Ariah's Audio Library

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.

> Please note that versions before 1.2.0 are not officially supported. They did not have settings the user could easily change, and they had at least a couple functions that just didn't work. Therefore, every function in this documentation is credited to only being available at earliest from 1.2.0 onward.

Contents

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 that 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`

Versions 1.2.0+

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`

Versions 1.2.0+

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`

Versions 1.2.0+

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`

Versions 1.2.0+

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

`__AriahAudioLibrary__FALLOFF_REF_DIST`

Versions 1.2.0+

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`

Versions 1.2.0+

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`

Versions 1.2.0+

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]

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 easily 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

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]);
```

Versions 1.2.0+

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 this *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]);
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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 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]);
```

Versions 1.2.0+

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 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]);
```

Versions 1.2.0+

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 this 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);
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

Description

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

```
bgm_set_pitch(pitch);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

Description

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

Time

```
bgm_set_time(time);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

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_is_playing();
```

Versions 1.2.0+

Description

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

```
bgm_is_paused();
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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.

Note that in order for this function to work, you must include the line `audio_listener_orientation(0, 0, -1, 0, 1, 0);` in the *Create* event of either your player or a universal controller object. You must also include the line `audio_listener_position(x, y, 0);` in a *Step* event of your player *(I use Begin Step)*.

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

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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);
```

Versions 1.2.0+

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 same 'sound' asset, this function will return the volume of the first match it finds.

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

Versions 1.2.0+

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);
```

Versions 1.2.0+

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);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

Description

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

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

Versions 1.2.0+

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]);
```

Versions 1.2.0+

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.

> Note that in order for this function to work, you must include the line `audio_listener_orientation(0, 0, -1, 0, 1, 0);` in the *Create* event of either your player or a universal controller object. You must also include the line `audio_listener_position(x, y, 0);` in a *Step* event of your player *(I use Begin Step)*.

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

Versions 1.2.0+

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)*.
>
> *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**.

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

Versions 1.2.0+

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 originating 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]);
```

Versions 1.2.0+

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);
```

Versions 1.2.0+

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 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.

```
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 the 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);
```

Versions 1.2.0+

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();
```

Versions 1.2.0+

Description

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

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

Versions 1.2.0+

Arguments

> *fade (optional)* ~ a number (in milliseconds), the time over which to fade 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.

1. From the *GameMaker* documentation. ↵

2. From the *GameMaker* documentation. ↵

3. From the *GameMaker* documentation. ↵