Micro Audio

Unity audio manager

Micro Audio is a powerful and streamlined audio manager for the Unity game engine.

Setup

Right-click on the hierarchy window > `Microlight` > `Micro Audio` > `Micro Audio Manager`.
Or drag&drop prefab from the `Prefabs` folder into the scene.

The library requires `MicroAudioManger` to be present in the scene. Only one is needed in the starting scene and then it will persist between scenes because the manager is set as **`DontDestroyOnLoad`**.

YouTube tutorial video.

About

Micro Audio library eases the management of audio in your game. It saves You time by doing core audio features for you.

- Manages audio layers
- Easily save and load audio volume settings
- Play sound effects from anywhere in the project. Background stuff is managed automatically
- Play a single music clip or create your own playlist of audio clips
- Shuffle your music playlist or turn on crossfade when changing your music tracks
- Easily add more audio layers for your own project needs

API

API is static and simplified so references are not required, simply call any method from anywhere in your project.

Properties	Description
`Mixer`	*Returns audio mixer
`MasterMixerGroup`	*Returns master mixer group
`MusicMixerGroup`	*Returns music mixer group
`SFXMixerGroup`	*Returns sound effects mixer group
`UIMixerGroup`	*Returns UI mixer group
`MasterVolume`	Master layer volume
`MusicVolume`	Music layer volume
`SFXVolume`	Sound effects layer volume
`UIVolume`	UI layer volume
`CurrentTrackProgress`	*Progress of current music track (0-1)
`ActiveMusicGroup`	*Returns currently active MicroSoundGroup
`MusicPlaylist`	*List of ActiveMusicGroup clips index order
`MusicCurrentPlaylistIndex`	*MusicPlaylist index of current clip
`MainMusicAudioSource`	*AudioSource for current music track
`CrossfadeMusicAudioSource`	*AudioSource for fading out track

Methods	Description
`SaveSettings`	Saves volume settings
`LoadSettings`	Loads and applies volume settings
`PlayOneTrack`	Plays one music track
`PlayMusicGroup`	Plays music playlist
`NextTrack`	Plays next track in playlist
`PreviousTrack`	Plays previous track in playlist
`SelectTrack`	Plays track at specified index in playlist
`SetCrossfadeDuration`	Sets duration of crossfade feature
`PlayEffectSound`	Plays sound effect
`PlayUISound`	Plays UI sound effect
`GetDelayStatusOfSound`	Gets delay stats of specified AudioSource

Events	Description
`OnMusicStart`	When music track started
`OnMusicEnd`	When music track finishes
`OnCrossfadeStart`	When crossfade of tracks started
`OnCrossfadeEnd`	When crossfade of tracks ended
`OnNewPlaylist`	When new playlist is generated

'*' = read only

Demo

The project contains a demo scene where library features are presented and how to use the API. Feel free to explore the scene. `DemoSceneManager.cs` script showcases how to use the library.

Features

Mixer

Micro Audio has integrated mixers for easy control of volume in your games. By default, there are 4 layers (master, music, sounds, UI) but more can easily be added for project needs. Mixer and audio layers can be accessed by static property: `MicroAudio.MasterMixerGroup;`. The volume of the audio layer can also be changed with static properties like: `MicroAudio.MasterVolume=0.5f;`. Master affects all other layers and the sounds (SFX) layer affects the UI layer.

- Master
    - Music
    - Sounds
        - UI

Music

With `PlayOneTrack(AudioClip clip, bool loop = true, float crossfade = 0f);` method, only one music clip can be played. The method allows additional customization like looping and crossfade duration for this change only.

`PlayMusicGroup(MicroSoundGroup group, bool shuffle = false);` sets specified sound group as the new playlist, the playlist can then be shuffled which will be shuffled again each time the playlist reaches the end and starts again. `PlayMusicGroup(MicroSoundGroup group, float crossfadeTime, bool shuffle = false);` allows to also set crossfade duration which can also be set with `SetCrossfadeDuration(float crossfadeDuration)`.

`NextTrack()`, `PreviousTrack()` and `SelectTrack(int index)` are controls for playlist. Note: `SelectTrack(int index)` is an index of the playlist, if the playlist is shuffled playlist track doesn't have to correspond to the same track in `MicroSoundGroup`.

Various info can be extracted from a music player like `CurrentTrackProgress` which tells % (0-1) of current track progress. See the `API` section for more info.

