor información acerca de los controles. Cuando el Joystick se inicializa la cola de eventos de pygame recibirá la entrada de comandos del Joystick.

Es seguro llamar a este método mas de una vez.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el Joystick.

```
Joystick.quit(): return None
```

Este método deshabilita el Joystick. Después de esto, la cola de eventos de pygame no recibirá mas eventos del dispositivo.

Es seguro llamar a esta función mas de una vez.

- buscar código donde se use esta función.

Editar

get_init

Consulta si el Joystick está incializado.

```
Joystick.get_init(): return bool
```

Retorna `True` si ya se ha llamado al método init() en este objeto.

- buscar código donde se use esta función.

Editar

get_id

Obtiene el identificador del Joystick.

```
Joystick.get_id(): return int
```

Retorna en número identificador que representa este dispositivo. Este valor es el mismo que se ha indicado al constructor del Joystick(). Este método se puede llamar de forma segura mientras el Joystick no esté inicializado.

- buscar código donde se use esta función.

Editar

get_name

Obtiene el nombre de sistema del Joystick.

```
Joystick.get_name(): return string
```

Retorna el nombre de sistema para este dispositivo de Joystick. Se desconoce que nombre dará el sistema al Joystick, aunque debería ser un nombre único que identifica al dispositivo. Este método se puede llamar con seguridad mientras el Joystick no esté inicializado.

- buscar código donde se use esta función.

Editar

get_numaxes

Retorna el número de mandos de posición de un joystick.

`Joystick.get_numaxes(): return int`

Retorna el número de mandos de posición de un Joystick. Generalmente será de mandos para la posición (horizontal y vertical). Controles como aletas de avión o frenos se manejan como mandos de posición adicionales.

Los eventos `pygame.JOYAXISMOTION` estarán en el rango de -1.0 a 1.0. Un valor como 0.0 significa que la posición de movimiento está centrada. Los controles tradicionales generarán eventos con valores como -1, 0 o 1 sin valores intermedios. Los joysticks antiguos y analógicos no siempre usarán el rango completo de -1 a 1, y el valor del centro será algún área cerca de 0. Los joysticks analógicos generalmente son imprecisos, lo que generará un montón de eventos de movimiento muy pequeños.

- buscar código donde se use esta función.

Editar

get_axis

Obtiene la posición actual.

`Joystick.get_axis(axis_number): return float`

Retorna la posición actual del control de un joystick. El valor estará en el rengo de -1 a 1, donde un valor como 0 será el centro. Seguramente tendrá que tener en cuenta cierta tolerancia en estos valores para manejar cualquier vibración.

FIX

El parámetro `axis_number` debe ser un número entero desde 0 a `get_numaxes() - 1`.

- buscar código donde se use esta función.

Editar

get_numballs

Obtiene el número de *trackballs* en un Joystick.

`Joystick.get_numballs(): return int`

Retorna el número de dispositivos de *trackballs* en un Joystick. Estos dispositivos funcionan de forma similar a un mouse aunque no tienen posición absoluta, solo tienen cantidades relativas de movimiento.

El evento `pygame.JOYBALLMOTION` se enviará cuando gire el dispositivo de *trackball*. Este evento reportará la cantidad de movimiento del *trackball*.

- buscar código donde se use esta función.

Editar

get_ball

Obtiene la posición relativa del *trackball*.

`Joystick.get_ball(ball_number): return x, y`

Retorna el movimiento relativo de un *trackball* de Joystick. El valor retornado es un par (x, y) almacenando el movimiento relativo desde la última llamada a `get_ball`.

El número indicado en el parámetro `ball_number` debe ser un número entero entre 0 y `get_numballs() -1`.

- buscar código donde se use esta función.

Editar

get_numbuttons

Obtiene el número de botones de un joystick.

`Joystick.get_numbuttons(): return int`

Retorna el número de botones que se pueden pulsar en un joystick. Estos botones tienen un estado booleano (activado o desactivado).

Los botones generan eventos `pygame.JOYBUTTONDOWN` y `pygame.JOYBUTTONDOWN` cuando se pulsan o sueltan.

- buscar código donde se use esta función.

Editar

get_button

Obtiene el estado actual de un botón.

`Joystick.get_button(button): return bool`

Retorna el estado actual de un botón de joystick.

- buscar código donde se use esta función.

Editar

get_numhats

Retorna el número de controles de dirección en un joystick.

`Joystick.get_numhats(): return int`

Retorna el número de direccionales de posición en un joystick. Los direccionales de un joystick son como pequeñas flechas que tienen dos direcciones.

Los eventos `pygame.JOYHATMOTION` se generan cuando los direccionales cambian de posición.

El atributo `position` para el evento contiene un par de valores que pueden ser -1, 0 o 1. Una posición como (0, 0) significa que el direccional está centrado.

- buscar código donde se use esta función.

Editar

get_hat

Obtiene la posición de un direccional del joystick

`Joystick.get_hat(hat_number): return x, y`

Retorna la posición actual de un direccional. La posición se da como dos valores que representan la posición X e Y del direccional. (0, 0) representa el centro. Un valor de -1 significa izquiera o abajo y un valor como 1 significa derecha o arriba. Entonces, (-1, 0) significa izquierda, (1, 0) significa derecha, (0, 1) significa arriba, (1, 1) significa arriba a la derecha etc.

Este valor es entero, cada coordenada puede ser -1, 0 o 1 pero nunca un valor intermedio.

El número de direccional debe ser un número entre 0 y `get_numhats() -1`

- buscar código donde se use esta función.

key

Este modulo contiene funciones para gestionar el dispositivo de teclado.

- get_focused
- get_pressed
- get_mods
- set_mods
- set_repeat
- get_repeat
- name

La cola de eventos obtiene eventos como *pygame.KEYDOWN* y *pygame.KEYUP* cuando se pulsan o sueltan las teclas del teclado respectivamente. Cada evento tiene una atributo llamado key que es un identificador en entero que representa cada tecla del teclado.

El evento *pygame.KEYDOWN* tiene un atributo adicional llamado *unicode*, y otro *scancode*. *unicode* representa un único caracter que es la traducción completa del caracter ingresado por teclado. Teniendo en cuenta las teclas de composición y mayúsculas. *scancode* representa el código de tecla específico de la plataforma. Este código podría ser diferente de un teclado a otro, aunque es útil para la selección de teclas raras como las teclas multimedia.

Existen muchas constantes de teclado que se utilizadan para representar teclas. La siguiente es una lista con todas esas constantes:

Constante	ASCII	Nombre habitual
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 ctrl
K_LCTRL	left ctrl
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

El telcado tambien tiene una lista de estados de combinaciones que pueden ser montados a través de una lógica binaria.

- 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

Editar

get_focused

```
pygame.key.get_focused() -> return bool
```

Esta función devuelve verdadero cuando la ventana de visualización contiene el foco del teclado. Si se necesita que la ventana no pierda el foco del teclado, se puede utilizar pygame.event.set_grab para capturar todas las entradas de teclado.

- buscar código donde se use esta función.

Editar

get_pressed

```
pygame.key.get_pressed() -> return bools
```

Devuelve una secuencia de valores lógicos (booleans) representando el estado de cada tecla en el teclado. Utilice los valores de constante de tecla como índice de la secuencia. Un valor verdadero significa que el botón esta presionado.

Tenga en cuenta que obtener la lista de las teclas pulsadas con esta función no es la forma apropiada de gestionar la entrada de texto por parte del usuario. No hay forma de conocer el orden de las teclas pulsadas, y las pulsaciones muy rápidas de teclas pueden pasar desapercibidas entre dos llamadas a get_pressed. Tampoco hay forma de trasladar las teclas pulsadas a un valor de caracter completamente imprimible.

Vea el evento pygame.KEYDOWN de la cola de eventos para desarrollar esta funcionalidad.

- buscar código donde se use esta función.

Editar

get_mods

```
pygame.key.get_mods() -> return int
```

Devuelve un entero representando una mascara con todos las teclas que estan siendo presionadas. Utilizando lógica binaria puedes verificar si una tecla como shift está pulsada, el estado de capslock y más.

- buscar código donde se use esta función.

Editar

set_mods

```
pygame.key.set_mods(int) -> return None
```

Crea una mascara binaria con las constantes de las teclas que deseas imponer sobre tu programa.

- buscar código donde se use esta función.

Editar

set_repeat

```
pygame.key.set_repeat() -> return None
pygame.key.set_repeat(delay, interval) -> return None
```

Cuando la funcionalidad de repetición de teclas está habilitada en el teclado, las teclas que quedan presionadas generan múltiples eventos de tipo pygame.KEYDOWN. El parrámetro *delay* (tiempo de retraso) es el número de milisegundos transcurridos antes de enviar el primer evento. Luego el resto de los eventos se enviarán en ese intervalo de milisegundos.

Si no especifica argumentos la repetición de teclas quedará deshabilitada.

Cuando se inicializa pygame la repetición de teclas está deshabilitada.

- buscar código donde se use esta función.

Editar

get_repeat

```
pygame.key.get_repeat() -> return (delay, interval)
```

Esta función cumple una tarea similar a set_repeat, solo que nos muestra el intervalo de milisegundos en lugar de definirlo.

- buscar código donde se use esta función.

Editar

name

```
pygame.key.name(key) -> return string
```

Devuelve el nombre descriptivo de la constante del telclado.

- buscar código donde se use esta función.

mask

Módulo de pygame para gestionar máscaras de imágenes.

- from_surface

Otras páginas:

- Mask

Útiles para detectar colisiones entre píxeles perfecta. Una máscara utiliza 1 bit por *píxel* para almacenar qué partes de una imagen pueden colisionar.

Este módulo es nuevo en pygame 1.8.

Editar

from_surface

Retorna una máscara (objeto Mask) para la superficie dada.

```
pygame.mask.from_surface(Surface, threshold = 127) -> Mask
```

Interpreta las partes transparentes de la imagen como no-colisionables y las partes opacas como colisionables.

La transparencia de cada pixel se analiza verificando si es mas grande que el valor indicado por `threshold`.

Si la superficie ha sido procesada mediante `Surface.set_colorkey` entonces el valor indicado por `threshold` no se utilizará.

- buscar código donde se use esta función.

Mask

Objeto de pygame para representar máscaras de bit 2D.


```
pygame.Mask((width, height)): return Mask
```

Editar

get_size

Retorna el tamaño de una máscara.

```
Mask.get_size() -> width,height
```

Editar

get_at

Retorna un valor distinto de 0 si el bit en la posición (x, y) es colisionable.

```
Mask.get_at( (x,y) ) -> int
```

Las coordenadas comienzan en (0, 0), que es la parte superior izquierda, al igual que ocurre con las superficies.

Editar

set_at

Define el valor de colisión de la máscara en la posición `x` e `y` indicada.

```
Mask.set_at( (x,y),value)
```

Editar

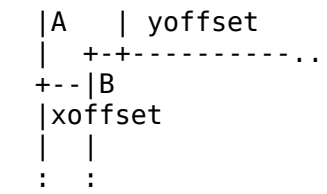
overlap

Retorna el punto de intersección si las máscaras están super-posicionadas por el desplazamiento dato. O bien retorna None si no están super-posicionadas.

```
Mask.overlap(othermask, offset) -> x,y
```

La prueba de superposición utiliza los siguientes desplazamientos (que pueden ser negativos):

```
  +----+----------..
```



Editar

overlap_area

Retorna el número de *pixeles* super-posicionados.

```
Mask.overlap_area(othermask, offset) -> numpixels
```

Puede ver cuantos *pixeles* están super-posicionados con otra máscara dada. Puede ser utilizado para ver en que dirección colisionan las cosas, o para ver en que grado (de cantidad) colisionan las máscaras.

Editar

get_bounding_rects

Retorna una lista de rectángulos delimitadores de las regiones de *pixeles* que definen colisión.

```
Mask.get_bounding_rects() -> Rects
```

Obtiene un rectángulo de regiones conectadas de *pixeles* que definen colisión. El rectángulo de colisión representa los *pixeles* conectados dentro de su región.

mixer

Módulo de pygame para cargar y reproducir sonidos.

Otras páginas:

Este módulo contiene clases para crear objetos `Sound` y controlar la reproducción de audio. El módulo `mixer` es opcional y depende de `SDL_mixer`. Antes de usar este módulo su programa debería verificar si el módulo `pygame.mixer` está disponible.

El módulo `mixer` tiene un número limitado de canales para reproducir sonidos. Generalmente los programas le piden a `pygame` que comience a reproducir un sonido y `pygame` selecciona un canal de audio disponible de forma automática. Por defecto hay 8 canales simultáneos, aunque los programas complejos pueden obtener un control mas preciso de la cantidad de canales y utilizarlos.

Toda la reproducción de sonido se realiza en segundo plano (en hilos diferentes). Cuando comienza a reproducir un objeto `Sound`, esta llamada retornará inmediatamente mientras el sonido continúa sonando. Un objeto `Sound` también se puede reproducir varias veces.

El módulo `mixer` también tiene un canal de *streaming* que se utiliza para reproducir música y se accede a él a través del módulo `pygame.mixer.music`.

El módulo `mixer` se debe inicializar como los otros módulos de pygame, aunque tiene algunas condiciones adicionales. La función `pygame.mixer.init()` toma varios argumentos opcionales para controlar la frecuencia de reproducción. `pygame` colocará por defecto valores razonables, aunque no realizará conversión de frecuencias, por lo tanto `mixer` debería iniciarse para coincidir con la frecuencia y calidad de los recursos de audio.

Nota: Para no tener sonidos con pequeñas demoras, utilice un tamaño de buffer mas pequeño. El valor por defecto se ha seleccionado para reducir la posibilidad de sonidos a rasguños (como en los discos antiguos) en algunos equipos. Puede cambiar el valor por defecto del tamaño de buffer llamando a la función `pygame.mixer.pre_init` antes de llamar a `pygame.mixer.init` o `pygame.init`. Por ejemplo:

```
pygame.mixer.pre_init(44100,-16,2, 1024)
```

Note que el valor por defecto cambió de 1024 a 3072 en pygame 1.8.

Editar

init

Inicializa el módulo mixer.

```
pygame.mixer.init(frequency=22050, size=-16, channels=2, buffer=3072): return
None
```

Inicializar el módulo mixer para cargar y reproducir sonido. Los valores por defecto de los argumentos se pueden reemplazar utilizando atributos de audio específicos. El argumento `size` representa cuantos bits se usarán para cada muestra de audio. Se usarán valores de muestra con signo si se especifica un valor negativo, en otro caso se usarán muestras de audio sin signo.

El argumento `channels` se usa para especificar cuando usar el modo *estéreo* y cuando el modo *mono*. 1 indica *mono* y 2 *estéreo*. No se permiten otros valores.

El argumento `buffer` controla el número de muestras internas que se usarán en el mezclador de sonido. El valor por defecto debería funcionar en la mayoría de los casos. Este valor se puede reducir para disminuir la latencia, aunque podría ocurrir una pérdida de calidad en el sonido. También se puede aumentar para asegurarse que la reproducción nunca se detenga, aunque esto impone latencia. El valor de `buffer` debe ser potencia de dos.

Algunos equipos necesitan que el módulo `pygame.mixer` se inicialice después de los módulos de video. El módulo de orden superior tiene precaución de esto automáticamente, pero no envía ningún argumento especial la iniciar `mixer`. Para resolver esto, el módulo `mixer` tiene una función llamada `pygame.mixer.pre_init()` para definir los valores por defecto antes de llamar a `pygame.init`

Es seguro llamar a esta función mas de una vez, pero después de haber inicializado el módulo `mixer` no puede cambiar los argumentos de reproducción sin antes llamar a `pygame.mixer.init()`

- buscar código donde se use esta función.

Editar

pre_init

Define con anterioridad los argumentos de `mixer.init`.

```
pygame.mixer.pre_init(frequency=0, size=0, channels=0, buffersize=0): return
None
```

Cualquier argumento cambia los valores por defecto que se utilizan cuando se llama a `pygame.mixer.init()`. La mejor forma de personalizar el módulo es llamar a esta función antes de llamar a `pygame.init`.

- buscar código donde se use esta función.

Editar

quit

Deshabilita el módulo `mixer`.

```
pygame.mixer.quit(): return None
```

Deshabilita el módulo `pygame.mixer`. Se detendrá toda la reproducción de audio y los objetos *Sound* cargados podrían ser incompatibles con el mezclador si éste se modifica luego.

Editar

get_init

Consulta si se ha inicializado el módulo `mixer`.

```
pygame.mixer.get_init(): return (frequency, format, channels)
```

Esta función retorna los argumentos de reproducción si el módulo está iniciado. Se retornará `None` si el módulo no se ha iniciado.

Editar

stop

Detiene la reproducción de todos los canales de sonido.

```
pygame.mixer.stop(): return None
```

Detiene la reproducción de todos los canales del mezclador activos.

- buscar código donde se use esta función.

Editar

pause

Detiene la reproducción de forma temporal en todos los canales de sonido.

```
pygame.mixer.pause(): return None
```

Detendrá de forma temporal todas las reproducciones en los canales del mezclador activos. La reproducción de sonido se puede reanudar mas tarde llamando a la función `pygame.mixer.unpause()`.

- buscar código donde se use esta función.

Editar

unpause

Reanuda la reproducción de audio en pausa.

```
pygame.mixer.unpause(): return None
```

Reanudará todos los canales de sonido activos luego de que estos se han puesto en pausa.

- buscar código donde se use esta función.

Editar

fadeout

Reduce el sonido gradualmente de todos los sonidos antes de detenerlos.

`pygame.mixer.fadeout(time): return None`

Deducirá el volumen de todos los canales de sonidos en el tiempo indicado por por el argumento `milliseconds`. El sonido se detendrá una vez que el volumen llegue a su menor valor.

- buscar código donde se use esta función.

Editar

set_num_channels

Define el número total de canales para reproducir.

`pygame.mixer.set_num_channels(count): return None`

Define el número de canales disponibles para el mezclador. El valor por defecto es 8. Puede aumentar o disminuir este valor. Se reduce este valor todos los sonidos que estén sonando en los canales a eliminar se detendrán.

Editar

get_num_channels

Obtiene el número total de canales.

`pygame.mixer.get_num_channels(): return count`

Retorna el número de canales para reproducir activos.

