Ariah's Audio Library 1.3.0

This written documentation for Ariah's Audio Library is intended to be your guide to using the library. If you would prefer to learn through video documentation, I will be uploading a video to my YouTube channel in the near future where I detail how to use the library and give a tour of the code base. This file will be uploaded with a link to the video when it is released.

This documentation covers version 1.3.0 of Ariah's Audio Library, and may include and/or exclude functions that are present in other versions. Check out the Current Version's documentation or the Version History page for documentation on a specific version of the library.

Contents

What's New?

With version 1.3.0 comes a rewrite of this documentation as well as a few nice additions to the library itself.

Version 1.3.0

Long Term Stable (LTS)

Since the last time this library was updated, *GameMaker* came out with a Long Term Stable (LTS) release of the program, aiming to maintain stability through the long term for beefier projects that may need that stability throughout development.

This project will currently only (officially) support this LTS release (Version 2022.0), as the more current versions of *GameMaker* have added official support for looping sections of audio tracks. I may still make a secondary version for the main branch of *GameMaker*, but it isn't a priority right now and wouldn't be consistently updated.

Documentation

- The documentation was rewritten from the ground up, not so intentionally due to the original markdown file being lost in a hard drive failure.
- The decision was made to transition the documentation away from being a living document that accomadates all versions to a series of documents curated for each version. The initial version of the documentation, covering versions 1.0.0, 1.1.0, 1.2.0, and 1.2.1 of the library, is available on the Version History page.
- The most up-to-date version of the documentation will now be at a permanent link.

Ariah's Audio Library

- The **BGM** function `bgm_is_playing()` has been given an optional argument for specifying a specific track.
  - If the argument is specified, it will return *true* if the specified track is currently playing, and *false* if its paused, if another track is playing, or if no track is playing at all.
  - If the argument is not specified, the function maintains its previous behavior: it will return *true* if any **BGM** is currently playing, and *false* if the current track is paused or if no track is playing at all.
- Made the **BGM** functions consistent when referring to time. Every function now uses time in milliseconds.
  - This change affects the functionality of `bgm_set_loop_info()`, `bgm_get_loop_info()`, `bgm_get_length()`, `bgm_set_time()`, and `bgm_get_time()`.
  - There is a new option in the settings called `LEGACY_USE_SECONDS_IN_BGM` that defaults to *false*, but when set to *true* restores the original functionality of these functions, using seconds instead of milliseconds. If you are updating an existing project, you can set this new option to *true* and nothing will be broken.
- Using Positional Audio has been made slightly easer, you can either set the position for the listener manually with `set_listener_x()`, `set_listener_y()`, and/or `set_listener_position()`, or you can set the listener to follow an object, such as your player, with `set_listener_following()`.

For GameMaker 2022.0

License

Ariah's Audio Library is licensed under CC0. You have the right to use this library in your own game, even a commercial one, without providing attribution.

Overview

Ariah's Audio Library aims to be an easy to use upgrade to *GameMaker*'s audio system. It adds features such as advanced track looping, easy positional audio, and easy access to a "global audio volume".

This library is built around the idea of there being three types of audio in a typical game: **background music** (**BGM**), **sound effects** (**SFX**), and **background sounds** (**BGS**).

BGM is limited to one track at a time. Attempting to play a **BGM** after one is already playing will result in the currently playing track stopping (or fading into the new track). **BGM** is, self-evidently, used for the music in your game. Individual tracks played as **BGM** can loop or not loop, and **BGM** supports custom loop points, allowing you to play a track with an intro without including that intro in the loop. **BGM** *cannot* be played at a point, it always plays globally.

There can be as many **SFX** tracks playing at one time as you would like. **SFX** *can't* loop, but it *can* be played at a point in the world. **SFX** is used for small sounds the play as a reaction to something the player does, like selecting a menu option, slashing a sword, or getting hit by an enemy.

BGS is similar to **SFX** in the sense that you can have as many tracks playing at one time as you would like. **BGS** *always* loops the entire track (this does not support custom loop points like **BGM**). **BGS** can also be played at a point in the world, like **SFX**. **BGS** is used for background noise that can play under music, such as the noise generated by rain, cicadas, or fire.

