Sonity Documentation

Sonity 1.1.2 Documentation FLYTTAD TILL GITBOOK YO

Audio Middleware for Unity SKRIV INTE HÄR

Table of Contents

1 Getting Started

1.1 What is Sonity?

Sonity is an audio middleware giving you full control of your sound, all within Unity.
Balancing ease of use with advanced features it will help you to quickly make your game sound good.

- **New in 1.1.0:** Steam Audio & PlayMaker integration.
- **New in 1.0.6:** Increase the volume of your sounds above 0dB.
- Components for triggering sounds without code.
- Live-edit and multi-edit everything with scriptable objects.
- Use SoundParameters to control your sound in real time.
- Automatic asset creation and reference finder.
- Music functionality like playlists, stems and intros.
- Visually debug sounds live in-game.
- Custom DSP filters and distortion per voice.
- Highly optimized system with IL2CPP support.
- Fully documented source code provided.
- Developed and used in commercial games.
- Crafted with my experience as a game audio professional.

Note about licences:
The Sonity Unity Asset Store license is for a single user at a time.
If multiple people use Sonity, you need multiple licenses.

Helpful Links and Info

- Website
- Free Trial - Unity Asset Store
- Paid Version - Unity Asset Store
- Video Tutorials

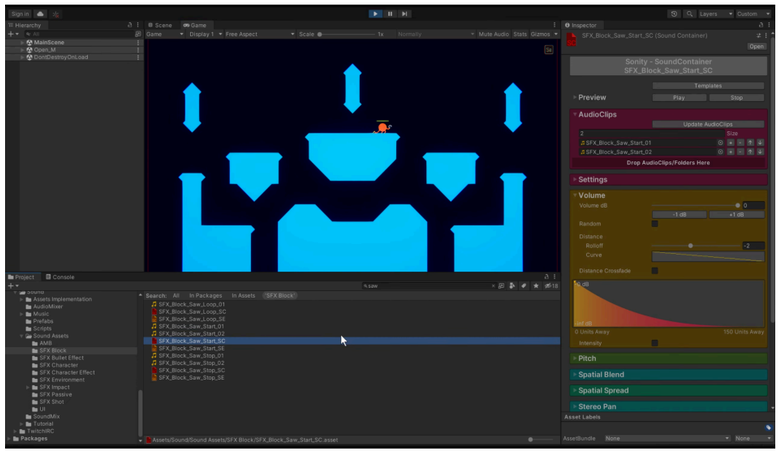
- Documentation
- Discord
- FAQ

1.2 Video Tutorials

Sonity - Showcase

Link to video
0:00 Intro
0:19 Features
1:31 Outro

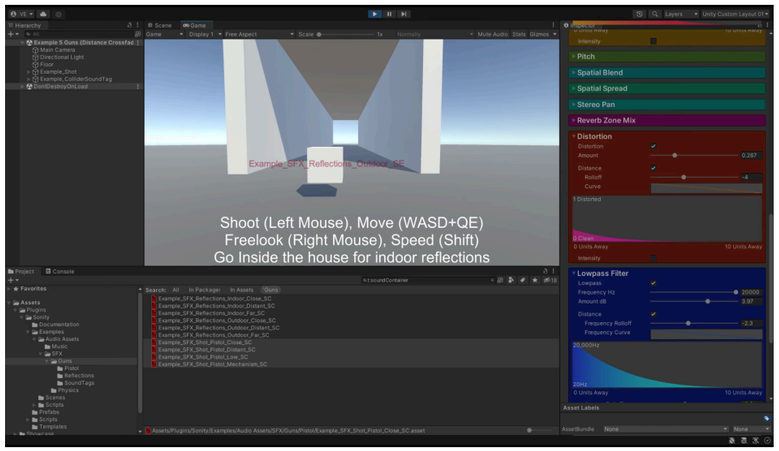
Sonity - Quickstart Tutorial



Link to video
0:00 Intro
0:19 Installation

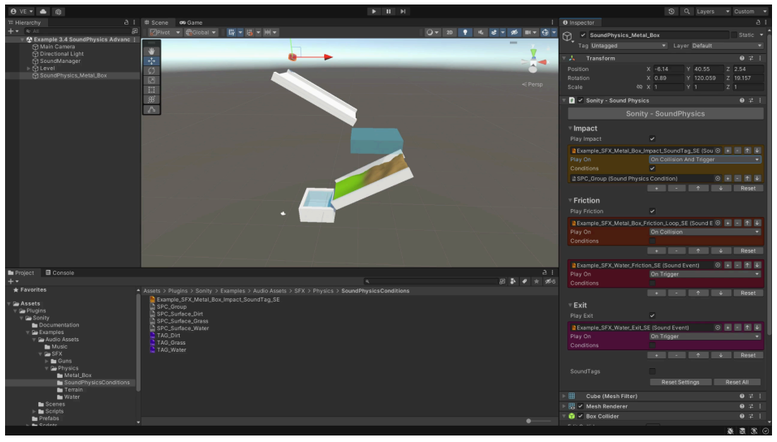
0:49 Asset Creation
1:41 Asset Editing
2:53 SoundTrigger Implementation
3:15 C# Implementation
4:34 Outro

Sonity - Realistic Gun Sounds Tutorial



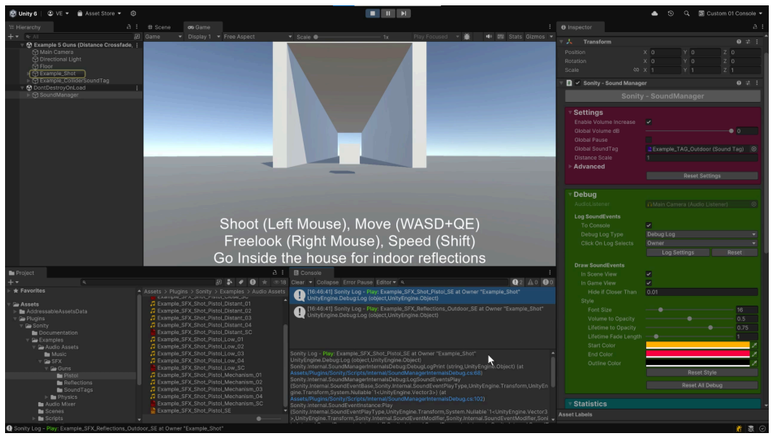
Link to video
0:00 Intro
0:45 Distance Crossfades
3:22 Pistol Shot Editing
4:30 Adding SoundTags
5:44 Play Shot Sound in C#
6:42 Set Global SoundTags
8:07 Mixing Reflections
8:46 Using the Lowpass Filter
10:00 Using Distortion
10:41 Outro

Sonity - SoundPhysics & Intensity Tutorial



Link to video
0:00 Introduction
1:45 SoundPhysics Basic: Impact and Intensity, Crossfading velocity layers
9:55 SoundPhysics Intermediate: Friction, Exit and OnTrigger
16:56 SoundPhysics Advanced A: Using SoundPhysicsConditions
21:52 SoundPhysics Advanced B: Using SoundTags
32:09 Outro

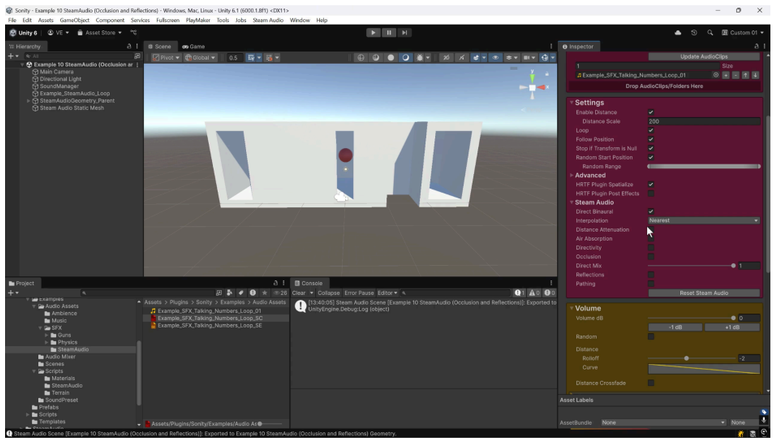
Sonity 1.0.6 - The Big Volume & Debug Update



Link to video
0:15 Introduction
0:29 SoundEvent Base Settings
1:02 AudioMixerGroup Legacy Conversion
1:34 Enable Volume Increase
3:41 Log SoundEvents
4:16 SoundEvents Debug Settings

4:44 Editor Tools
5:21 Editor Tools - Reference Finder
5:34 Editor Tools - Select Same Type
5:54 Editor Tools - Selection History
6:29 Notes Fields
6:49 SoundPresets
8:44 AudioListenerDistance
10:37 Outro

Sonity 1.1.1 - Steam Audio, PlayMaker + tips & tricks



Link to video
0:08 Whats New
0:25 Steam Audio Integration
5:40 PlayMaker Integration
6:37 2D/3D Information Panel
8:04 Tip: Volume Increase with AudioMixerGroups
9:32 Tip: Legacy support for PlayMusic and Play2D
10:13 Outro

1.3 Installation

1. After purchase, go to the Package Manager, show “My Assets” and search for “Sonity”.
2. Then click on “Sonity” and press “Download”, then “Import”.
3. It should be located at “Assets\Plugins\Sonity”
4. If materials are pink, select all “ExampleMaterial” and upgrade to URP with:
   Edit -> Rendering -> Materials -> Convert Selected Built-in Materials to URP.

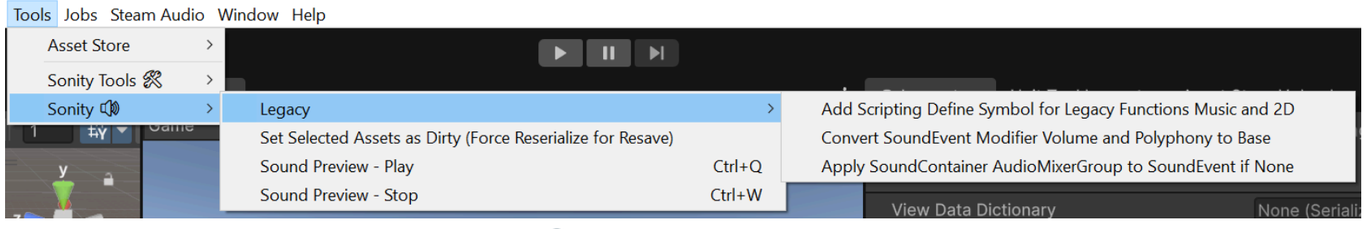
1.4 Upgrade

1. Backup your project before adding or updating Sonity.
2. Delete the whole Sonity folder if it says so in the Sonity Changelog.txt.
3. Import and install the latest version of Sonity from the Asset Store.

4. Restart Unity if needed.

Legacy Support

Sonity has legacy support for deprecated functions and conversions which can upgrade old assets to new standards.



All legacy conversion tools are located at: “Tools/Sonity 🔊/Legacy/”

- Add Scripting Define Symbol for Legacy Functions Music and 2D (Renamed Music/2D Functions in 1.1.0)
- Convert SoundEvent Modifier Volume and Polyphony to Base (Added SoundEvent Base Settings in 1.0.6)
- Apply SoundContainer AudioMixerGroup to SoundEvent if None (Removed SoundContainer AudioMixerGroup in 1.0.6)

> **💡 TIP - Upgrading**
>
> If you’re upgrading from an earlier version you might need to remove the whole “Sonity” folder in the project.
> And if you´re still having trouble, try removing the “.asmdef” files, search for “t:asmdef Sonity.” and then re-import those files again.
> Also, if you have your own .asmdef files in your project, you might have to reference both Sonity Runtime .asmdef files to use Sonity.

How to Upgrade Free Trial

Just delete the “Sonity” folder and import the paid version of Sonity, all assets created with the free trial will continue to work.

Supported Unity Versions

- **Unity 6, 2022, 2021, 2020, 2019 and IL2CPP**
  Tested, full functionality.

- **Unity 2018.3.0**
  Tested, everything works except “Draw SoundEvents (In Scene View)” and Terrain example.
  Minor GUI tweaks for compatibility.

- **WebGL**
  Everything works except DSP (distortion/lowpass/highpass).

- **Disabling Domain Reloading**
  Disabling domain reloading in the “enter play mode options” is supported since Sonity 1.0.3.

- **Support for Addressables**
  Adding and deleting SoundEvents in runtime is supported since Sonity 1.0.5.
  Object comparison with cached GUIDs is supported since Sonity 1.0.6.

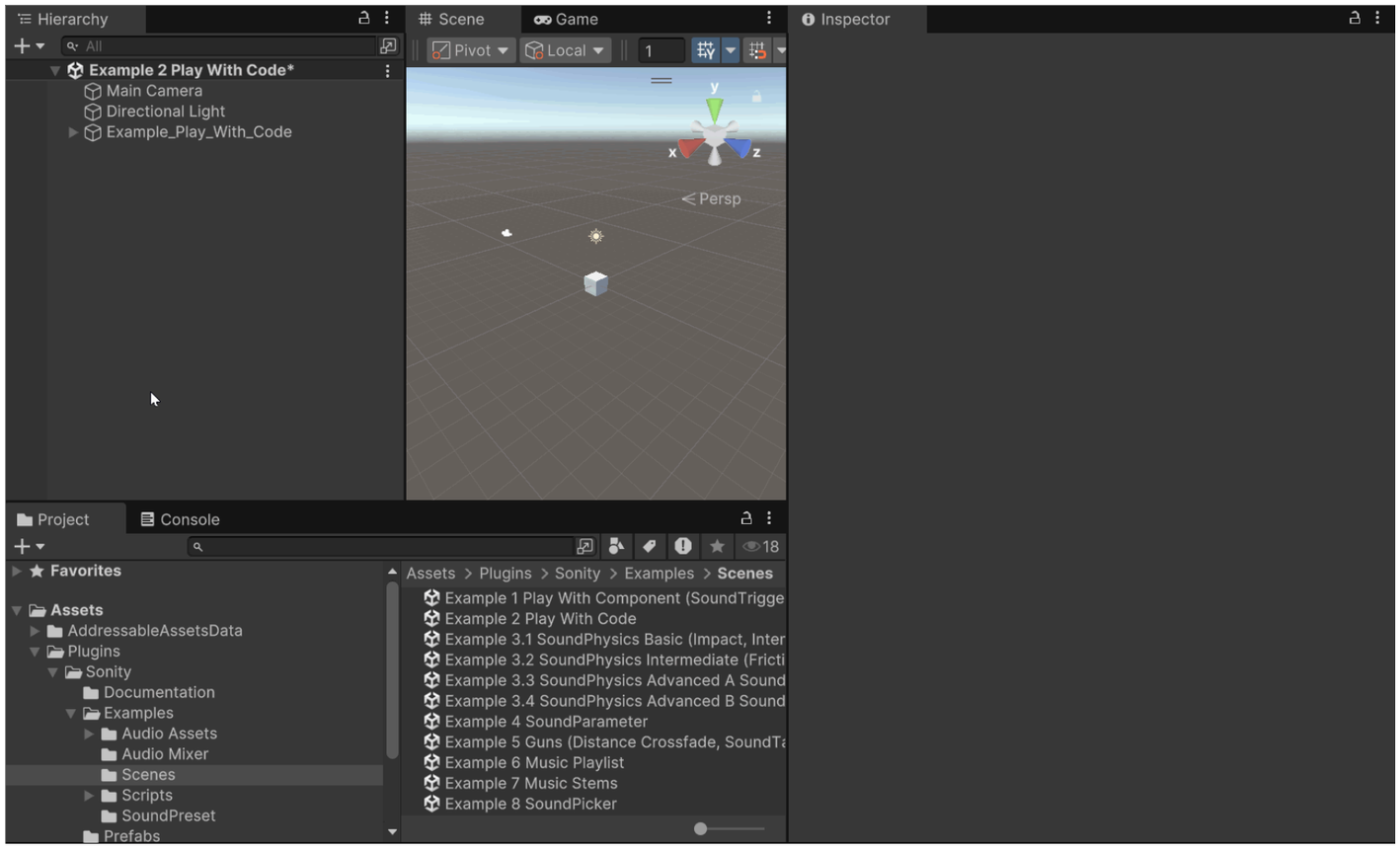
Addressable AudioMixer support since Sonity 1.0.6.

1.4 Adding the SoundManager

The SoundManager is used to play sounds and manage global settings.
An instance in the scene is required to play SoundEvents.
You can either add the pre-made prefab called "SoundManager" or add the "Sonity - Sound Manager" component to an empty GameObject in the scene.

> *Tip: Don't forget to set up the Distance Scale to match the scale of your project.*

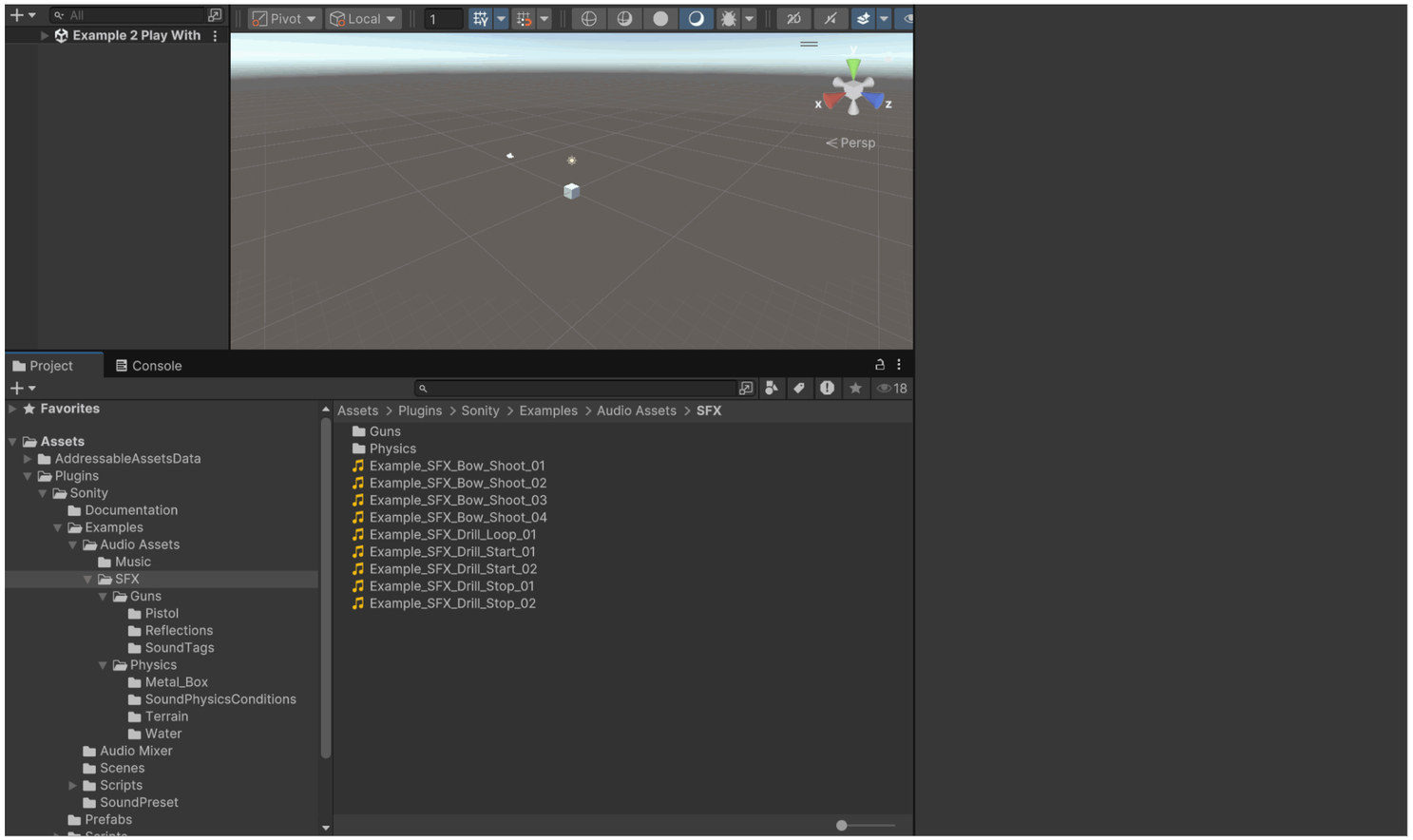
More info at SoundManager Component.



