Version 1.3.0

Introduction

Eazy Sound Manager is a **simple** tool which aims to make sound and music management in games easier. Playing a single audio clip is now as easy as calling one API function. The API can handle multiple music, game and UI sound effects at the same time while still giving you the option to interrupt previous audio clips when needed. Audio clips can be one shot, or looping.

Moreover, Eazy Sound Manager has the option to make music persist through multiple scenes, as well as add fade in/out transitions. Different global settings for music, game sound effects and UI sound effects are also implemented. However, each audio has its own volume setting which is always relative to its global volume.

Features

- Play multiple audio clips
- Play music, game sound effects and UI sound effects
- Play/Stop/Pause/Resume all or individual audio clips
- Loop music
- Fade in and fade out transitions
- Global volume settings
- Music persistence across scenes
- 2D & 3D (spatial) audio support
- Audio pool
- Runtime API (Fully documented)
- No setup needed
- **Full C# source code**

Compatibility

- Multi-platform support (Windows, Mac, Linux, Android, iOS)
- Unity 5.6 or higher

Download

You can download the plugin from the Unity Asset store

Installation

Installation is super easy. You can get Eazy Sound Manager working in just a few minutes. First, download the Unity package from the Unity Asset Store. When the download is finished, the Import window should appear, listing all the files and folders in the package. Make sure you have them all selected, and press **Import**. Unity should now start importing the assets in the package.

Upgrade from older versions (< 1.3)

If you are already using an older version of Eazy Sound Manager, you will need to make a few changes to use the new version.

- Change namespace to `Hellmade.Sound` in your scripts.
- Change `SoundManager` to `EazySoundManager` in your code.
- You can delete the old Eazy Sound Manager folders from your Unity projects, but make sure you have imported all the required files for this version.
- Access to the `AudioSource` from `Audio` objects is now deprecated. Almost all `Audiosource` parameters can be accessed directly from the `Audio` object now. Check the API reference for further details.
- Several names for parameters have changed to enforce the correct naming conventions (Mostly changed capitalizations). Check the API reference for further details.

Content

The imported package includes all the scripts for Eazy Sound Manager, as well as demo scenes for demonstrations. Some editor resources are also included. The main folder will be stored at `Assets/Hellmade/Eazy Sound Manager`

- **Demo files:** This folder included example scenes and all demo files `Assets/Hellmade/Eazy Sound Manager/Demo`
- **Docs:** This folder included a copy of the manual `Assets/Hellmade/Eazy Sound Manager/Docs`
- **Scripts:** All the scripts required for Eazy Sound Manager `Assets/Hellmade/Eazy Sound Manager/Scripts`

Getting Started

Setting Eazy Sound Manager up and using it is super easy and straightforward. Eazy Sound Manager can only be used from code using the very simple API.

Setup

No setup is needed to get Eazy Sound Manager running. If you have imported all the required files, you are ready to go!

DO NOT ATTACH THE SCRIPT ON ANY GAMEOBJECT

Make sure to always include the namespace Hellmade.Sound when using Eazy Sound Manager from code. Just use this on the top of every script that uses Eazy Sound Manager:

`using Hellmade.Sound;`

Play Music & Sounds

To play a new audio, just use `PlayMusic` for background music, `PlaySound` for sound fx, and `PlaySoundUI` for UI sound fx. These functions return a unique `AudioID`, which can be later used to access the created Audio. Note that playing a new music audio, will stop the previous one. On the other hand, multiple sounds can be played at the same time.

All Play functions have several overloads which allow you to use only the settings that are relevant to you. Please check the API reference for more details.

Example

```
int backgroundMusicID = EazySoundManager.PlayMusic(musicAudioClip, 0.7f, true, false,
1, 1);
```

The above example will play a music audio (musicAudioClip), which will have 0.7 volume, will be looped, will not persist across scenes, and will have fade in and out transitions of 1 second.