Importing

After downloading the library from the official download page, open your *GameMaker* project and select **Tools** -> **Import Local Package** from the toolbar.

Select the file ending in **.yymps**, click "*Add All*" and then click "*Import*".

Settings

One you have successfully imported the library, you will see a new folder in your Asset Browser called "*Audio*". It is recommended you create a folder named something along the lines of "*Libraries*" where you can drag your "*Audio*" folder (and the folder for any other libraries you might install) into.

Inside the "*Audio*" folder, you will find an object named **audio**, and a script named **ariah_audio_lib_settings**. You do ***not*** need to modify the **audio** object in any way.

Inside **ariah_audio_lib_settings**, you will find a few functions prefixed with "*__AriahAudioLib__*". These are the settings that you can modify.

Audio Priority Settings

These settings deal with the priority at which to play each of the three types of audio. *GameMaker*'s audio system uses a priority variable for playing audio that's used when too many audio channels are being utilized. The lowest priority sounds stop playing first, in favor of the higher priority sounds currently playing.

The native *GameMaker* function `audio_channel_num()` can be used to change the number of audio channels supported. The default is **128**.

`__AriahAudioLibrary__BGM_PRIORITY`

This function by default returns **2**. You may change what this function returns to whatever you wish the priority for **BGM** to be.

`__AriahAudioLibrary__SFX_PRIORITY`

This function by default returns **3**. You may change what this function returns to whatever you wish the priority for **SFX** to be.

`__AriahAudioLibrary__BGS_PRIORITY`

This function by default returns **1**. You may change what this function returns to whatever you wish the priority for **BGS** to be.

Audio Falloff Settings

These settings are used when playing sounds at a location in the game. They specifically have to do with how the audio fades out as the player gets further away from the point at which sound is playing.

If you don't understand how these work, you can safely leave them as the defaults.

`__AriahAudioLibrary__FALLOFF_MODEL`

This function by default returns the build in macro `audio_falloff_linear`. You can view the various possibilities on the official *GameMaker* documentation, here.

`__AriahAudioLibrary__FALLOFF_REF_DIST`

This function by default returns **100**. This value "...is the distance from the listener under which the volume for the sound playing would normally drop by half before being influenced by [falloff] factor or the specified maximum distance."[1]

`__AriahAudioLibrary__FALLOFF_MAX_DIST`

This function by default returns **300**. This value "...sets the distance where there will no longer be any attenuation of the source sound. This can be the point at which the sound is no longer heard *or* the point at which the sound volume no longer decreases below the minimum threshold defined by the model chosen."[2]

`__AriahAudioLibrary__FALLOFF_FACTOR`

This function by default returns **1**. This value "...is used in distance attenuation based on the inverse distance model and sets the final minimum threshold for a sound with falloff."[3]

Legacy Settings

These settings are introduced when something major changes that breaks some existing code. If you're making a new project, it's best not to worry about these. If you're updating, look into if the change affects your game and enable one if you need the reversion.

`__AriahAudioLibrary__LEGACY_USE_SECONDS_IN_BGM`

This function by default returns **false**. In versions prior to 1.3.0, some bgm functions handled time in seconds instead of milliseconds like everything else. This was adjusted in version 1.3.0 so that every function uses milliseconds. Setting this to **true** reverts this change to be compatible with existing projects.

Getting Started

In order to use this Audio Library, you must put the included **audio** object into the first room of your game.

Once you've done that, you can use any of the included functions anywhere in your game.

If you're looking to just jump into it and don't want to sift through the function index, here's a few quick tips to get you started.

Playing Audio

You can easliy play audio of any kind with the following functions:

```
audio.bgm_play(music);
audio.sfx_play(sound);
audio.bgs_play(background);
```

These functions will just work with no additional parameters. **BGS** will always loop and **SFX** will never loop. **BGM** will loop the entire track by default, but you can either...

1. Disable looping by adding the argument `false`.

   ```
   audio.bgm_play(music, false);
   ```

2. Set looping information for the track first, including the times (in seconds) for both the beginning and end of the loopable segment of the track.