1.5 Asset Creation

The asset hierarchy for playing sounds is:
AudioClip > SoundContainer > SoundEvent
If you have a bunch of AudioClips in your project, just select them (or their folder).
Then right click to bring up the "Create" > "Sonity" > "Create Assets from Selection" menu and press the desired button.

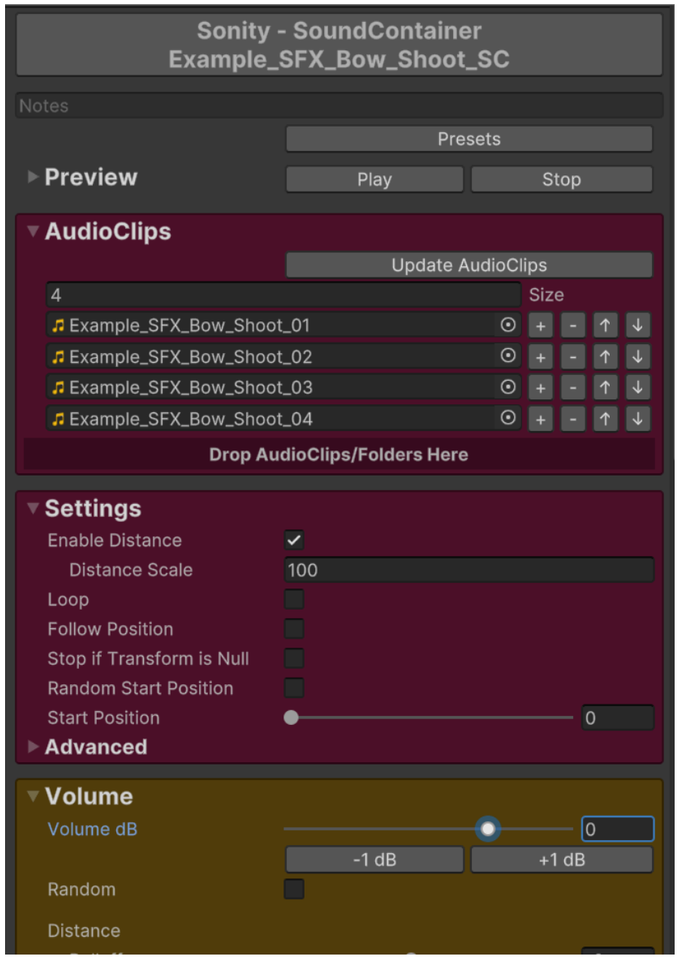
More info at [Create Assets From Selection].



1.6 Asset Editing

The [SoundContainers] and [SoundEvents] contain the settings of how your sounds are played.
They are made of scriptable objects, so all changes you make in runtime are updated live and saved.
All Sonity scriptable objects and components are multi-object editable.

SoundContainer Intro

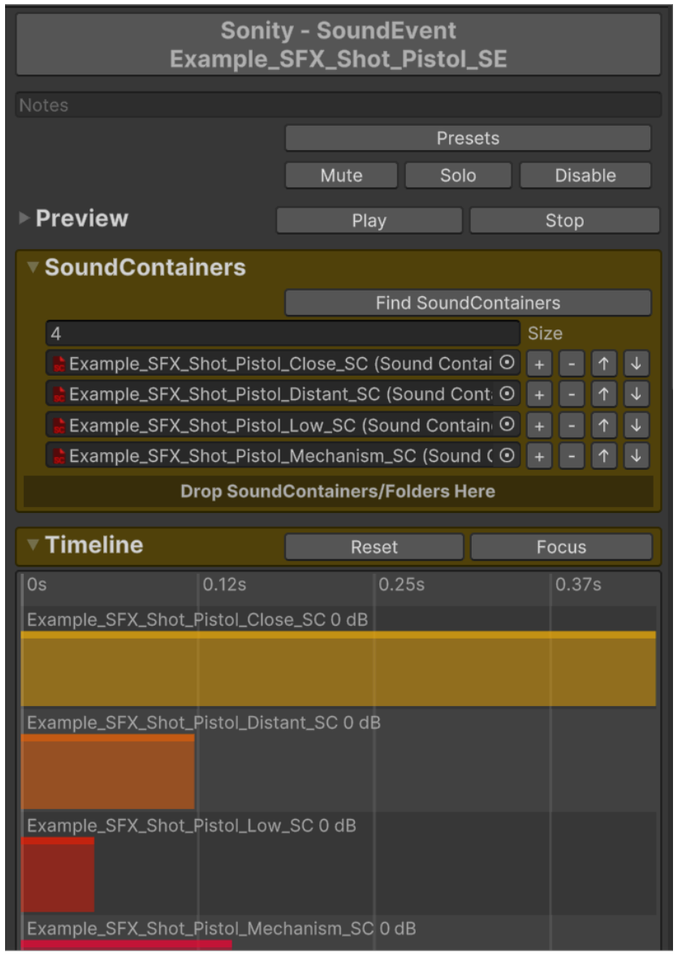


SoundContainers are the building blocks of Sonity.
They contain AudioClips and options of how the sound should be played.
All SoundContainers are multi-object editable.

Overview:
Looping
Volume
Pitch
Spatial Blend
Spatial Spread
Stereo Pan
Reverb Zone Mix
Distortion
Lowpass Filter
Highpass Filter

❗ *Tip: Don't forget to use the presets.*
More info at SoundContainer.

SoundEvent Intro



SoundEvents are what you play in Sonity.
They contain SoundContainers and options of how the sound should be played.
All SoundEvents are multi-object editable.

Overview:
Timeline
Modifier volume, pitch, fade in/out, etc.
Intensity.
Trigger another SoundEvent on Play, Stop, Tail, Tag.

More info at SoundEvent.

1.7 Shortcuts

These are the default shortcuts used by Sonity and Sonity Tools.
You can customize them at Edit -> Shortcuts.

Shortcut	Command
	Assets/Create/Sonity 🎧 /Create Assets From Selection/SC from AudioClips Multiple
	Assets/Create/Sonity 🎧 /Create Assets From Selection/SC from AudioClips Single
Ctrl+Shift+Q	Assets/Create/Sonity 🎧 /Create Assets From Selection/SC+SE from AudioClip Group Multiple SE
Ctrl+Shift+W	Assets/Create/Sonity 🎧 /Create Assets From Selection/SC+SE from AudioClip Group Single SE
	Assets/Create/Sonity 🎧 /Create Assets From Selection/SE from SC Multiple
	Assets/Create/Sonity 🎧 /Create Assets From Selection/SE from SC Single
	Assets/Create/Sonity 🎧 /SoundContainer
	Assets/Create/Sonity 🎧 /SoundDataGroup
	Assets/Create/Sonity 🎧 /SoundEvent
	Assets/Create/Sonity 🎧 /SoundMix
	Assets/Create/Sonity 🎧 /SoundPhysicsCondition
	Assets/Create/Sonity 🎧 /SoundPolyGroup
	Assets/Create/Sonity 🎧 /SoundPreset
	Assets/Create/Sonity 🎧 /SoundTag
	Assets/Create/Sonity 🎧 /SoundVolumeGroup
Ctrl+Alt+Shift+F	Assets/Sonity Tools 🧩 /Reference Finder 🔍 /Find References for Selected Assets
Ctrl+Alt+A	Assets/Sonity Tools 🧩 /Reference Finder 🔍 /Open Reference Finder Window
Ctrl+Alt+Shift+A	Assets/Sonity Tools 🧩 /Select Same Type 👆 /In Folder
	Assets/Sonity Tools 🧩 /Select Same Type 👆 /In Subfolders
U	Assets/Sonity Tools 🧩 /Selection History 📜 /Back
Shift+U	Assets/Sonity Tools 🧩 /Selection History 📜 /Forward
	Component/Scripts/Sonity/Entity Sound Manager
	Component/Scripts/Sonity/Entity Sound Trigger Authoring
	Component/Sonity 🎧 /Sonity - Audio Listener Distance
	Component/Sonity 🎧 /Sonity - Sound Physics
	Component/Sonity 🎧 /Sonity - Sound Physics 2D
	Component/Sonity 🎧 /Sonity - Sound Physics 2D No Friction
	Component/Sonity 🎧 /Sonity - Sound Physics No Friction
	Component/Sonity 🎧 /Sonity - Sound Trigger
	Component/Sonity/Template - Sound Music Manager
	Component/Sonity/Template - Sound Play UI
	Component/Sonity/Template - Sound Volume Manager
Ctrl+Q	Tools/Sonity 🎧 /Sound Preview - Play
Ctrl+W	Tools/Sonity 🎧 /Sound Preview - Stop

	Tools/Sonity Tools 🧩 /Reference Finder 🔍 /Find References for Selected Assets
	Tools/Sonity Tools 🧩 /Reference Finder 🔍 /Open Reference Finder Window
	Tools/Sonity Tools 🧩 /Select Same Type 👆 /In Folder
	Tools/Sonity Tools 🧩 /Select Same Type 👆 /In Subfolders
	Tools/Sonity Tools 🧩 /Selection History 📜 /Back
	Tools/Sonity Tools 🧩 /Selection History 📜 /Forward

Create Assets From Selection Shortcuts

Ctrl+Shift+Q - SC+SE from AudioClip Group Multiple SE
Ctrl+Shift+W - SC+SE from AudioClip Group Single SE

Preview Shortcuts

Ctrl+Q - Play (Previews the selected SoundEvent or SoundContainer).
Ctrl+W - Stop (Stops any playing preview of SoundEvents and SoundContainers, press two times to skip fade out).

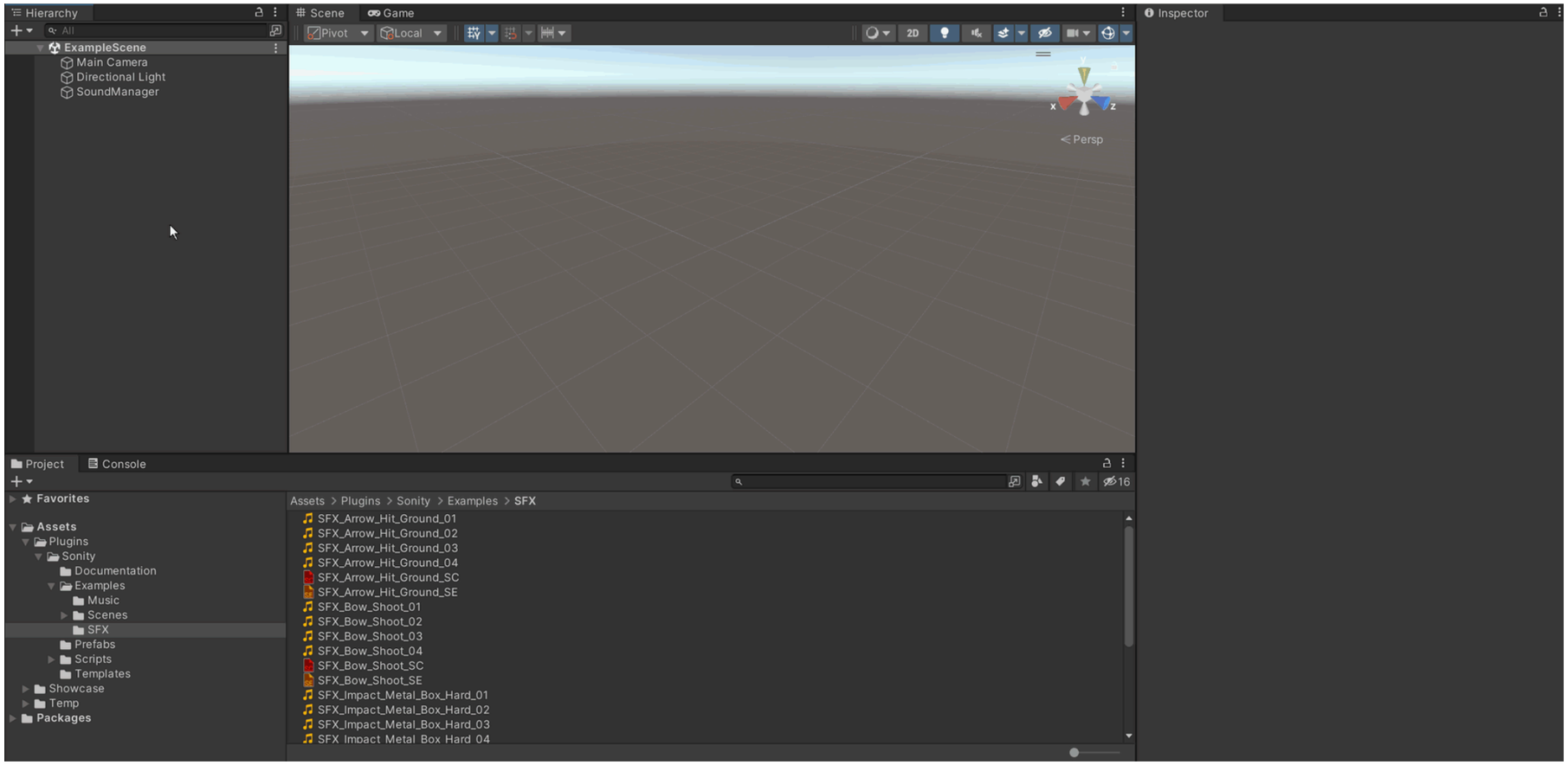
Timeline Shortcuts & Controls

Zoom: Mousewheel scroll.
Pan: Mousewheel hold and drag (or left mouse button also if outside of item).
Volume: Hold and drag top of item up/down or click on the volume to write the decibel value.
Move item: Hold and drag on item left/right.
Focus on items: F

1.8 Playing and Stopping Sounds with SoundTrigger Basics

To play a SoundEvent, create a gameobject and add a SoundTrigger component to it.
Then select the SoundEvent you want to play and when you want it to play like in the example below:

More info at SoundTrigger.



1.9 Playing and Stopping Sounds with C# Basics

To play a SoundEvent, create a C# script and include the “Sonity” namespace.
Then make a public “SoundEvent” property, select the desired SoundEvent in the inspector.
Use the functions of the SoundEvent to play and stop your SoundEvents like in the example below:

More info at SoundEvent Functions and the SoundManager Functions.

Example code:

```
using UnityEngine;
using Sonity;

public class PlayStopExample : MonoBehaviour {

    public SoundEvent soundEvent;

    void PlayExample() {
```

```
        // Plays the SoundEvent at the position of the transform
        soundEvent.Play(transform);
    }

    void StopExample() {
        // Stops the SoundEvent playing at the transform
        soundEvent.Stop(transform);
    }
}
```

1.10 Templates & Examples

Sonity contains examples and script templates of advanced functionality which you can use in your game.

Example_AudioMixer.mixer

It is an example AudioMixer with groups such as Example_AMB, Example_MUS, Example_SFX, Example_UI and Example_VO.

TemplateSoundMusicManager.cs

Template of a singleton music playback system.
Add to a GameObject in the scene and use like this:

```
SonityTemplate.TemplateSoundMusicManager.Instance.PlayMainMenu();
```

TemplateSoundPlayUI.cs

Template of a singleton used to play e.g. UI sounds.
Useful when you want to play SoundEvents in a lot of places through code.
Add to a GameObject in the scene and use like this:

```
SonityTemplate.TemplateSoundPlayUI.Instance.PlayButtonClick();
```

TemplateSoundVolumeManager.cs

Template of a singleton AudioMixer volume controller.
The default settings work with the provided "Example_AudioMixer.mixer".
Add to a GameObject in the scene and use like this:

```
SonityTemplate.TemplateSoundVolumeManager.Instance.SetVolumeMaster(1f);
```

1.11 Unity Native Audio Support

Sonity is built around the native audio system in Unity, so everything that Unity audio supports Sonity also supports.
You can use built in features like:

AudioClip

Settings for your audio data.

> **💡 TIP - AudioClip Settings**
>
> Music and longer sounds should probably be set to streaming as to use less memory.
> If you have problems with e.g. stuttering audio, try changing the Load Type of the AudioClips.
> Read more on: How To - Optimize AudioClip Settings.

AudioMixer

For channel based effects like reverb sends, compression, ducking etc. Contains AudioMixerGroups, which SoundContainers and SoundEvents can be assigned to output into.
Can also expose parameters to enable real time editing of e.g. volume and reverb effects (see for example TemplateSoundVolumeManager).

> **💡 TIP - AudioMixer Performance Tips**
>
> If you have issues with audio performance you might want to make sure you're not using an unnecessary amount of AudioMixerGroups (like a hundred of them).
> Because changing AudioMixerGroup for a Voice uses performance (when playing a new Voice Sonity tries to find an unused Voice which uses the same AudioMixerGroup).
> Use AudioMixerGroups when you want effects per group or e.g. ducking or for larger groups of sounds, but maybe not a have a unique one for each sound.

Audio Mixer Snapshot

For switching between different effect settings in the Audio Mixer.

Reverb Zones

For reverb with different settings in different zones.

Audio Project Settings

For global settings of the audio like Buffer Size (latency), Speaker Mode (5.1, surround etc) and Spatializer Plugin (VR, vertical spatialization, steam audio, oculus/meta spatializer etc).

> **💡 TIP - Audio Buffer Size & Voice Count**
>
> If you experience audio performance problems/stutter/dropouts you might want to raise your DSP Buffer Size to Good Latency or Best Performance.
> And maybe lower the number of voices your project is allocated.
> 32 Real Voices should be sufficient for most games and you really never need to go above 64 Real Voices in my experience.