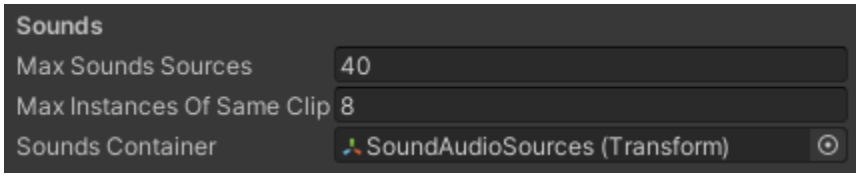
Music audio sources can be left as is but crossfade time can set the starting value for crossfade duration. Calling the `PlayMusicGroup` method with the crossfade parameter will change this value.

Sounds

The sound effects feature allows for the playing of sound effects. Sound effects are auto-generated and are global, which means AudioSource is not required as a parameter to play the sound effect but is rather generated by a script. If a project requires 3D space sounds, sources for playing can be passed as parameters in methods.

`PlayEffectSound(AudioClip clip, float delay = 0f, float volume = 1f, AudioSource src = null)` and UI counterpart, play sound effects in their respective audio layer. The playing of sound effects can be delayed and info about its delay can be accessed with the `GetDelayStatusOfSound(AudioSource src)` method by passing the audio source. Methods for playing sound effects return AudioSource which is used for playing the sound effect.

The number of active sound sources can be limited in settings but setting the value to 0 will set its status to unlimited. Max instances of the same clip allow only a certain amount of the same sound effects to be active at the same time. The sounds container is a container in which sound effects will be spawned, no need to change it but can be changed.



Sound Fade

Sound fade is a feature that can be used independently from the Micro Audio library. The feature will work as long as there is a Micro Audio Manager in the scene. Simply create a new instance of the class passing audio source, start and end volume, and time over which fade will happen.

The fade can be paused by `IsPlaying = false`, killed by `Kill()`, or fast forwarded to the end by `Skip()`.

Properties	Description
`IsPlaying`	Is fading paused or not
`Source`	*AudioSource that is being faded
`StartVolume`	*Start volume
`EndVolume`	*Target volume
`OverSeconds`	*Duration of the fade
`Progress`	*Current progress of the fade (0-1)

Methods	Description
`Kill`	Kills fade at current progress
`Skip`	Skips fade to end, setting end value

Delay Sound

Delay sound, like sound fade can be used independently and allows playing sound effects at the later time.

Delay can't be paused but can be killed by `Kill()` and fast forwarded to the end by `Skip()`. Delay can however be reset to 0 to start over again with the `ResetTimer()` method.

Properties	Description
`Source`	*AudioSource that is being delayed
`Delay`	*Duration of the delay
`Progress`	*Current progress of the delay (0-1)

Methods	Description
`Kill`	Kills delay without playing sound
`Skip`	Finishes delay early
`ResetTimer`	Resets timer back to 0

Crossfade

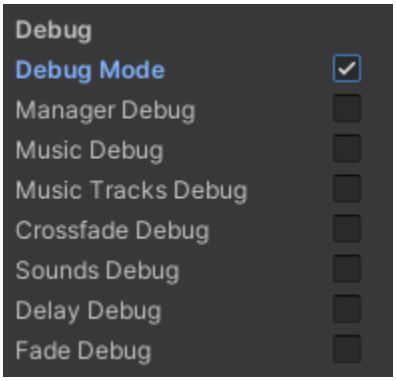
Crossfade allows the current song to gradually lower its volume at the end while the new song increases its volume and slowly takes over. The duration of the effect can be set by the `SetCrossfadeDuration` method or passed in as a parameter of the `PlayMusicGroup` method. The parameter for `PlayOneTrack` is independent of this feature and works only for that one transition.

Save/Load

Saving and loading of volume settings is done through the `PlayerPrefs` feature from Unity. Simply call the `SaveSettings` and `LoadSettings` method. An example of saving and loading volume settings can be inspected in a pre-built demo scene. In the demo scene volume is saved as soon as it is changed but it can be saved on applying volume settings or when leaving the options screen.

Debug

The library offers debug mode if there is some problem. Debug mode is very customizable and offers focusing on most likely culprit. By default only debug mode is visible but when debug mode is enabled, additional options appear.



Option	Description
`Debug Mode`	Turn debug mode on or off
`Manager Debug`	Messages from core manager
`Music Debug`	Messages about music feature, like playlist
`Music Tracks Debug`	Info about music tracks and their status
`Crossfade Debug`	Status of crossfade feature
`Sounds Debug`	Messages from sound effects
`Delay Debug`	Status of sound delay feature
`Fade Debug`	Status of sound fade feature

Contact

For any questions please send e-mail to: microlightgames@gmail.com