pygame documentation

search

Most useful stuff:
Color | display | draw | event | font | image | key | locals | mixer | mouse | Rect | Surface | time | music | pygame

Advanced stuff: cursors | joystick | mask | sprite | transform | BufferProxy | freetype | gfxdraw | midi | PixelArray | pixelcopy | sndarray | surfarray | math

Other: camera | controller | examples | fastevent | scrap | tests | touch | version

pygame.mixer.music

pygame module for controlling streamed audio

pygame.mixer.music.load
Load a music file for playback

pygame.mixer.music.unload
Unload the currently loaded music to free up resources

pygame.mixer.music.play
Start the playback of the music stream

pygame.mixer.music.rewind
restart music

pygame.mixer.music.stop
stop the music playback

pygame.mixer.music.pause
temporarily stop music playback

pygame.mixer.music.unpause
resume paused music

pygame.mixer.music.fadeout

pygame.mixer.music.get_volume
get the music volume

pygame.mixer.music.get_busy
check if the music stream is playing

pygame.mixer.music.set_pos
set position to play from

pygame.mixer.music.get_pos
get the music play time

pygame.mixer.music.queue
queue a sound file to follow the current

pygame.mixer.music.set_endevent
have the music send an event when playback stops

pygame.mixer.music.get_endevent
get the event a channel sends when playback stops

The music module is closely tied to **pygame.mixer**. Use the music module to control the playback of music in the sound mixer.

The difference between the music playback and regular Sound playback is that the music is streamed, and never actually loaded all at once. The mixer system only supports a single music stream at once.

On older pygame versions, `MP3` support was limited under Mac and Linux. This changed in pygame `v2.0.2` which got improved MP3 support. Consider using `OGG` file format for music as that can give slightly better compression than MP3 in most cases.

`pygame.mixer.music.load()`

Load a music file for playback

`load(filename) → None`

the music playing.

If you are loading from a file object, the namehint parameter can be used to specify the type of music data in the object. For example: `load(fileobj, "ogg")`.

Changed in pygame 2.0.2: Added optional `namehint` argument

Search examples for pygame.mixer.music.load

pygame.mixer.music.**unload**()

Unload the currently loaded music to free up resources

`unload() → None`

This closes resources like files for any music that may be loaded.

New in pygame 2.0.0.

Search examples for pygame.mixer.music.unload

pygame.mixer.music.**play**()

Start the playback of the music stream

`play(loops=0, start=0.0, fade_ms=0) → None`

This will play the loaded music stream. If the music is already playing it will be restarted.

`loops` is an optional integer argument, which is `0` by default, which indicates how many times to repeat the music. The music repeats indefinitely if this argument is set to `-1`.

`start` is an optional float argument, which is `0.0` by default, which denotes the position in time from which the music starts playing. The starting position depends on the format of the music played. `MP3` and `OGG` use the position as time in seconds. For `MP3` files the start time position selected may not be accurate as things like variable bit rate encoding and ID3 tags can throw off the timing calculations. For `MOD`

denotes the period of time (in milliseconds) over which the music will fade up from volume level `0.0` to full volume (or the volume level previously set by **set_volume()**). The sample may end before the fade-in is complete. If the music is already streaming `fade_ms` is ignored.

Changed in pygame 2.0.0: Added optional `fade_ms` argument

Search examples for pygame.mixer.music.play

pygame.mixer.music.**rewind**()

restart music

`rewind() → None`

Resets playback of the current music to the beginning. If **pause()** has previously been used to pause the music, the music will remain paused.

Note: **rewind()** supports a limited number of file types and notably `WAV` files are NOT supported. For unsupported file types use **play()** which will restart the music that's already playing (note that this will start the music playing again even if previously paused).

Search examples for pygame.mixer.music.rewind

pygame.mixer.music.**stop**()

stop the music playback

`stop() → None`

Stops the music playback if it is currently playing. endevent will be triggered, if set. It won't unload the music.