Audio Listener

Need to be placed in the scene for audio to be heard. Is usually placed on the main camera or player.

1.12 How To

This section is filled with answers for commonly asked questions.

How To - Optimize AudioClip Settings

Here are some general settings you'd want for most games get lower RAM usage, disk read and CPU usage.

Unity Manual - AudioClip (Read more about the other settings).

AudioClip - Overall Settings
Compression Format:Vorbis
Quality: 60. Which is ~192 kbit/s for Vorbis. You can go lower if you'd want to save space but it comes at a loss of quality.
Sample Rate Setting: Should be the same as the sample rate of the platform, which today mostly is 48kHz (this is so that it doesnt have to do a sample rate conversion) when playing.
Also, if the sounds are created in 48kHz there is more high frequency content which makes it sound better when pitching down vs 44.1kHz.

❗ *Tip: If you select multiple AudioClips, you can see their total "Imported Size" which is the size they'll have after all your format and quality settings are applied.*

AudioClip - Vorbis Quality
Unity Vorbis Quality 1 is VBR Quality 0 (not -2). Mono is half the bitrate of stereo and 4 channel files (for surround) are twice the bitrate of stereo.

Unity Vorbis Quality	VBR Target kbit/s stereo	VBR Target kbit/s mono	VBR Range kbit/s stereo	Noise Test kbit/s stereo	Size % vs 100	Size % vs 60	Comment
100	500	250	500 - 1000	554	100%	260%	Uneccessarely big, don't use this
90	320	160	320 - 500	435	64%	167%	Uneccessarely big
80	256	128	256 - 320	382	51%	133%	Use if you want to mazimize quality & have the disk space and RAM
70	224	112	224 - 256	309	45%	117%	
60	192	96	192 - 224	232	38%	100%	Use this as a default, it is a good balance between quality and size
50	160	80	160 - 192	173	32%	83%	Sounds like near CD quality
40	128	64	128 - 160	128	26%	67%	
30	112	56	112 - 128	117	22%	58%	Smaller & better sound than .mp3 128kbps
20	96	48	96 - 112	102	19%	50%	
10	80	40	80 - 96	87	16%	42%	Sounds acceptable. Use if you are desparate for saving space and RAM
1	64	32	64 - 80	71	13%	33%	Looses a lot of quality, noticeably stereo separation and bass

Notes: Noise test is calculated using VBR export & size comparison to CBR with a known bitrate.
Sources: Recommended Encoder Settings and Vorbis Quality. The bitrates are approximate.

AudioClip - Load Type
Stereo files are double the size of mono files. So a 60s 2 channel sound is the same size as a 120s 1 channel sound. The lengths are approximated with stereo files in mind.

Length Seconds ~	Play Frequency	Load Type	Pros	Cons	Example Sounds
Short - Less than 1s	High	Decompress On Load	Uses less CPU	Uses 10x RAM so only use for short sounds.	Player sounds, footsteps, impacts etc Commonly used for mobile platforms
Short - Less than 1s	Low	Compressed In Memory	Uses less RAM		
Medium - More than 1s less than 60s	High & Low	Compressed In Memory	Uses less RAM		
Long - More than 60s	High	Compressed In Memory	Uses less disk read	Try if you have problems with audio delays Watch out for load times and RAM usage	Localized 3D ambiences etc
Long - More than 60s	Low	Streaming	Uses less RAM	Many platforms can only handle less than 10 simultaneously streaming sounds	Music, background 2D ambiences etc

Note: You want to stream the largest files you have, but not too many at the same time.
So the 60s threshold for long sounds can be lowered to e.g. 30s or even lower if not many sounds are +60s.

AudioClip - Preload Audio Data
Makes the sound load at startup, which might create long load times on harddrives and slower systems.
Is useful to combine with Load In Background so it doesn't interrupt the main thread.

Length in seconds	Play Frequency	Preload Audio Data	Pros	Cons	Example Sounds
Short - Less than 1s	High	Yes	Less frequent disk read	Use only for frequent & short sounds Combine with Load in Background	Player sounds, footsteps, impacts etc
Short - Less than 1s	Low	No			
Medium & Long - More than 1s	Low	No			

Note: I've had problems where if I don't Preload Audio Data, there might be clicks heard in the end of sounds.
"Prevent End Clicks" in the SoundContainer mitigates this.

AudioClip - Load In Background
When enabled, the AudioClip is loaded without stalling the main thread and any playing is delayed until the AudioClip is loaded.
Useful to combine with Preload Audio Data so it doesn't interrupt the main thread.

How To - Debug Sounds and Performance

If you want to debug audio performance or what sounds are playing, you can use:
- Unity built in Profiler
- SoundManager Statistics

Unity Profiler

Read more here.
Because Sonity is built upon Unity native AudioSources, you can use the built in Profiler.

Example picture of the audio part of the profiler in "Detailed" mode (bottom left) with deep profile enabled:

Object	Asset	Volume	VU Level	↑Audibility	Group	Priority	Plays	3D	Paused	Muted	Virtual	OneShot	Looped	Distance
Voice 10	SFX_Shot_Basic_Single_05	-16.8 dB		-16.8 dB	SFX	237	2	YES	NO	NO	NO	NO	NO	50.8
Voice 15	SFX_Shot_Basic_Single_03	-16.9 dB		-16.9 dB	SFX	237	2	YES	NO	NO	NO	NO	NO	51.3
Voice 4	SFX_Environment_Saw_Loop_01	-16.9 dB		-16.9 dB	SFX	237	4	YES	NO	NO	NO	NO	YES	51.3
Voice 8	SFX_Environment_Saw_Damage_01	-17.0 dB		-17.0 dB	SFX	237	1	YES	NO	NO	NO	NO	NO	51.5
Voice 0	SFX_Environment_Saw_Loop_01	-17.0 dB		-17.0 dB	SFX	237	4	YES	NO	NO	NO	NO	YES	51.5
Voice 3	SFX_Impact_Environment_Basic_04	-17.7 dB		-17.7 dB	SFX	237	1	YES	NO	NO	NO	NO	NO	51.8
Voice 13	SFX_Environment_Box_Impact_Soft_08	-19.4 dB		-19.4 dB	SFX	241	1	YES	NO	NO	NO	NO	NO	49.5
Voice 2	SFX_Character_Die_01	-21.3 dB		-21.3 dB	SFX	244	1	YES	NO	NO	NO	NO	NO	52.6
Voice 9	SFX_Character_Reload_In_Progress_Loop_01	-22.7 dB		-22.7 dB	SFX	246	3	YES	NO	NO	NO	NO	YES	51.7
Voice 1	SFX_Character_Reload_In_Progress_Loop_01	-23.1 dB		-23.1 dB	SFX	246	3	YES	NO	NO	NO	NO	YES	51.9
Voice 7	SFX_Character_Land_07	-23.7 dB		-23.7 dB	SFX	247	1	YES	NO	NO	NO	NO	NO	51.7
Voice 6	SFX_Character_Land_06	-32.0 dB		-32.0 dB	SFX	252	1	YES	NO	NO	NO	NO	NO	51.7
Voice 5	SFX_Environment_Rope_Creak_Loop_01	-43.8 dB		-43.8 dB	SFX	254	100	YES	NO	NO	NO	NO	YES	63.4
Voice 11	SFX_Character_Reload_In_Progress_Loop_01	-52.9 dB		-52.9 dB	SFX	255	3	YES	NO	NO	NO	NO	YES	122.2
Voice 12	SFX_Environment_Box_Impact_Hard_06	-79.3 dB		-79.3 dB	SFX	255	1	YES	NO	NO	NO	NO	NO	49.5

SoundManager Statistics

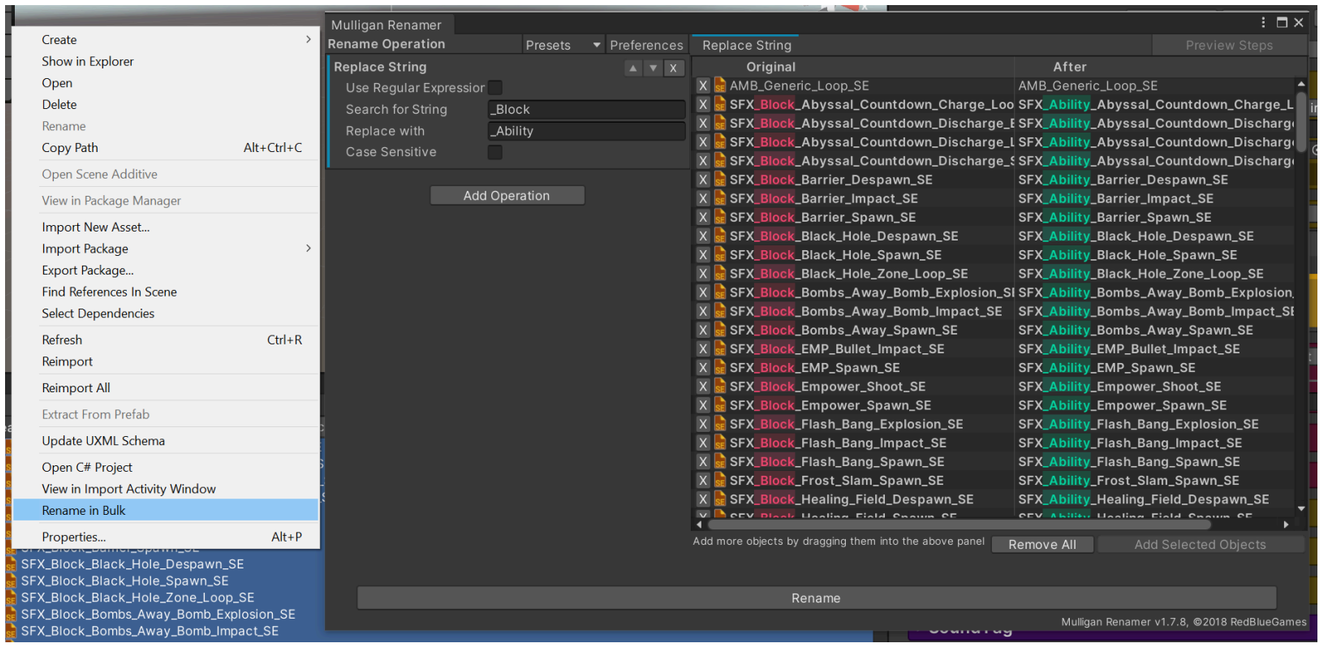
Read more here.

Example picture of the the SoundManager:

How To - Rename A Lot of Assets

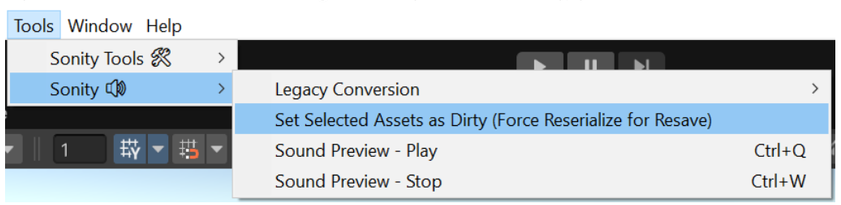
If you need to rename a lof of assets in Unity I can recommend the free tool called “Mulligan Renamer”.
Download at Unity Asset Store or Github.

Example of use:

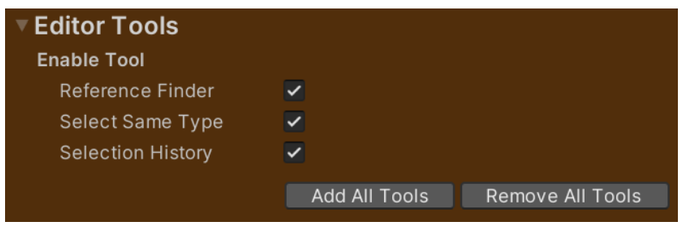


How To - Force Resave Assets

If you want to force save all changes for any assets in Unity, you can use the tool "Set Selected Assets as Dirty (Force Reserialize for Resave)".

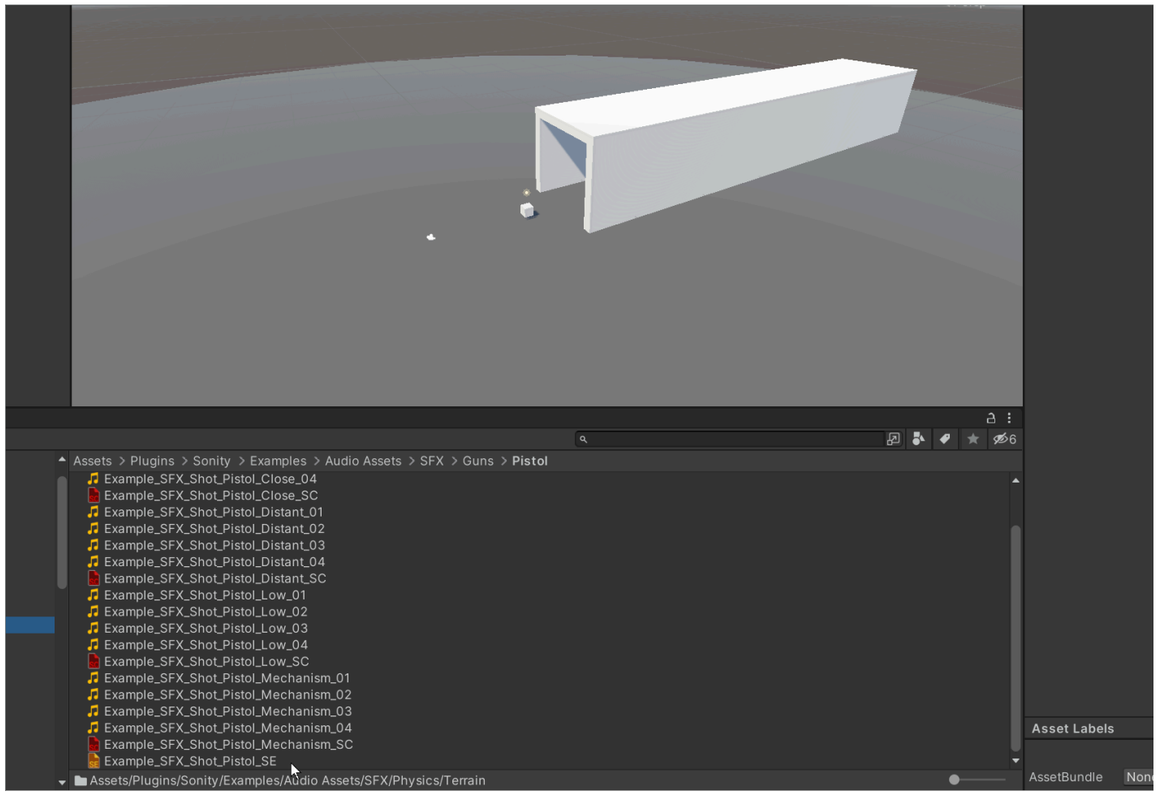


1.13 SoundManager Editor Tools



In the SoundManager there is a part which is dedicated to Editor Tools.
This is a collection of nifty editor tools which you can use to speed up your workflow.
They are enabled with the use of “Script Define Symbols” in Project Settings -> Player -> Other Settings -> Script Compilation -> Script Define Symbols.
This way they can be bound to default hotkeys and they won’t hog up your toolbars if you don’t use them.

Editor Tool - Reference Finder



Reference Finder is an editor tool for finding where an asset is referenced by another asset.
It is enabled with the use of the Script Define Symbol "SONITY_ENABLE_EDITOR_TOOL_REFERENCE_FINDER".
This is assigned in Project Settings -> Player -> Other Settings -> Script Compilation -> Script Define Symbols.
"Find References for Selected Assets" is useful for e.g. figuring out to which (if any) prefabs a SoundEvent is assigned to.
"Open Reference Finder Window" is useful for figuring out e.g. which AudioClips are completely unreferenced and therefore unused.

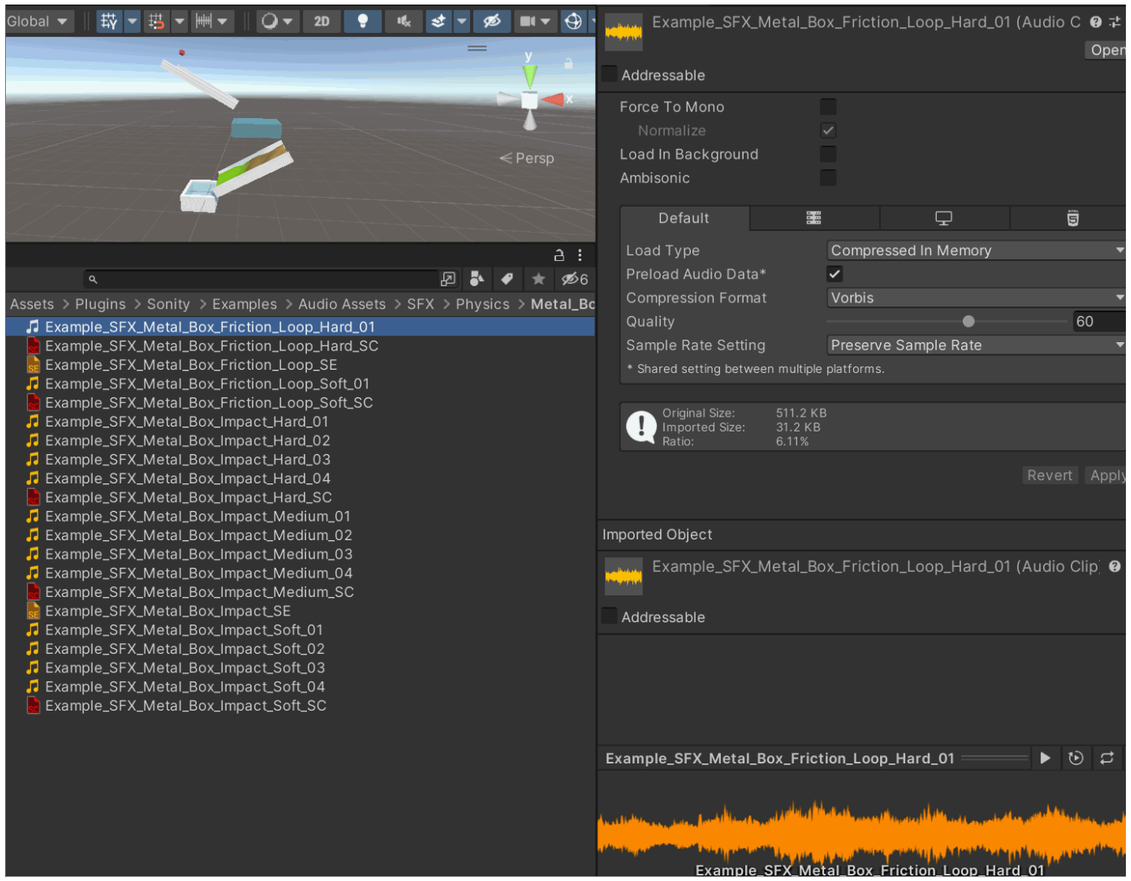
❕ *Tip: Close the window before renaming or moving any file in the project or it might lag a lot.*