Editar

set_reserved

Reserva canales para que no comiencen a ser utilizados automáticamente.

`pygame.mixer.set_reserved(count): return None`

El módulo `mixer` puede preservar cualquier número de canales para que no se seleccionen automáticamente al reproducir Sonidos. No se detendrán los sonidos que actualmente estén sonando en los canales reservados.

Esto permite a la aplicación reservar un número específico de canales para sonidos importantes que no deberían interrumpirse y tener garantizado un canal para reproducirse.

Editar

find_channel

Encuentra un canal libre.

`pygame.mixer.find_channel(force=False): return Channel`

Encuentra y retorna un objeto `Channel` inactivo. Si no hay canales inactivos retornará `None`. Si no hay canales inactivos y el argumento `foce` vale `True` entonces encontrará el canal que tiene el sonido que mas tiempo se a reproducido y lo retornará.

Si el módulo tiene canales reservados de la función `pygame.mixer.set_reserved()` entonces esos canales no se retornarán aquí.

Editar

get_busy

Consulta si algún sonido a comenzado a reproducirse.

`pygame.mixer.get_busy(): return bool`

Retorna `True` si el módulo está reproduciendo en alguno de los canales. Retornará `False` si el mezclador está libre.

music

Módulo de pygame para controlar la reproducción de música.

El módulo music está muy relacionado con el módulo `pygame.mixer`. Use el módulo music para controlar la reproducción de música en el módulo mixer.

La diferencia entre la reproducción de música y la reproducción de un sonido es que la música se reproduce mientras se carga, y nunca se carga completamente de una vez. El módulo mixer soporta solamente la reproducción de una música a la vez.

Editar

load

Carga un archivo de música para reproducir.

```
pygame.mixer.music.load(filename): return None
```

Cargará el archivo de música y lo preparará para reproducir. Se detendrá cualquier música si se estaba reproduciendo. Esta función no comenzará a reproducir la música.

Tenga en cuenta que la música solo se puede cargar a partir del nombre de un archivo, no se puede cargar desde objetos `file` como las otras funciones para cargar recursos de pygame.

- buscar código donde se use esta función.

Editar

play

Comienza a reproducir un flujo de música.

```
pygame.mixer.music.play(loops=0, start=0.0): return None
```

Reproducirá la música que se ha cargado. Si la música ya estaba sonando entonces se reiniciará.

El argumento `loops` controla el número de veces que se debe reproducir la canción. Por ejemplo, `play(5)` causará que la canción suene una vez y luego se repita 5 veces; es decir, unas 6 veces. Si el argumento `loop` vale -1 se repetirá la reproducción indefinidamente.

El argumento de posición `start` controla a partir de donde comenzará a reproducirse. La posición de inicio depende del formato de música utilizado. Los formatos MP3 y OGG utilizan la posición como tiempo medido en segundos. La música en formato MOD usará `start` como el número de patrón. Si no se puede definir la posición de inicio se lanzará la excepción `NotImplementedError`.

- buscar código donde se use esta función.

Editar

rewind

Reinicia la música.

```
pygame.mixer.music.rewind(): return None
```

Reinicia la reproducción de la música actual para iniciar desde el principio.

- buscar código donde se use esta función.

Editar

stop

Detiene la reproducción de música.

```
pygame.mixer.music.stop(): return None
```

Detiene la reproducción de música si se está reproduciendo.

- buscar código donde se use esta función.

Editar

pause

Detiene de forma temporal la reproducción.

```
pygame.mixer.music.pause(): return None
```

Detiene de forma temporal la reproducción de música. Con la función `pygame.mixer.music.unpause()` puede continuar la reproducción.

- buscar código donde se use esta función.

Editar

unpause

Continúa reproduciendo una canción en pausa.

```
pygame.mixer.music.unpause(): return None
```

Continúa la reproducción de una canción luego de que esta se ha pausado.

- buscar código donde se use esta función.

Editar

fadeout

Detiene la reproducción de música luego de reducir su volumen.

`pygame.mixer.music.fadeout(time): return None`

Detendrá la reproducción de música luego de haber reducido el volumen en el tiempo especificado (en milisegundos).

Note que esta función puede bloquear el programa mientras altera el volumen.

- buscar código donde se use esta función.

Editar

set_volume

Define el volumen de la música.

`pygame.mixer.music.set_volume(value): return None`

Define el volumen de la reproducción de música. El argumento `value` es un número entre 0.0 y 1.0. Se redefine el nivel de volumen cuando se carga una nueva música.

- buscar código donde se use esta función.

Editar

get_volume

Obtiene el volumen de la música.

`pygame.mixer.music.get_volume(): return value`

Retorna el volumen actual para el mezclador. El valor debe estar entre 0.0 y 1.0.

- buscar código donde se use esta función.

Editar

get_busy

Consulta si se está reproduciendo música.

`pygame.mixer.music.get_busy(): return bool`

Retorna `True` cuando se está reproduciendo música. Retornará `False` si el módulo está desocupado.

- buscar código donde se use esta función.

Editar

get_pos

Obtiene el tiempo de reproducción.

```
pygame.mixer.music.get_pos(): return time
```

Obtiene el número de milisegundos desde que el módulo ha comenzado a reproducir música. El tiempo que se retorna solo representa cuanto tiempo a estado reproduciendo música, este valor no tiene en cuenta cualquier desplazamiento de posición inicial.

- buscar código donde se use esta función.

Editar

queue

Pone en cola un archivo de música para seguir al actual.

```
pygame.mixer.music.queue(filename): return None
```

Esta función carga un archivo de música y lo pone en una cola. Un archivo que se encuentra en la cola comenzará a sonar cuando la música actual termine normalmente. La cola de canciones se perderá si la música actual se interrumpe o intercambia.

El siguiente ejemplo reproducirá una canción de *Bach* seis veces y luego reproducirá una de *Mozart*:

```
pygame.mixer.music.load('bach.ogg')
pygame.mixer.music.play(5)               # Reproduce 6 veces, no 5.
pygame.mixer.music.queue('mozart.ogg')
```

- buscar código donde se use esta función.

Editar

set_endevent

Hace que el módulo envíe un evento cuando la reproducción termine.

```
pygame.mixer.music.set_endevent(): return None
pygame.mixer.music.set_endevent(type): return None
```

Esta función hace que pygame emita una señal (con ayuda de la cola de eventos) cuando el módulo termine de reproducir. El argumento determina que tipo de evento se quiere emite.

El evento se emitirá cada vez que una canción termine, no solo la primera vez. Para anular la emisión de eventos llame a esta función sin argumentos.

- buscar código donde se use esta función.

Editar

get_endevent

Obtiene el evento que un canal emite cuando termina de reproducir.

```
pygame.mixer.music.get_endevent(): return type
```

Retorna el tipo de evento que será enviado cada vez que el módulo termine de reproducir. La función retornará `pygame.NOEVENT` si no hay un evento asociado.

- buscar código donde se use esta función.

Channel

Genera un objeto `Channel` (canal de audio) para controlar la reproducción de audio.

Editar

Channel

`pygame.mixer.Channel(id): return Channel`

Retorna un objeto `Channel` para uno de los canales actuales. El argumento `id` debe ser un valor entre 0 y el valor que devuelve `pygame.mixer.get_num_channels()`.

El objeto `Channel` se puede usar para obtener un control preciso sobre la reproducción de sonidos. Un canal solo puede reproducir un objeto `Sound` a la vez. Como pygame por defecto maneja por su cuenta estos objetos, utilizarlos es completamente opcional.

Editar

play

Reproduce un sonido en un canal específico.

`Channel.play(Sound, loops=0, maxtime=0, fade_ms=0): return None`

Comenzará a reproducir un sonido en un canal específico. Se interrumpirá cualquier sonido que esté sonando en este canal.

El argumento `loops` tiene el mismo significado que en `Sound.play()`: es el número de veces que se repetirá el sonido luego de sonar la primera vez. Si vale 3, entonces el sonido se reproducirá 4 veces (la primera y luego 3 veces mas). Si `loops` vale -1, entonces el sonido se repetirá indefinidamente.

Como en `Sound.play()`, el argumento `maxtime` se puede usar para detener la reproducción de sonido luego de un tiempo determinado (indicado en milisegundos).

Al igual que `Sound.play()`, el argumento `fade_ms` se puede usar para alterar progresivamente el volumen de un sonido.

Editar

stop

Detiene la reproducción de sonido en el canal de audio.

```
Channel.stop(): return None
```

Detiene la reproducción de sonido en el canal. Luego de interrumpir la reproducción el canal quedará disponible para que nuevos sonidos puedan sonar en él.

Editar

pause

Detiene temporalmente la reproducción de un canal.

```
Channel.pause(): return None
```

Detiene de forma temporal la reproducción de sonido en un canal. Este sonido se puede reanudar nuevamente mediante la función `Channel.unpause()`.

Editar

unpause

Reanuda la reproducción en un canal.

```
Channel.unpause(): return None
```

Continúa la reproducción en un canal que está en pausa.

Editar

fadeout

Detiene la reproducción luego de reducir el volumen.

```
Channel.fadeout(time): return None
```

Detiene la reproducción de un canal luego de reducir progresivamente el volumen de un sonido en el tiempo indicado por el argumento `time` (en milisegundos).

Editar

set_volume

Define el volumen de un canal de reproducción.

```
Channel.set_volume(value): return None
Channel.set_volume(left, right): return None
```

Define el volumen, o latencia, de un sonido. El nivel de volumen se reinicia cuando un canal comienza a reproducir nuevamente. Esta función solo afecta al sonido actual. El argument `value` debe ser un número entre 0.0 y 1.0

Si se pasa un solo argumento, este se interpretará como el volumen de ambos parlantes. Si se pasan dos argumento y el módulo `mixer` usa el modo estéreo, entonces el primer argumento será el volumen del parlante izquierdo y el segundo argumento será el volumen del parlante derecho. (Si el segundo argumento es None, entonces el primer argumento se interpretará como el volumen de ambos parlantes.)

Si el canal está reproduciendo un sonido en donde se ha llamado a la función `set_volume()`, se tendrán en cuenta las dos llamadas. Por ejemplo:

```
sound = pygame.mixer.Sound("s.wav")
channel = s.play()      # El sonido suena al máximo de volúmen.
sound.set_volume(0.9)   # Ahora al 90% de si volumen.
sound.set_volume(0.6)   # Ahora al 60% (el anterior valor de descarta).
channel.set_volume(0.5) # Ahora al 30% (porque 0.6 * 0.5 = 0.3).
```

Editar

get_volume

Obtiene el volumen del canal de reproducción.

`Channel.get_volume(): return value`

Retorna el volumen del canal para el sonido que es está reproduciendo. Esta función no tiene en cuenta la separación estéreo que se ha utilizado en `Channel.set_volume`. El objeto `Sound` también tiene su propio volumen que se mezcla con el del canal.

Editar

get_busy

Consulta si el canal está activo.

`Channel.get_busy(): return bool`

Retorna `True` si el canal está reproduciendo audio, y retornará `False` si el canal está inactivo.

Editar

get_sound

Obtiene el sonido que se está reproduciendo actualmente.

`Channel.get_sound(): return Sound`

Retorna el objeto `Sound` que se está reproduciendo en este canal. Retornará `None` si el canal está inactivo.

Editar

queue

Coloca un objeto `Sound` en la cola para seguir al actual.

```
Channel.queue(Sound): return None
```

Un objeto `Sound` comienza a reproducirse inmediatamente después de otro si se coloca en la cola de reproducción del canal. Cada canal solo puede tener un sonido en cola al mismo tiempo. El sonido en cola solo se reproducirá si el sonido actual finaliza normalmente. En otro caso, si se llama a `Channel.stop()` o `Channel.play()`, el sonido en cola se cancelará.

El sonido comenzará a reproducirse inmediatamente si no hay otro sonido en curso.

Editar

get_queue

Retorna cualquier objeto `Sound` que esté en cola.

```
Channel.get_queue(): return Sound
```

Se retornará el sonido que esté en cola para este canal. Una vez que el sonido comienza a reproducirse ya no estará en la cola de reproducción.

Editar

set_endevent

Hace que el canal envíe un evento cuando la reproducción finalice.

```
Channel.set_endevent(): return None
Channel.set_endevent(type): return None
```

Cuando se define un evento de terminación para el canal, se enviará un evento a la cola de eventos cada vez que un sonido termine de reproducirse en este canal (no solo la primera vez). Use la función `pygame.event.get()` para recibir el evento de terminación una vez que halla sido enviado.

Note que si ha llamado a `Sound.play(n)` o `Channel.play(sound, n)`, el evento de terminación se enviará una sola vez, luego de reproducirse “n+1” veces (vea la documentación de `Sound.play`).

Se enviará el evento de terminación inmediatamente si se llama a `Channel.stop()` o `Channel.play()` mientras el sonido está en reproducción.

El argumento `type` indica el identificador de evento para enviar a la cola de eventos. Puede ser válido usar cualquier tipo de evento, aunque una buena elección debería ser optar por un valor entre `pygame.locals.USEREVENT` y `pygame.locals.NUMEVENTS`.

El canal dejará de enviar eventos si se llama a esta función sin argumentos.

Editar

get_endevent

Obtiene el evento que un canal emite cuando finaliza la reproducción.

```
Channel.get_endevent(): return type
```

Retorna el tipo de evento que se enviará cada vez que el canal termina de reproducir un objeto `Sound`. Esta función retornará `pygame.NOEVENT` si el canal no tiene asignado un evento para emitir.

Sound

Genera un nuevo objeto Sound a partir de un archivo.

Editar

Sound

```
pygame.mixer.Sound(filename): return Sound
pygame.mixer.Sound(buffer): return Sound
pygame.mixer.Sound(object): return Sound
```

Carga un nuevo sonido a partir de un nombre de archivo, un archivo de python o un objeto de almacenamiento que se pueda leer. Se realizará un ajuste limitado de frecuencia para que coincida con los argumentos de inicialización del módulo `mixer`.

El objeto Sound representa los datos de sonido actual. Los métodos que cambian el estado del objeto de sonido lo harán en todas las instancias de ese objeto.

El sonido puede cargarse desde un archivo de audio OGG o desde un archivo WAV sin compresión.

Nota: La memoria asignada para los datos se copiará internamente, dada de información será compartida entre el archivo y el objeto de sonido.

El constructor `pygame.mixer.Sound(buffer)` es nuevo en pygame 1.8.

Editar

play

Comienza a reproducir el sonido.

```
Sound.play(loops=0, maxtime=0, fade_ms=0): return Channel
```

Comienza a reproducir el sonido en un canal disponible (por ejemplo, en los parlantes de la computadora). Se elegirá de forma forzada el canal, por lo tanto la reproducción se podría detener el sonido en curso si es necesario.

El argumento `loops` controla cuantas veces de deberá repetir el sonido luego de haber sonado por primera vez. Un valor como 5 significa que el sonido será reproducido una vez, y luego se repetirá cinco veces mas, por lo tanto sonará seis veces en total. El valor por defecto (cero) significa que el sonido no se repetirá, y solo sonará una vez. Si `loops` se define a -1 el sonido se repetirá constantemente (aunque podrá llamar a `stop()` para detenerlo).

El argumento `maxtime` se puede usar para detener la reproducción luego del numero de

milisegundos indicado.

El argumento `fade_ms` hará que el sonido comience a reproducirse desde el valor de volumen 0 y aumente de volumen hasta el máximo valor en el tiempo indicado. Note que el sonido podría termina antes de que el aumento de volumen se complete.

Este método retornará un objeto Channel con el canal que ha sido seleccionado.

Editar

stop

Detiene la reproducción de sonido.

```
Sound.stop(): return None
```

Detendrá la reproducción de este sonido en cualquiera de los canales activos.

Editar

fadeout

Detiene el sonido luego de reducir el volumen.

```
Sound.fadeout(time): return None
```

Detendrá la reproducción del sonido luego de reducir el volumen en los milisegundos indicados en el argumento `time`. El sonido se desvanecerá y detendrá en todos los canales de reproducción activos.

Editar

set_volume

Define el volumen de reproducción para el sonido.

```
Sound.set_volume(value): return None
```

Este método definirá el volumen de reproducción para este sonido. Esto afectará inmediatamente al sonido si este se está reproduciendo. También afectará al sonido en sus futuras reproducciones. El argumento `value` es un valor comprendido ente 0.0 y 1.0.

Editar

get_volume

Obtiene el volumen de reproducción.

```
Sound.get_volume(): return value
```

Retorna un valor de 0.0 a 1.0 representando el volumen para este sonido.

Editar

get_num_channels

Retorna cuantas veces se está reproduciendo este sonido.

```
Sound.get_num_channels(): return count
```

Retorna el numero de canales activos donde se está reproduciendo este sonido.

Editar

get_length

Obtiene la longitud del sonido.

```
Sound.get_length(): return seconds
```

Retorna la longitud del sonido en segundos.

Editar

get_buffer

Obtiene un objeto de almacenamiento para modificar el sonido.

```
Sound.get_buffer(): return BufferProxy
```

Retorna un objeto de almacenamiento para modificar el sonido. Este objeto se puede usar para acceso directo y manipulación.

Esta funcionalidad es nueva en pygame 1.8

mouse

Módulo de pygame para utilizar el mouse.

- get_pressed
- get_pos
- get_rel
- set_pos
- set_visible
- get_focused
- set_cursor
- get_cursor

Las funciones de este módulo se pueden usar para obtener el estado actual del dispositivo de mouse. Estas funciones también pueden alterar el cursor del sistema.

Cuando se define el modo de video, la cola de eventos comenzará a recibir eventos de mouse. Los botones del mouse generan los eventos `pygame.MOUSEBUTTONDOWN` y `pygame.MOUSEBUTTONUP`. Estos eventos contienen un atributo `button` que representa el botón que se ha pulsado. La rueda del mouse generará eventos `pygame.MOUSEBUTTONDOWN`, el atributo `button` será 4 si la rueda gira hacia arriba y 5 cuando la rueda gire hacia abajo. Se generará el evento `pygame.MOUSEMOTION` en cualquier momento que se mueva el mouse. El movimiento de mouse se divide en pequeños y precisos eventos `MOUSEMOTION`. Los eventos de movimiento que no se eliminan de la cola de eventos apropiadamente son la razón principal de que la cola de eventos se llene.

