Micro Audio

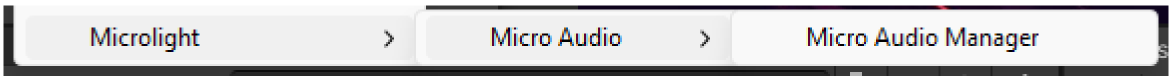
Powerful and streamlined library for managing audio in your game.

Setup

Download '**Microlight**' folder and place it inside '**Assets**' folder of your project.

Micro audio can also be found on Unity Asset Store: https://assetstore.unity.com/packages/slug/272359

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



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

Video tutorial can be found here: https://youtu.be/LryjQARKRMs

About

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

- Manages audio channels
- 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 channels for your own project needs
- Infinity sounds which replace monotonous loops
- Source code which allows to edit manager to your needs

API

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

```
Example code:
MicroAudio.PlayUISound(button_click);
```

Properties	Description
`Mixer`	*Returns audio mixer
`MasterMixerGroup`	*Returns master mixer group
`MusicMixerGroup`	*Returns music mixer group
`SoundsMixerGroup`	*Returns sounds mixer group
`SFXMixerGroup`	*Returns sound effects mixer group
`UIMixerGroup`	*Returns UI mixer group
`MasterVolume`	Master channel volume
`MusicVolume`	Music channel volume
`SoundsVolume`	Sounds channel volume
`SFXVolume`	Sound effects channel volume
`UIVolume`	UI channel volume
`MusicAudioSource`	*AudioSource for current music track
`CrossfadeAudioSource`	*AudioSource for fading out track
`CurrentTrackProgress`	*Progress of current music track (0-1)
`MusicGroup`	*Active MicroSoundGroup containing clips for playlist
`MusicPlaylist`	*List of indexes of MusicGroup clips in playing order
`MusicPlaylistIndex`	*Current index of MusicPlaylist
`ShufflePlaylist`	*Was playlist shuffled or not
`IsMusicPaused`	*Is music being paused

Properties	Description
`IsSoundsPaused`	*Are sounds being paused

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
`PauseMusic`	Pauses music playback (and fades)
`ResumeMusic`	Resumes music playback (and fades)
`ToggleMusicPause`	Pauses or resumes music playback based on current state
`StopMusic`	Stops music playback
`PlayEffectSound`	Plays sound effect
`PlayUISound`	Plays UI sound effect
`GetDelayStatusOfSound`	Gets delay stats of specified AudioSource
`PauseSounds`	Pauses ALL sounds (includes delayed and infinity)
`ResumeSounds`	Resumes ALL sounds (includes delayed and infinity)
`StopSounds`	Stops ALL sounds (includes delayed and infinity)
`PlayInfinityEffectSound`	Plays infinity sound on SFX channel
`PlayInfinityUISound`	Plays infinity sound on UI channel

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

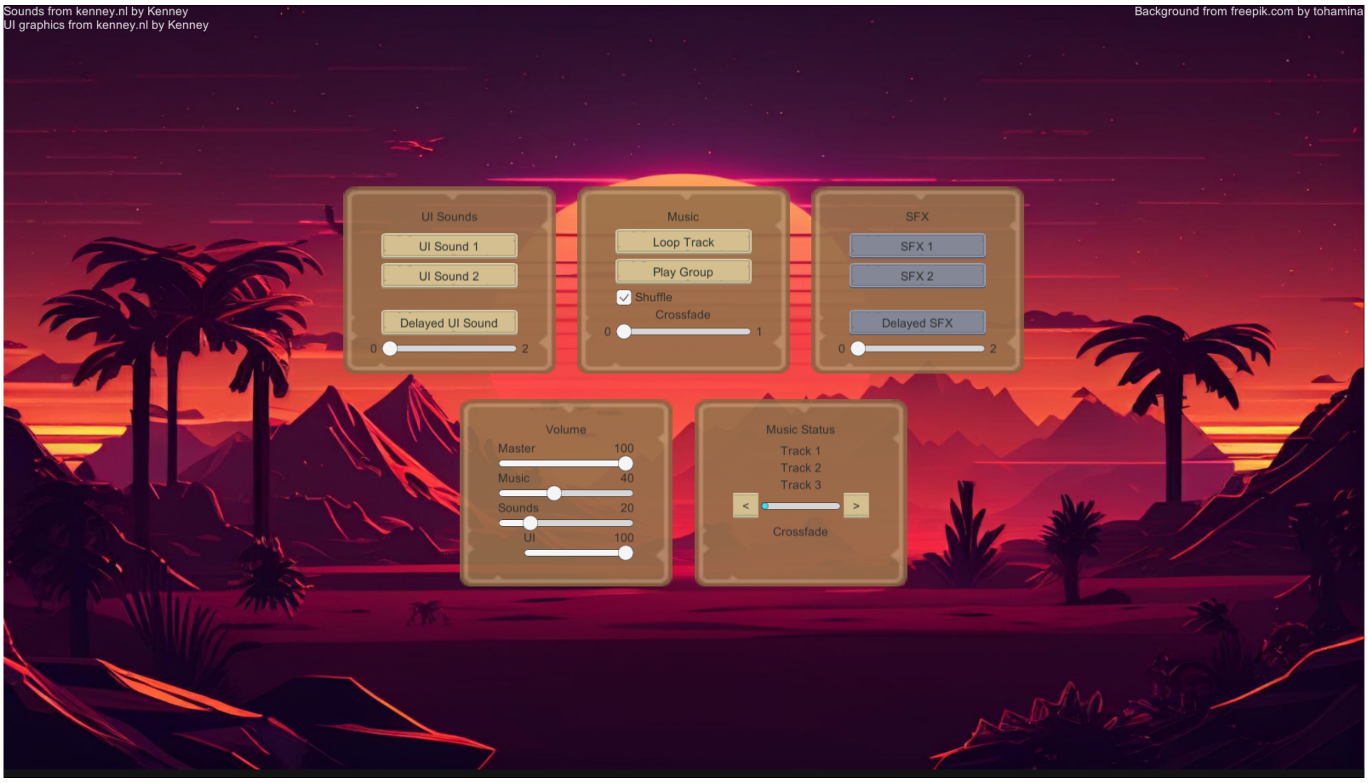
Events	Description
`OnMusicPausedChanged`	When music has been paused/resumed
`OnMusicStopped`	When music playback has been stopped