Toolbar Actions
Tools/Sonity Tools 🛠️/Reference Finder 🔍/Open Reference Finder Window
Tools/Sonity Tools 🛠️/Reference Finder 🔍/Find References for Selected Assets
Assets/Sonity Tools 🛠️/Reference Finder 🔍/Open Reference Finder Window
Assets/Sonity Tools 🛠️/Reference Finder 🔍/Find References for Selected Assets

Default Shortcuts
Ctrl+Shift+Alt+F - Reference Finder 🔍/Find References for Selected Assets

Reference Finder is made by Alexey Perov and licensed under a free to use and distribute MIT license.
Link to Github page

Editor Tool - Select Same Type

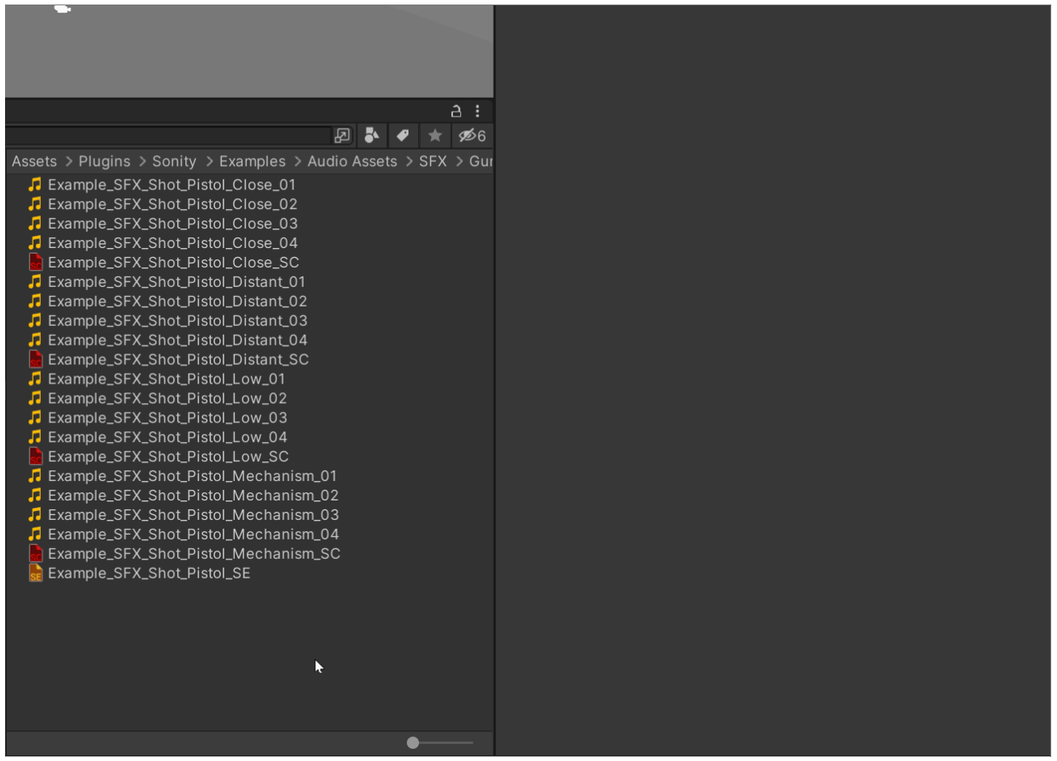


Select Same Type is an editor tool for quickly selecting all assets of the same type in a folder which enables you to quickly edit a lot of assets.
It is enabled with the use of the Script Define Symbol "SONITY_ENABLE_EDITOR_TOOL_SELECT_SAME_TYPE".
This is assigned in Project Settings -> Player -> Other Settings -> Script Compilation -> Script Define Symbols.

Toolbar Actions
Tools/Sonity Tools 🛠️/Select Same Type 👊/In Same Folder
Tools/Sonity Tools 🛠️/Select Same Type 👊/In Subfolders
Assets/Sonity Tools 🛠️/Select Same Type 👊/In Same Folder
Assets/Sonity Tools 🛠️/Select Same Type 👊/In Subfolders

Default Shortcuts
Ctrl+Alt+A - Select Same Type 👊/In Same Folder
Ctrl+Alt+Shift+A - Select Same Type 👊/In Subfolders

Editor Tool - Selection History



Selection History is an editor tool for quickly undoing and redoing selections.
This enables you to quickly move between objects you've previously selected.
It is enabled with the use of the Script Define Symbol "SONITY_ENABLE_EDITOR_TOOL_SELECTION_HISTORY".
This is assigned in Project Settings -> Player -> Other Settings -> Script Compilation -> Script Define Symbols.

Toolbar Actions
Tools/Sonity Tools 🛠️/Selection History 📜/Back
Tools/Sonity Tools 🛠️/Selection History 📜/Forward
Assets/Sonity Tools 🛠️/Selection History 📜/Back
Assets/Sonity Tools 🛠️/Selection History 📜/Forward

Default Shortcuts
U - Selection History 📜/Back
Shift+U - Selection History 📜/Forward

Selection History is made by Matthew Miner and licensed under a free to use and distribute MIT license.
[Link to Github page]

1.14 Sonity Changelog

Here is a list of all the changes made to each released version of Sonity.

1.1.1 - 2D/3D Info & Fixes (2025 September)
Please remove the whole Sonity folder before upgrading.
New Feature: Added 3D/2D sound info for the SoundContainer and SoundEvents for better overview.
New Feature: Added SoundContainer Distance Scale info and batch replace when multiple objects are selected.
New Feature: Added SoundContainer preset "UI" and and fixed so all Presets set Reverb Zone Mix.
Legacy: Support for renamed music & 2D functions with "Tools/Sonity 🔊/Legacy/Add Scripting Define Symbol for Legacy Functions Music and 2D"
Fix: If materials are pink, select all "ExampleMaterial" and upgrade to URP with Edit -> Rendering -> Materials -> Convert Selected Built-in Materials to URP.
Fix: Fixed so Log SoundEvents Owner will show the proper owner when Polyphony Mode Limited Globally is used.
Fix: Fixed bug where steam audio sources werent force-updated, which made occlussion not work.
Fix: Fixed SoundEvent mute, solo and disable color transparency.
Fix: SoundPreset when creating assets added SetDirty.
Fix: Editor Sound Preview AudioListener fixed HideFlags.
Fix: Added nullchecks to SoundManager Log and changed the default values.
Fix: Changed so all asmdefs use GUIDs.
Documentation: Added info about legacy support in documentation.

1.1.0 - Steam Audio & PlayMaker Integration (2025 June)
Please remove the whole Sonity folder before upgrading.
Rename: Renamed Play2D() to UIPlay() and PlayMusic() to MusicPlay() with the same pattern for all all other Music & 2D/UI functions.
New Feature: Added integration for Steam Audio, enabling features like occlussion, reflection and propagation.
New Feature: Added integration for PlayMaker, adding actions to implement sounds using visual scripting.
New Feature: Added async startup, enabling quicker load times when using the addressable AudioMixer.
New Feature: Added Example scene for AudioListenerDistance Override.
Fix: Renamed "Adressable" to "Addressable" for eg SONITY_ENABLE_ADDRESSABLE_AUDIOMIXER.
Fix: Fixed bug where Music/UI Stop, Pause, Unpause wouldn't work if Limited Globally polyphony was used.
Fix: Fixed bug where WebGL build could give the error "An invalid seek position was passed to this function".
Fix: Fixed so when SoundContainers are added, removed or changed in the SoundEvent at runtime it will update live in the editor.
Fix: Fixed bug where StopEverything, StopAllAtOwner, StopAtPosition might give an error.
Fix: Fixed bug where virtualized loops random start position wouldn't be randomized.
Fix: Fixed bug where volume increase settings mismatch would make the sound preview not work properly.
Fix: Fixed bug where play on Trigger On Stop in combination with StopAllAtOwner might cause errors.
Fix: Fixed bug where SoundEventCopy would copy the wrong volumeRatio value.
Fix: Fixed so Enable Volume Increase volume is set on awake.
Fix: Fixed so SoundTrigger initialises when it is triggered.
Fix: Polished the TemplateSoundVolumeManager.
Fix: Fixed the SoundPreset default settings.
Fix: Fixed bug where GetAngleToAudioListener would throw warnings.
Fix: Fixed so that the AudioListener is reset in edit mode when previewing sounds.
Fix: Changed so GetLastPlayedClipLength returns Mathf.Infinity when its null (returning 0 caused problems when TriggerOnTail and delay was combined).

1.0.6 - The Big Volume & Debug Update (2024 November)

Please remove the whole Sonity folder before upgrading because scripts were moved.
Support: Added support for Unity 6 (velocity replaced with linearVelocity).
New Feature: Added support to increase the volume of SoundEvents and SoundContainers by +24 dB each and SoundVolumeGroups by +12 dB.
New Feature: Added SoundEvent Base Volume & Polyphony settings (convert old to new with "Tools/Sonity 🔊/Convert SoundEvent Modifier Volume and Polyphony to Base").
New Feature: Added SoundVolumeGroup objects for linked SoundEvent volume control.
New Feature: Added SoundManager Debug Log SoundEvents to console.
New Feature: Added debug settings per SoundEvent for e.g. hiding or changing color of debug drawing and logging.
New Feature: Added new editor tools "Selection History", "Reference Finder" and "Select Same Type".
New Feature: Added SoundContainer setting "HRTF Spatialize" for external spatializer plugin support.
New Feature: Added fields for the AudioListener and the AudioListenerDistance for easy debugging.
New Feature: Added support for having the AudioMixer as an addressable asset in the SoundManager.
New Feature: Added "Note" fields for all scripable objects.
New Feature: Added addressable support for scriptable object asset comparison by adding custom GUIDs to all scriptable objects.
New Feature: Added SoundEvent.GetContainsLoop().
New Feature: Added SoundEvent.GetLastPlayedAudioSource().
New Feature: Added SoundManager Debug Draw SoundEvents added "Hide Closer Than" in game view setting.
Documentation: Added links in the documentation changelog.
Fix: Removed AudioMixerGroup from the SoundContainer. Convert with: "Tools/Sonity 🔊/Legacy/Apply SoundContainer AudioMixerGroup to SoundEvent if None".
Fix: Fixed so preview uses Global Volume.
Fix: Fixed bug where SoundPreset SoundEvent creation would return null SoundTag.
Fix: Removed ducking limiter from AudioMixer example because it sometimes seemed to make audio silent when lots of things were playing.
Fix: Better find AudioListener & AudioListenerDistance where disabled objects doesnt count.
Fix: Fixed problem where sound would be stopped when RunInBackground is disabled and not restart when Unity regains focus.
Fix: Removed debug of missing SoundManager when application is quitting.
Fix: Added checks and info that the SoundPicker is not nestable in an array or custom class because it is built upon CustomPropertyDrawer.
Fix: Cleaned up SoundMix functions.
Fix: Fixed the icons for the scriptable object with new higher resolution versions and better import settings.
Fix: Replaced deprecated FindObjectOfType and FindObjectsOfType with newer versions when supported.
Fix: Fixed example scenes Unity downgrade bug where BoxCollider sizes were set to size 2.
Documentation: Fixed info about SoundMix and AudioMixerGroup.
Change: Made Advanced subcategory for Settings in SoundContainers, SoundEvents and the SoundManager.
Change: Removed the create SoundPolyGroup for SE shortcuts.
Change: Fixed the default SoundManager Instance Statistics settings.
Change: Increased the SoundContainer "ReverbZoneMix Distance Increase" from 0.1 to 0.5.

1.0.5.1 - Fixes (2024 May)
Fix: Fixed bug where AudioListener wasn't checked if it was disabled properly.
Fix: Fixed bug where SoundPreset copied SoundEvent modifiers and SoundEvent SoundTag settings by reference instead of by value.

1.0.5 - SoundPhysics Update (2023 December)
New Tutorial: SoundPhysics & Intensity Tutorial (https://youtu.be/ehnwL678N4E)
New Feature: Support for addressables (deleting SoundEvents in runtime).
New Feature: SoundPhysics is expanded with support for Is Trigger and exit sounds.
New Feature: New SoundPhysicsConditions object for advanced reusable conditions and terrain layer support.
New Feature: Added a SoundPreset object enabling custom presets for SoundEvents and SoundContainers.
New Feature: Added Pause/Unpause functions e.g. Pause/Unpause AllAtOwner, Everywhere, Everything, 2D, All2D, Music, AllMusic.

New Feature: Added SoundManager functions and settings Global Pause and Global Volume.
New Feature: Added SoundManager setting for Sound Time Scale and default is changed from RealtimeSinceStartup to UnscaledTime.
New Feature: Added SoundEvent settings Ignore Local Pause and Ignore Global Pause.
New Feature: Added GetMusic and Get2D functions like e.g. GetMusicSoundEventState().
New Feature: Better UnityEvent support by adding SoundEvent functions with less parameters like StopAllowFadeOut and StopImmediate etc.
Deprication: SoundPhysics is reworked with arrays of SoundEvents, automatic conversion system is added in the editor.
Deprication: SoundPhysics is reworked to use object references so old tags must be remade.
Documentation how to: Added How To - Optimize AudioClip Settings.
Documentation how to: Added How To - Rename A Lot of Assets.
Documentation how to: Added How To - Debug Sounds and Performance.
Documentation: Documented SoundTrigger and SoundPicker functions.
Documentation: Updated info on AudioMixerGroup performance tips.
Documentation: Added video tutorial links.
Change: SoundPhysics is split for performance reasons into SoundPhysics, SoundPhysics2D, SoundPhysicsNoFriction, SoundPhysics2DNoFriciton.
Change: Renamed SoundTrigger enum StopNoFadeOut to StopImmediate.
Change: Renamed StopAllAt2D() to StopAll2D().
Change: Renamed tools and create assets to "Sonity 🔊".
Change: Renamed GetLastPlayedClipTime() to GetLastPlayedClipTimeSeconds.
Addition: Added GetLastPlayedClipTimeRatio().
Addition: Added SoundTrigger Actions: Pause, Pause Forced, Unpause.
Addition: Added SoundEventState.Paused.
Addition: Added SoundPhysics example scenes including friction, exit, Trigger On etc.
Addition: Added Intensity Scale Add and Scale Multiplier and Scale Max to 1 button.
Addition: Added Intensity Debug Zoom for easier visual debugging.
Addition: Added Tools "Set Selected Assets as Dirty (Force Reserialize for Resave)".
Addition: Expanded Example_AudioMixerWithLimiter with AMB, MUS, SFX, UI, VO.
Fix: SoundContainer distance scale is hidden when distance is disabled.
Fix: Increased the contrast of the SoundEvent and SoundContainer icons.
Fix: Fixed so automatic crossfades work with sound names ending with "loop".
Fix: Fixed SoundParameterIntensity continuous threshold stopping sounds and restarting loops.
Fix: Fixed problem where SoundPhysics friction didnt work as expected.
Fix: Fixed bug where SoundParameters weren't nullchecked.
Fix: Fixed so SoundContainer volume can be increased when multiple objects are selected and volume is at -0 dB.
Fix: Optimized SoundEvent recorded intensity drawing.
Fix: Replaced all ToLower with ToLowerInvariant for greater language support.
Fix: Fixed bug with create assets filename extension.
Fix: Fixed nullcheck in SoundPicker for SoundManager.Instance.
Legacy support: Fixed Unity 2018 compability by fixing: asmdef to non guid mode, SubsystemRegistration, GetContact.impulse, editor enum and header names.

1.0.4 - Free Trial Release (2023 June)
New Feature: Free trial version released.
Fix: Fixed bug where audio positions were based on the previous frame.
Fix: Fixed bug where disabled SoundEvents played in builds.
Fix: Fixed scripts for free trial version.

1.0.3 - Support for Disabling Domain Reloading (2023 June)

New Feature: Support for disabling domain reloading in the “enter play mode options”.
New Feature: Override Listener Distance with the AudioListenerDistance component.
Fix: Parented all Sonity scene objects created in runtime under the SoundManager.
Fix: Improved documentation on how to bind shortcuts.
Fix: Fixed SoundTriggerCustomEditor example code.
Fix: Fixed preview TriggerOnTail tail length 0 bug.

1.0.2.1 - Hotfix (2023 May)
Hotfix: Fixed error in SoundContainer settings editor.