   ```
   audio.bgm_set_loop_info(music, 6.000, 78.000);
   audio.bgm_play(music);
   ```

   Please note that the time of the ending of the loopable segment ***cannot*** be the ending of the track. *In GameMaker, you **cannot** reliably predict when a track is ending, so the track must continue for at least another second to be safe.*

   I would recommend adding a couple seconds from the beginning of the loopable segment to the end of your track for the smoothest loop.

For further information, please refer to the Function Index.

Positional Audio

It's quite easy to play both **SFX** and **BGS** sounds positionally in your world.

In order to set this up, you must either...

- ...set the x and y of the listener with `audio.set_listener_x()` and `audio.set_listener_y()` or with `audio.set_listener_position()`.
- ...set an object for the listener to follow with `audio.set_listener_following()`.

Then, you just need to play the sounds with one of the following:

```
audio.sfx_play_at(sound, x, y, [fade], [volume], [pitch]);
audio.bgs_play_at(background, x, y, [fade], [volume], [pitch]);
```

And it will simply work!

Function Index

In order to use the following functions, you must prefix them with `audio.` just as was done in the Getting Started section.

Background Music (BGM)

Playing, Pausing, and Stopping

```
bgm_play(music, [loop], [fade], [volume], [pitch]);
```

Arguments

> ***music*** ~ a 'sound' asset, the sound to play as **BGM**.
>
> ***loop*** *(optional)* ~ a boolean, whether or not to loop the track *(default: **true**)*.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade in the new track. If another track is already playing, this number is also the amount of time to fade out the previous track *(default: **0**)*.
>
> ***volume*** *(optional)* ~ a number (1 = 100%), the desired volume of this *specific* track *(default: **1**)*.
>
> ***pitch*** *(optional)* ~ a number (1 = 100%), the desired pitch of htis *specific* track *(default: **1**)*.

Description

You can use this function to *play* an audio track as **BGM**. If another track is already playing as **BGM**, that track will stop (or fade out) and the new one will start (or fade in).

```
bgm_stop([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade out the currently playing track.

Description

You can use this function to stop the currently playing **BGM** track.

```
bgm_pause([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade out the currently playing track.

Description

You can use thsi function to pause the currently playing **BGM** track. Both the track itself and the time the track was paused at is saved, to be resumed when `bgm_resume()` is called. If another track is played as **BGM** after one is paused, the paused track and the saved time will be removed from memory in favor of the new track.

```
bgm_resume([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade back in the currently paused track.

Description

You can use this function to resume the currently paused **BGM** track. If you have paused a track with `bgm_pause()`, it will resume at the same time as you paused it at unless the time was changed with `bgm_set_time()`. *If no track has been paused, this function will do nothing*.

Looping

```
bgm_set_loop_info(track, [start], [end]);
```

Arguments

> ***track*** ~ a 'sound' asset, the sound to set loop information for.
>
> ***start*** *(optional)* ~ a number (in seconds), the time at which the loopable segment of the track begins *(default: **undefined**)*.
>
> ***end*** *(optional)* ~ a number (in seconds), the time at which the loopable segment of the track ends *(default: **undefined**)*.

Description

You can use this function to set the loopable portion of a track. If a track is played as **BGM** and looped without using this function first, the entire track will be looped. If the function was used to set the loop information, only the defined segment of the track will loop.

If either the ***start*** or ***end*** arguments are not given or are *undefined*, any existing loop information previously set for this track will be deleted.

```
bgm_get_loop_info(track);
```

Arguments

> ***track*** ~ a 'sound' asset, the sound to get loop information for.

Description

You can use this function to get loop information for a track. Returns an array with two elements, the beginning of the loopable segment and the end of the loopable segment (in seconds).

If no loop information was set for the track, it will return *undefined*.

Volume and Pitch

```
bgm_set_global_volume(volume, [fade]);
```

Arguments

> ***volume*** ~ a number (1 = 100%), the desired global volume for **BGM** tracks.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing tracks should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the global volume for **BGM** tracks. The global volume affects *every* **BGM** track, so it's recommended this is the value you use when adding a volume slider to your menu.

Each **BGM** track's volume is calculated by multiplying the global bgm volume with that track's individual volume setting. They are both **1** by default.