El mouse entrará en modo virtual si se oculta el cursor y se define la entrada del mismo como exclusiva de la ventana. Este modo ocasiona que los movimiento relativos no se detendrán en los bordes de la pantalla. Vea las funciones `pygame.mouse.set_visible()` y `pygame.event.set_grab()` para configurarlo así.

Editar

get_pressed

Obtiene el estado de los botones del mouse.

```
pygame.mouse.get_pressed(): return (button1, button2, button3)
```

Retorna una secuencia de valores booleanos representado el estado de todos los botones del mouse. Un valor `True` significa que el botón está pulsado al momento de hacer la llamada a esta función.

Nota: para obtener todos los eventos del mouse es mejor usar `pygame.event.wait()` o bien `pygame.event.get()` y consultar todos esos eventos para ver si estos son `MOUSEBUTTONDOWN`, `MOUSEBUTTONUP` o `MOUSEMOTION`.

Note que en X11 algunos servidores X usan emulación del botón central. Se emitirá un evento del botón 2 cuando pulse al mismo tiempo los botones 1 y 3.

Recuerde llamar a la función `pygame.event.get()` antes de usar esta función. De otra forma no funcionará.

- buscar código donde se use esta función.

Editar

get_pos

Obtiene la posición del cursor de mouse.

`pygame.mouse.get_pos(): return (x, y)`

Retorna la posición X e Y del cursor de mouse. La posición será relativa a la esquina superior izquierda de la pantalla. El cursor podría estar localizado fuera de la ventana, aunque esta función solo retornará la posición si está dentro de la ventana.

- buscar código donde se use esta función.

Editar

get_rel

Obtiene la cantidad de movimiento del mouse.

`pygame.mouse.get_rel(): return (x, y)`

Retorna la cantidad de movimiento en X e Y desde la llamada previa a esta función. El movimiento relativo del cursor de mouse está restringido a los bordes de la pantalla, aunque hay una forma de evitar esto usando un modo de mouse virtual. Este modo virtual se describe al principio de la página.

- buscar código donde se use esta función.

Editar

set_pos

Define la posición del cursor.

`pygame.mouse.set_pos([x, y]): return None`

Define la posición actual del mouse. Si el cursor del mouse está visible entonces *saltará* a la nueva posición inmediatamente. Mover el mouse generará un nuevo evento `pygame.MOUSEMOTION`.

- buscar código donde se use esta función.

Editar

set_visible

Oculta o muestra el cursor del mouse.

`pygame.mouse.set_visible(bool): return bool`

El cursor del mouse será visible si el argumento `bool` es `True`. Esta función retornará el estado de visibilidad anterior del cursor.

- buscar código donde se use esta función.

Editar

get_focused

Consulta si la ventana está recibiendo la entrada del mouse.

`pygame.mouse.get_focused(): return bool`

Retorna `True` cuando pygame está recibiendo los eventos del mouse (o, en la terminología de las ventanas, está *activo* o tiene *foco*).

Esté método es mas útil cuando funciona en una ventana. En cambio, en un modo de pantalla completa, este método siempre retornará `True`.

Note que en sistemas MS Windows, la ventana que tiene el foco del mouse también tendrá el foco del teclado. Pero bajo sistemas el servidor X (GNU/Linux por ejemplo), una ventana podría recibir los eventos del mouse y otra ventana los eventos del teclado. La función `pygame.mouse.get_focused()` indica que ventana de pygame recibe los eventos del mouse.

- buscar código donde se use esta función.

Editar

set_cursor

Define la imagen para el cursor del mouse de sistema.

`pygame.mouse.set_cursor(size, hotspot, xormasks, andmasks): return None`

Cuando el cursor del mouse está visible, se mostrará como un mapa de bits en blanco y negro usando un vector de bits. El argumento `size` es una secuencia que contiene el ancho y alto del cursor, `hotspot` es una secuencia que contiene el punto de control, `xormasks` es una secuencia de bytes que contiene la máscara de datos *xor*. Por último `andmasks` es una secuencia de bytes que contiene los datos de bits del cursor.

El ancho tiene que ser múltiplo de 0, y los vectores deben tener el tamaño correcto para el ancho y alto indicado. De otra forma se lanzará una excepción.

Vea el módulo `pygame.cursor` para obtener ayuda sobre como crear vectores personalizados o por defecto para cursores del sistema.

- buscar código donde se use esta función.

Editar

get_cursor

Obtiene la imagen para el cursor del mouse de sistema.

`pygame.mouse.get_cursor(): return (size, hotspot, xormasks, andmasks)`

Obtiene información acerca del cursor de sistema del mouse. El valor de retorno tiene la misma información que los argumentos indicados en `pygame.mouse.set_cursor()`.

- buscar código donde se use esta función.

movie

Módulo de pygame para reproducir video *mpeg*.

Otras páginas:

- Movie

Pygame puede reproducir audio y video desde archivos de video codificados con MPEG1. La reproducción de video ocurre en segundo plano (en un hilo diferente), lo que hace mas sencillo de manejar la reproducción.

El módulo `pygame.movie` actualmente no funciona en sistemas Windows. Y como no hay forma de tenerlo funcionado ahí, una alternativa es utilizar la biblioteca http://www.pymedia.org.

El audio para el objeto Movie deberá tener control total sobre el sistema de sonido. Esto significa que el módulo `pygame.mixer` deberá estar deshabilitado si se quiere reproducir el audio del video. La solución habitual es llamar a `pygame.mixer.quit()` antes de reproducir el video. El módulo `mixer` se puede iniciar mas tarde cuando el video termine.

La imagen del video se imprimirá en pantalla arriba de todo lo que esté en la ventana. Para imprimir el contenido del video como gráficos normales puede: crear una superficie fuera de la pantalla y definirla como la superficie destino del video. Luego, imprimir esta superficie sobre la pantalla una vez por cuadro.

Movie

Carga un archivo de película MPEG.

Editar

Movie

```
pygame.movie.Movie(filename): return Movie
pygame.movie.Movie(object): return Movie
```

Carga un nuevo flujo de película MPEG a partir de un archivo u objeto tipo archivo de python. El objeto Movie funciona de manera similar a los objetos Sound de pygame.mixer.

Los objetos Movie tienen asignada una superficie destino. La película se dibujará sobre esta superficie en un hilo de segundo plano. Si la superficie de video es la pantalla principal, entonces se intentará utilizar la aceleración de video por hardware. La superficie destino por defecto será la pantalla principal.

Editar

play

Comienza a reproducir una película.

```
Movie.play(loops=0): return None
```

Comienza a reproducir la película, El sonido y el video comenzarán a reproducirse solo si están habilitados. El argumento opcional `loops` controla cuantas veces se repetirá la película. Un valor de -1 para `loops` significa que la película se reproducirá por siempre.

Editar

stop

Detiene la reproducción de la película.

```
Movie.stop(): return None
```

Detiene la reproducción de una película. Tanto la reproducción de sonido como de video se detendrán en la posición actual.

Editar

pause

Detiene de manera temporal, o reanuda, la reproducción de la película.

```
Movie.pause(): return None
```

Esta función detiene o reanuda la reproducción de la película.

Editar

skip

Avanza la posición de reproducción de la película.

```
Movie.skip(seconds): return None
```

Avanza la reproducción de la película en el tiempo especificado. Esta función se puede llamar antes de comenzar a reproducir para definir el tiempo de inicio. Solamente se puede avanzar hacia adelante, no hacia atrás. El argumento puede ser un número real.

Editar

rewind

Reinicia la reproducción.

```
Movie.rewind(): return None
```

Cambia la posición de la película para llevarla al principio. La película comenzará a reproducirse desde el principio.

Se lanzará la excepción `ValueError` si la película no se puede reiniciar. Si esta operación falla el objeto `movie` se considerará inválido.

Editar

render_frame

Define el cuadro de video actual.

```
Movie.render_frame(frame_number): return frame_number
```

Esta función recibe un numero de cuadro para imprimir. Intentará mostrar el cuadro de la película

indicado a la superficie destino. Y retorna el número de cuadro real que se muestra.

Editar

get_frame

Obtiene el cuadro de video actual.

```
Movie.get_frame(): return frame_number
```

Retorna el número de cuadro del video actual.

Editar

get_time

Obtiene el tiempo de reproducción actual.

```
Movie.get_time(): return seconds
```

Retorna el tiempo de reproducción actual como un valor real indicado en segundos. Este método parece estar actualmente con un error y siempre retorna 0.0

Editar

get_busy

Consulta si la película se está reproduciendo.

```
Movie.get_busy(): return bool
```

Retorna `True` si la película se está reproduciendo.

Editar

get_length

Obtiene la duración de la película en segundos.

```
Movie.get_length(): return seconds
```

Retorna la duración de la película en segundos usando un número real.

Editar

get_size

Obtiene la resolución de un video.

```
Movie.get_size(): return (width, height)
```

Obtiene la resolución de un video. La película se ajustará al tamaño de cualquier superficie, aunque esta función reportará el tamaño natural del video.

Editar

has_video

Consulta si el archivo de película contiene video.

```
Movie.get_video(): return bool
```

Retorna `True` cuando el archivo de película que se abrió contiene información de video.

Editar

has_audio

Consulta si el archivo de película contiene video.

```
Movie.get_audio(): return bool
```

Retorna `True` cuando el archivo de película que se abrió contiene información de audio.

Editar

set_volume

Define el volumen de reproducción de audio.

```
Movie.set_volume(value): return None
```

Define el nivel de volumen para esta película. El argumento `value` es un número entre 0.0 y 1.0. Se deshabilitará y no procesará el sonido si se coloca el volumen en 0.

Editar

set_display

Define la superficie de video destino.

```
Movie.set_display(Surface, rect=None): return None
```

Define la superficie destino para el video de la película. También puede indicar un argumento rectángulo para definir la posición, el video se moverá y adaptará al área indicada.

La decodificación de video se deshabilitará si se pasa `None` como superficie destino.

Overlay

Objeto de pygame para gráficos de video overlay


```
pygame.Overlay(format, (width, height)): return Overlay
```

El objeto `Overlay` provee soporte para acceder a superficies overlay de hardware. Los *overlay* de video no usan el formato estándar RGB y pueden usar diferentes resoluciones de datos para crear un sola imagen.

Los objectos `Overlay` representan el acceso a bajo nivel a la pantalla de hardware. Para usar este objeto debe conocer los detalles técnicos de los `overlays` de video.

El formato de `Overlay` determina el tipo de datos que se usaran. No todo el hardware soportará todos los tipos de formatos. Esta es una lista de los tipos de formato disponibles.

- YV12_OVERLAY
- IYUV_OVERLAY
- YUV2_OVERLAY
- UYVY_OVERLAY
- YVYU_OVERLAY

Los argumentos `width` y `height` controlan el tamaño de datos de la imagen `overlay`. Esta imagen se puede mostrar en cualquier tamaño, no solo en el tamaño original.

Los objetos `overlay` no se pueden ocultar, y siempre se verán sobre el contenido de la pantalla.

Editar

display

Define los datos de pixel del `overlay`.

```
Overlay.display( (y, u, v) ): return None
Overlay.display(): return None
```

Muestra los datos `yuv` en los planos `overlay` de SDL. Los argumentos `y`, `u`, y `v` pueden ser cadenas o datos binarios. Los datos deben estar en el formato correcto que se utilizó para crear el objeto.

Si no se indican argumentos, el objeto se dibujará con los datos actuales. Esto puede ser útil cuando el objeto no está acelerado por hardware.

No se validan las cadenas, y cadenas de tamaño inapropiado podrían interrumpir el programa.

Editar

set_location

Controla donde se mostrará el `overlay`.

```
Overlay.set_location(rect): return None
```

Define la posición de la superficie. El `overlay` se mostrará en una posición relativa a la superficie de pantalla principal. Esta operación no dibuja nuevamente el `overlay`, este se actualizará en la siguiente llamada a `Overlay.display()`.

Editar

get_hardware

Consulta si el objeto `Overlay` tiene aceleración por hardware.

```
Overlay.get_hardware(rect): return int
```

Retorna un valor `True` cuando el objeto `Overlay` esté acelerado por hardware. Se utilizará impresión por software si la plataforma no soporta aceleración por hardware.

PixelArray

Objeto de pygame para obtener acceso directo a los pixels de las superficies.

Editar

PixelArray

`pygame.PixelArray(Surface): return PixelArray`

La clase `PixelArray` envuelve una superficie y provee acceso directo a sus pixels, usando una matriz de dos dimensiones. Soporta operaciones como `slicing` (tajadas o rebanadas), manipulación de pixel o fila y asignaciones aunque no se permiten las operaciones de suma, resta, multiplicación o división…

Si bien es posible asignar tanto colores en valor entero como tuplas RGB(A), la clase `PixelArray` utilizará internamente enteros para la representación de elementos. Por ello, la consulta de ciertos colores se debe realizar usando el método `Surface.map_rgb()` de la superficie.

```
pxarray = pygame.PixelArray (surface)
# Check, if the first pixel at the topleft corner is blue
if pxarray[0][0] == surface.map_rgb ((0, 0, 255)):
    ...
```

Los pixels se pueden manipular usando valores enteros o tuplas con las componentes de color.

```
pxarray[x][y] = 0xFF00FF
pxarray[x][y] = (255, 0, 255)
```

Si está trabajando sobre una rebanada (o *slice*) puede usar secuencias arbitrarias u otros objetos `PixelArray` para modificar los pixels. De todas formas tienen que coincidir con el tamaño del objeto.

```
pxarray[a:b] = 0xFF00FF                   # define todos los pixels a 0xFF00FF.
pxarray[a:b] = (0xFF00FF, 0xAACCEE, ... ) # El primer pixel a 0xFF00FF,
                                          # el segundo a 0xAACCEE.
pxarray[a:b] = ( (255, 0, 255), (170, 204, 238), ...) # igual que arriba.
pxarray[a:b] = ( (255, 0, 255), 0xAACCEE, ...)        # igual que arriba.

pxarray[a:b] = otherarray[x:y]            # note que el tamaño de las rebanadas
debe coincidir.
```

Tenga en cuenta que algo como:

```
  pxarray[2:4][3:5] = ...
```

No causará una manipulación rectangular. En lugar de ello primero se reducirá a una matriz de dos columnas, que luego se reducirá en columnas una vez mas, lo que ocasionará una falla de tipo

`IndexError`. Esto se debe al mecanismo de rebanadas (*slicing*) de python y es un comportamiento absolutamente correcto. En cambio, lo que puede hacer es crear una rebanada de una sola columna primero y luego manipularla de la siguiente forma

```
pxarray[2][3:5] = ...
pxarray[3][3:5] = ...
```

También puede usar las habilidades de sub-indices, para realizar una manipulación rectangular o crear una vista de una parte del objeto `PixelArray`. Puede crear una vista diferente de manera simple creando “sub-arrays” mediante los sub-indices.

```
# Genera un nuevo objeto PixelArray otorgando una vista diferente
# de la superficie o matriz original.

newarray = pxarray[2:4,3:5]
otherarray = pxarray[::2,::2]
```

Los sub-indices también se puede usar para realizar manipulaciones de pixel rectangulares en lugar de recorrer los ejes `x` e `y` como se hacía mas arriba.

```
pxarray[::2,:] = (0, 0, 0)            # Convierte a negro cada segunda columna.
```

El objeto `PixelArray` bloquea la superficie mientras existe, por eso tiene que eliminarlo de manera explícita cuando ya no lo use y la superficie debería realizar las operaciones en el mismo momento.

Esta funcionalidad es nueva en pygame 1.8, y los sub-indices son nuevos en pygame 1.8.1

Editar

surface

Obtiene la superficie que utiliza el objeto `PixelArray`.

```
PixelArray.surface: Return Surface
```

Retorna la superficie para la que se creó el objeto `PixelArray`.

Editar

make_surface

Genera una nueva superficie a partir del objeto `PixelArray` actual.

```
PixelArray.make_surface (): Return Surface
```

Genera una nueva superficie a partir del objeto `PixelArray`. Esta superficie puede ser diferente de la original dependiendo del tamaño del objeto, el orden de pixel etc.

```
# Genera una nueva superficie invertida sobre el eje vertical.
sf = pxarray[:,::-1].make_surface ()
```

Esta función es nueva en pygame 1.8.1

Editar

replace

Replaces the passed color in the PixelArray with another one.

```
PixelArray.replace (color, repcolor, distance=0, weights=(0.299, 0.587, 0.114)): Return None
```

Reemplaza los pixels que tienen el color `repcolor` por el color `color`.

Usa una formula de distancia euclidea simple para calcular la distancia entre los colores. Los espacios de distancia van desde 0.0 a 1.0 y se usan como umbral para detectar el color. Esto causa que la operación de reemplazo tome pixels con un color similar, pero no exactamente el mismo.

Esta es una operación *in place*, lo que significa que afecta directamente a los pixels del objeto `PixelArray`.

Esta función es nueva en pygame 1.8.1

Editar

extract

Extrae el color indicado del objeto `PixelArray`.

```
PixelArray.extract (color, distance=0, weights=(0.299, 0.587, 0.114)): Return PixelArray
```

Extrae el color indicado cambiando todos los pixels que coinciden a blanco, mientras que los pixels que no coinciden se cambian a color negro. Esta función retorna un nuevo objeto `PixelArray` con la máscara de color blanco y negro.

Usa una formula de distancia euclidea simple para calcular la distancia entre los colores. Los espacios de distancia van desde 0.0 a 1.0 y se usan como umbral para detectar el color. Esto causa que la operación de reemplazo tome pixels con un color similar, pero no exactamente el mismo.

Esta función es nueva en pygame 1.8.1

Editar

compare

Compara el objeto PixelArray con otro.

```
PixelArray.compare (array, distance=0, weights=(0.299, 0.587, 0.114)): Return PixelArray
```

Compara el contenido del objeto `PixelArray` con otro objecto `PixelArray`. Retorna un nuevo objeto `PixelArray` con la máscara de color blanco y negro que indica las diferencias (en blanco) y similitudes (en negro). Los dos objetos `PixelArray` deben tener dimensiones y cantidad de colores idénticas.