Preparing Audio

An `Audio` can also be created and initialized without immediately playing it. This can be useful in cases where you just need to prepare and create all or some of your Audios beforehand, and use them later on. Just use any of the Prepare functions and save a reference of the `Audio` object.

```
public Audio fightSceneAudio;

int fightSceneMusicID = EazySoundManager.PrepareMusic(fightSceneAudioClip, 1f, true,
false, 0.5f, 1);
fightSceneAudio = SoundManager.GetAudio(fightSceneMusicID);
...
fightSceneAudio.Play();
```

Accessing Audios

As stated above, each Audio object has its own `AudioID`. This ID can be used to access them at any time after creation. You can play, stop or pause Audios individually.

```
Audio backgroundMusicAudio = EazySoundManager.GetAudio(backgroundMusicID);
backgroundMusicAudio.Stop();
```

If you already know the type of your Audio, you can also use the type-specific GetAudio functions, `GetMusicAudio`, `GetSoundAudio` and `GetUISoundAudio`.

Another way to access `Audio` objects, is to search using the `AudioClip`.

```
public AudioClip backgroundMusicClip;
...
Audio backgroundMusicAudio = EazySoundManager.GetAudio(backgroundMusicClip);
backgroundMusicAudio.Stop();
```

Moreover, several settings like the fade in/ fade out speed, audio volume or whether to keep looping can be changed by accessing an `Audio` object.

```
backgroundMusicAudio.SetVolume(0.5f);
backgroundMusicAudio.Loop = false;
```

3D Audio

Eazy Sound Manager also supports playing 3D (spatial) audio. Only difference with 2D audio, is that you need to specify a transform to be the source of the audio (if 3D audio is not desired, just use null as source transform).

```
int gunShootSoundID = SoundManager.PlaySound(gunShootClip, 1f, false, gunTransform);
```

You can easily set min and max distances, or any other 3D Audio setting.

```
Audio gunShootAudio = SoundManager.GetAudio(gunShootSoundID);
gunShootAudio.Min3DDistance = 1f;
gunShootAudio.Max3DDistance = 10f;
gunShootAudio.SpatialBlend = 0.8f;
```

Global Volumes

The volume of an `Audio` is also affected by the global volumes. There are four global volumes:

- `GlobalVolume`: This affects all Audios
- `GlobalMusicVolume`: This affects only the Music Audios. It is also affected by the GlobalVolume.
- `GlobalSoundsVolume`: This affects only the Sound FX Audios. It is also affected by the GlobalVolume.
- `GlobalUISoundsVolume`: This affects only the UI Sound FX Audios. It is also affected by the GlobalVolume.

```
EazySoundManager.GlobalMusicVolume = 0.5f;
```

Ignore Duplicates

It is possible to automatically ignore a Play request if it is a duplicate. For example, if you try to play a music clip that is already playing, it will be ignored, and therefore not played again. Just set the IgnoreDuplicate option to `True` for any `Audio` type to achieve this behavior

```
EazySoundManager.IgnoreDuplicateMusic = True;
```

API Reference

You can access the API reference online at: http://www.hellmadegames.com/projects/eazynetchecker/docs/api-reference/html/N_Hellmade_Net.htm

You can also find an offline version included in the package

Please use the online version whenever you can, since it will always be up to date faster.

Support

If you need help, have a question or want to recommend future features, please feel free to contact us and we will get back to you as soon as possible.

You can either send us an email, or even contact us using Facebook:

Email: support@hellmadegames.com

Facebook: https://www.facebook.com/hellmadegames

Change Log

[1.3.0] - 12-30-2018

Added

- Audio Pooling
- Optimizations

Changed

- Namespace to *Hellmade.Sound*
- *SoundManager* to *EazySoundManager*
- Code organization for better reusability
- Several parameters names to enforce correct naming conventions

Removed

- Access to the AudioSource component. Almost all AudioSource options can be now accessed directly from the Audio object.