```
bgm_get_global_volume();
```

Description

You can use this function to get the current global volume for **BGM** tracks. The global volume affects *every* **BGM** track, so it's recommended this is the value you use when adding a volume slider to your menu.

```
bgm_set_volume(volume, [fade]);
```

Arguments

> ***volume*** ~ a number (1 = 100%), the desired volume for the currently playing **BGM** track.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing tracks should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the volume for the currently playing **BGM** track.

Each **BGM** track's volume is calculated by multiplying this number with the global bgm volume. They are both **1** by default.

```
bgm_get_volume();
```

Description

You can use this function to get the volume for the currently playing **BGM** track.

```
bgm_set_pitch(pitch);
```

Arguments

> ***pitch*** ~ a number (1 = 100%), the desired pitch for the currently playing **BGM** track.

Description

You can use this function to change the pitch for the currently playing **BGM** track.

```
bgm_get_pitch();
```

Description

You can use this function to get the pitch for the currently playing **BGM** track.

```
bgm_set_time(time);
```

Arguments

> ***time*** ~ a number (in seconds), the time at which to set the currently playing **BGM** track.

Description

You can use this function to set the time for the currently playing **BGM** track. If this function is called while the current **BGM** track is paused, the track will resume at the new time once `bgm_resume()` is called.

```
bgm_get_time();
```

Description

You can use this function to get the time for the currently playing **BGM** track. If this function is called while the current **BGM** track is paused, it will return the time at which the track was paused.

Misc.

```
bgm_get_length();
```

Description

You can use this function to get the length of the currently playing **BGM** track. If you have set loop information for this track via `bgm_set_loop_info()`, this will return the time of the end of the loop rather than the end of the track.

```
bgm_is_playing([track]);
```

Arguments

> ***track*** *(optional)* ~ a 'sound' asset, the track to check if it's currently playing *(default: **undefined**)*

Description

You can use this function to get whether or not a specific **BGM** track is currently playing. If no track is specified, it gets whether or not any **BGM** track is currently playing. This will return *false* if either the specified track is not playing, no track was specified and no track is playing, or the current **BGM** track is paused.

```
bgm_is_paused();
```

Description

You can use this function to get whether or not a **BGM** track is currently paused. This will return *false* if either the current **BGM** track is playing, or if there is no current **BGM** track.

```
bgm_update_gain([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade the currently playing **BGM** track to the current volume settings *(default: **0**)*.

Description

You can use this function to update the volume of the currently playing **BGM** track if the volumes were updated while the track is playing.

> Note that this function is called automatically by functions like `bgm_set_volume()`, so you would only need to call this function if you modify `gb_vol_bgm` or `music_volume` directly which is ***not*** recommended.

Sound Effects (SFX)

Playing and Stopping

```
sfx_play_at(sound, x, y, [fade], [volume], [pitch]);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound to play as a **SFX**.
>
> ***x*** ~ a number, the x coordinate at which to play the sound.
>
> ***y*** ~ a number, the y coordinate at which to play the sound.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade in the sound effect *(default: **0**)*.
>
> ***volume*** *(optional)* ~ a number (1 = 100%), the desired volume of this *specific* sound effect *(default: **1**)*.
>
> ***pitch*** *(optional)* ~ a number (1 = 100%), the desired pitch of this *specific* sound effect *(default: **1**)*.

Description

You can use this function to play an audio track as a **SFX** at a given location within your room.

```
sfx_play(sound, [fade], [volume], [pitch]);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound to play as a **SFX**.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade in the sound effect *(default: **0**)*.
>
> ***volume*** *(optional)* ~ a number (1 = 100%), the desired volume of this *specific* sound effect *(default: **1**)*.
>
> ***pitch*** *(optional)* ~ a number (1 = 100%), the desired pitch of this *specific* sound effect *(default: **1**)*.

Description

You can use this function to *play* an audio track as a **SFX**.

```
sfx_stop(sound, [fade]);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound to stop.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade out the SFX sound. *(default: **0**)*.

Description

You can use this function to stop a specific sound. If multiple **SFX** sounds are playing from the same sound asset, it will stop all of the sounds originating from that asset.