Usa una formula de distancia euclidea simple para calcular la distancia entre los colores. Los espacios de distancia van desde 0.0 a 1.0 y se usan como umbral para detectar el color. Esto causa que la operación de reemplazo tome pixels con un color similar, pero no exactamente el mismo.

Esta función es nueva en pygame 1.8.1.

Rect

Objeto de pygame que se utiliza para almacenar coordenadas rectangulares.

Editar

Rect

```
pygame.Rect(left, top, width, height): return Rect
pygame.Rect((left, top), (width, height)): return Rect
pygame.Rect(object): return Rect
```

Pygame utiliza objetos Rect para almacenar y manipular areas rectangulares. Un objeto Rect se puede crear a partir de una combinación de valores `izquierda`, `arriba`, `ancho` y `alto`. También se pueden crear desde objetos python que ya sean un objeto Rect o tengan un atributo de nombre `rect`.

Cualquier función de pygame que requiera un argumento Rect también acepta cualquiera de eso valores para construir un rectángulo. Esto hace mas sencillo crear objetos Rect en el aire como argumentos a funciones.

Los métodos de Rect que cambian la posición o el tamaño del rectángulo retornan una nueva copia del objeto con los cambios realizados. El rectángulo original no se modifica. Algunos métodos tiene una versión alternativa de estas funcionalidades pero que actúan sobre el objeto mismo y retornan None. Estos métodos alternativos se denotan con el sufijo “ip”.

El objeto Rect tiene varios atributos virtuales que se pueden usar para mover o alinear un rectángulo.

- top, left, bottom, right
- topleft, bottomleft, topright, bottomright
- midtop, midleft, midbottom, midright

- center, centerx, centery
- size, width, height
- w,h

Todos estos atributos se pueden asignar así:

```
rect1.right = 10
rect2.center = (20,30)
```

Asignar un valor a `size`, `width` o `height` cambia las dimensiones del rectángulo, todas las otras asignaciones mueven el rectángulo pero sin alterar el tamaño. Note que algunos atributos son números enteros y otros son pares de números enteros.

Si un rectángulo tiene atributos `width` o `height` distintos de cero, entonces retornarán True por una consulta distinta de cero. Algunos métodos retornan un rectángulo con tamaño 0 para representar un rectángulo inválido.

Las coordenadas para objetos Rect siempre son números enteros. Los valores de tamaño se pueden programar para tener valores negativos, pero estos se consideran no válidos para la mayoría de las operaciones.

Existen varios métodos para consultar colisiones con otros rectángulos. La mayoría de los contenedores de python se pueden utilizar para buscar colisiones entre varios rectángulos contra uno.

El área cubierta por un rectángulo no incluye los *pixeles* que se encuentran en el límite inferior y derecho. Si el borde inferior de un rectángulo es igual al borde superior de otro rectángulo (por ejemplo `rect1.bottom = rect2.top`), los dos ocupan la misma linea en pantalla pero no se superponen, y la consulta a `rect1.colliderect(rect2)` retornará `False`.

Editar

move

Mueve el rectángulo.

```
Rect.move(x, y): return Rect
```

Retorna un nuevo rectángulo que está desplazado en la cantidad indicada. Los argumento `x` e `y` pueden ser cualquier valor entero, positivo o negativo.

- buscar código donde se use esta función.

Editar

move_ip

Mueve el rectángulo afectando al objeto receptor.

```
Rect.move_ip(x, y): return None
```

Similar al método move, pero opera afectando al objeto receptor.

- buscar código donde se use esta función.

Editar

inflate

Aumenta o disminuye el tamaño del rectángulo.

```
Rect.inflate(x, y): return Rect
```

Retorna un nuevo rectángulo con el tamaño alterado en la cantidad indicada. El rectángulo se mantiene centrado en el mismo punto. Los valores negativos reducen el tamaño del rectángulo.

- buscar código donde se use esta función.

Editar

inflate_ip

Aumenta o disminuye el tamaño del rectángulo afectando al objeto receptor.

```
Rect.inflate_ip(x, y): return None
```

Similar al método inflate, pero modifica al objeto receptor.

- buscar código donde se use esta función.

Editar

clamp

Mueve el rectángulo dentro e otro.

```
Rect.clamp(Rect): return Rect
```

Retorna una nuevo rectángulo desplazado para estar completamente dentro de otro rectángulo dado como parámetro. Si el rectángulo es demasiado grande para caber dentro del otro rectángulo, se posicionará en el mismo centro que el rectángulo del argumento, pero su tamaño no se cambiará.

- buscar código donde se use esta función.

Editar

clamp_ip

Mueve el rectángulo dentro e otro.

```
Rect.clamp_ip(Rect): return None
```

Similar al método clamp, pero afecta al objeto receptor.

- buscar código donde se use esta función.

Editar

clip

Recorta un rectángulo dentro de otro.

```
Rect.clip(Rect): return Rect
```

Retorna un nuevo rectángulo que se recorta para estar completamente dentro de otro indicado por parámetro. Si los dos rectángulos no están en colisión, se retornará un rectángulo de tamaño 0.

- buscar código donde se use esta función.

Editar

union

Une dos rectángulo para convertirse en uno.

```
Rect.union(Rect): return Rect
```

Retorna un nuevo rectángulo que cubre completamente el área de otros dos rectángulos dados como parámetro. Pueden haber áreas dentro del nuevo rectángulo que no estén en el área de los originales.

- buscar código donde se use esta función.

Editar

union_ip

Une dos rectángulo para convertirse en uno, afectando al objeto receptor.

```
Rect.union_ip(Rect): return None
```

Similar al método union, pero que afecta al rectángulo receptor.

- buscar código donde se use esta función.

Editar

unionall

La unión de varios rectángulos.

```
Rect.unionall(Rect_sequence): return Rect
```

Retorna la unión de un rectángulo con una secuencia de varios rectángulos.

- buscar código donde se use esta función.

Editar

unionall_ip

La unión de varios rectángulos, afectando al objeto receptor.

```
Rect.unionall_ip(Rect_sequence): return None
```

Similar al método unionall, pero afectando al objeto receptor.

- buscar código donde se use esta función.

Editar

fit

Cambia el tamaño o mueve el rectángulo pero respetando su aspecto.

```
Rect.fit(Rect): return Rect
```

Retorna un nuevo rectángulo desplazado y redimensionado para caber dentro de otro. Se preserva la proporción de aspecto del rectángulo original, por lo tanto el nuevo rectángulo puede ser mas pequeño que el ancho o alto del rectángulo final.

- buscar código donde se use esta función.

Editar

normalize

Corrige los tamaños negativos.

```
Rect.normalize(): return None
```

Invertirá el ancho o alto de un rectángulo si tiene algún tamaño negativo. El rectángulo conservará la misma posición, pero con los lados alternados.

- buscar código donde se use esta función.

Editar

contains

Consulta si un rectángulo está dentro de otro.

```
Rect.contains(Rect): return bool
```

Retorna `True` cuando el argumento está completamente dentro del objeto receptor.

- buscar código donde se use esta función.

Editar

collidepoint

Consulta si un punto está dentro de un rectángulo.

```
Rect.collidepoint(x, y): return bool
Rect.collidepoint( (x,y) ): return bool
```

Retorna `True` si el punto dado está dentro del rectángulo. Un punto sobre el costado derecho o inferior del rectángulo no se considera que está dentro del rectángulo.

- buscar código donde se use esta función.

Editar

colliderect

Consulta si dos rectángulos están en contacto

```
Rect.colliderect(Rect): return bool
```

Retorna `True` si cualquier parte de alguno de los rectángulos están en contacto (excepto los bordes superior+inferior o derecha+izquierda)

- buscar código donde se use esta función.

Editar

collidelist

Consulta si un rectángulo entra en contacto con una lista.

```
Rect.collidelist(list): return index
```

Consulta si el rectángulo colisiona con otro de una secuencia de rectángulos. Se retorna el índice de la primer colisión que se encuentra. Se retorna el índice -1 si no se encuentran colisiones.

- buscar código donde se use esta función.

Editar

collidelistall

Consulta si todos los rectángulos en una lista entran en contacto.

```
Rect.collidelistall(list): return indices
```

Retorna una lista de todos los índices que contienen rectángulos que colisionan con el rectángulo. Se retorna una lista vacía si no se encuentran rectángulos en colisión.

- buscar código donde se use esta función.

Editar

collidedict

Consulta si un rectángulo en un diccionario entra en contacto con el rectángulo.

```
Rect.collidedict(dict): return (key, value)
```

Retorna la clave y el valor del primer elemento de diccionario que colisiona con el rectángulo. Se retorna `None` si no se encuentran colisiones.

Los objetos Rect no se pueden usar como clave en los diccionarios, solamente se pueden utilizar como valores.

- buscar código donde se use esta función.

Editar

collidedictall

Consulta si todos los rectángulos en un diccionario entran en contacto.

```
Rect.collidedictall(dict): return [(key, value), ...]
```

Retorna una lista de todos los parece de clave y valor que colisionan con un rectángulo. Se retorna un diccionario vacío si no se encuentran colisiones.

Los objetos Rect no se pueden usar como clave en los diccionarios, solamente se pueden utilizar como valores.

- buscar código donde se use esta función.

scrap

Módulo de pygame que ofrece soporte para el portapapeles (*clipboard*).

El módulo `scrap` se utiliza para obtener o colocar cosas en el portapapeles. De forma que pueda copiar y pegar datos entre pygame y otro tipo de aplicaciones. Este módulo define algunos tipos de datos básicos:

- SCRAP_PPM
- SCRAP_PBM
- SCRAP_BMP
- SCRAP_TEXT

Estos tipos de datos básicos se pueden colorar en el portapapeles y permiten usar tipos de datos propios. SCRAP_PPM, SCRAP_PBM y SCRAP_BMP son adecuados para compartir datos de superficies gráficas con otras aplicaciones, mientra que SCRAP_TEXT es indicado para intercambiar texto plano.

Los tipos de datos SCRAP_* hacen referencia a los siguientes tipos MIME, y pygame los registra correctamente como los tipos de datos por defecto del sistema operativo:

Tipo pygame	Tipo MIME	Datos
SCRAP_TEXT	text/plain	para texto plano
SCRAP_PBM	image/pbm	para datos de imagen PBM
SCRAP_PPM	image/ppm	para datos de imagen PPM
SCRAP_BMP	image/bmp	para datos de imagen BMP

Dependiendo de la plataforma, cuando los datos se colocan dentro del portapapeles se registran algunos tipos de datos adicionales de forma automática para garantizar un comportamiento de intercambio consistente con otras aplicaciones. Los siguientes tipos indicados en la lista se puede usar como cadenas para enviarse a las respectivas funciones del módulo `pygame.scrap`.

En las plataformas Windows se soportan tipos de datos adicionales de forma automática y se resuelven sus definiciones internas:

- text/plain;charset=utf-8 Para texto en formato UTF-8
- audio/wav Para audio en formato WAV
- image/tiff Para imágenes en formato TIFF

En las plataformas X11 se soportan los siguientes tipos de datos adicionales:

- UTF8_STRING Para texto en formato UTF-8
- text/plain;charset=utf-8 Para texto en formato UTF-8
- COMPOUND_TEXT Para texto COMPOUND

Como se ha comentado anteriormente, usted puede definir sus propios tipos de datos para el portapapeles, de todas formas estos podrían ser inutilizables por otras aplicaciones. Así, la información que se envía al portapapeles usando:

```
pygame.scrap.put ("own_data", data)
```

sólo podría usarse por aplicaciones que consulten el portapapeles buscando el tipo de datos “own_data”.

Este módulo es **experimental**:

Es una funcionalidad nueva en pygame 1.8, solo funciona en Window, X11 y Mac OS X. En Mac OS X solo funciona el intercambio de texto, los otros tipos de datos estarán disponibles en la siguiente versión.

Editar

init

Inicializa el módulo scrap.

```
scrap.init () -> None
```

Intenta inicializar el módulo `scrap` y lanza un excepción si falla. Tenga en cuenta que este módulo necesita tener una superficie de visualización, por lo tanto debe asegurarse de haber adquirido una anteriormente usando la función `pygame.display.set_mode()`'.

Editar

get

Obtiene los datos para el tipo indicado del portapapeles.

```
scrap.get (type) -> string
```

Retorna los datos para el tipo de dato especificado desde el portapapeles. Los datos se retornan como cadenas y podrían necesitar futuros arreglos. Se retornará `None` si no hay datos del tipo de dato indicado.

```
text = pygame.scrap.get (SCRAP_TEXT)

if text:
    # Hacer cosas con el texto
else:
    print "Parace que no hay texto en el portapapeles."
```

Editar

get_types

Obtiene una lista de los tipos de portapapeles disponibles.

```
scrap.get_types () -> list
```

Obtiene una lista de cadenas con los identificadores de los tipos de portapapeles disponibles. Cada identificador se puede usar en el método `scrap.get()` para obtener el contenido del portapapeles en el tipo específico. Se retornará una lista vacía si no hay datos en el portapapeles.

```
types = pygame.scrap.get_types ()
for t in types:
    if "text" in t:
        # Hay cierto contenido con la palabra "text", posiblemente
```

```
        # sea texto, por lo tanto se impimirá:
        print pygame.scrap.get (t)
```

Editar

put

Coloca datos dentro del portapapeles.

```
scrap.put(type, data) -> None
```

Coloca información para un tipo de dato específico en el portapapeles. Los datos deben estar en formato de cadena. El argumento `type` debe ser una cadena que identifica el tipo de dato que se colocará en el portapapeles. Este argumento puede ser uno de los valores nativos como `SCRAP_PBM`, `SCRAP_PPM`, `SCRAP_BMP`, `SCRAP_TEXT` o un identificador de cadena de su tipo de dato personalizado.

Este método lanza una excepción si el contenido indicado no se puede colocar en el portapapeles.

Un ejemplo:

```
fp = open ("example.bmp", "rb")
pygame.scrap.put (SCRAP_BMP, fp.read ())
fp.close ()
# Ahora puede obtener la información desde otras aplicaciones
# usando el portapapeles.

pygame.scrap.put (SCRAP_TEXT, "A text to copy")
pygame.scrap.put ("Plain text", "A text to copy")
```

Editar

contains

Consulta si un cierto tipo de dato se encuentra en el portapapeles.

```
scrap.contains (type) -> bool
```

Retorna `True` si el tipo de dato que se ha enviado está disponible en el portapapeles. En otro caso retornará `False`.

```
if pygame.scrap.contains (SCRAP_TEXT):
    print "Hay texto en el portapapeles."
if pygame.scrap.contains ("own_data_type"):
    print "Existe información personalizada en el portapapeles."
```

Editar

lost

Consulta si el portapapeles está en dominio de la aplicación.

```
scrap.lost() -> bool
```

Retorna `True` si el portapapeles está siendo utilizado por otra aplicación.

```
if pygame.scrap.lost ():
   print "No hay contenido para mi, el portapapeles está siendo utilizado por
alguien mas..."
```

Editar

set_mode

Define el modo de acceso al portapapeles.

```
scrap.set_mode(mode) -> None
```

Define el modo de acceso al portapapeles. Este modo solo es de interés para los entornos X11, donde están disponibles los modos de selección por mouse (SRAP_SELECTION) y el portapapeles (SCRAP_CLIPBOARD). Solicitar el modo `SCRAP_SELECTION` en otros entornos no causará ninguna diferencia.

Se lanzará un excepción `ValueError` si se envía un valor diferente a `SCRAP_CLIPBOARD` o `SCRAP_SELECTION`.

surfarray

Módulo de pygame para acceder a los pixels de una superficie usando una matriz como interfaz.

Funciones para convertir datos de pixel entre matrices y superficies de pygame. Este módulo solo funcionará cuando pygame puede tener acceso a los paquetes externos Numpy o Numeric.

Cada pixel se almacena como un único valor entero para representar los colores rojo, verde y azul. Las imágenes de 8 bits usan un valor de referencia a una paleta de colores. Los pixels que pueden representar mas colores usan un proceso de empaquetamiento de bits para agrupar tres o cuatro valores en un único número.

La matriz se organiza primero por su eje `X` y luego su eje `Y`. Las Matrices que tratan a los pixels como números empaquetados se definen como Matrices de dos dimensiones. Este módulo también puede separar las componentes de color en mas dimensiones; estos tipos de matrices se definen como matrices de tres dimensiones, donde el último indice indica con 0 el componente rojo, con 1 el componente verde y con 2 el azul.

Los sistemas de matriz que se utilizan son:

- numeric
- numpy

Por defecto se utilizará `Numeric`, siempre y cuando esté instalado. En otro caso se usará `numpy` si está instalado. Se lanzará la excepción `ImportError` si ninguno de los dos paquetes está instalado.

El sistema que se utilizará se puede cambiar en tiempo de ejecución usando el método `use_arraytype()`, que necesita como argumento el sistema de matriz a utilizar.

Nota: numpy y Numeric no son completamente compatibles. Algunas manipulaciones de vectores pueden funcionar en un sistema, pero tener un comportamiento diferente o incluso no funcionar en otro.

Además, a diferencia de Numeric, numpy puede usar enteros sin signo de 16 bits. Las imágenes con datos de 16 bits se tratarán como números sin signo. En cambio, Numeric siempre utilizará enteros con signo para la representación, es importante tener esto en mente, ya que usted puede estar utilizando funciones del módulo y sorprenderse por los valores.

El soporte para `numpy` es nuevo en pygame 1.8

array2d

Copia los pixels en una matriz de dos dimensiones.

```
pygame.surfarray.array2d(Surface): return array
```

Copia los pixels desde una superficie a una matriz de dos dimensiones. La profundidad de colores de la superficie controlará el tamaño de los valores enteros, y funcionará para cualquier tipo de formato de pixel.

Esta función bloqueará temporalmente la superficie (vea la función `Surface.lock` para mas detalles).

Editar

pixels2d

Genera una referencia a los pixels en una matriz de dos dimensiones.

```
pygame.surfarray.pixels2d(Surface): return array
```

Genera una nueva matriz de dos dimensiones que referencia directamente a los valores de pixels en una superficie. Cualquier cambio en la matriz afectará a los pixels de la superficie. Esta es una operación rápida ya que no se duplican los datos.

No se pueden crear matrices de referencia para superficies de 24 bits, pero si de cualquier otro tipo de superficies.

