pygame

Python 2.x 2.4–2.7 , Python 3.x 3.2–3.6

Pygame is the go-to library for making multimedia applications, especially games, in Python. The official website is http://www.pygame.org/ .

Examples

Installing pygame

With `pip` :

```
pip install pygame
```

With `conda` :

```
conda install -c tlatorre pygame=1.9.2
```

Direct download from website : http://www.pygame.org/download.shtml

You can find the suitable installers fro windows and other operating systems.

Projects can also be found at http://www.pygame.org/

Pygame's mixer module

The `pygame.mixer` module helps control the music used in `pygame` programs. As of now, there are 15 different functions for the `mixer` module.

Initializing

Similar to how you have to initialize `pygame` with `pygame.init()` , you must initialize `pygame.mixer` as well.

By using the first option, we initialize the module using the default values. You can though, override these default options. By using the second option, we can initialize the module using the values we manually put in ourselves. Standard values:

```
pygame.mixer.init(frequency=22050, size=-16, channels=2, buffer=4096)
```

To check whether we have initialized it or not, we can use `pygame.mixer.get_init()` , which returns `True` if it is and `False` if it is not. To quit/undo the initializing, simply use `pygame.mixer.quit()` . If you want to continue playing sounds with the module, you might have to reinitialize the module.

Possible Actions

As your sound is playing, you can pause it tempoparily with `pygame.mixer.pause()` . To resume playing your sounds, simply use `pygame.mixer.unpause()` . You can also fadeout the end of the sound by using `pygame.mixer.fadeout()` . It takes an argument, which is the number of milliseconds it takes to finish fading out the music.

Channels

You can play as many songs as needed as long there are enough open channels to support them. By default, there are 8 channels. To change the number of channels there are, use `pygame.mixer.set_num_channels()` . The argument is a non-negative integer. If the number of channels are decreased, any sounds playing on the removed channels will immediately stop.

To find how many channels are currently being used, call `pygame.mixer.get_channels(count)` . The output is the number of channels that are not currently open. You can also reserve channels for sounds that must be played by using `pygame.mixer.set_reserved(count)` . The argument is also a non-negative integer. Any sounds playing on the newly reserved channels will not be stopped.

You can also find out which channel isn't being used by using `pygame.mixer.find_channel(force)` . Its argument is a bool: either True or False. If there are no channels that are idle and `force` is False, it will return `None` . If `force` is true, it will return the channel that has been playing for the longest time.

Syntax

```
pygame.mixer.init(frequency=22050, size=-16, channels=2, buffer=4096)
```

```
pygame.mixer.pre_init(frequency, size, channels, buffer)
```

```
pygame.mixer.quit()
pygame.mixer.get_init()
pygame.mixer.stop()
pygame.mixer.pause()
pygame.mixer.unpause()
pygame.mixer.fadeout(time)
pygame.mixer.set_num_channels(count)
pygame.mixer.get_num_channels()
pygame.mixer.set_reserved(count)
pygame.mixer.find_channel(force)
pygame.mixer.get_busy()
```

Parameters

Parameter	Details
count	A positive integer that represents something like the number of channels needed to be reserved.
force	A boolean value (`False` or `True`) that determines whether `find_channel()` has to return a channel (inactive or not) with `True` or not (if there are no inactive channels) with `False`

Remarks