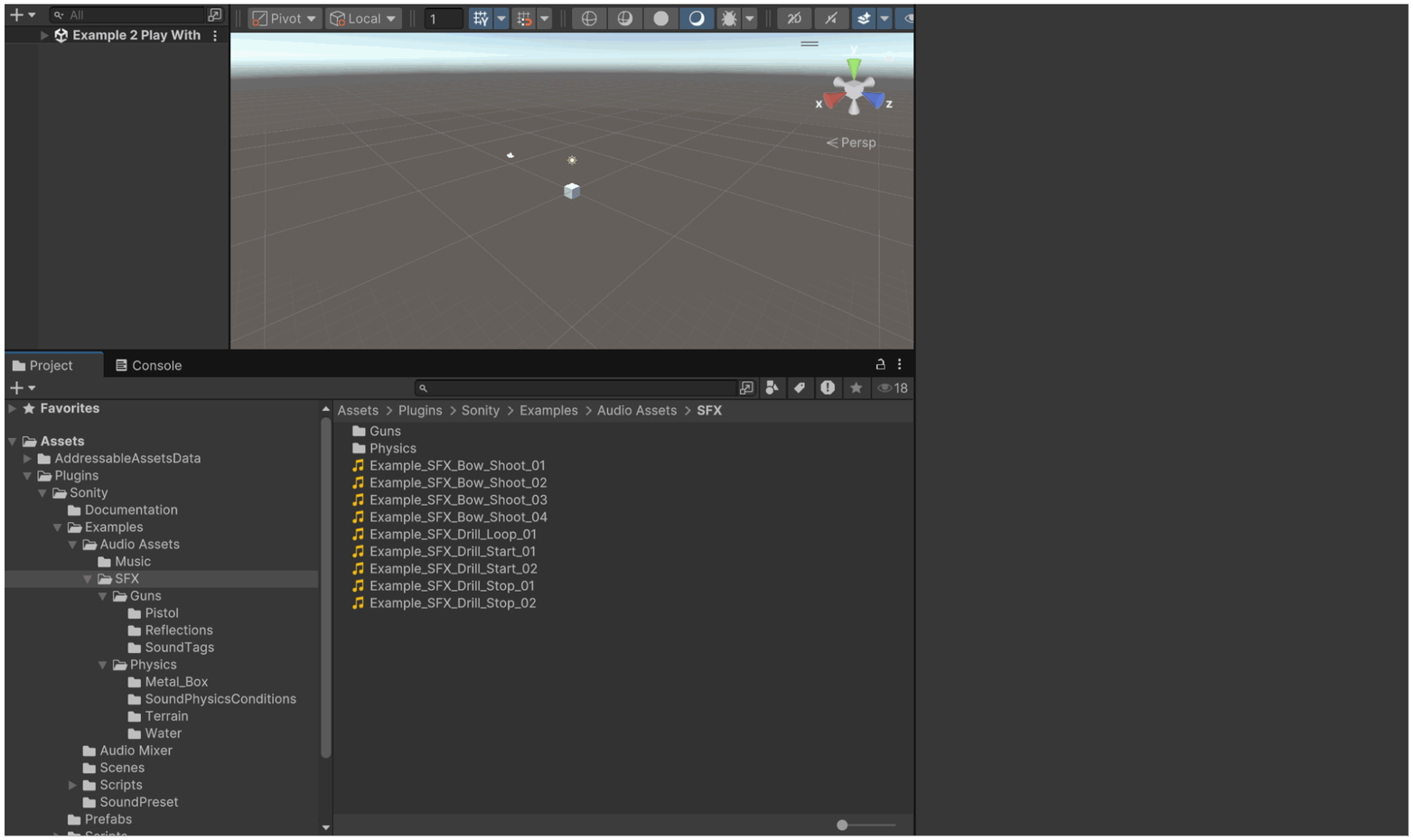
1.0.2 - Mixed Fixes (2023 May)
New Feature: Pass Parameter setting in the SoundEvent for passing SoundParameters to sub SoundEvents.
New Feature: Added GetSpectrumData() to get the spectrum data from AudioSources.
Support: Added support for using multiple disabled AudioListeners.
Support: Added support for SoundTrigger Custom Override and added examples in the documentation.
Support: Fixed support in examples and the timeline for the new Input System Package.
Fix: Fixed bug where SoundEvents with global polyphony couldn't be stopped.
Fix: Fixed nullcheck in SoundPicker for SoundManager.Instance.
Fix: The SoundManager and sub objects are now moved to Vector3.zero at start.
Fix: SoundPhysics removed collider requirement warning.
Fix: Preparing code for trial .dll version.
Fix: Fixed Find Assets OrderBy bug.
Example: Added example scene showcasing SoundPicker.
Documentation: Added tips and info about "Unity Native Audio Support" in documentation.
Documentation: SoundPhysics added info about rigidbody requirements.
Documentation: Updated asmdef info in documentation.
Documentation: Fixed documentation SoundParameter incorrect example code.
Documentation: Fixed DontDestroyOnLoad tooltip.
Documentation: Fixed Modifier tooltips.
Documentation: Fixed SoundParameterIntensity summary.
Documentation: Fixed Play2D summary.

1.0.1 - Gun Tutorial (2022 November)
New Feature: Automatic asset creator finds common denominator name.
New Feature: SoundContainer presets added set automatic looping and automatic crossfades.
Example: Added new examples used in the new gun tutorials.
Documentation: Updated documentation.
Documentation: Minor spelling corrections of tooltips and documentation.
Change: Changed so timeline zooms when holding down control and scrolling with mousewheel.
Change: Changed so SoundEvent SoundTag always use modifiers.
Change: Trigger On Tail default length changed to 0.
Change: Trigger On Tail added warning if length is less than shortest AudioClip.
Fix: Fixed bug with serialized cache bools.
Fix: Fixed bug with SoundTag position.
Fix: Fixed bug with preview SoundEvent distance scale.
Fix: Fixed so debug logs only print in developent builds and the editor.

Fix: Fixed nullcheck for camera in "Draw SoundEvents in Game View".
Fix: SoundEvent and SoundManager GetMaxLength removed forceUpdate.
Fix: All helpboxes now respect GUI Warnings setting.
Fix: Optimized SoundManager update.
Fix: Cleared SoundEventInstanceDictionaryValue.cs
Fix: Changed template scripts to useDontDestroyOnLoad to public.

1.0.0 - First Release (2022 October)
Release.

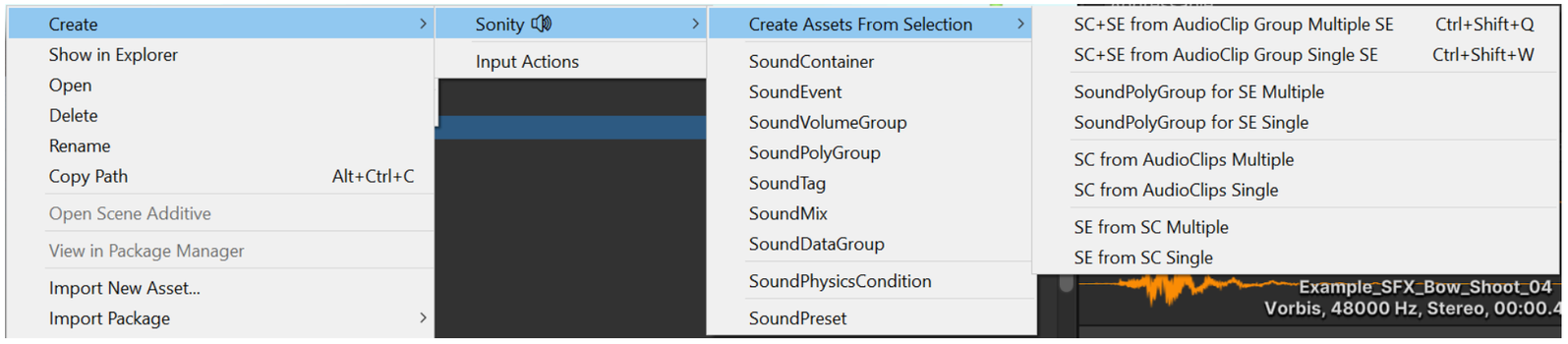
2 Create Assets From Selection



Right-click in the project menu to bring up the “Create Assets from Selection” menu.
With this you can easily create all your sound assets quickly if your AudioClips are named right.

- It works with folders as well as files.
- If a SoundContainers name contains “loop” it will automatically enable “Loop”, “Follow Position”, “Stop if Transform is Null” and “Random Start Position”.

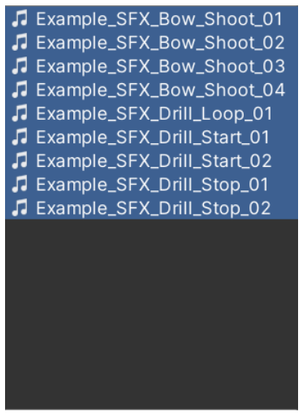
- It will automatically find out if you use space, hyphen “-” or underscore “_” in your filenames and will style the file ending the same way (e.g. “_SC”, “-SC” etc.)



Shortcuts
SC+SE from AudioClip Group Multiple SE: Ctrl+Shift+Q
SC+SE from AudioClip Group Single SE: Ctrl+Shift+W
SoundPolyGroup for SE Multiple: Ctrl+Alt+Shift+Q
SoundPolyGroup for SE Single: Ctrl+Alt+Shift+W
They are rebindable in the Shortcuts window.

AudioClip Group
AudioClip groups are sounds with the same name, but different numbers in the end e.g 01, 02, 03 etc.

Create SoundContainers and multiple SoundEvents
In this example we select the bow and drill sounds and run “SC+SE from AudioClip Group Multiple SE”:

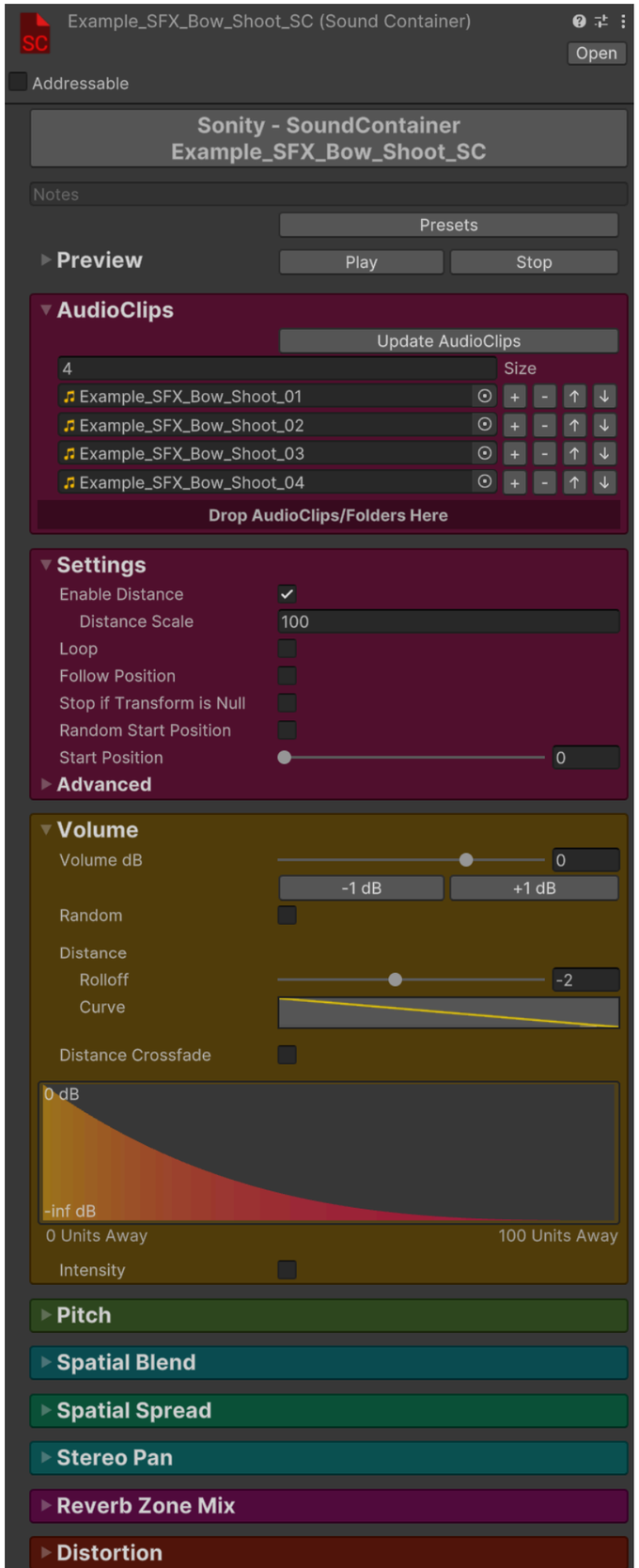


This creates 4 SoundContainers and 4 SoundEvents, one per AudioClip group.

Create SoundContainers and a single SoundEvent
In this example we select the shoot sounds and run “SC+SE from AudioClip Group Single SE”:

They contain 4 different groups which will create 4 SoundContainers, one per group and a single SoundEvent.
It will try to find a common denominator of the names of the SoundContainers for naming the SoundEvent.

3 SoundContainer


Example_SFX_Bow_Shoot_SC (Sound Container)
Open
Addressable
Sonity - SoundContainer
Example_SFX_Bow_Shoot_SC
Notes
Presets
Preview
Play
Stop
AudioClips
Update AudioClips
4
Size
Example_SFX_Bow_Shoot_01
Example_SFX_Bow_Shoot_02
Example_SFX_Bow_Shoot_03
Example_SFX_Bow_Shoot_04
Drop AudioClips/Folders Here
Settings
Enable Distance
Distance Scale
100
Loop
Follow Position
Stop if Transform is Null
Random Start Position
Start Position
0
Advanced
Volume
Volume dB
0
-1 dB
+1 dB
Random
Distance
Rolloff
-2
Curve
Distance Crossfade
0 dB
-inf dB
0 Units Away
100 Units Away
Intensity
Pitch
Spatial Blend
Spatial Spread
Stereo Pan
Reverb Zone Mix
Distortion

SoundContainers are the building blocks of Sonity.
They contain AudioClips and options of how the sound should be played.
All SoundContainers are multi-object editable.

Here is a description of controls shared by multiple fields.

Rolloff
The power of the rolloff.
0 is linear.

Curve
Curve of the value over distance or intensity.
Distance: From 0 (close) to 1 (distant).
Intensity: From 0 (soft) to 1 (hard).

Strength
How much effect the distance/intensity should have.

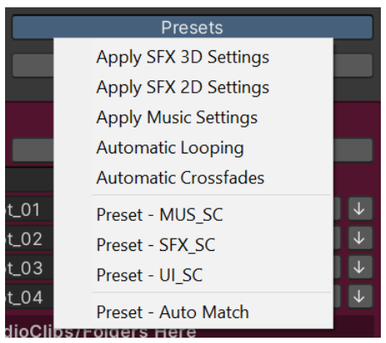
Increase
Increases the amount.

3.1 SoundContainer Notes

Example note:

At the top of the SoundContainer there is a text field which you can fill with notes.

3.2 SoundContainer Presets



The “Presets” shown in the example picture above need to be created with the SoundPreset object.
This info also shows up on button hover:

SFX 3D
Enable Distance = true

Spatial Blend = 1
Never Steal Voice = false
Never Steal Voice Effects = false
Pitch Random = true
Priority = 0.5
Reverb Zone Mix = -0 dB

SFX 2D
Enable Distance = false
Spatial Blend = 0
Never Steal Voice = false
Never Steal Voice Effects = false
Pitch Random = true
Priority = 0.5
Reverb Zone Mix = -0 dB

SFX 2D
Enable Distance = false
Spatial Blend = 0
Never Steal Voice = false
Never Steal Voice Effects = false
Pitch Random = false
Priority = 0.5
Reverb Zone Mix = Negative Infinity dB

Music
Enable Distance = false
Spatial Blend = 0
Never Steal Voice = true
Never Steal Voice Effects = true
Pitch Random = false
Volume Random = false
Priority = 1
Reverb Zone Mix = Negative Infinity dB

Automatic Looping
If the name of the selected SoundContainers contains “loop” then it will automatically enable “Loop”, “Follow Position”, “Stop if Transform is Null” and “Random Start Position”.

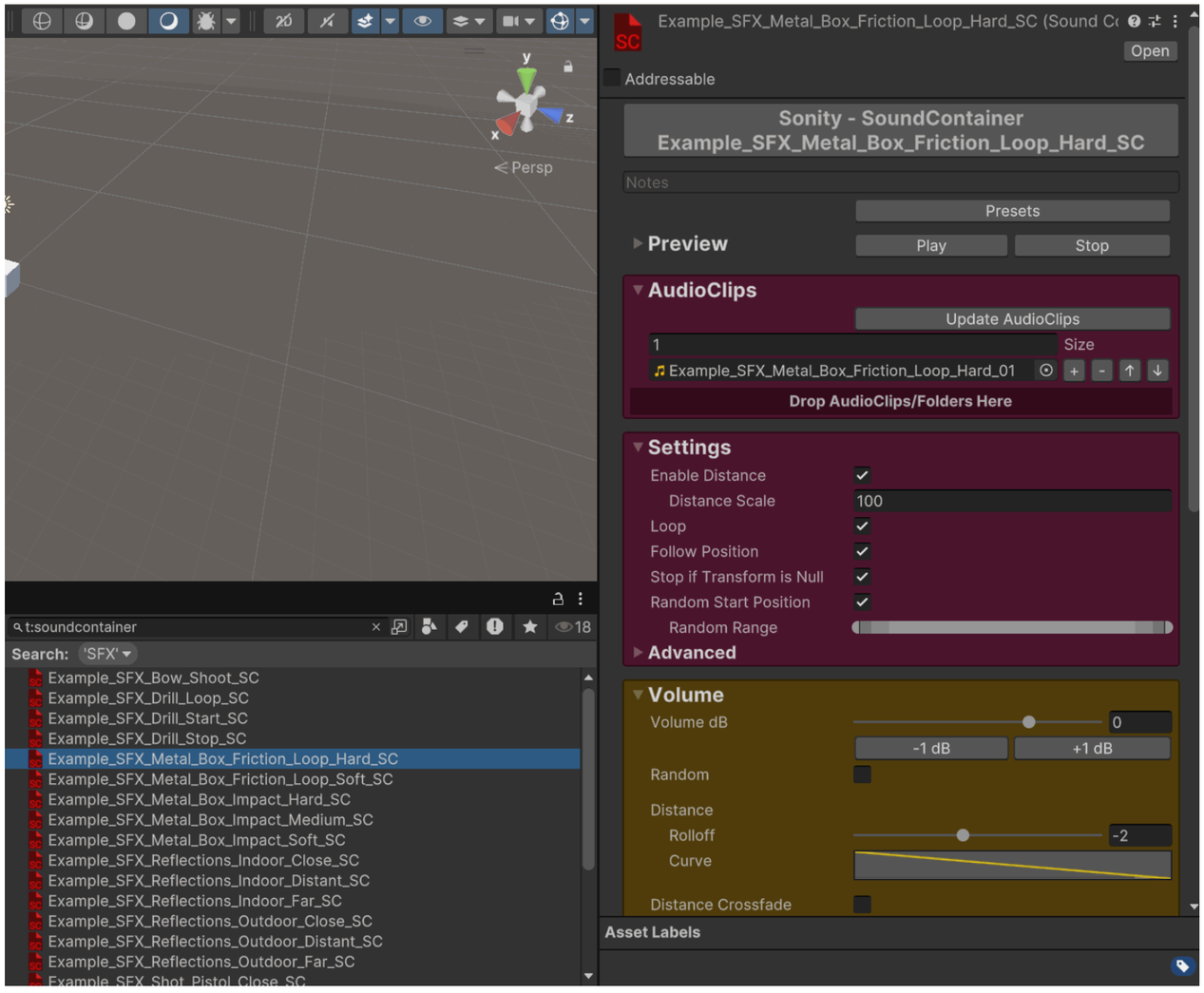
Automatic Crossfades
If the names of the selected SoundContainers end in certain combinations it will automatically set up distance or intensity crossfades.
It works on multiple groups at the same time.
These are the combinations and their result:

Distance Crossfade:
Close + Distant + Far = 3 layers
Close + Distant = 2 layers

Close + Far” = 2 layers

Intensity Crossfade:
Soft + Medium + Hard = 3 layers
Soft + Hard = 2 layers

Example usage of Automatic Crossfades:



SoundPresets
Create SoundPreset objects to get custom settings which you can apply to your SoundContainers and SoundEvents.
Either manually select the preset you want to apply in the dropdown, or use “Auto Match” to automatically apply the settings based on the name of the asset.

3.3 SoundContainer Preview

With the preview you can listen to the sounds in the editor.
The preview handle enables you to move the sound around.

Preview Shortcuts
Play: Ctrl+Q (Previews the selected SoundEvent or SoundContainer).
Stop: Ctrl+W (Stops any playing preview of SoundEvents and SoundContainers, press two times to skip fade out).
They are rebindable in the Shortcuts window.

Play
Previews the SoundContainer.
Does not work if Unity cannot build the project, or if the game is paused.
The default shortcut is Ctrl+Q.

Stop
Press two times to skip fade out.
The default shortcut is Ctrl+W.