La superficie a la que se hace referencia permanecerá bloqueada mientras dure la vida del objeto matriz (vea el método `Surface.lock` para mas detalles).

Editar

array3d

Copia los pixels en una matriz de tres dimensiones.

```
pygame.surfarray.array3d(Surface): return array
```

Copia los pixels desde una superficie a una matriz de tres dimensiones. La profundidad de colores de la superficie controlará el tamaño de los valores enteros, y funcionará para cualquier tipo de formato de pixel.

Esta función bloqueará temporalmente la superficie (vea la función `Surface.lock` para mas detalles).

Editar

pixels3d

Genera una referencia a los pixels en una matriz de tres dimensiones.

```
pygame.surfarray.pixels3d(Surface): return array
```

Genera una nueva matriz de tres dimensiones que referencia directamente a los valores de pixels en una superficie. Cualquier cambio en la matriz afectará a los pixels de la superficie. Esta es una

operación rápida ya que no se duplican los datos.

Esta operación solo funciona en superficies que tienen formatos de 24 o 32 bits. No se puede crear referencias a formatos de pixel mas bajos.

La superficie a la que se hace referencia permanecerá bloqueada mientras dure la vida del objeto matriz (vea el método `Surface.lock` para mas detalles).

Editar

array_alpha

Copia los valores alphas a una matriz de dos dimensiones.

```
pygame.surfarray.array_alpha(Surface): return array
```

Copia los valores alpha de los pixel (el grado de transparencia) desde una superficie a una matriz de dos dimensiones. Esta operación funcionará para cualquier formato de superficie. Las superficies que no tienen valores alpha retornarán una matriz con todos los valores opacos.

Esta función bloqueará temporalmente la superficie (vea la función `Surface.lock` para mas detalles).

Editar

pixels_alpha

Genera una referencia a los pixels alpha en una matriz de dos dimensiones.

```
pygame.surfarray.pixels_alpha(Surface): return array
```

Genera una nueva matriz de dos dimensiones que referencia directamente a los valores alpha de una superficie. Cualquier cambio en la matriz afectará a los pixels de la superficie. Esta es una operación rápida ya que no se duplican los datos.

Solo puede funcionar en superficies de 32 bits con valores alpha por cada pixel.

La superficie a la que se hace referencia permanecerá bloqueada mientras dure la vida del objeto matriz (vea el método `Surface.lock` para mas detalles).

Editar

array_colorkey

Copia los valores clave de una superficie en una matriz de dos dimensiones.

```
pygame.surfarray.array_colorkey(Surface): return array
```

Genera una nueva matriz con los valores de transparencia por color clave de cada pixel. El pixel será completamente transparente si coincide con el valor del color clave; en caso contrario el pixel será completamente opaco.

Esta operación funcionará sobre cualquier tupo de formato de superficie. Si la imagen no tiene color calve se retornará una matriz completamente sólida.

Esta función bloqueará temporalmente la superficie.

Editar

make_surface

Convierte una matriz en una nueva superficie.

`pygame.surfarray.make_surface(array): return Surface`

Genera una nueva superficie que coincide con los datos y el formato en la matriz. La matriz puede ser de dos o tres dimensiones con cualquier cantidad de valores enteros.

Editar

blit_array

Dibuja directamente sobre los valores de una matriz.

`pygame.surfarray.blit_array(Surface, array): return None`

Copia valores directamente desde una matriz sobre una superficie. Esta operación es mas rápida que convertir la matriz en una superficie y luego imprimirla. La matriz debe tener las mismas dimensiones que la superficie y la operación reemplazará complemente todos los valores de los pixels.

Esta función bloqueará temporalmente la superficie como los nuevos valores que se copian.

Editar

map_array

Convierte una matriz de tres dimensiones a una matriz de dos dimensiones.

`pygame.surfarray.map_array(Surface, array3d): return array2d`

Convierte una matriz de tres dimensiones en una matriz de dos dimensiones. Esta función usará el formato de la superficie dada para controlar la conversión. Los formatos de superficie basados en una paleta no están soportados.

Editar

use_arraytype

Define el sistema de matriz que se usará para matrices de superficie.

`pygame.surfarray.use_arraytype (arraytype): return None`

Usa el tipo de matriz indicado para el resto de las funciones del módulo. Los tipos de matriz disponibles son:

- numeric
- numpy

Se lanzará una excepción `ValueError` si el tipo de matriz solicitado no está disponible.

Esta función es nueva a partir de pygame 1.8

Editar

get_arraytype

Obtiene el tipo de matriz utilizado actualmente.

```
pygame.surfarray.get_arraytype (): return str
```

Retorna el tipo de matriz que que se utiliza actualmente. Este valor será uno de los que forman parte de la tupla que devuelve `get_arraytypes()` e indica que sistema de matriz que se usará para la creación de matrices.

Esta función es nueva en pygame 1.8

Editar

get_arraytypes

Obtiene los sistema de matriz actualmente soportadas.

```
pygame.surfarray.get_arraytypes (): return tuple
```

Consulta que tipo de sistemas de vector están disponibles y los retorna como una tupla de cadenas de caracteres. Los valores de la tupla se pueden usar directamente en el método `pygame.surfarray.use_arraytype()`. Si no hay sistema de matriz disponible se podría obtener `None` como retorno.

Esta función es nueva en pygame 1.8.

Surface

Objeto de pygame para representar imágenes.

- Surface
- blit
- convert
- convert_alpha
- copy
- fill
- set_colorkey
- get_colorkey
- set_alpha
- get_alpha
- lock
- unlock
- mustlock
- get_locked
- get_locks
- get_at
- set_at
- get_palette
- get_palette_at
- set_palette
- set_palette_at
- map_rgb
- unmap_rgb
- set_clip
- get_clip
- subsurface
- get_parent
- get_abs_parent
- get_offset
- get_abs_offset
- get_size
- get_width
- get_height
- get_rect
- get_bitsize
- get_bytesize
- get_flags
- get_pitch
- get_masks
- set_masks
- get_shifts
- set_shifts
- get_losses
- get_bounding_rect
- get_buffer

Editar

Surface

```
pygame.Surface( (width, height), flags=0, depth=0, masks=None): return Surface
pygame.Surface( (width, height), flags=0, Surface): return Surface
```

Un objeto Surface de pygame se utiliza para representar cualquier imagen. La superficie tiene un formato de pixel y resolución fija. Las superficies con pixeles de 8 bits usan una paleta de 256 colores.

Invoque `pygame.Surface()` para crear un nuevo objeto image. La superficie será completamente negra. El único argumento requerido es el tamaño. La superficie se creará con el formato que mejor coincida con la pantalla actual si no se especifican los argumentos adicionales.

El formato de pixel se puede controlar especificando la profundidad de colores o una superficie existente. El argumento `flags` es una combinación de características adiciones para la superficie. Puede utilizar cualquier combinación de estas:

- HWSURFACE: Genera la imagen en la memoria de video.
- SRCALPHA: El formato de pixel incluirá transparencias por pixel.

Ambas opciones son solo una solicitud, tal vez no sea posible para todos los modos de video.

Los usuarios avanzados pueden combinar un conjunto de opciones con un valor `depth`. El argumento `masks` es un conjunto de 4 números enteros que especifica cuales bits representan a cada color en el pixel. Las superficies normales no requieren el argumento `mask`.

Las superficies pueden tener varios atributos adicionales como planos alpha, colores clave o recortes. Estas funciones afectan principalmente a la forma en que se imprime la superficie sobre otra. Las rutinas `blit` intentarán usar aceleración de hardware cuando sea posible, en caso contrario usarán métodos de impresión por software muy optimizados.

Existen tres tipos de transparencia en pygame: colores clave, transparencia de superficie, y transparencia de pixel. La transparencia de superficie se puede combinar con colores clave, pero las imágenes con transparencia de pixel no puede usar los otros modos. La transparencia por color clave hace transparente un solo color. No se imprimirán los pixeles que coincidan con el color clave. La transparencia de superficie es un valor individual que cambia la transparencia de la imagen completa. Una transparencia de superficie de 255 será opaca mientras que un valor de 0 será completamente transparente.

La transparencia de pixel es diferente porque se almacena el valor de transparencia de cada pixel. Esto permite crear efectos de transparencia mas precisos, pero es algo mas lento. La transparencia de pixel no se puede mezclar con los otros tipos de transparencia.

Existe soporte para acceder a los pixels de la superficie. El acceso pixels en superficies de hardware es lento y no se recomienda. Estos métodos son adecuados para acceso simple, pero serán considerablemente lentos cuando realice mucho trabajo de pixels con ellos. Si planea realizar mucho trabajo a nivel de pixels se recomienda usar el módulo `pygame.surfarray` que puede tratar a las superficies como vectores de varias dimensiones (y es bastante rápido).

Cualquier función que acceda directamente a los datos de pixels de la superficie necesitarán que la superficie esté bloqueada. Estas funciones pueden bloquear (`block()`) o desbloquear (`unlock()`) las superficies por ellas mismas si ayuda, pero, si habrá una sobrecarga muy grande de múltiples bloqueos o desbloqueos si se llama a esta función muchas veces. Es mejor bloquear manualmente la superficie antes de llamar a las funciones muchas veces, y luego desbloquear la superficie cuando se halla finalizado. Todas las funciones que necesitan bloquear la superficie lo indican en la documentación. Recuerde dejar la superficie bloqueada solo mientras sea necesario.

Los pixels de la superficie se almacenan internamente como un número individual que tiene todos los colores agrupados. Use las funciones `Surface.map_rgb()` y `Surface.unmap_rgb()`

para convertir entre valores individuales (rojo, verde y azul) en colores agrupados para la superficie.

Las superficies también pueden ser una referencia a una sección de otra superficie. Se generan con el método `Surface.subsurface()`. Cualquier cambio en alguna de las dos superficie que verá reflejado en ambas.

Cada superficie contiene un área de recorte. Por defecto el área de recorte cubre la superficie entera. Si esta área de modifica, todas las operaciones de dibujado solo afectarán un área mas pequeña.

Editar

blit

Dibuja una imagen sobre otra.

```
Surface.blit(source, dest, area=None, special_flags = 0): return Rect
```

Dibuja una superficie `source` sobre otra. La impresión se puede posicionar usando el argumento `dest`. `dest` puede ser un par de coordenadas representando la esquina superior izquierda o bien un rectángulo, cuya esquina superior izquierda representará la posición destino de impresión. El tamaño de rectángulo destino no afectará la impresión.

Se puede pasar un rectángulo opcional como argumento `area`. Este representa una porción mas pequeña de la superficie `source` a imprimir.

La opción `special_flags` puede tomar los siguientes valores:

- BLEND_ADD
- BLEND_SUB
- BLEND_MULT
- BLEND_MIN
- BLEND_MAX
- BLEND_RGBA_ADD
- BLEND_RGBA_SUB
- BLEND_RGBA_MULT
- BLEND_RGBA_MIN
- BLEND_RGBA_MAX
- BLEND_RGB_ADD
- BLEND_RGB_SUB
- BLEND_RGB_MULT
- BLEND_RGB_MIN
- BLEND_RGB_MAX

Note, tal vez se agreguen mas opciones de impresión en el futuro.

El rectángulo retornado representa el área de los pixels afectados, excluyendo cualquier pixel fuera de la superficie destino o el área de recorte.

Se ignorarán los pixels alpha cuand se imprima sobre una superficie de 8 bits.

La opción `special_flags` es nueva en pygame 1.8.

Editar

convert

Cambia el formato de pixel de una imagen.

```
Surface.convert(Surface): return Surface
Surface.convert(depth, flags=0): return Surface
Surface.convert(masks, flags=0): return Surface
Surface.convert(): return Surface
```

Genera una nueva copia de la superficie con un formato de pixel modificado. El nuevo formato de pixel se puede determinar a partir de otra superficie existente. Otra posibilidad es especificar los argumentos `depth`, `flags` y `mask`, de manera similar a `pygame.Surface()`

La superficie nueva tendrá el mismo formato de pixel de la pantalla si no envía ningún argumento. Este formato será el mas rápido de imprimir. Es una buena idea convertir todas las superficies antes de imprimirlas varias veces.

La superficie convertida podría no tener pixels alpha, dado que serán eliminados si la superficie original los tenía. Vea la función `Surface.convert_alpha()` para crear o preservar superficies con canal alpha.

Editar

convert_alpha

```
Surface.convert_alpha(Surface): return Surface
Surface.convert_alpha(): return Surface
```

Genera una nueva copia de la superficie con el formato de pixel deseado. La superficie nueva tendrá un formato adecuado para imprimirse mas rápidamente el formato indicado con canal alpha. Si no se especifica el argumento `surface`, la nueva superficie se optimizará para el formato de pantalla actual.

A diferencia del método `Surface.convert()`, el formato de pixel para la imagen nueva podría no ser exactamente el mismo que se pide, aunque se optimizará para imprimirse sobre la superficie destino.

Editar

copy

Genera una nueva copia de la superficie.

```
Surface.copy(): return Surface
```

Hace una copia duplicada de un superficie. La superficie nueva tendrá el mismo formato de pixel, paletas de colores y configuración de transparencia que la original.

Editar

fill

Pinta una superficie con un color solido.

```
Surface.fill(color, rect=None, special_flags=0): return Rect
```

Pinta la superficie con un color solido. Se pintará la superficie entera si no se especifica el argumento `rect`. El argumento `rect` limitará la modificación al área especificada. La operación

de pintado también se limitará por el área de recorte de la superficie.

El argumento `color` puede ser una secuencia `RGB`, `RGBA` o un índice de una paleta de colores. Si usa el formato `RGB` se ignorará el componente alpha (una parte de `RGBA`) a menos que la superficie use transparencia por pixel (atributo `SRCALPHA`).

A partir de pygame 1.8.0 puede usar las siguientes opciones adicionales:

- BLEND_ADD
- BLEND_SUB
- BLEND_MULT
- BLEND_MIN
- BLEND_MAX

Y a partir de pygame 1.8.1 se suman:

- BLEND_RGBA_ADD
- BLEND_RGBA_SUB
- BLEND_RGBA_MULT
- BLEND_RGBA_MIN
- BLEND_RGBA_MAX
- BLEND_RGB_ADD
- BLEND_RGB_SUB
- BLEND_RGB_MULT
- BLEND_RGB_MIN
- BLEND_RGB_MAX

Esta función retornará el área afectada de la superficie.

Editar

set_colorkey

Define el color clave de transparencia.

```
Surface.set_colorkey(Color, flags=0): return None
Surface.set_colorkey(None): return None
```

Define el color clave para la superficie. Cuando imprima esta superficie sobre otra, cualquier pixel que tenga el mismo color que el color clave no se imprimirá. El argumento `color` puede ser un color `RGB` o un indice de una paleta de colores. Si se envía `None` como argumento entonces se deshabilitará el color clave.

Se ignorará el color clave si la superficie tiene un formato para usar valores alpha por pixel. La transparencia por color clave se puede mezclar con la transparencia a nivel de superficie.

Se puede definir el argumento opcional `flags` a `pygame.RLEACCEL` para obtener mejor rendimiento en pantallas que no tengan aceleración de video. Una superficie `RLEACCEL` puede ser mas lenta de modificar, pero se imprimirá mas rápido.

Editar

get_colorkey

Obtiene el color clave de transparencia actual.

```
Surface.get_colorkey(): return RGB or None
```

Retorna el color clave actual de la superficie. Si la superficie no tiene color clave la función retornará `None`.

Editar

set_alpha

Define el valor de transparencia para toda la superficie.

```
Surface.set_alpha(value, flags=0): return None
Surface.set_alpha(None): return None
```

Define el valor de transparencia para la superficie. Cuando se imprima esta superficie sobre otra los pixels se dibujarán ligeramente transparentes. El valor de transparencia es un número entero de 0 a 255, 0 representa completamente transparente y 255 completamente opaco. Se deshabilitará la transparencia de la superficie si se pasa `None` como valor de transparencia.

Esta transparencia es diferente a la transparencia por pixel. Se ignorará este valor de transparencia si la superficie contiene pixels con transparencia. Si la superficie contiene transparencia por pixel, cuando llame a esta función con el argumento `None` se deshabilitará esa transparencia por pixel.

El argumento adicional `flags` se puede definir como `pygame.RLEACCEL` para obtener mayor rendimiento en pantallas que no tengan aceleración de video. Una superficie `RLEACCEL` será mas lenta de modificar, aunque será mas rápido imprimirla sobre otra.

Editar

get_alpha

Obtiene el valor de transparencia de la superficie.

```
Surface.get_alpha(): return int_value or None
```

Retorna el valor de transparencia actual para la superficie. Se retornará `None` si el valor de transparencia no está definido.

Editar

lock

Bloquea la memoria de la superficie para acceder a sus pixels.

```
Surface.lock(): return None
```

Bloquea los datos de pixel de una superficie para acceder a ellos. En la superficies aceleradas, los datos de pixels podrían estar almacenados en memoria de video volátil o en formas no lineales bajo compresión. Cuando se bloquea una superficie la información de pixels se convierte en un formato accesible. El código que lee o escribe valores de pixels necesitará que la superficie se bloquee para realizar esas tareas.

Las superficies no deberían permanecer bloqueadas mas de lo necesario. Una superficie bloqueada podría no mostrarse o ser manipulada por pygame.

No todas las superficies necesitan bloquease. El método `Surface.mustlock()` puede determinar si la superficie requiere bloquease. De todas formas, no hay desventaja al bloquear o

desbloquear una superficie que no lo necesita.

Todas las funciones de pygame bloquearán o desbloquearán automáticamente los datos de la superficie si es necesario. Si una sección de código hace varias llamas para modificar la superficie, entonces se bloqueará y desbloqueará muchas veces la superficie. Por este motivo, es mucho mas útil bloquear la superficie manualmente, luego modificarla muchas veces y luego desbloquearla manualmente.

Es seguro anidar llamas para bloquear y desbloquear. La superficie solo se desbloqueará después de soltar el último bloqueo.

Editar

unlock

Desbloquea la memoria de la superficie del acceso a pixels.

```
Surface.unlock(): return None
```

Desbloquea los datos de pixels de la superficie luego de que ha sido bloqueada. La superficie desbloqueada podrá imprimirse nuevamente por pygame. Vea la documentación de `Surface.lock()` para mas detalles.