Search examples for pygame.mixer.music.stop

pygame.mixer.music.**pause**()

temporarily stop music playback

`pause() → None`

pygame.mixer.music.**unpause**()

resume paused music

unpause() → None

This will resume the playback of a music stream after it has been paused.

Search examples for pygame.mixer.music.unpause

pygame.mixer.music.**fadeout**()

stop music playback after fading out

fadeout(time) → None

Fade out and stop the currently playing music.

The time argument denotes the integer milliseconds for which the fading effect is generated.

Note, that this function blocks until the music has faded out. Calls to **fadeout()** and **set_volume()** will have no effect during this time. If an event was set using **set_endevent()** it will be called after the music has faded.

Search examples for pygame.mixer.music.fadeout

pygame.mixer.music.**set_volume**()

set the music volume

set_volume(volume) → None

Set the volume of the music playback.

The volume argument is a float between 0.0 and 1.0 that sets the volume level. When new music is loaded the volume is reset to full volume. If volume is a negative value it will be ignored and the volume will remain set at the current level. If the volume argument is greater than 1.0, the volume will be set to 1.0.

get the music volume

`get_volume() → value`

Returns the current volume for the mixer. The value will be between `0.0` and `1.0`.

Search examples for pygame.mixer.music.get_volume

`pygame.mixer.music.`**`get_busy()`**

check if the music stream is playing

`get_busy() → bool`

Returns True when the music stream is actively playing. When the music is idle this returns False. In pygame 2.0.1 and above this function returns False when the music is paused. In pygame 1 it returns True when the music is paused.

Changed in pygame 2.0.1: Returns False when music paused.

Search examples for pygame.mixer.music.get_busy

`pygame.mixer.music.`**`set_pos()`**

set position to play from

`set_pos(pos) → None`

This sets the position in the music file where playback will start. The meaning of "pos", a float (or a number that can be converted to a float), depends on the music format.

For `MOD` files, pos is the integer pattern number in the module. For `OGG` it is the absolute position, in seconds, from the beginning of the sound. For `MP3` files, it is the relative position, in seconds, from the current position. For absolute positioning in an `MP3` file, first call **`rewind()`**.

Other file formats are unsupported. Newer versions of SDL_mixer have better positioning support than earlier ones. An SDLError is raised if a particular format does not support positioning.

pygame.mixer.music.**get_pos**()

get the music play time

`get_pos() → time`

This gets the number of milliseconds that the music has been playing for. The returned time only represents how long the music has been playing; it does not take into account any starting position offsets.

pygame.mixer.music.**queue**()

queue a sound file to follow the current

`queue(filename) → None`

`queue(fileobj, namehint="", loops=0) → None`

This will load a sound file and queue it. A queued sound file will begin as soon as the current sound naturally ends. Only one sound can be queued at a time. Queuing a new sound while another sound is queued will result in the new sound becoming the queued sound. Also, if the current sound is ever stopped or changed, the queued sound will be lost.

If you are loading from a file object, the namehint parameter can be used to specify the type of music data in the object. For example: `queue(fileobj, "ogg")`.

The following example will play music by Bach six times, then play music by Mozart once:

```
pygame.mixer.music.load('bach.ogg')
pygame.mixer.music.play(5)        # Plays six times, not five!
pygame.mixer.music.queue('mozart.ogg')
```

Changed in pygame 2.0.2: Added optional `namehint` argument

have the music send an event when playback stops

`set_endevent() → None`

`set_endevent(type) → None`

This causes pygame to signal (by means of the event queue) when the music is done playing. The argument determines the type of event that will be queued.

The event will be queued every time the music finishes, not just the first time. To stop the event from being queued, call this method with no argument.

Search examples for pygame.mixer.music.set_endevent

`pygame.mixer.music.`**`get_endevent()`**

get the event a channel sends when playback stops

`get_endevent() → type`

Returns the event type to be sent every time the music finishes playback. If there is no endevent the function returns `pygame.NOEVENT`.

Search examples for pygame.mixer.music.get_endevent

Edit on GitHub