`OnSoundFinished`	When any sound has finished playing
`OnSoundsPausedChange`	When pause for all sounds has been called
`OnSoundsStopped`	When stop for all sounds has been called

'*' = read only

Demo

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



Features

Mixer

Micro Audio has integrated mixers for easy control of volume in your games. By default, there are 5 channels (master, music, sounds, SFX and UI) but more can easily be added for project needs. Mixer and audio channels can be accessed by static property: `MicroAudio.MasterMixerGroup;`. The volume of the audio channels can also be changed with static properties like: `MicroAudio.MasterVolume=0.5f;`. Master affects all other channels and the sounds channel affects the SFX and UI channels.

- Master
  - Music
  - Sounds
    - SFX
    - UI

Music

With `PlayOneTrack(AudioClip clip, bool loop = true, float volume = -1f, float crossfade = -1f);` method, only one music clip can be played. The method allows additional customization like track volume, looping and crossfade duration. Leaving value at default (-1) means that the previous value will be used.

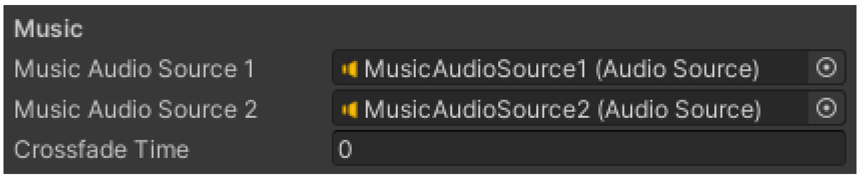
`PlayMusicGroup(MicroSoundGroup group, bool shuffle = false, float volume = -1f, float crossfade = -1f, bool bypassCrossfade = false);` sets specified sound group as the new playlist, the playlist can then be shuffled which will be shuffled again each time the playlist reaches the end and starts again. Like `PlayOneTrack`, leaving the values at default value (-1) means that the old values will be used.

`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`.

`PauseMusic`, `ResumeMusic` and `ToggleMusicPause` allow for control over the music tracks playback. Music can be paused at any time and resumed later and this will affect crossfade feature also. `StopMusic` will however stop music and all of its components (crossfade) and can not be resumed.

Various info can be extracted from 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 `PlayOneTrack` or `PlayMusicGroup` method with the crossfade parameter will change this value.


Music
Music Audio Source 1: MusicAudioSource1 (Audio Source)
Music Audio Source 2: MusicAudioSource2 (Audio Source)
Crossfade Time: 0

Example: music control implementation

```
public void LoopOne() {
    MicroAudio.PlayOneTrack(musicLoopTrack);
}
public void PlayMusicGroup() {
    MicroAudio.PlayMusicGroup(musicGroup, crossfadeSlider.value,
shuffleToggle.isOn);
}
public void NextTrack() {
    MicroAudio.NextTrack();
}
public void PreviousTrack() {
    MicroAudio.PreviousTrack();
}
public void Stop(){
    MicroAudio.StopMusic();
}
```

Example: tracking progress of music track implementation

```
    private void Update() {
        trackProgressSlider.value = MicroAudio.CurrentTrackProgress;
    }
```

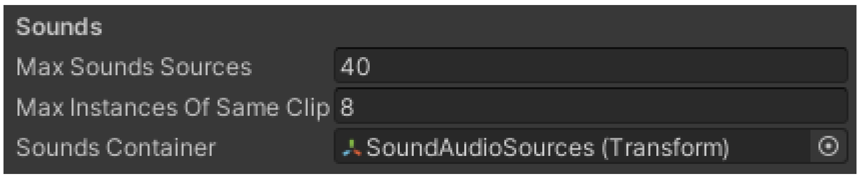
Sounds

The sound effects feature allows for the playing of sound effects. Audio sources for 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, float pitch = 1f, bool loop = false, AudioSource src = null` and UI counterpart, play sound effects in their respective audio channel. 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.