Todas las funciones de pygame bloquearán o desbloquearán automáticamente los datos de la superficie si es necesario. Si una sección de código hace varias llamas para modificar la superficie, entonces se bloqueará y desbloqueará muchas veces la superficie. Por este motivo, es mucho mas útil bloquear la superficie manualmente, luego modificarla muchas veces y luego desbloquearla manualmente.

Es seguro anidar llamas para bloquear y desbloquear. La superficie solo se desbloqueará después de soltar el último bloqueo.

Editar

mustlock

Verifica si la superficie necesita bloquearse.

```
Surface.mustlock(): return bool
```

Retorna `True` si la superficie se debe bloquear para acceder a sus datos de pixel. Usualmente las superficies de software pura no necesitan bloquease. Este método no se usa con frecuencia, dado que es seguro y mas rápido directamente bloquear todas las superficies como sea necesario.

Todas las funciones de pygame bloquearán o desbloquearán automáticamente los datos de la superficie si es necesario. Si una sección de código hace varias llamas para modificar la superficie, entonces se bloqueará y desbloqueará muchas veces la superficie. Por este motivo, es mucho mas útil bloquear la superficie manualmente, luego modificarla muchas veces y luego desbloquearla manualmente.

Editar

get_locked

Consulta si la superficie está bloqueada.

```
Surface.get_locked(): return bool
```

Retorna `True` cuando la superficie está bloqueada. Esta función no se fija o preocupa sobre cuantas veces se ha bloqueado la superficie.

Editar

get_locks

Obtiene los bloqueos de la superficie.

```
Surface.get_locks(): return tuple
```

Retorna los bloqueos existentes para la superficie.

Editar

get_at

Obtiene el valor de color de un pixel

```
Surface.get_at( (x, y) ): return Color
```

Retorna el valor de color RGBA en la posición indicada. Si la superficie no tiene transparencia por pixel, entonces el valor alpha del color será siempre 255 (completamente opaco). Se lanzará una excepción `IndexError` si la posición del pixel está fuera del área de la superficie.

Obtener y definir los pixels de a uno a la vez es una tarea generalmente lenta para ser utilizada en un juego o una situación de tiempo real.

Esta función bloqueará y desbloqueará la superficie temporalmente como sea necesario.

Editar

set_at

Define el valor de color para un pixel.

```
Surface.set_at( (x, y), Color): return None
```

Define el valor de color RGBA o entero (si utiliza paleta) de un pixel. Si la superficie no tiene transparencia por pixel, entonces el valor alpha se ignorará. No tendrá ningún efecto definir pixels fuera del área total o el área de recorte de la superficie.

Obtener y definir los pixels de a uno a la vez es una tarea generalmente lenta para ser utilizada en un juego o una situación de tiempo real.

Esta función bloqueará y desbloqueará la superficie temporalmente como sea necesario.

Editar

get_palette

Obtiene la paleta de colores para una superficie de 8 bits.

`Surface.get_palette(): return [RGB, RGB, RGB, ...]`

Retorna una lista de a lo máximo 256 elementos de color que representan los colores utilizados en una superficie de 8bits. La lista que se retorna es una copia de la paleta, y los cambios que se realicen en esta copia no tendrán efecto sobre la superficie.

Editar

get_palette_at

Obtiene el color de una entrada de la paleta de colores.

`Surface.get_palette_at(index): return RGB`

Retorna los valores rojo, verde y azul de un elemento de la paleta de colores de la superficie. El argumento `index` debería ser un valor entre 0 y 255.

Editar

set_palette

Define la paleta de colores para una superficie de 8 bits.

`Surface.set_palette([RGB, RGB, RGB, ...]): return None`

Define la paleta completa para una superficie de 8 bits. Esta función reemplazará los colores de la paleta existente. Se puede enviar una paleta parcial, de modo que solo se reemplazarán los primeros elementos de la misma.

Esta función no hace nada sobre una superficie con mas de 8 bits por pixel.

Editar

set_palette_at

Define el color para un solo elemento en una paleta de colores.

`Surface.set_at(index, RGB): return None`

Define el valor de un color en la paleta de una superficie. El valor del argumento `index` debería estar ente 0 y 255.

Esta función no hace nada sobre una superficie con mas de 8 bits por pixel.

Editar

map_rgb

Convierte un color en un formato empaquetado.

`Surface.map_rgb(Color): return mapped_int`

Convierte un color RGBA en un número entero empaquetado para esta superficie. El número entero retornado no contendrá mas bits que la profundidad de color de la superficie. Estos valores

empaquetados no se usan con frecuencia dentro de pygame, aunque se pueden pasar como argumento a varias funciones que soliciten una superficie y un color.

Vea la documentación del objeto `Surface` para obtener mas información acerca de los colores y los formatos de pixel.

Editar

unmap_rgb

Convierte el valor de un entero empaquetado en un color.

```
Surface.map_rgb(mapped_int): return Color
```

Convierte un color de formato empaquetado en un conjunto de componentes de color RGB para esta superficie. Estos valores empaquetados no se usan con frecuencia dentro de pygame, aunque se pueden pasar como argumento a varias funciones que soliciten una superficie y un color.

Vea la documentación del objeto `Surface` para obtener mas información acerca de los colores y los formatos de pixel.

Editar

set_clip

Define el área de recorte para la superficie.

```
Surface.set_clip(rect): return None
Surface.set_clip(None): return None
```

Cada superficie tiene una área de recorte activa. Este recorte es un rectángulo que representa a los pixels que se pueden modificar en una superficie. Toda la superficie se podrá modificar si se pasa `None` como área de recorte.

El área de recorte está siempre limitada al área de la superficie en sí misma. Si el rectángulo de recorte es mas grande, entonces se encogerá para caber dentro de la superficie.

Editar

get_clip

Obtiene el área de recorte actual de la superficie.

```
Surface.get_clip(): return Rect
```

Retorna una rectángulo que representa el área de recorte. La superficie siempre retornará un rectángulo válido que nunca estará por afuera de los bordes de la superficie. La superficie retornará el área completa de la misma si no se ha definido un área de recorte.

Editar

subsurface

Genera una nueva superficie que hace referencia a su pariente.

```
Surface.subsurface(Rect): return Surface
```

Retorna una nueva superficie que comparte sus pixels con su superficie pariente. La nueva superficie se considera hija de la original. Las modificaciones a los pixels de cualquier de las dos superficies afectará a la otra. La información de la superficie como el área de recorte o los colores clave son únicos para cada superficie.

La nueva superficie heredará la paleta, colores clave y configuración de transparencia de su padre.

Es posible tener cualquier número de *sub-superficies* y *sub-sub-superficies* de una superficie. También es posible tener una *sub-superficie* de la pantalla principal si el modo de video no está acelerado por software.

Vea las funciones *Surface.get_offset()* y *Subsurfaces.get_parent()* para aprender mas acerca del estado de una *sub-superficie*.

Editar

get_parent

Encuentra el padre de una *sub-superficie*.

```
Surface.get_parent(): return Surface
```

Retorna el padre de una *sub-superficie*. Si el receptor no es una *sub-superficie* entonces se retornará None.

Editar

get_abs_parent

Obtiene el padre de mayor nivel de una superficie.

```
Surface.get_abs_parent(): return Surface
```

Retorna el padre de una *sub-superficie*. Se retornará `None` si el receptor no es una *sub-superficie*.

Editar

get_offset

Encuentra la posición la superficie hija dentro del la superficie padre.

```
Surface.get_offset(): return (x, y)
```

Obtiene la posición de desplazamiento de una *sub-superficie* dentro de la superficie padre. Retornará (0, 0) si la superficie no es una *sub-superficie*.

Editar

get_abs_offset

Obtiene la posición absoluta de una superficie hija en relación a su padre de mayor nivel.

```
Surface.get_abs_offset(): return (x, y)
```

Obtiene la posición desplazamiento de una superficie hija en relación a superficie padre de nivel superior. Retornará (0, 0) si la superficie no es una *sub-superficie*.

Editar

get_size

Obtiene las dimensiones de una superficie.

```
Surface.get_size(): return (width, height)
```

Retorna el ancho y alto de una superficie medida en pixels.

Editar

get_width

Obtiene el ancho de una superficie.

```
Surface.get_width(): return width
```

Retorna el ancho de una superficie medida en pixels.

Editar

get_height

Obtiene la altura de una superficie.

```
Surface.get_height(): return height
```

Retorna la altura de una superficies medida en pixels.

Editar

get_rect

Obtiene el área rectangular de una superficie.

```
Surface.get_rect(*\*kwargs): return Rect
```

Retorna un nuevo rectángulo que cubre la superficie entera. Este rectángulo siempre comenzará en la posición (0, 0) y tendrá el ancho y alto idéntico al tamaño de la imagen.

Puede pasar valores clave como argumentos a esta función. Estos argumentos se aplicarán a los atributos del rectángulo antes de ser retornado. Un ejemplo podría ser `mysurf.get_rect(center=(100,100))` para crear un rectángulo de la superficie con centro en la posición (100, 100).

Editar

get_bitsize

Obtiene la profundidad de colores en bits del formato de pixel de la superficie.

```
Surface.get_bitsize(): return int
```

Retorna el número de bits utilizados para representar cada pixel. Este valor podría no coincidir exactamente con el número de bytes usados por pixel. Por ejemplo, una superficie de 15 bits requiere 2 bytes completos.

Editar

get_bytesize

Obtiene el número de bytes utilizados por pixel de la superficie.

```
Surface.get_bytesize(): return int
```

Retorna el número de bytes utilizados por pixel.

Editar

get_flags

Obtiene las opciones adicionales utilizadas por la superficie.

```
Surface.get_flags(): return int
```

Retorna el conjunto de propiedades de la superficie. Cada propiedad es un bit en la máscara de bits `flags`. Las propiedades habituales son `HWSURFACE`, `RLEACCEL`, `SRCALPHA` y `SRCCOLORKEY`.

Esta es una lista mas completa de propiedades. La lista completa de estas propiedades se puede encontrar en el archivo sdl_video.h.

Atributo	Máscara de bits	Descripción
SWSURFACE	0×00000000	La superficie está en la memoria de sistema
HWSURFACE	0×00000001	La superficie está en la memoria de video
ASYNCBLIT	0×00000004	Usa impresión asíncrona cuando sea posible

y algunas disponibles para la función `pygame.display.set_mode()`:

Atributo	Máscara de bits	Descripción
ANYFORMAT	0×10000000	Permite cualquier formato de pixel o color
HWPALETTE	0×20000000	La superficie tiene una paleta exclusiva
DOUBLEBUF	0×40000000	Define un modo de video con Double Buffer
FULLSCREEN	0×80000000	La superficie opera en modo pantalla completa
OPENGL	0×00000002	Genera un contexto de impresión para OpenGL
OPENGLBLIT	0x0000000A	Genera un contexto de impresión 2D para Opengl (en desuso)
RESIZABLE	0×00000010	El modo de video se puede redimensionar
NOFRAME	0×00000020	No usar borde o título para esta ventana

otras utilizadas internamente (de solo lectura)

Atributo	Máscara de	Descripción

	bits	
HWACCEL	0×00000100	La operación de impresión usa aceleración por hardware
SRCCOLORKEY	0×00001000	La operación de impresión usa un color clave para simular transparencia
RLEACCELOK	0×00002000	Atributo privado
RLEACCEL	0×00004000	La superficie está empaquetada con el formato RLE
SRCALPHA	0×00010000	La operación de impresión utiliza la transparencia original
PREALLOC	0×01000000	La superficie utiliza memoria pre-solicitada

Editar

get_pitch

Obtiene el número de bytes utilizados por cada fila de la superficie.

```
Surface.get_pitch(): return int
```

Retorna el número de bytes que separan a cada fila de pixels en una superficie. Las superficies en memoria de video no siempre se almacenan en forma lineal. Las *sub-superficies* también podrían tener un `pitch` mas grande que su verdadera longitud.

Este valor no es necesario para el uso habitual de pygame.

Editar

get_masks

Obtiene la máscara de bits necesaria para convertir entre un color RGB y un color empaquetado.

```
Surface.get_masks(): return (R, G, B, A)
```

Retorna la máscara de bits que se utiliza para empaquetar cada color en un número entero.

Este valor no se necesita para el uso normal de pygame.

Editar

set_masks

Define la máscara de bits necesaria para convertir entre un color RGB en un color en formato empaquetado.

```
Surface.set_masks( (r,g,b,a) ): return None
```

Esta función no se necesita para el uso habitual de pygame.

Es una función nueva a partir de pygame 1.8.1.

Editar

get_shifts

Obtiene los bits de intercambios necesarios para convertir un color RGB en un color en formato

empaquetado.

```
Surface.get_shifts(): return (R, G, B, A)
```

Retorna los bits de intercambio de pixel necesarios para convertir el formato de cada color.

Este valor no se necesita para el uso habitual de pygame.

Editar

set_shifts

Define los bits de intercambios necesarios para convertir un color RGB en un color en formato empaquetado.

```
Surface.get_shifts( (r,g,b,a) ): return None
```

Esta función no se necesita para el uso habitual de pygame.

Esta función es nueva a partir de pygame 1.8.1.

Editar

get_losses

Obtiene los bits significativos que se usan para convertir el formato de color.

```
Surface.get_losses(): return (R, G, B, A)
```

Retorna el número menos significativo de bits agrupados por cada color en un número entero empaquetado.

Este valor no se necesita para el uso habitual de pygame.

Editar

get_bounding_rect

Obtiene el rectángulo mas pequeño que contiene información.

```
Surface.get_bounding_rect(min_alpha = 1): return Rect
```

Retorna la región rectangular mas pequeña que contiene todos los pixels en una superficie que tienen un atributo de transparencia mas grande o igual que le indicado por argumento.

Esta función bloqueará y desbloqueará temporalmente la superficie.

Esta función es nueva en pygame 1.8.

Editar

get_buffer

Obtiene un objeto buffer para los pixels de la superficie.

```
Surface.get_buffer(): return BufferProxy
```

Retorna un objeto buffer para los pixels de la superficie. El buffer se puede usar para acceder y manipular directamente los pixels.

Este método bloquea y desbloquea la superficie de forma implícita. El bloqueo a la superficie se anulará una vez que el objeto `BufferProxy` sea eliminado.

Esta función es nueva en pygame 1.8.

sprite

Módulo de pygame con clases básicas de objetos para juegos.

- spritecollide
- collide_rect
- collide_rect_ratio
- collide_circle
- collide_circle_ratio
- collide_mask
- groupcollide
- spritecollideany

Otras paginas:

- Sprite
- Group
- GroupSingle
- LayeredDirty
- DirtySprite
- LayeredUpdates
- OrderedUpdates
- RenderUpdates

Este módulo contiene varias clases simples para ser utilizadas dentro de los juegos. Hay una clase principal `Sprite` y varias clases `Group` que contienen sprites. El uso de estas clases es completamente opcional cuando se usa pygame. Las clases son bastante livianas y solo proveen un punto de partida para el código que es común en la mayoría de los juegos.

Se espera que la clase `Sprite` se utilice como clase base para los diferentes tipos de objetos en el juego. También hay una clase `Group` básica que simplemente almacena sprites. Un juego podría crear nuevos tipos de clases `Group` que operen sobre instancias de objetos `Sprite` personalizadas.

La clase `Sprite` básica puede dibujar los sprites que contiene sobre una superficie. El método `Group.draw()` requiere que cada que cada `Sprite` tenga los atributos `image` y `rect`. El método `Group.clear()` requiere estos mismos atributos para poder borrar todos los sprites con fondo de pantalla. También hay grupos mas avanzados, por ejemplo `pygame.sprite.RenderUpdates()` y `pygame.sprite.OrderedUpdates()`.

Por último, este módulo contiene varias funciones de colisión. Estas ayudan a encontrar sprites que tienen rectángulos de colisión en contacto.

Los grupos están diseñados para ser muy eficientes al insertar y eliminar sprites de ellos. También permiten pruebas sencillas para ver si un sprite está dentro del grupo. Un determinado `Sprite` puede estar en cualquier número de grupos. Un juego podría usar algunos grupos para controlar la impresión de objetos, y un conjunto separado de grupos para controlar la interacción o el movimiento del personaje. Considere mantener los sprites dentro de grupos organizados en lugar de agregar atributos de tipo a los objetos sprite. Esto le ayudará a encontrarlos mas fácilmente en el juego.

Los sprites y los grupos manejan sus relaciones con los métodos `add()` y `remove()`'. Estos métodos pueden aceptar una o varias instancias de objetos. Los inicializadores para estas clases también aceptan uno o varios objetos para insertar. Es seguro agregar y eliminar el mismo `Sprite` de un grupo.

Mientras es posible diseñar clases de sprite y grupos que no deriven desde las clases `Sprite` y `AbstractGroup` de mas arriba, es extremadamente que herede de ellas cuando agregue una clase de grupo o sprite.

Los sprites no son seguros para operar desde diferentes hilos, por lo tanto debe bloquearlos usted mismo si está usando hilos.

Editar

spritecollide

Encuentra sprites en el grupo que están en contacto con otro sprite.

```
pygame.sprite.spritecollide(sprite, group, dokill, collided = None): return
Sprite_list
```

Retorna una lista que contiene todos los sprites en un grupo que están colisionando con otro sprite. La intersección se determina comparando el atributo `Sprite.rect` de cada sprite.

El argumento `dokill` es un valor booleando. Si vale `True` todos los sprites que colisionan se eliminarán del grupo.

El argumento `collided` es una función que se utiliza para calcular si dos sprites están en contacto, esta función debería tomar dos sprites como agumentos y retornar un valor `True` o `False` indicado si están colisionando. Si no se especifica el valor para el argumento, todos los sprites deberán tener un valor `rect`, que es el rectángulo del área de sprite, que se usará para calcular l colisión.

Funciones de colisión:

- collide_rect
- collide_rect_ratio
- collide_circle
- collide_circle_ratio
- collide_mask

- buscar código donde se use esta función.

Editar

collide_rect

Detección de colisión entre dos sprites, usando rectángulos.

```
pygame.sprite.collide_rect(left, right): return bool
```

Consulta la colisión entre dos sprites. Usa la función `colliderect` del módulo `rect` para calcular la colisión. Está diseñada para ser enviada como una función de colisión a las funciones generales de colisión. Los sprites deben tener atributos `rect`.