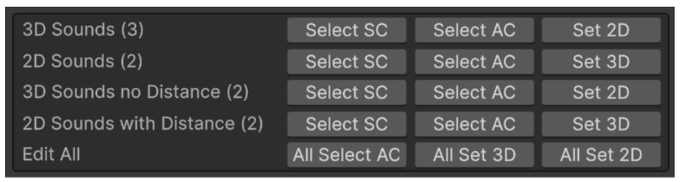
Reset
Resets the preview settings.

Intensity
Controls the intensity value of the played SoundContainers.

AudioMixerGroup
Only used for preview.

3.4 SoundContainer Spatialization Info

Show and edit spatialization info for the selected SoundContainers.



3D Sounds
All SoundContainers with Spatial Blend > 0 and Distance Enabled.

2D Sounds
All SoundContainers with Spatial Blend = 0 and Distance Disabled.

3D Sound no Distance
All SoundContainers with Spatial Blend > 0 and Distance Disabled.

2D Sound with Distance
All SoundContainers with Spatial Blend = 0 and Distance Enabled.

Select SC
Selects SoundContainers matching the specific settings.

Select All SC
Selects All SoundContainers in the same folder disregarding subfolders.

Select AC
Selects AudioClips of the SoundContainers matching the specific settings.

Set 3D
Sets the selected SoundContainers to Spatial Blend 1 and Distance Enabled.

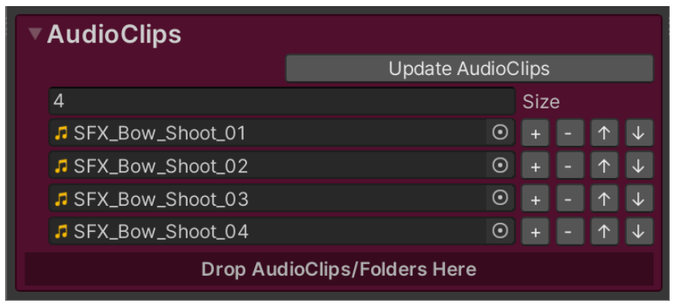
Set 2D
Sets the selected SoundContainers to Spatial Blend 0 and Distance Disabled.

All Select AC
Selects all AudioClips of the selected SoundContainers.

All Set 3D
Sets all selected SoundContainers to Spatial Blend 1 and Distance Enabled.

All Set 2D
Sets all selected SoundContainers to Spatial Blend 0 and Distance Disabled.

3.5 SoundContainer AudioClips



This is where you assign the AudioClips which are going to be used when the SoundContainer is played.

Update AudioClips

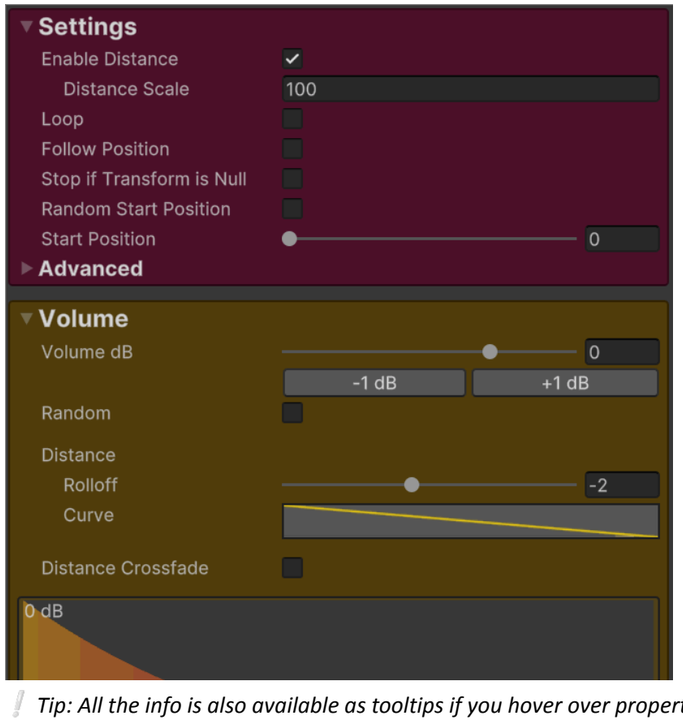
Updating AudioClips can be used to automatically add/remove variations.
Can be used on multiple items at once (tip: search “t:soundContainer”).

Refresh AudioClip Group
Adds all AudioClips with the same name as the first AudioClip (disregarding numbers, e.g. 01 02).

Find AudioClip Group
Automatically finds all AudioClips containing the same name as this SoundContainer (disregarding _SC, numbers).
If no matching AudioClips are found, it will try and remove one character at the end of the name at a time until it finds a hit.

Drop AudioClips/Folders Here
Here you can drag and drop AudioClips or folders with AudioClips in them.

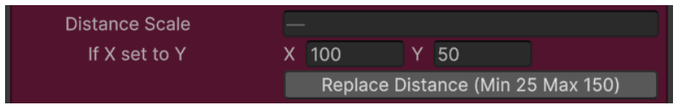
3.6 SoundContainer Settings



> *Tip: All the info is also available as tooltips if you hover over properties.*

Enable Distance
Otherwise the SoundContainer will not be affected by distance (disable for music etc).

Distance Scale
Range scale multiplier. It is multiplied by the Distance Scale of the SoundManager.



If X set to Y
Shows up if you have select multiple SoundContainers with Distance Enabled.
Useful for batch editing a lot of assets where you want to change the Distance Scale for many sounds.

Replace Distance
Replaces any Distance Scale matching X with the Y value.
Also shows the Min and Max value for Distance Scale of the selected objects.

Loop

Makes the sound loop.
If you use "Create Assets from Selection" and the AudioClip name contains "loop".
Then it will automatically enable “Loop”, “Follow Position”, “Stop if Transform is Null” and “Random Start Position”.

Follow Position
If the SoundContainer should follow the given Transform position.

Stop if Transform is Null
Automatically stops the sound if the Transform it's played at is destroyed (either the owner or position Transform).
Useful safety precaution for loops.

Random Start Position
Starts the sound at a random position.
Overrides the Start Position setting.
Useful for loops.

> **Random Range**
> Min/max range within the sound can start at.

Start Position
0 is the start and 1 is the end.

Advanced Settings

Reverse
If enabled the AudioClip will be played backwards.
Make sure to set the start position to the end.
Reverse is only supported for AudioClips which are stored in an uncompressed format or will be decompressed at load time.

Play Order
Determines in which order the AudioClips will be played.

> **Global Random**
> All SoundEvents will share the same global random AudioClip pool, which ensures less repetition.
> Uses a pseudo random function remembering half of the length of available AudioClips it last played to avoid repetition.
>
> **Local Random**
> Same as global random except its per SoundEvent owner.

If enabled all will share the same global random AudioClip pool, which ensures less repetition.
Otherwise random AudioClip per owner is used.
The randomizer either way uses a pseudo random function remembering which AudioClips it last played to avoid repetition.

Priority
The priority the Voice has when Voice stealing.
Also the priority the Voice Effects has when Voice Effects stealing.

1 is high priority, 0.5 is default priority and 0 is low priority.
It's multiplied with the volume of the Voice when evaluating final priority.

Lock Axis
Locks the selected axis to the selected position.
Useful for 2D games if you want to lock the sound to a position along an axis.

> **Axis**
> The axis to lock.
>
> **Position**
> The position to set the locked axis to.

Prevent End Clicks
If enabled it will fade out the volume 0.1 seconds before the end of the AudioClip to prevent clicks.
If the AudioClips is shorter than 0.1 seconds or set to loop the fade will be skipped.
DC offsets and some settings in Unity make an AudioClip click at the end.
❗ *Tip: If you still experience sporadic clicks, try changing the Load Type of the AudioClips to e.g. "Compressed In Memory", it might help.*

Never Steal Voice
The SoundManager will never steal this Voice if the Voice Limit is reached (use on music etc).

Never Steal Voice Effects
The SoundManager will never steal the Voice Effects on this Voice if the Voice Effect Limit is reached (use on music etc).

Doppler Amount
How much the pitch of the sound is changed by the relative velocity between the AudioListener and the AudioSource.

Bypass Reverb Zones
Bypasses any reverb zones

Bypass Voice Effects
Bypasses any effects on the AudioSource, e.g. Distortion and Filters.
Voice effects are automatically bypassed if you don't have distortion/lowpass/highpass enabled.

Bypass Listener Effects
Bypasses any effects on the listener

HRTF Plugin Spatialize
Enables spatialization using external third party plugins.
You need to add the plugin and select it in the project audio settings.
Example of third party spatializers:

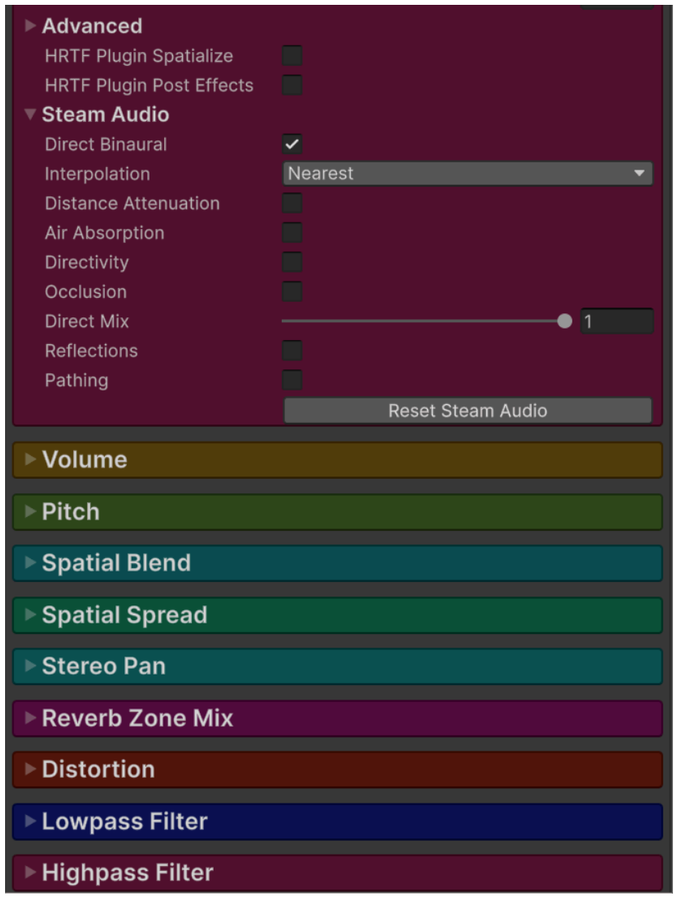
- Steam Audio Spatializer
- Oculus Spatializer Unity
- Google Resonance Audio Spatializer

- Vive 3DSP Audio SDK Spatializer
- Microsoft Spatializer
- Apple PHASE Spatializer
- Qualcomm 3D Spatializer

HRTF Plugin Post Effects
If the spatialization plugin effect happens before before or after any effects.

3.7 SoundContainer Settings Steam Audio

If Steam Audio is enabled in the SoundManager Settings.



HRTF Plugin Spatialize
Spatialize makes the sound be affected by Steam Audio with effects like occlusion, reflections and more.
Disable Spatialize for 2D sounds like music etc.

> ❕ **Steam Audio Source**
>
> This part of the text is mostly as the one from the official Valve Steam Audio Source Component Reference.
> I've added information which was missing on some parts.

Direct Binaural
If checked, HRTF-based binaural rendering will be used to spatialize the source. This requires 2-channel (stereo) audio output.
If unchecked, panning will be used to the spatialize the source using the user's speaker layout.
Binaural rendering provides improved spatialization at the cost of slightly increased CPU usage.

Interpolation
Controls how HRTFs are interpolated when the source moves relative to the listener.
Nearest: Uses the HRTF from the direction nearest to the direction of the source for which HRTF data is available.
The fastest option, but can result in audible artifacts for certain kinds of audio clips, such as white noise or engine sounds.
Bilinear: Uses an HRTF generated after interpolating from four directions nearest to the direction of the source, for which HRTF data is available.
This may result in smoother audio for some kinds of sources when the listener looks around, but has higher CPU usage (up to 2x).

Perspective Correction
If checked, perspective correction (based on the projection matrix of the current main camera) is applied to this source during spatialization.
This can improve the perceived positional accuracy in non-VR applications. See Steam Audio Settings for more details.
Requires Enable Perspective Correction to be checked in Steam Audio Settings.

Distance Attenuation
If checked, distance attenuation will be calculated and applied to the Audio Source.
This takes into account the Spatial Blend setting on the Audio Source, so if Spatial Blend is set to 2D, distance attenuation is effectively not applied.

> **Input**
> Specifies how the distance attenuation value is determined.
> Curve Driven: Distance attenuation is controlled by the Volume curve on the Audio Source.
> Physics Based: A physics-based distance attenuation model is used.
> This is an inverse distance falloff. The curves defined on the Audio Source are ignored.

Air Absorption
If checked, frequency-dependent distance based air absorption will be calculated and applied to the Audio Source.

> **Input**
> Specifies how the air absorption values (which are 3-band EQ values) are determined.
> Simulation Defined: Uses a physics-based air absorption model.
> This is an exponential falloff, with higher frequencies falling off faster with distance than lower frequencies.
> User Defined: Uses the values specified in the Air Absorption Low, Air Absorption Mid, and Air Absorption High sliders as the EQ values.
> The air absorption value will not automatically change with distance to the source.
> You are expected to control the Air Absorption Low, Air Absorption Mid, and Air Absorption High sliders using a custom script to achieve this effect.
>
> > **Low**
> > The low frequency (up to 800 Hz) EQ value for air absorption.
> > Only used if Air Absorption Input is set to User Defined.

0 = low frequencies are completely attenuated,
1 = low frequencies are not attenuated at all.

Mid
The middle frequency (800 Hz - 8 kHz) EQ value for air absorption.
Only used if Air Absorption Input is set to User Defined.
0 = middle frequencies are completely attenuated,
1 = middle frequencies are not attenuated at all.

High
The high frequency (8 kHz and above) EQ value for air absorption.
Only used if Air Absorption Input is set to User Defined.
0 = high frequencies are completely attenuated,
1 = high frequencies are not attenuated at all.

Directivity
If checked, attenuation based on the source's directivity pattern and orientation will be applied to the Audio Source.

Input
Specifies how the directivity attenuation value is determined.
Simulation Defined. Uses a dipole directivity model.
You can control the dipole shape using the Dipole Weight and Dipole Power properties.
User Defined. Uses the value specified in the Directivity Value property.
This value will not automatically change as the source rotates.
You are expected to control the Directivity Value property using a custom script to achieve this effect.

Dipole Weight
Blends between monopole (omnidirectional) and dipole directivity patterns.
0 = pure monopole (sound is emitted in all directions with equal intensity),
1 = pure dipole (sound is focused to the front and back of the source).
At 0.5, the source has a cardioid directivity, with most of the sound emitted to the front of the source.
Only used if Directivity Input is set to Simulation Defined.

Dipole Power
Controls how focused the dipole directivity is.
Higher values result in sharper directivity patterns.
Only used if Directivity Input is set to Simulation Defined.

Value
The directivity attenuation value.
Only used if Directivity Input is set to User Defined.
0 = sound is completely attenuated,
1 = sound is not attenuated at all.

Occlusion
If checked, attenuation based on the occlusion of the source by the scene geometry will be applied to the Audio Source.

Input
Specifies how the occlusion attenuation value is determined.
Simulation Defined. Uses ray tracing to determine how much of the source is occluded.
User Defined. Uses the Occlusion Value slider to control occlusion.
The occlusion value will not automatically change based on surrounding geometry.
You are expected to control the Occlusion Value slider using a custom script to achieve this effect.
This option is intended for integrating your own occlusion model with Steam Audio.

Type
Specifies how rays should be traced to model occlusion.
Raycast: Trace a single ray from the listener to the source. If the ray is occluded, the source is considered occluded.
Volumetric: Trace multiple rays from the listener to the source based on the Occlusion Radius setting.
The proportion of rays that are occluded determine how much of the direct sound is considered occluded.
Transmission calculations, if enabled, are only applied to the occluded portion of the direct sound.

Radius
The apparent size of the sound source.
The larger the source radius, the larger an object must be in order to fully occlude sound emitted by the source.

Samples
The number of rays to trace from the listener to various points in a sphere around the source.
Only used if Occlusion Type is set to Volumetric.
Increasing this number results in smoother transitions as the source becomes more (or less) occluded.
This comes at the cost of increased CPU usage.

Value
The occlusion attenuation value.
Only used if Occlusion Input is set to User Defined.
0 = sound is completely attenuated,
1 = sound is not attenuated at all.

Transmission
If checked, a filter based on the transmission of sound through occluding scene geometry will be applied to the Audio Source.

Type
Specifies how the transmission filter is applied.
Frequency Independent. Transmission is modeled as a single attenuation factor.
Frequency Dependent. Transmission is modeled as a 3-band EQ.

Input
Specifies how the transmission attenuation or EQ values are determined.
Simulation Defined. Uses ray tracing to determine how much of the sound is transmitted.
User Defined. Uses the Transmission Low, Transmission Mid, and Transmission High sliders to control transmission.
The transmission values will not automatically change based on surrounding geometry.
You are expected to control the sliders using a custom script to achieve this effect.

This option is intended for integrating your own occlusion and transmission model with Steam Audio.

Low
The low frequency (up to 800 Hz) EQ value for transmission.
Only used if Transmission Input is set to User Defined.
0 = low frequencies are completely attenuated,
1 = low frequencies are not attenuated at all.

Mid
The middle frequency (800 Hz to 8 kHz) EQ value for transmission.
Only used if Transmission Input is set to User Defined.
0 = middle frequencies are completely attenuated,
1 = middle frequencies are not attenuated at all.

High
The high frequency (8 kHz and above) EQ value for transmission.
Only used if Transmission Input is set to User Defined.
0 = high frequencies are completely attenuated,
1 = high frequencies are not attenuated at all.

Attenuation
The attenuation value for transmission.
Only used if Transmission Input is set to User Defined.
0 = the sound is completely attenuated,
1 = the sound is not attenuated at all.

Max Surfaces
The maximum number of transmission surfaces, starting from the closest surface to the listener,
whose transmission coefficients will be considered when calculating the total amount of sound transmitted.
Increasing this value will result in more accurate results when multiple surfaces lie between the source and the listener, at the cost of increased CPU usage.

Direct Mix
The contribution of the direct sound path to the overall mix for this Audio Source.
Lower values reduce the contribution more.

Reflections
If checked, reflections reaching the listener from the source will be simulated and applied to the Audio Source.

Type
Specifies how reflections should be simulated for this source.
Realtime. Rays are traced in real-time, and bounced around the scene to simulate sound reflecting from the source and reaching the listener.
This allows for smooth variations, and reflections off of dynamic geometry, at the cost of significant CPU usage.
Baked Static Source. The source is assumed to be static, and the listener position is used to interpolate reflected sound from baked data.
This results in relatively low CPU usage, but cannot model reflections off of dynamic geometry, and requires more memory and disk space.
Baked Static Listener. The listener is assumed to be static, and the source position is used to interpolate reflected sound from baked data.
This results in relatively low CPU usage, but cannot model reflections off of dynamic geometry, and requires more memory and disk space.