Just like music, sounds can be paused or stopped at any time with `PauseSounds` and `StopSounds`. However this feature is currently a bit limited. Pausing sounds doesn't allow for new sounds to be played. Pausing or stopping sounds will also affect other sounds features like delay of the sounds and Infinity Sound features. If You need more control over your sounds, for now those solutions will need to be custom but there are plans to implement grouping sound effects or pausing just specific channels.

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.


Sounds

Max Sounds Sources	40
Max Instances Of Same Clip	8
Sounds Container	SoundAudioSources (Transform)

Example: UI sound effects implementation

```
    public void UIButton1() {
        MicroAudio.PlayUISound(uiClip1);
    }
    public void UIButton2() {
```

```
    MicroAudio.PlayUISound(uiClip2);
}
public void UIButtonDelay() {
    MicroAudio.PlayUISound(uiClipDelayed, uiDelaySlider.value);
}
```

Example: Displaying message when sound effect ends

```
AudioSource transactionSource = MicroAudio.PlayUISound(transactionSound);
MicroAudio.OnSoundFinished += TransactionSoundFinished;

...

void TransactionSoundFinished(AudioSource src){
    if(src == transactionSource){
        DisplayMessage("Transaction complete!");
        MicroAudio.OnSoundFinished -= TransactionSoundFinished;
    }
}
```

Infinity Sound

Infinity sound is the new feature in Micro Audio manager. If you need some looping sound it can get stale and repetative really quickly. Infinity sound lets you define base loop sound that is played entire time and define list of random sounds which will be played while Infinity Sound is playing. Example would be actions that happen often and need sound effects but you don't know exact duration of the action like digging ground with wood or stone shovel, filling water tank of various sizes and so on. It also allows for defining start and end sounds which will always be the same.

Effect require setting up MicroInfinitySoundGroup which will be passed to the MicroAudio API function `PlayInfinityEffectSound` and `PlayInfinityUISound`. These functions return `MicroInfinityInstance` which can be controlled further.

MicroInfinityInstance API:

Properties	Description
`IsPaused`	*Is instance paused or not (read-only)

Methods	Description
`Pause`	Pauses Infinity Sound
`Resume`	Resumes paused Infinity Sound
`Stop`	Stops Infinity Sound

Events	Description
`OnEnd`	When Infinity Sound stops

Events	Description
`OnPausedChanged`	When Infinity Sound gets paused/resumed

Currently Infinity Sound doesn't support custom audio sources (for 3D sound purposes) but it is planned feature.

Sound Fade

Sound fade is a feature that can be used independently from the Micro Audio manager. The feature will work as long as there is a Micro Audio Manger 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 `IsPaused = false`, killed by `Kill()`, or fast forwarded to the end by `SkipToEnd()`.

Properties	Description
`IsPaused`	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
`SkipToEnd`	Skips fade to end, setting end value

Events	Description
`OnFadeEnd`	When fade reaches end sets final value
`OnPausedChanged`	When fade gets paused/resumed
`OnDestroy`	When fade ends, gets called even if killed

Delay Sound

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

Delay can be paused with `IsPaused = false`, killed by `Kill()` and fast forwarded to the end by `SkipToEnd()`. Delay can also be reset to 0 to start over again with the `ResetTimer()` method.

Example of delayed sound implementation can be seen under Sounds section.

Properties	Description

Properties	Description
`Source`	*AudioSource that is being delayed
`Delay`	*Duration of the delay
`Progress`	*Current progress of the delay (0-1)
`IsPaused`	Is delay paused or not

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

Events	Description
`OnDelayEnd`	When delay reaches end plays sound
`OnPausedChanged`	When delay gets paused/resumed
`OnDestroy`	When delay ends, gets called even if killed

Crossfade

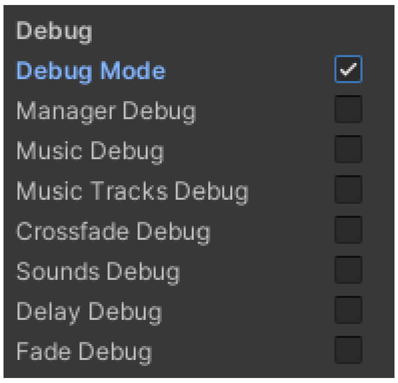
Crossfade allows the current song to gradually lower its volume at the end while the new song increases its volume and slowly takes over. The duration of the effect can be set by the `SetCrossfadeDuration` method or passed in as a parameter of the `PlayMusicGroup` method. The parameter for `PlayOneTrack`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



Debug

Debug Mode	✓
Manager Debug	
Music Debug	
Music Tracks Debug	
Crossfade Debug	
Sounds Debug	
Delay Debug	
Fade Debug	

Option	Description
`Debug Mode`	Turn debug mode on or off
`Manager Debug`	Messages from core manager
`Music Debug`	Messages about music feature, like playlist
`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
`Infinity Debug`	Status of Infinity Sound feature