Esta función es nueva en pygame 1.8.0

- buscar código donde se use esta función.

Editar

collide_rect_ratio

Detección de colisión entre dos sprites, usando rectángulos reducidos.

`pygame.sprite.collide_rect_ratio(ratio): return collided_callable`

Verifica colisiones entre dos sprites usando usa versión reducida de los rectángulos de sprite.

Se generan con un radio, y la instancia retornada está diseñada para ser enviada como una función de colisión a las funciones generales de colisión.

El argumento `ratio` es un número real, 1.0 indica que será del mismo tamaño, 2.0 es el doble de grande y 0.5 es de la mitad de tamaño.

Esta función es nueva en pygame 1.8.1

- buscar código donde se use esta función.

Editar

collide_circle

Detección de colisión entre dos sprites, usando círculos.

`pygame.sprite.collide_circle(left, right): return bool`

Verifica la colisión entre dos sprites, verificando si dos círculos centrados en los sprites están en contacto. Si el sprite tiene un atributo `radius` este se usará para crear en círculo, en caso de que no exista se creará un círculo lo suficientemente grande para contener todo el rectángulo del sprite indicado por el atributo `rect`. Esta función está diseñada para ser enviada como función de colisión a las funciones generales de colisión. Los sprites deben tener los atributos `rect` y `radius` (este último es opcional).

Esta función es nueva en pygame 1.8.1

- buscar código donde se use esta función.

Editar

collide_circle_ratio

Detección de colisión entre dos sprites, usando una versión reducida del círculo.

`pygame.sprite.collide_circle_ratio(ratio): return collided_callable`

Verifica colisiones entre dos sprites usando usa versión reducida de los círculos de sprite.

Se generan con un radio, y la instancia retornada está diseñada para ser enviada como una función de colisión a las funciones generales de colisión.

El argumento `ratio` es un número real, 1.0 indica que será del mismo tamaño, 2.0 es el doble de grande y 0.5 es de la mitad de tamaño.

El objeto creado verifica la existencia de colisión entre dos sprites, comprobando si los dos círculos con centro en los sprites están en contacto luego de haberlos alterado de tamaño. Lo los sprites tienen un atributo `radius` este se usará para crear el círculo, en otro caso se creará un círculo lo suficientemente grande grande para contener por completo el rectángulo de sprite según su atributo `rect`. Está diseñada para ser enviada como función de colisión a las funciones generales de

colisión. Los sprites deben tener los atributos `rect` y `radius` (este último es opcional).

Esta función es nueva en pygame 1.8.1

- buscar código donde se use esta función.

Editar

collide_mask

Detección de colisión entre dos sprites usando máscaras.

`pygame.sprite.collide_mask(SpriteLeft, SpriteRight): return bool`

Verifica la colisión entre dos sprites, probando si sus máscaras de bits se superponen. Si el sprite tiene un atributo `mask`, este atributo se usará como máscara, en otro caso se creará la máscara a partir de la imagen del sprite. Esta función está diseñada para ser enviada como función de colisión a las funciones generales de colisión. Los sprites deben tener un atributo `rect` y un atributo opcional de nombre `mask`.

Esta función es nueva en pygame 1.8.0

- buscar código donde se use esta función.

Editar

groupcollide

Encuentra todos los sprites que colisionan entre dos grupos.

`pygame.sprite.groupcollide(group1, group2, dokill1, dokill2): return Sprite_dict`

Esta función encontrará intersecciones entre todos los sprites de dos grupos. Las intersecciones se determinan comparando los atributos `Sprite.rect` de cada `Sprite`.

Cada sprite dentro del grupo `group1` se agrega al diccionario de retorno como clave. El valor de cada elemento será una lista de los sprites del grupo `group2` que colisionan con el primero.

Si algunos de los argumentos `dokill` vale `True`, se eliminaran los sprites en colisión de sus respectivos grupos.

- buscar código donde se use esta función.

Editar

spritecollideany

Consulta simple para ver si un sprite colisiona con algún otro en el grupo.

`pygame.sprite.spritecollideany(sprite, group): return bool`

Consulta si el sprite dado colisiona con algún sprite en el grupo. La intersección se determina comparando el atributo `Sprite.rect` de cada sprite.

Esta prueba de colisión puede ser mas rápida que `pygame.sprite.spritecollideany()` dado que tiene menos trabajo para hacer. Retornará al encontrar la primer colisión.

- buscar código donde se use esta función.

Sprite

Una clase base para objetos del juego visibles.

Editar

Sprite

`pygame.sprite.Sprite(*groups): return Sprite`

Es la clase base para los objetos visibles del juego. Las clases derivadas sobre-escribirán el método `Sprite.update()` y asignarán un valor a los atributos `Sprite.image` y `Sprite.rect`. El constructor puede aceptar cualquier número de objetos `Group` a donde se insertará el objeto.

Cuando genera una subclase de Sprite, asegúrese de llamar al constructor de la clase base antes de agregar el *Sprite* a los grupos.

Editar

update

Método para control el comportamiento del sprite.

`Sprite.update(*args):`

La implementación por defecto de este método no hace nada; es solo un *hueco* conveniente que puede sobreescribir. Este método se llama desde `Group.update()` con cualquier argumento que se le envíe.

No hay necesidad de usar este método si no está utilizando el método del mismo nombre en la clase `Group`.

Editar

add

Inserta el sprite a los grupos.

`Sprite.add(*groups): return None`

Se puede pasar cualquier número de grupos como argumentos. El *Sprite* se insertará en los grupos, a menos que ya forme parte de ellos.

Editar

remove

Elimina un sprite de los grupos.

```
Sprite.remove(*groups): return None
```

Se puede pasar cualquier número de grupos como argumentos. El *Sprite* será eliminado de los grupos a los que pertenezca.

Editar

kill

Elimina el *Sprite* de todos los grupos.

```
Sprite.kill(): return None
```

El *Sprite* será eliminado de todos los grupos a los que pertenezca. No se cambiará nada acerca del estado del *Sprite*. Es posible continuar usando el *Sprite* luego de haber llamado a este método, incluyendo agregarlo a otros grupos.

Editar

alive

Consulta si el sprite pertenece a algún grupo.

```
Sprite.alive(): return bool
```

Retorna `True` cuando el *Sprite* pertenece a uno o mas grupos.

Editar

groups

Lista los grupos que contienen este sprite.

```
Sprite.groups(): return group_list
```

Retorna una lista de todos los grupos que contienen a este sprite.

Group

Clase contenedora para varios Sprites.

Editar

Group

`pygame.sprite.Group(*sprites): return Group`

Un contenedor simple para objetos Sprite. Se puede heredar de esta clase para crear contenedores que tengan comportamiento mas específico. El constructor toma cualquier número de sprites para agregar en el grupo. El grupo soporta las siguientes operaciones estándar de python:

Operador python	Descripción
in	Consulta si un Sprite está incluido
len	Cuenta el número de sprites contenidos
bool	Consulta si alguno de los sprites están incluidos
iter	Itera a través de todos los sprites

Los sprites en el grupo no están ordenados, por lo tanto recorrer o dibujar los sprite se realizar sin un orden particular.

Editar

sprites

Retorna una lista de los sprites contenidos en el grupo.

`Group.sprites(): return sprite_list`

Retorna una lista con todos los sprites contenidos en el grupo. También puede obtener un iterador para el grupo, aunque no podrá iterar en el grupo mientras lo modifica.

Editar

copy

Duplica el grupo.

`Group.copy(): return Group`

Genera un nuevo grupo con los mismos sprites que el original. Si ha creado una nueva clase heredando desde Group, el nuevo objeto será de la misma clase que el original. Esto funciona solamente si el constructor de la clase derivada toma los mismos argumentos que la clase Group.

Editar

add

Agrega sprites al grupo.

`Group.add(*sprites): return None`

Agrega cualquier número de sprites a este grupo. Esta función solo agregará sprites que aún no sean miembros del grupo.

Cada argumento `sprite` también puede ser una iterador conteniendo sprites.

Editar

remove

Elimina un sprite del grupo.

`Group.remove(*sprites): return None`

Elimina cualquier número de sprites del grupo. Esta función solo elimina sprites que son miembros actuales del grupo.

Cada argumento `sprite` puede ser un iterador conteniendo Sprites.

Editar

has

Consulta si un grupo contiene sprites.

`Group.has(*sprites): return None`

Retorna `True` si el grupo contiene todos los sprites dados. Esto es similar a utilizar el operador *in* en el grupo (`if sprite in group: …`), que consulta si un sprite individual pertenece al grupo.

Cada argumento `sprite` puede ser un iterador conteniendo Sprites.

Editar

update

Llama al método `update` en los sprites contenidos.

`Group.update(*args): return None`

Llama al método `update())` en todos los sprites incluidos en el grupo. La clase base `Sprite` tiene un método `update` que toma cualquier número de argumentos y no hace nada. Los

argumento que se pasan a `Group.update()` se pasarán a cada Sprite.

No hay forma de obtener el valor de retorno de los métodos `update` de los sprites.

Editar

draw

Dibuja las imágenes de sprites.

```
Group.draw(Surface): return None
```

Dibuja los sprites contenidos sobre el argumento `Surface`. Para ello utiliza el atributo `Sprite.image` para la superficie fuente y `Sprite.rect` para la posición.

El grupo no almacena los sprites en orden, por lo tanto el orden al momento de dibujar es arbitrario.

Editar

clear

Dibuja un fondo sobre los sprites.

```
Group.clear(Surface_dest, background): return None
```

Borra los sprites usados en la última llamada a `Group.draw()`. La superficie destino se limpia pintando con el fondo de pantalla sobre las posición anterior del sprite.

El fondo de pantalla es generalmente una Superficie que tiene las mismas dimensiones que la superficie destino. De todas formas, también puede ser un nombre de función que tome dos argumentos, la superficie destino y un area a limpia. La función `background` se llamará varias veces para limpiar la pantalla.

Este es un ejemplo de una función que limpiará los sprites con un color rojo.

```
def clear_callback(surf, rect):
    color = 255, 0, 0
    surf.fill(color, rect)
```

Editar

empty

Elimina todos los sprites.

```
Group.empty(): return None
```

Elimina todos los sprites de este grupo.

GroupSingle

Grupo contenedor que almacena un solo Sprite.

- groupsingle

Editar

GroupSingle

`pygame.sprite.GroupSingle(sprite=None): return GroupSingle`

El contenedor `GroupSingle` solo almacena un Sprite. Cuando se agrega un nuevo sprite, el anterior se elimina.

Existe una propiedad especial, `GroupSingle.sprite`, que accede al sprite que un grupo contiene. Puede ser None cuando el grupo está vacío. Incluso la propiedad se puede utilizar para asignarle un sprite y que éste se almacene dentro del contenedor GroupSingle.

LayeredDirty

El grupo LayeredDirty está diseñado para objetos DirtySprite, y es subclase de LayeredUpdates.

Editar

LayeredDirty

`pygame.sprite.LayeredDirty(*spites, * *kwargs): return LayeredDirty`

Este grupo requiere objetos `pygame.sprite.DirtySprite` o cualquier tipo de sprite que tenga los siguientes atributos.

- image
- rect
- dirty
- visible
- blendmode (vea la documentación de `DirtySprite`).

Este grupo utiliza una técnica de marcas `dirty`, por lo tanto es mas rápido que los grupos `pygame.sprite.RenderUpdates` si usted tiene varios sprites estáticos. Además cambia automáticamente entre dos modalidades de actualización: pantalla completa o actualización `dirty rectangles`, por lo tanto no tendrá que preocuparse de que tendría que ser mas rápido.

Al igual que en `pygame.sprite.Group`, puede especificar algunos atributos adicionales a través de parámetros.

- _use_update: Puede valer True o False, por defecto vale False.
- _default_layer: layer o capa por defecto en donde se insertarán los sprites que no tienen layer.
- _time_threshold: tiempo de tolerancia para alternar entre el modo de actualización de pantalla completa y `dirty rectangles`, por defecto vale 1000./80 == 1000/fps.

Esta funcionalidad es nueva en pygame 1.8.0

Editar

draw

Dibuja todos los sprites en el orden correcto sobre la superficie indicada.

`LayeredDirty.draw(surface, bgd=None): return Rect_list`

También puede especificar un fondo de pantalla. Si el fondo de pantalla ya está definido entonces el argumento `bgd` no tendrá efecto.

Editar

clear

Se usa para definir un fondo de pantalla.

```
LayeredDirty.clear(surface, bgd): return None
```

Editar

repaint_rect

Dibuja nuevamente el área indicada.

```
LayeredDirty.repaint_rect(screen_rect): return None
```

El argumento `screen_rect` está en coordenadas de pantalla.

Editar

set_clip

Define el área donde se puede dibujar. Para descartar este recorte solo indique `None` como argumento.

```
LayeredDirty.set_clip(screen_rect=None): return None
```

Editar

get_clip

Obtiene el área donde se dibujará, conocida como área de recorte.

```
LayeredDirty.get_clip(): return Rect
```

Editar

change_layer

Cambia la capa o layer de un sprite.

```
change_layer(sprite, new_layer): return None
```

El sprite tendría que haber sido insertado para dibujarse. Esta tarea no se verifica.

Editar

set_timing_treshold

Define la tolerancia en milisegundos.

```
set_timing_treshold(time_ms): return None
```

Por defecto esto vale 1000./80, donde 80 es la cantidad de cuadros por segundo (fps) que se quiere para cambiar a modo de actualización por pantalla completa.

DirtySprite

Una subclase de `Sprite` con mas características y funcionalidad.

- DirtySprite

Editar

DirtySprite

`pygame.sprite.DirtySprite(*groups): return DirtySprite`

Esta es una lista de todos los atributos adicionales de `DirtySprite` con sus valores por defecto.

Editar

dirty = 1

- Si vale 1, el sprite de dibujará y luego el atributo se colocará en 0 otra vez.
- Si vale 2 entonces el sprite siempre se dibujará (en cada cuadro, el indicador no se colocará a cero).
- 0 significa que no representa cambios por lo tanto no se pintará nuevamente.

Editar

blendmode = 0

- Es el argumento `special_flags` de la operación `blit`, `blendmodes`.

Editar

source_rect = None

- rectángulo origen que se usará, recuerde que esta área será relativa a la esquina superior izquierda (0,0) del atributo `self.image`.

Editar

visible = 1

- normalmente vale 1, si se define a 0 no se volverá a dibujar (también debe definir el atributo `dirty` para limpiarlo de la pantalla).

Editar

layer = 0

- Es un valor de solo lectura que se lee cuando se agrega a un grupo de la clase `LayeredRenderGroup`, par mas detalles vea la documentación de `LayeredRenderGroup`.

LayeredUpdates

El grupo LayeredUpdates maneja capas (layers) que dibujan como grupos `OrderedUpdates`.

Editar

LayeredUpdates

`pygame.sprite.LayeredUpdates(*spites, * *kwargs): return LayeredUpdates`

Este grupo es completamente compatible con la clase `pygame.sprite.Sprite`.

Usted puede definir la capa por defecto a través del argumento `kwargs` usando el parámetro de nombre `default_layer` y un número entero para indicar la capa. La capa por defecto es 0.

Si el sprite que se agrega al grupo tiene un atributo de nombre `layer`, entonces se utilizará ese atributo para determinar en que capa se va a agregar el sprite. Si el argumento `kwarg` contiene el parámetro `layer` entonces los sprites se agregarán a esa capa (ignorando el atributo `sprite.layer`). Se utilizará la capa por defecto para insertar los sprites si no se utiliza ninguna de las formas descritas mas arriba.

Esta funcionalidad es nueva en pygame 1.8.0

Editar

add

Agrega un sprite o una secuencia de sprites a un grupo.

`LayeredUpdates.add(*sprites, * *kwargs): return None`

Si el sprite que se agrega al grupo tiene un atributo de nombre `layer`, entonces se utilizará ese atributo para determinar en que capa se va a agregar el sprite. Si el argumento `kwarg` contiene el parámetro `layer` entonces los sprites se agregarán a esa capa (ignorando el atributo `sprite.layer`). Se utilizará la capa por defecto para insertar los sprites si no se utiliza ninguna

de las formas descritas mas arriba.

Editar

sprites

Retorna una lista de sprites ordenados (primero los de atrás, luego los del frente).

```
LayeredUpdates.sprites(): return sprites
```

Editar

draw

Dibuja todos los sprites ordenados sobre la superficie indicada.

```
LayeredUpdates.draw(surface): return Rect_list
```

Editar

get_sprites_at

Retorna una lista de los todos los sprites en esa posición.

```
LayeredUpdates.get_sprites_at(pos): return colliding_sprites
```

Los sprites mas abajo se retornan primero, y por último los que están mas arriba.

Editar

get_sprite

Retorna el sprite que está en el índice indicado por `idx`.

```
LayeredUpdates.get_sprite(idx): return sprite
```

Emite la excepción `IndexOutOfBounds` si el argumento `idx` no está comprendido en el rango.

Editar

remove_sprites_of_layer

Elimina todos los sprites de la capa o layer y los retorna como una lista.

```
LayeredUpdates.remove_sprites_of_layer(layer_nr): return sprites
```

Editar

layers

Retorna una lista de las capas definidas, ordenadas desde mas lejanas a mas cercanas.

```
LayeredUpdates.layers(): return layers
```

Editar

change_layer

Cambia la capa de un sprite.

```
LayeredUpdates.change_layer(sprite, new_layer): return None
```

El sprite se debería haber agregado para dibujar. Esta tarea no se verifica.

Editar

get_layer_of_sprite

Retorna la capa o layer en la que se encuentra el sprite actualmente.

```
LayeredUpdates.get_layer_of_sprite(sprite): return layer
```

Se retornará la capa por defecto si el sprite no se encuenta en el grupo.

Editar

get_top_layer

Retorna la capa superior.

```
LayeredUpdates.get_top_layer(): return layer
```

Editar

get_bottom_layer

Retorna la capa inferior.

```
LayeredUpdates.get_bottom_layer(): return layer
```

Editar

move_to_front

Trae un sprite a la capa superior.

```
LayeredUpdates.move_to_front(sprite): return None
```

Trae el sprite al frente cambiando la capa del sprite a la capa mas alta (agrega el sprite al final de esa capa).