Use Distance Curve
Is shown when distance attenuation is enabled and is set to curve driven.

Baked Source
If Reflections Type is set to Baked Static Source, the position and orientation of the GameObject specified in this field will be used as the position and orientation of the source.

Apply HRTF
If checked, applies HRTF-based 3D audio rendering to reflections.
Results in an improvement in spatialization quality when using convolution or hybrid reverb, at the cost of slightly increased CPU usage. Default: off.
Warning, this might introduce distortions for louder sounds.

Mix Level
The contribution of reflections to the overall mix for this Audio Source.
Lower values reduce the contribution more.

Pathing
If checked, shortest paths taken by sound as it propagates from the source to the listener will be simulated,
and appropriate spatialization will be applied to the Audio Source for these indirect paths.

Probe Batch
When simulating pathing, the baked data stored in this probe batch will be used to look up paths from the source to the listener.

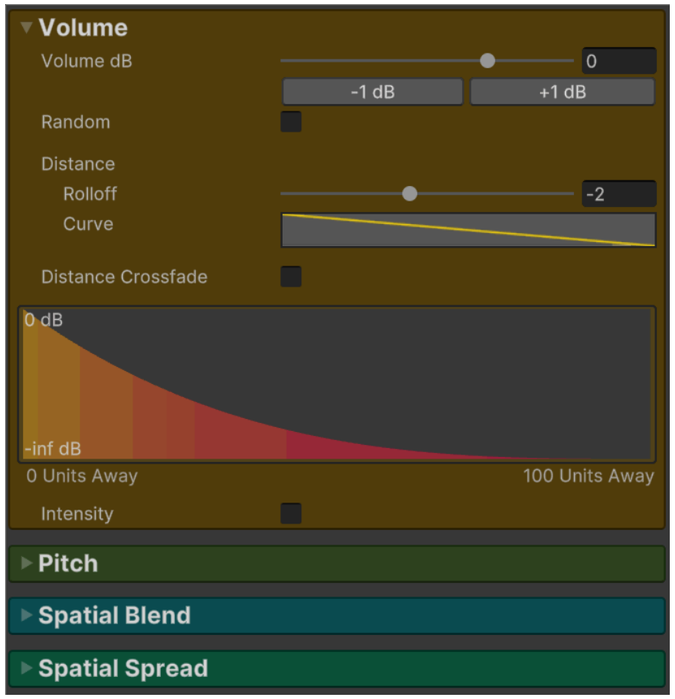
Validation
If checked, each baked path from the source to the listener is checked in real-time to see if it is occluded by dynamic geometry.
If so, the path is not rendered.

Find Alternate
If checked, if a baked path from the source to the listener is found to be occluded by dynamic geometry,
alternate paths are searched for in real-time, which account for the dynamic geometry.

Apply HRTF
If checked, applies HRTF-based 3D audio rendering to pathing.
Results in an improvement in spatialization quality, at the cost of slightly increased CPU usage. Default: off.

Mix Level
The contribution of pathing to the overall mix for this Audio Source.
Lower values reduce the contribution more.

3.8 SoundContainer Volume



Volume dB
Volume offset in decibel.
If you want to be able to raise the volume and not just lower it you have 2 options:

> **Option A:**
> Enable "Enable Volume Increase" in the SoundManager.
> This will enable you to raise the volumes of SoundContainers and SoundEvents by +24 dB each and SoundVolumeGroups by +12dB.
>
> **Option B:**
> Select all the SoundContainers and lower the volume by -20 dB with the -1dB button.
> Then to compensate you can increase the global volume with an Audio Mixer (which you then can set to +20 dB).

-1 dB
Lowers the relative volume of all the selected SoundContainers.
Useful for example if you want to raise the volume of one SoundContainer and keep the relative volume.
Because then you can lower all of them to get more headroom.
If multiple SoundContainers are selected it will show the lowest volume.

+1 dB
Raises the relative volume of all the selected SoundContainers.

Useful for example if you want to set the loudest volume to 0 dB but keep the relative volumes.
If multiple SoundContainers are selected it will show the highest volume.

Distance Crossfade
With distance crossfade you can easily crossfade between different sounds over distance.
For e.g. gunshots you could add sounds with close, distant and far perspectives.
You'd set the "Layers" setting to 3 for all the SoundContainers.
Then you'd set "This Is" of close to 1, distant to 2 and far to 3.

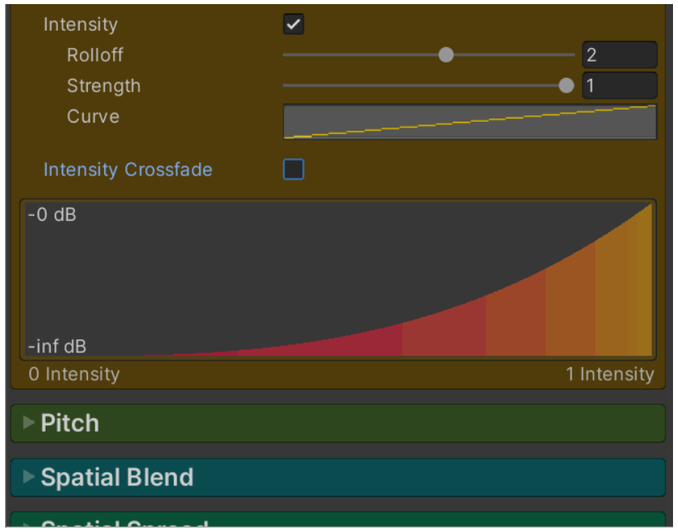
> **Layers**
> The number of layers the crossfade is based on. You must have at least 2 layers.
>
> **This is**
> Which layer this is. Set up with other SoundContainers for the other layers. Lower numbers are closer and higher are more distant.

Intensity
Changes the volume over intensity.
Use on for example physics sounds where you pass the velocity with a SoundParameterIntensity.

Intensity Crossfade
With intensity crossfade you can easily crossfade between different sounds over intensity.
For e.g. impacts you could add sounds with hard, medium and soft variations.
You'd set the "Layers" setting to 3 for all the SoundContainers.
Then you'd set "This Is" of hard to 3, medium to 2 and soft to 1.

> **Layers**
> The number of layers the crossfade is based on. You must have at least 2 layers.
>
> **This is**
> Which layer this is. Set up with other SoundContainers for the other layers. Higher numbers are harder and lower numbers are softer.

Intensity Crossfade example:



3.9 SoundContainer Pitch

Controls the pitch (in semitones).

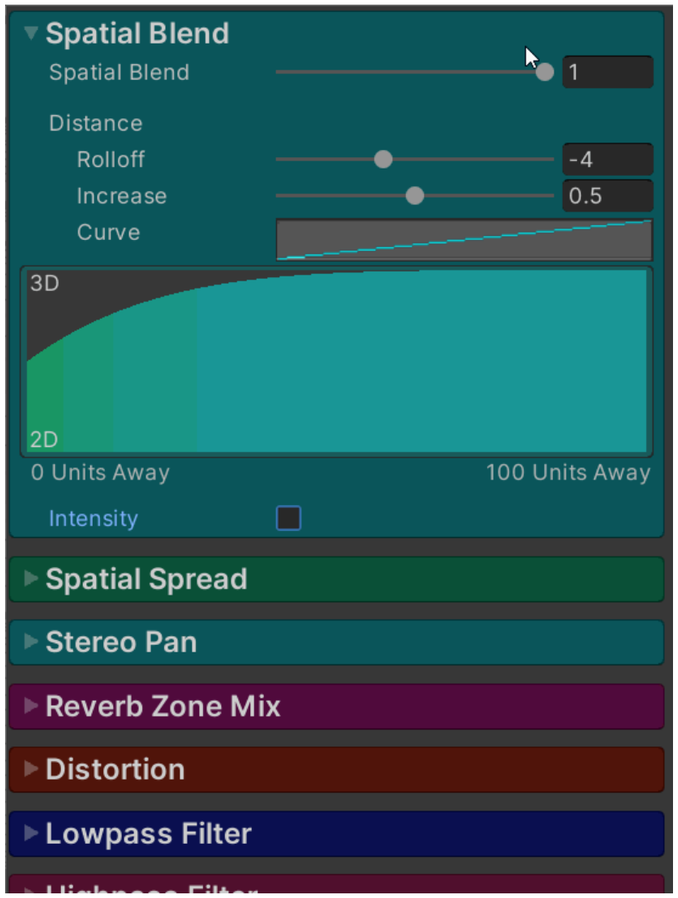
Intensity
Changes the pitch over intensity.
Use on for example car sounds where you pass the speed with a SoundParameterIntensity.

Low st
The lowest intensity in semitones. Range -128 to 128.

High st

The highest intensity in semitones. Range -128 to 128.

3.10 SoundContainer Spatial Blend



Controls the spatial blend. 0 is 2D (not spatialized) and 1 is 3D (spatialized).

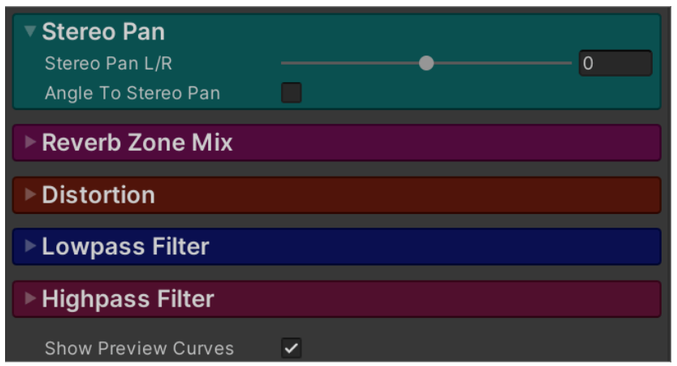
Intensity
Changes the spatial blend over intensity.

3.11 SoundContainer Spatial Spread

Controls the spatial spread (in degrees).
Only the 3D (spatialized) part of the sound is affected by the spatial spread.

Intensity
Changes the spatial spread over intensity.

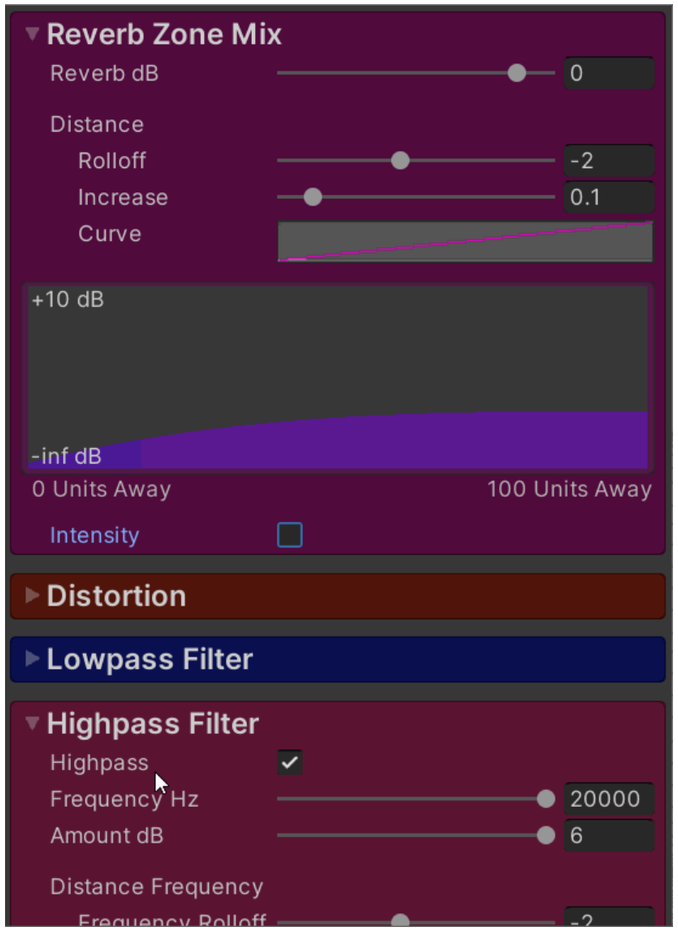
3.12 SoundContainer Stereo Pan



Controls the stereo pan. -1 is left and 1 right.
Only the 2D part of the sound is affected by the stereo pan.

Angle To Stereo Pan
Pans the sound depending on the angle between the Voice and the AudioListener.

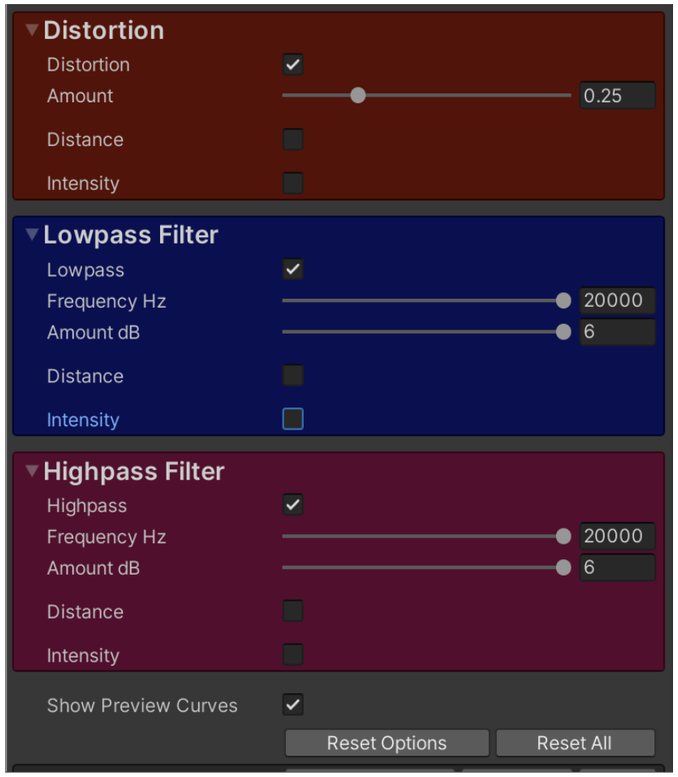
3.13 SoundContainer Reverb Zone Mix



Controls the reverb zone mix (in decibel).

Intensity
Changes the reverb zone mix over intensity.

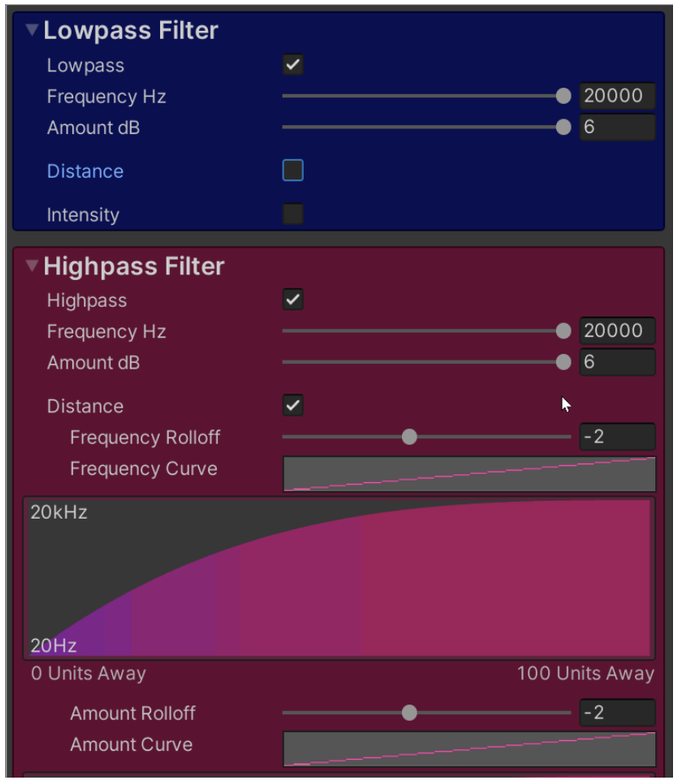
3.14 SoundContainer Distortion



Waveshaper type distortion.
0 is clean, 1 is distorted.
SoundContainer Voice Effects are applied per Voice.
If distortion amount is 0 the effect is disabled internally for performance.
The number of active Voice Effects are limited by the “Voice Effect Limit” on the SoundManager.
DSP effects are not available in WebGL.

Intensity
Changes the distortion over intensity.
Use on for example explosion sounds where you pass the size of the explosion with a SoundParameterIntensity.

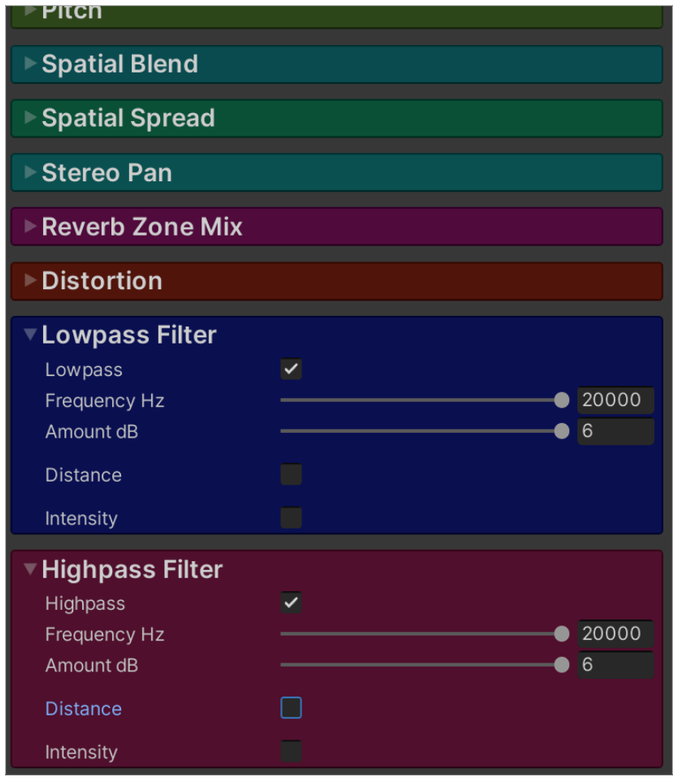
3.15 SoundContainer Lowpass



Lowpass filter with a variable amount.
Maximum of 6dB per octave.
SoundContainer Voice Effects are applied per Voice.
If frequency is 20,000 Hz or amount is 0 dB the effect is disabled internally for performance.
The number of active Voice Effects are limited by the “Voice Effect Limit” on the SoundManager.
DSP effects are not available in WebGL.

Intensity
Changes the lowpass over intensity.
Use on for example occluded sounds where you pass the amount of occlusion with SoundParameterIntensity.