```
sfx_stop_all([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade out the **SFX** sounds *(default: **0**)*.

Description

You can use this function to stop *all currently playing* **SFX** sounds.

Volume and Pitch

```
sfx_set_global_volume(volume, [fade]);
```

Arguments

> ***volume*** ~ a number (1 = 100%), the desired global volume for **SFX* sounds.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing **SFX** sounds should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the global volume for **SFX** sounds. The global volume affects *every* **SFX** sound, so it's recommended this is the value you use when adding a volume slider to your menu.

Each **SFX** sound's volume is calculated by multiplying the global sfx volume with that sound's individual volume setting. They are both 1 by default.

```
sfx_get_global_volume();
```

Description

You can use this function to get the current global volume for **SFX** sounds. The global volume affects *every* **SFX** sound, so it's recommended this is the value you use when adding a volume slider to your menu.

```
sfx_set_volume(sound, volume, [fade]);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound for which to change the volume.
>
> ***volume*** ~ a number (1 = 100%), the desired volume for the given **SFX** sound.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing sounds should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the volume for a given **SFX** sound. If multiple **SFX** sounds were played from the same 'sound' asset, they all will be affected by calling this function.

Each **SFX** sound's volume is calculated by multiplying this number with the global sfx volume. They are both 1 by default.

```
sfx_get_volume(sound);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound for which to get the volume.

Description

You can use this function to get the volume for the given **SFX** sound. If multiple **SFX** sounds were played from the save 'sound' asset, this function will return the volume of the first match it finds.

```
sfx_set_pitch(sound, pitch);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound for which to set the pitch.
>
> ***pitch*** ~ a number (1 = 100%), the desired pitch for the currently playing **SFX** sound.

Description

You can use this function to change the volume for a given **SFX** sound. If multiple **SFX** sounds were played from the same 'sound' asset, they all will be affected by calling this function.

```
sfx_get_pitch(sound);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound for which to get the pitch.

Description

You can use this function to get the pitch for the given **SFX** sound. If multiple **SFX** sounds were played from the same 'sound' asset, this function will return the pitch of the first match it finds.

Misc.

```
sfx_is_playing(sound);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound to check if it's playing.

Description

You can call this function to check if a given **SFX** sound is currently playing.

```
sfx_is_any_playing();
```

Description

You can call this function to check if any **SFX** sound is currently playing.

```
sfx_update_gain([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade currently playing **SFX** sounds to the current volume settings *(default: **0**)*.

Description

You can use this function to update the volume of the currently playing **SFX** sounds if the volumes were updated while the track is playing.

> Note that this function is called automatically by functions like `sfx_set_volume()` so you would only need to call this function if you modify `gb_vol_sfx` or `sound_volumes` directly which is ***not*** recommended.

Background Sounds (BGS)

Playing and Stopping

```
bgs_play_at(background, x, y, [fade], [volume], [pitch]);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound to play as a **BGS**.
>
> ***x*** ~ a number, the x coordinate at which to play the sound.
>
> ***y*** ~ a number, the y coordinate at which to play the sound.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade in the background sound *(default: **0**)*.
>
> ***volume*** *(optional)* ~ a number (1 = 100%), the desired volume of this *specific* background sound *(default: **1**)*.
>
> ***pitch*** *(optional)* ~ a number (1 = 100%), the desired pitch of this *specific* background sound *(default: **1**)*.

Description

You can use this function to *play* an audio track as a **BGS** at a given location within your room.

```
bgs_play(background, [fade], [volume], [pitch]);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound to play as a **BGS**.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the amount of time to fade in the background sound *(default: **0**)*.
>
> ***volume*** *(optional)* ~ a number (1 = 100%), the desired volume of this *specific* background sound *(default: **1**)*.

Description

you can use this function to *play* an audio track as a **BGS**.

```
bgs_stop(background, [fade]);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound to stop.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade out the **BGS** sound. *(default: **0**)*.

Description

You can use this function to stop a specific sound. If multiple **BGS** sounds are playing from the same sound asset, it will stop all of the sounds orginating from that asset.