Editar

move_to_back

Mueve el sprite a la capa inferior.

```
LayeredUpdates.move_to_back(sprite): return None
```

Mueve el sprite a la capa inferior, produciendo que el sprite aparezca detrás de los demás y se genere una capa adicional.

Editar

get_top_sprite

Retorna el sprite que se encuentra mas arriba.

```
LayeredUpdates.get_top_sprite(): return Sprite
```

Editar

get_sprites_from_layer

Retorna todos los sprites de una capa, ordenados según se han insertado.

```
LayeredUpdates.get_sprites_from_layer(layer): return sprites
```

Retorna todos los sprites de una capa, ordenados según se han insertado. Este método usa una búsqueda lineal y los sprites no se eliminarán de la capa.

Editar

switch_layer

Intercambia los sprites de la capa `layer1` a la capa `layer2`.

```
LayeredUpdates.switch_layer(layer1_nr, layer2_nr): return None
```

Los números de capa deben existir, aunque esto no se verifica en el método.

OrderedUpdates

Una clase como RenderUpdates que dibuja objetos Sprite en el orden en que fueron insertados.

- OrderedUpdates

Editar

OrderedUpdates

`pygame.sprite.OrderedUpdates(*spites): return OrderedUpdates`

Esta clase deriva de `pygame.sprite.RenderUpdates()`. Mantiene el orden en que se agregaron los objetos Sprite al grupo para imprimirlos. Esto produce que agregar y eliminar objetos del grupo sea un poco mas lento que en los Grupos normales.

RenderUpdates

Clase de grupo que utiliza el procedimiento 'dirty rectangles'.

Editar

RenderUpdates

`pygame.sprite.RenderUpdates(*sprites): return RenderUpdates`

Esta clase deriva desde `pygame.sprite.Group()`. Tiene un método `draw()` extendido para guardar las áreas modificadas de la pantalla.

Editar

draw

Dibuja las imágenes de objetos Sprite y guarda información de las áreas modificadas.

`RenderUpdates.draw(surface): return Rect_list`

Dibuja todos los objetos Sprite en la superficie, al igual que `Group.draw()`. Aunque este método retorna una lista de áreas rectangulares de la pantalla que han sido modificadas. Los cambios devueltos incluyen áreas de la pantalla que han sido afectadas por llamadas previas a `Group.clear`

La lista de objetos Rect devuelta se podría utilizar para llamar a `pygame.display.update`. Esto ayudará a mejorar el rendimiento del programa en modos de video gestionados por software. Este tipo de actualización solamente es útil en pantallas sin fondos animados.

sndarray

Módulo de pygame para acceder a muestras de sonido.

Contiene funciones para convertir desde objetos de sonido a vectores de Numeric o numpy. Este módulo estará disponible solo cuando pygame pueda utilizar los paquetes Numeric o numpy.

Los datos de sonido se componen de miles de muestras por segundo, y cada muestra es la amplitud de la onda en un momento particular del tiempo. Por ejemplo, en el formato 22-kHz, el elemento número 5 del vector es la amplitud de onda luego de 5/22000 segundos.

Cada muestra es un número de 8 o 16 bits, dependiendo del formato. Un archivo de sonido estéreo tiene dos valores por muestra, mientras que un archivo de sonido mono tiene solo uno.

Los sistemas de vector soportados son:

- numeric
- numpy

Por defecto se elije Numeric, si está instalado. En otro caso, se usará numpy (si está instalado). Si ninguno de los dos está instalado, el módulo notificará una excepción *ImportError*.

El sistema de vector se puede cambiar en tiempo de ejecución usando el método `use_arraytype()`, que requiere uno los sistemas especificados mas arriba como cadena.

Nota: numpy y Numeric no son completamente compatibles. Algunas manipulaciones de vectores pueden funcionar en un sistema, pero tener un comportamiento diferente o incluso no funcionar en otro.

Además, a diferencia de Numeric, numpy puede usar enteros sin signo de 16 bits. Si la muestra representa dados de 16 bits los sonidos con datos de 16 bits se tratarán como números sin signo. En cambio, Numeric siempre utilizará enteros con signo para la representación, es importante tener esto en mente, ya que usted puede estar utilizando funciones del módulo y sorprenderse por los valores.

Editar

array

Copia muestras de sonido en un vector.

```
pygame.sndarray.array(Sound): return array
```

Genera un nuevo vector para los datos de sonido y copia las muestras. El vector siempre tendrá el mismo formato retornado por `pygame.mixer.get_init()`

Editar

make_sound

Convierte un vector en un objeto Sound.

`pygame.sndarray.make_sound(array): return Sound`

Genera un nuevo objeto Sound reproducible desde un vector. Se debe inicializar el módulo mixer y el formato del vector debe ser similar al formato de audio de mixer.

Editar

use_arraytype

Define el sistema de vector que se debe utilizar para los vectores de sonido.

`pygame.sndarray.use_arraytype (arraytype): return None`

Utiliza el sistema de vector solicitado para las funciones del módulo. Actualmente los sistemas soportados son:

- numeric
- numpy

Se notificará la excepción *ValueError* si no está disponible el sistema solicitado.

Esta funcionalidad es nueva en pygame 1.8

Editar

get_arraytype

Obtiene el sistema de vector disponible.

`pygame.sndarray.get_arraytype (): return str`

Retorna el sistema de vector actualmente activo. Este será un valor de la tupla de `get_arraytypes()` e indica que sistema de vector se está utilizando para la creación de vectores.

Esta funcionalidad es nueva en pygame 1.8

Editar

get_arraytypes

Obtiene los sistemas de vector actualmente soportados.

`pygame.sndarray.get_arraytypes (): return tuple`

Verifica que sistemas de vector están disponibles y los retorna como una tupla de cadenas. Los valores de la tupla se pueden usar directamente en la función `pygame.sndarray.use_arraytype()`. Se retornará `None` si no se encuentra un sistema de vector soportado.

Esta funcionalidad es nueva en pygame 1.8

time

Módulo de pygame para gestionar tiempo.

- get_ticks
- wait
- delay
- set_timer

En otra pagina:

- Clock

El tiempo se representa en milisegundos, lo cual es equivalente a Segundos*1000. (Por lo que 2500 milisegundos son 2.5 segundos).

La mayoría de las plataformas tienen una resolución de tiempo limitada a 10 milisegundos (aproximadamente.)

Editar

get_ticks

```
pygame.time.get_ticks() -> int
```

Indica el tiempo en milisegundos desde que se llamó a *pygame.init()*. Siempre devuelve 0 antes de que pygame.init() se llame.

- buscar código donde se use esta función.

Editar

wait

```
pygame.time.wait(milisegundos) -> int
```

Hace una pausa. Está función duerme el programa para compartir la CPU con otros procesos. Ésta función es menos exacta que la función delay.

Devuelve el número actual de milisegundos usados.

- buscar código donde se use esta función.

Editar

delay

```
pygame.time.delay(milliseconds) -> int
```

Hace una pausa durante el número de milisegundos especificados. Esta función usa la CPU para realizar el retardo más exacto que wait.

La función devuelve el número actual de milisegundos usados.

- buscar código donde se use esta función.

Editar

set_timer

```
pygame.time.set_timer(event_id, milliseconds) -> None
```

Define un tipo de evento para que aparezca en la cola de eventos a intervalos de tiempo regulares. El primer evento no aparecerá a menos que la cantidad de tiempo especificado halla concluido.

Cada tipo de evento puede tener un temporizador separado asociado a él. Para definir el valor de *event_id* es mejor usar un valor entre *pygame.USEREVENT* y *pygame.NUMEVENTS*.

Para deshabilitar el temporizador para un *event* defina el argumento *milliseconds* a 0.

- buscar código donde se use esta función.

Clock

Objeto que gestiona tiempo.

Editar

Clock()

`pygame.time.Clock(): return Clock`

Crea un nuevo objeto Clock que puede ser usado para gestionar tiempo. El objeto Clock también provee varias funciones para controlar la velocidad (o *framerate*) de un juego.

- buscar código donde se use esta función.

Editar

Clock.tick

`Clock.tick(framerate=0): return milliseconds`

Este método debería llamarse una vez por actualización. Calcula internamente cuantos milisegundos han transcurrido desde la llamada anterior.

Si especifica el argumento *framerate* la función esperará el tiempo necesario para mantener al juego corriendo a la velocidad solicitada. Puede usarse para limitar la velocidad de ejecución del juego. Llamando al método con el valor 40 (*Clock.tick(40)*) una vez por actualización, el programa nunca funcionará a más de 40 cuadros por segundo de velocidad.

Note que esta función utiliza la función *SDL_Delay* que no es muy precisa en todas las plataformas, pero no usa mucho CPU. Use el método Clock.tick_busy_loop si desea un temporizador preciso.

- buscar código donde se use esta función.

Editar

Clock.tick_busy_loop

`Clock.tick_busy_loop(framerate=0): return milliseconds`

Este método debería llamarse una vez por actualización. Calcula internamente cuantos milisegundos han transcurrido desde la llamada anterior.

Si especifica el argumento *framerate* la función esperará el tiempo necesario para mantener al juego corriendo a la velocidad solicitada. Puede usarse para limitar la velocidad de ejecución del juego. Llamando al método con el valor 40 (*Clock.tick(40)*) una vez por actualización, el programa nunca funcionará a más de 40 cuadros por segundo de velocidad.

Note que esta función utiliza la función delay, que utiliza un montón de CPU en un bucle para asegurarse que el cálculo de tiempo es mas preciso.

- buscar código donde se use esta función.

Editar

Clock.get_time

```
Clock.get_time(): return milliseconds
```

Retorna el valor del argumento enviado al anterior llamado de Clock.tick. Este número representa el número de milisegundos transcurridos entre las dos anteriores llamadas a Clock.tick.

- buscar código donde se use esta función.

Editar

Clock.get_rawtime

```
Clock.get_rawtime(): return milliseconds
```

Similar a Clock.get_time, pero este no incluye ningún milisegundo utilizado mientras Clock.tick estaba esperando para limitar la velocidad de juego.

- buscar código donde se use esta función.

Editar

Clock.get_fps

```
Clock.get_fps(): return float
```

Calcula el rendimiento de tu juego (medido en cuadros por segundo). Este valor se calcula promediando las ultimas llamadas a Clock.tick.

- buscar código donde se use esta función.

transform

Módulo de pygame para transformar superficies.

Una transformación de superficie es una operación que mueve dimensiona los píxeles. Todos estas funciones toman una superficie sobre la que operan y devuelven una nueva superficie con los resultados.

Alguna de las transformaciones se consideran destructivas. Esto significa que cada vez que se realizan se pueden perder *pixeles* de datos. Los ejemplos comunes de esto son las operaciones de escalado y rotación. Por ese motivo, es mejor transformar nuevamente la imagen original que seguir transformando un imagen muchas veces; por ejemplo, imagine que está animando un resorte que salta, que se expande y contrae con el movimiento. Si aplica los cambios de tamaño incrementalmente a las imágenes anteriores perderá detalle; en lugar de eso siempre comience con la imagen original y cambie el tamaño de la misma al tamaño deseado.

Editar

flip

Invierte la imagen de manera horizontal y vertical.

```
pygame.transform.flip(Surface, xbool, ybool): return Surface
```

Puede invertir la superficie ya sea de forma vertical, horizontal, o ambas. Invertir una superficies es una tarea no destructiva y retorna una nueva superficie con el mismo tamaño.

- buscar código donde se use esta función.

Editar

scale

Altera el tamaño.

```
pygame.transform.scale(Surface, (width, height), DestSurface = None): return
Surface
```

Redimensiona una superficie a un nuevo tamaño. Esta es una operación rápida que no suaviza los resultados.

Se puede usar una superficie destino opcional, en lugar de tener que crear una nueva. Esto hace mas rápida la operación si tiene que cambiar el tamaño de algo muchas veces. De todas formas, la superficie destino debe tener el mismo tamaño que los parámetros `width` y `height` indicados. Además las superficie destino debe tener el mismo formato.

- buscar código donde se use esta función.

Editar

rotate

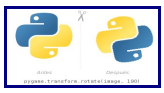
Aplica rotación a una imagen.

`pygame.transform.rotate(Surface, angle): return Surface`

Aplica una rotación inversa al sentido de las agujas del reloj sin suavizar. El argumento `angle` se representa en grados y puede ser cualquier número real. La magnitud negativa de `angle` rotará la imagen en sentido a las agujas del reloj.

A menos que las rotaciones se incrementen de a 90 grados, la imagen se rellenará para almacenarse en un nuevo tamaño. Si la imagen tiene *pixels* transparentes, el área de relleno será transparente. En otro caso pygame tomará un color que coincida con el valor clave de la imagen o el valor de la esquina superior izquierda.

- buscar código donde se use esta función.



Editar

rotozoom

Rotación y cambio de tamaño con filtro.

`pygame.transform.rotozoom(Surface, angle, scale): return Surface`

Es una transformación combinada de rotación y cambio de tamaño. La superficie resultado será una superficie suavizada de 32 bits. El argumento `scale` es un número real que será multiplicado por la resolución actual. El argumento `angle` es un numero real que representa la rotación contra las agujas del reloj mediada en grados. Un ángulo de rotación negativo girará la superficie en sentido a las agujas del reloj.

- buscar código donde se use esta función.

Editar

scale2x

Duplica el tamaño de la imagen de manera especial.

`pygame.transform.scale2x(Surface, DestSurface = None): Surface`

Retorna una nueva imagen que será del doble de tamaño que la original. Utiliza el algoritmo *AdvanceMAME Scale2X* que produce escalado de gráficos evitando el dentado o los cuadrados grandes.

Se puede usar una superficie destino opcional, en lugar de tener que crear una nueva. Esto hace mas rápida la operación si tiene que cambiar el tamaño de algo muchas veces. De todas formas, la superficie destino debe tener el mismo tamaño que los parámetros `width` y `height` indicados. Además las superficie destino debe tener el mismo formato.

- buscar código donde se use esta función.

Editar

smoothscale

Cambia el tamaño de una superficie con suavidad y de forma arbitraria.

```
pygame.transform.smoothscale(Surface, (width, height), DestSurface = None):
return Surface
```

Usa uno de dos algoritmos diferentes para alterar cada una de las dimensiones de la superficie. Al reducir la imagen, los *pixeles* destinos serán promedio de los *pixeles* originales que representan. Al aumentar la imagen, se utilizará un filtro bilineal. En las arquitecturas *amd64* y *i686*, se incluyen las rutinas MMX optimizadas que funcionarán mucho mas rápido que en otro tipo de equipos.

El tamaño es una secuencia de dos números que representa (`width`, `height`). Esta función solo opera con superficies de 24 o 32 bits. Se lanzará una excepción si la profundidad de color de la superficie original es menor a 24 bits.

`New in pygame 1.8`

Esta función es nueva en pygame 1.8.

- buscar código donde se use esta función.

Editar

chop

Obtiene una copia de una imagen con un área interior eliminada.

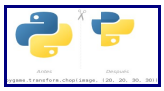
`pygame.transform.chop(Surface, rect): return Surface`

Extrae una porción de una imagen. Todos los *pixels* dentro del rectángulo dado se eliminarán. Las áreas diagonales (diagonales al rectángulo) se juntarán. La imagen original no se alterará por esta

operación.

Note: Si usted quiere cortar, esta función retorna la parte de la imagen sin el área del rectángulo; en su lugar puede imprimir con el rectángulo a una nueva superficie o copiar una *subsurface*.

- buscar código donde se use esta función.



Para ilustrar el ejemplo, imagine que define un rectángulo como se muestra en la siguiente imágen. La función `chop` eliminará tanto el rectángulo como la zona roja, creando una nueva superficie con los restos unidos.

Editar

laplacian

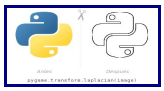
Busca los bordes en una superficie.

```
pygame.transform.laplacian(Surface, DestSurface = None): return Surface
```

Busca los bordes en una superficie usando el algoritmo laplacian.

Esta función es nueva en pygame 1.8.

- buscar código donde se use esta función.



Editar

average_surfaces

Busca la superficie mas similar a partir de muchas otras.

```
pygame.transform.average_surfaces(Surfaces, DestSurface = None): return Surface
```

Toma una secuencia de superficies y retorna la superficie con los colores mas parecidos a una dada.

Esta función es nueva en pygame 1.8.

- buscar código donde se use esta función.

Editar

threshold

Encuentra cuales y cuantos *pixels* en una superficie están dentro del umbral de un color.

```
pygame.transform.threshold(DestSurface, Surface, color, threshold = (0,0,0,0),
diff_color = (0,0,0,0), change_return = True, Surface =None): return
num_threshold_pixels
```

Puede definir la superficie destino donde todos los *pixeles* se cambian a `diff_color` cuando no están dentro del umbral `threshold`.

O se puede usar solo para contar los números de *pixeles* dentro del umbral si define `change_return` a `False`.

Cuando se define la tercera superficie, se usarán los colores de ella en lugar del tercer argumento `color`.

Puede definir un umbral, `threshold`, como `(r,g,b,a)`, donde `r`, `g`, `b` pueden ser diferentes umbrales. De forma que si quiere puede usar un umbral `r` de 40 y un umbral `b` de 2.

Esta función es nueva en pygame 1.8.

- buscar código donde se use esta función.

version

Módulo que contiene información de la versión.

- ver
- vernum

Este módulo se incorpora automáticamente dentro del paquete pygame y ofrece unas variables para verificar qué versión de pygame tiene el sistema.

Editar

ver

Número de versión como una cadena.

```
>>> pygame.version.ver
'1.8.1release'
```

Es la versión de pygame representada como una cadena. Puede contener un número de versión específico, por ejemplo “1.5.2”.

- buscar código donde se use esta función.

Editar

vernum

Número de versión en formato de enteros en una tupla.

```
>>> pygame.version.vernum
(1, 8, 1)
```

Esta variable de la versión se puede comparar fácilmente con otros números de versiones del mismo formato. Un ejemplo de verificación de versiones de pygame podría ser:

```
if pygame.version.vernum < (1, 5):
    print 'Cuidado, tiene una version muy antigua de pygame (%s)' %
pygame.version.ver
    disable_advanced_features = True
```

- buscar código donde se use esta función.