```
bgs_stop_all([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade out the **BGS** sounds *(default: **0**).

Description

You can use this function to stop *all currently playing* **BGS** sounds.

Volume and Pitch

```
bgs_set_global_volume(volume, [fade]);
```

Arguments

> ***volume*** ~ a number (1 = 100%), the desired global volume for **BGS** sounds.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing **BGS** sounds should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the global volume for **BGS** sounds. The global volume affects *every* **BGS** sound, so it's recommended this is the value you use when adding a volume slider to your menu.

Each **BGS** sound's volume is calculated by multiplying the global bgs volume with that sound's individual volume setting. They are both 1 by default.

```
bgs_get_global_volume();
```

Description

You can use this function to get the current global volume for **BGS** sounds. The global volume affects *every* **BGS** sound, so it's recommended this is the value you use when adding a volume slider to your menu.

```
bgs_set_volume(background, volume, [fade]);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound for which to change the volume.
>
> ***volume*** ~ a number (1 = 100%), the desired volume for the given **BGS** sound.
>
> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which any currently playing sounds should fade into their new volume *(default: **0**)*.

Description

You can use this function to change the volume for a given **BGS** sound. If multiple **BGS** sounds were played from the same 'sound' asset, they all will be affected by calling this function.

Each **BGS** sound's volume is calculated by multiplying this number with the global bgs volume. They are both **1** by default.

```
bgs_get_volume(background);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound for which to get the volume.

Description

You can use this function to get the volume for the given **BGS** sound. If multiple **BGS** sounds were played from the same 'sound' asset, this function will return the volume of the first match it finds.

```
bgs_set_pitch(background, pitch);
```

Arguments

> ***sound*** ~ a 'sound' asset, the sound for which to set the pitch.
>
> ***pitch*** ~ a number (1 = 100%), the desired pitch for the currently playing **BGS** sound.

Description

You can use this function to change the pitch for a given **BGS** sound. If multiple **BGS** sounds were played from the same 'sound' asset, they will all be affected by calling this function.

```
bgs_get_pitch(background);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound for which to get the pitch.

Description

You can use this function to get the pitch for a given **BGS** sound. If multiple **BGS** sounds were played from the same 'sound' asset, this function will return the pitch of the first match it finds.

Misc.

```
bgs_is_playing(background);
```

Arguments

> ***background*** ~ a 'sound' asset, the sound to check if it's playing.

Description

You can call this function to check if a given **BGS** sound is currently playing.

```
bgs_is_any_playing();
```

Description

You can call this function to check if *any* **BGS** sound is currently playing.

```
bgs_update_gain([fade]);
```

Arguments

> ***fade*** *(optional)* ~ a number (in milliseconds), the time over which to fade the currently playing **BGS** sounds to the current volume settings *(default: **0**)*.

Description

You can use this function to update the volume of the currently playing **BGS** sounds if the volumes were updated while the track is playing.

> Note that this function is called automatically by functions like `bgs_set_volume()` so you would only need to call this function if you modify `gb_vol_bgs` or `background_volumes` directly which is ***not*** recommended.

Positional Audio

```
set_listener_x(x);
```

Arguments

> ***x*** ~ a number, the desired x coordinate for the listener of positional audio.

Description

You can use this function to set the x coordinate for the listener of positional audio.

```
set_listener_y(y);
```

Arguments

> ***y*** ~ a number, the desired y coordinate for the listener of positional audio.

Description

You can use this function to set the y coordinate for the listener of positional audio.

```
set_listener_position(x, y);
```

Arguments

> ***x*** ~ a number, the desired x coordinate for the listener of positional audio.
>
> ***y*** ~ a number, the desired y coordinate for the listener of positional audio.

Description

You can use this function to set the x and y coordinates for the listener of positional audio. If the

---

```
set_listener_following(object);
```

Arguments

> ***object*** ~ an 'object' asset, the desired object for the listener of positional audio to follow.

Description

You can use this function to set the listener of positional audio to follow a specific object.

1. From the *GameMaker* documentation. ↩
2. From the *GameMaker* documentation. ↩
3. From the *GameMaker* documentation. ↩