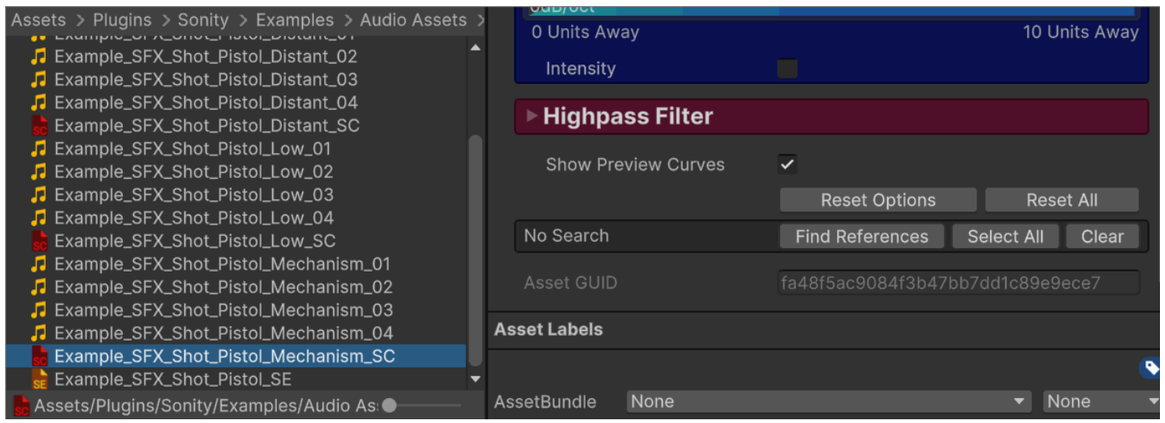
3.16 SoundContainer Highpass



Highpass filter with a variable amount.
Maximum of 6dB per octave.
SoundContainer Voice Effects are applied per Voice.
If frequency is 20 Hz or amount is 0 dB the effect is disabled internally for performance.
The number of active Voice Effects are limited by the “Voice Effect Limit” on the SoundManager.
DSP effects are not available in WebGL.

Intensity
Changes the highpass over intensity.
Use on for example occluded sounds where you pass the amount of occlusion with SoundParameterIntensity

3.17 SoundContainer Find References



Find References
Finds all the references to the SoundContainer.

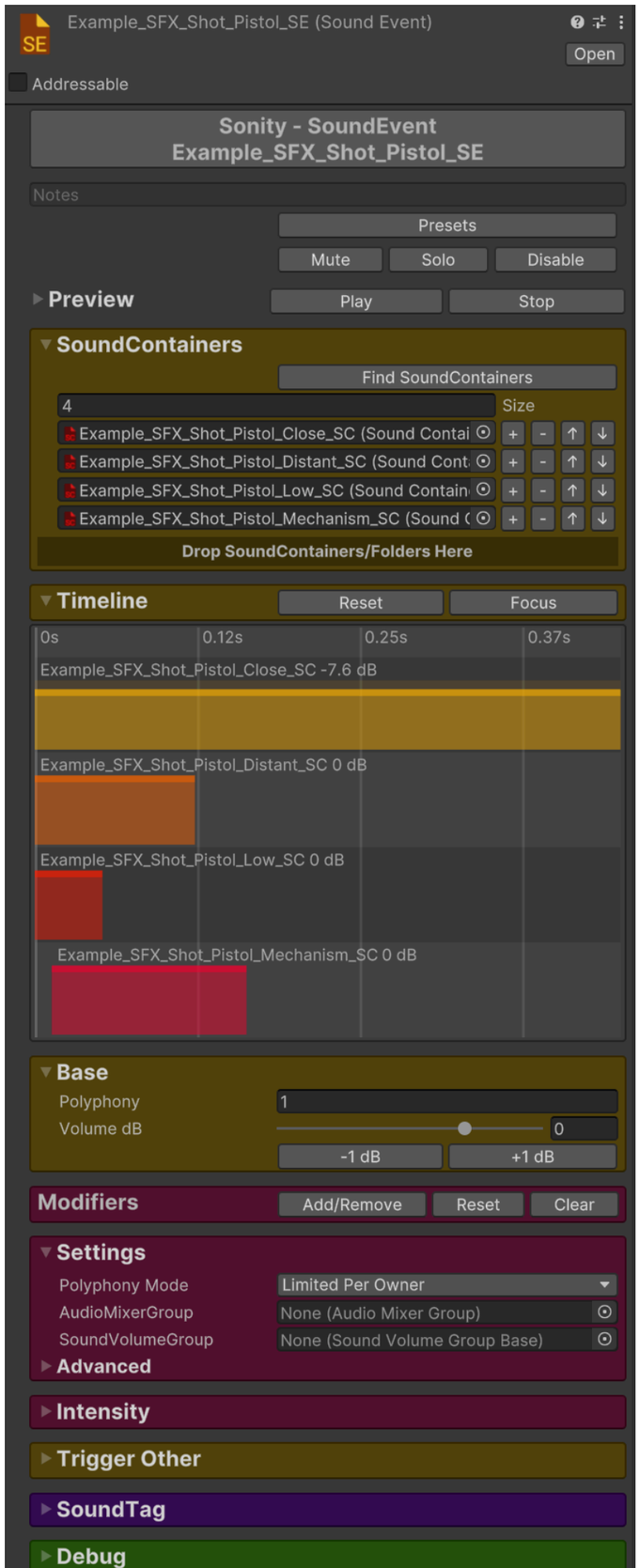
Select All
Selects all the assets with references to the SoundContainer.

Clear
Removes all the found references.

3.18 SoundContainer Asset GUID

The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

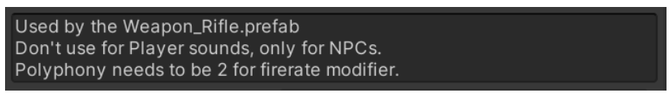
4 SoundEvent


Example_SFX_Shot_Pistol_SE (Sound Event)
Open
Addressable
Sonity - SoundEvent
Example_SFX_Shot_Pistol_SE
Notes
Presets
Mute
Solo
Disable
Preview
Play
Stop
SoundContainers
Find SoundContainers
4
Size
Example_SFX_Shot_Pistol_Close_SC (Sound Contai
Example_SFX_Shot_Pistol_Distant_SC (Sound Cont
Example_SFX_Shot_Pistol_Low_SC (Sound Contain
Example_SFX_Shot_Pistol_Mechanism_SC (Sound (
Drop SoundContainers/Folders Here
Timeline
Reset
Focus
0s
0.12s
0.25s
0.37s
Example_SFX_Shot_Pistol_Close_SC -7.6 dB
Example_SFX_Shot_Pistol_Distant_SC 0 dB
Example_SFX_Shot_Pistol_Low_SC 0 dB
Example_SFX_Shot_Pistol_Mechanism_SC 0 dB
Base
Polyphony
1
Volume dB
0
-1 dB
+1 dB
Modifiers
Add/Remove
Reset
Clear
Settings
Polyphony Mode
Limited Per Owner
AudioMixerGroup
None (Audio Mixer Group)
SoundVolumeGroup
None (Sound Volume Group Base)
Advanced
Intensity
Trigger Other
SoundTag
Debug

SoundEvents are what you play in Sonity.
They contain SoundContainers and options of how the sound should be played.
All SoundEvents are multi-object editable.

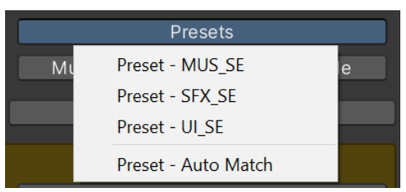
4.1 SoundEvent Notes

Example note:



At the top of the SoundEvent there is a text field which you can fill with notes.

4.1 SoundEvent Presets



The “Presets” shown in the example picture above need to be created with the SoundPreset object.

SoundPresets
Create SoundPreset objects to get custom settings which you can apply to your SoundContainers and SoundEvents.
Either manually select the preset you want to apply in the dropdown, or use “Auto Match” to automatically apply the settings based on the name of the asset.

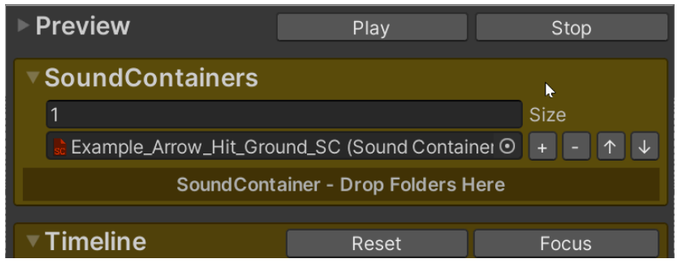
4.2 SoundEvent Mute, Solo, Disable

Mute
Mutes the SoundEvent.
Only affects the Unity Editor.

Solo
Mutes all other SoundEvents who don't have solo enabled.
The Solo property is not serialized (e.g. will be reset on start).
This is to prevent leaving a SoundEvent soloed by mistake which would make nothing else sound when the game runs.
Only affects the Unity Editor.

Disable
Disables the playing of the SoundEvent.
It is also disabled when building the project.

4.3 SoundEvent Preview



With the preview you can listen to the sounds in the editor.
The preview handle enables you to move the sound around.

Preview Shortcuts
Play: Ctrl+Q (Previews the selected SoundEvent or SoundContainer).
Stop: Ctrl+W (Stops any playing preview of SoundEvents and SoundContainers, press two times to skip fade out).
They are rebindable in the Shortcuts window.

Play
Previews the SoundEvent.
Does not work if Unity cannot build the project, or if the game is paused.
Preview doesn't play more than one level of TriggerOnPlay/Stop/Tail at the moment (there is full functionality ingame).
The default shortcut is Ctrl+Q.

Stop
Press two times to skip the fade out.
The default shortcut is Ctrl+W.

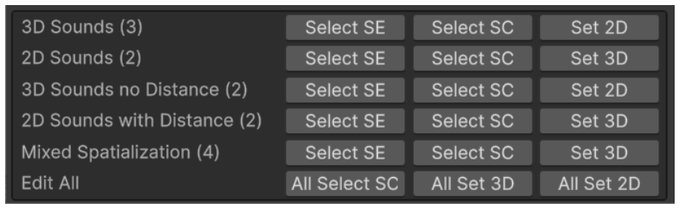
Reset
Resets the preview settings.

Intensity
Controls the intensity value of the played SoundContainers.

AudioMixerGroup
Only used for preview.

4.4 SoundEvent Spatialization Info

Show and edit spatialization info for the selected SoundEvents.



3D Sounds
All SoundEvents with SoundContainers which have Spatial Blend > 0 and Distance Enabled.

2D Sounds
All SoundEvents with SoundContainers which have Spatial Blend = 0 and Distance Disabled.

3D Sound no Distance
All SoundEvents with SoundContainers which have Spatial Blend > 0 and Distance Disabled.

2D Sound with Distance
All SoundEvents with SoundContainers which have Spatial Blend = 0 and Distance Enabled.

2D Sound with Distance
All SoundEvents with SoundContainers which have mixed Spatial Blend and Distance Enabled settings.

Empty
All SoundEvents which have no SoundContainers.

Select SE
Selects SoundEvents matching the specific settings.

Select All SE
Selects All SoundEvents in the same folder disregarding subfolders.

Select SC
Selects SoundContainers of the SoundEvents matching the specific settings.

Set 3D
Sets the SoundContainers of the selected SoundEvents to Spatial Blend 1 and Distance Enabled.

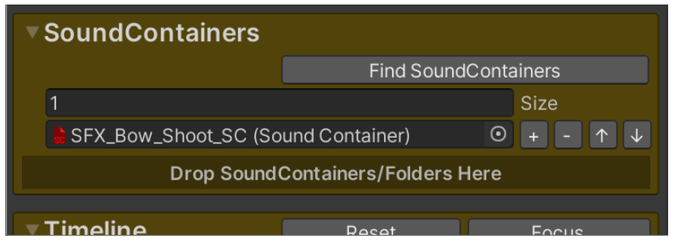
Set 2D
Sets the SoundContainers of the selected SoundEvents to Spatial Blend 0 and Distance Disabled.

All Select AC
Selects all SoundContainers of the selected SoundEvents.

All Set 3D
Sets all SoundContainers of the selected SoundEvents to Spatial Blend 1 and Distance Enabled.

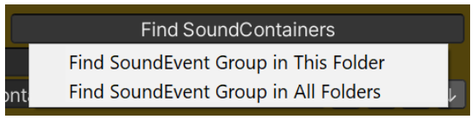
All Set 2D
Sets all SoundContainers of the selected SoundEvents to Spatial Blend 0 and Distance Disabled.

4.5 SoundEvent SoundContainers



This is where you assign the SoundContainers which are going to be used when the SoundEvent is played.

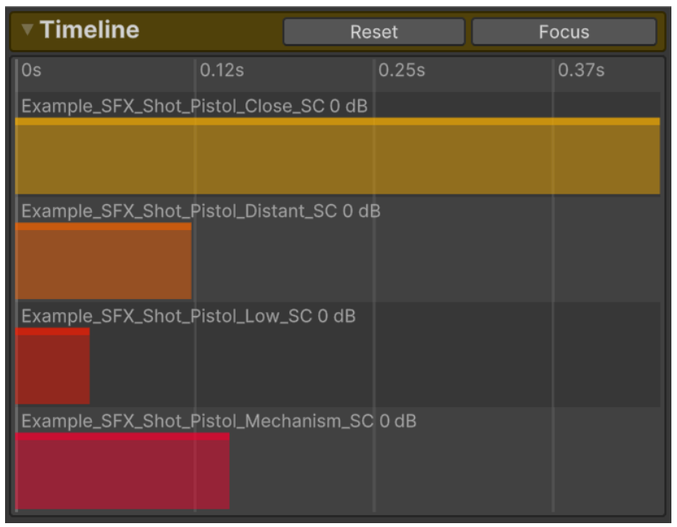
Find SoundContainers



Automatically finds all SoundContainers containing the same name as this SoundEvent (disregarding _SE, numbers).
Can be used on multiple items at once (tip: search "t:soundEvent").
If no matching SoundContainers are found, it will try and remove one character at the end of the name at a time until it finds a hit.";

Drop SoundContainers/Folders Here
Here you can drag and drop SoundContainers or folders with SoundContainers in them.

4.6 SoundEvent Timeline



In the timeline editor you can move the items in time and change the volume volume of individual SoundContainers.
The length of the item in the timeline is determined by the length of the longest AudioClip in the SoundContainer.
If you change the pitch, the length of the item will match the pitch.

Shortcuts and Controls
Zoom: Ctrl + mouse wheel scroll.
Pan: Mouse wheel hold and drag (or left mouse button hold and drag on the background).
Volume: Hold and drag top of item up/down or click on the volume to write the decibel value.
Move item: Hold and drag on item left/right.
Focus on items: F

4.7 SoundEvent Base

Contains base settings for the SoundEvent.

Polyphony
How many instances of the SoundEvent that can exist at the same transform.
Modifier polyphony overrides this value.

Volume dB
Volume offset in decibel.
If you want to be able to raise the volume and not just lower it you have 2 options:

Option A:
Enable "Enable Volume Increase" in the SoundManager.
This will enable you to raise the volumes of SoundContainers and SoundEvents by +24 dB each and SoundVolumeGroups by +12dB.

Option B:
Select all the SoundEvents and lower the volume by -20 dB with the -1dB button.
Then to compensate you can increase the global volume with an Audio Mixer (which you then can set to +20 dB).

> **💡 TIP - Convert from Old Modifier to Base**
> You can convert the old Modifier polyphony and volume to the new Base by selecting the assets and running:
> "Tools/Sonity 🔊/Convert SoundEvent Modifier Volume and Polyphony to Base"
> This will offset the Base volume with the Modifer volume and override the Base polphony with the Modifer polyphony.

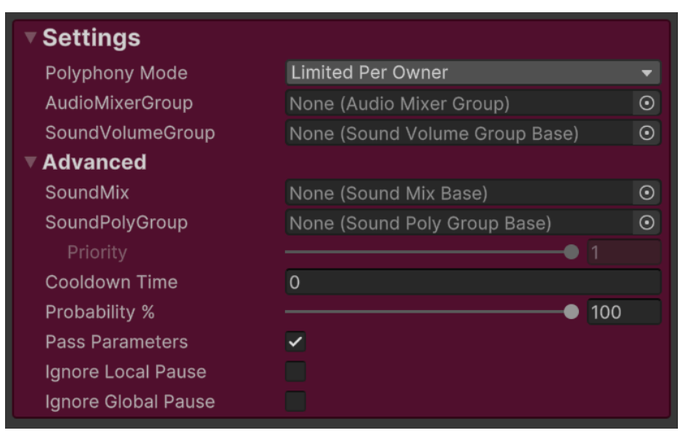
-1 dB
Lowers the relative volume of all the selected SoundEvents.
Useful for example if you want to raise the volume of one SoundEvent and keep the relative volume.
Because then you can lower all of them to get more headroom.
If multiple SoundEvents are selected it will show the lowest volume.

+1 dB
Raises the relative volume of all the selected SoundEvents.
Useful for example if you want to set the loudest volume to 0 dB but keep the relative volumes.
If multiple SoundEvents are selected it will show the highest volume.

4.8 SoundEvent Modifiers

Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
See Modifiers for more info.

4.9 SoundEvent Settings



Polyphony Mode

Limited Per Owner:
Useful if you want to limit polyphony e.g per player.
You can use e.g SoundEvent.PlayAtPosition(); to play a SoundEvent at one position with another owner.

Limited Globally:
Useful if you want to limit the polyphony globally e.g for bullet impacts.
This setting will change the old owner to the new position and set the new owner as the Transform of the SoundManager.Instance.
If you play the SoundEvent with either MusicPlay or UIPlay this setting will not have an effect.

❕ *Tip: If you want to limit the polyphony per owner and globally at the same time, you can use SoundPolyGroups.*

AudioMixerGroup
The AudioMixerGroup you want to output to.

> **💡 TIP - AudioMixerGroup Performance**
> If you have issues with audio performance you might want to make sure you're not using an unnecessary amount of AudioMixerGroups (like a hundred of them).
> Because changing AudioMixerGroup for a Voice uses performance (when playing a new Voice Sonity tries to find an unused Voice which uses the same AudioMixerGroup).
> Use AudioMixerGroups when you want effects per group or e.g. ducking or for larger groups of sounds, but maybe not a have a unique one for each sound.

SoundVolumeGroup
SoundVolumeGroup enables grouped volume control with live volume changes.
It is perfect for quickly changing the volume of a large group of sounds.
It also supports +12dB volume if you "Enable Volume Increase" in the SoundManager.

Advanced Settings

SoundMix
SoundMix enables hierarchical control of for example volume or distance scale etc.

SoundPolyGroup
SoundPolyGroup gives polyphony control grouped over different SoundEvent types.

Priority
Lower priority SoundEvents will be stolen first.
If "Skip Lower Priority" is enabled on the SoundPolyGroup this will determine if this SoundEvent will play or not when the Polyphony Limit is reached.

Cooldown Time
How quick this SoundEvent can be retriggered in seconds.
Is calculated using the time scale selected in the SoundManager.

Probability %

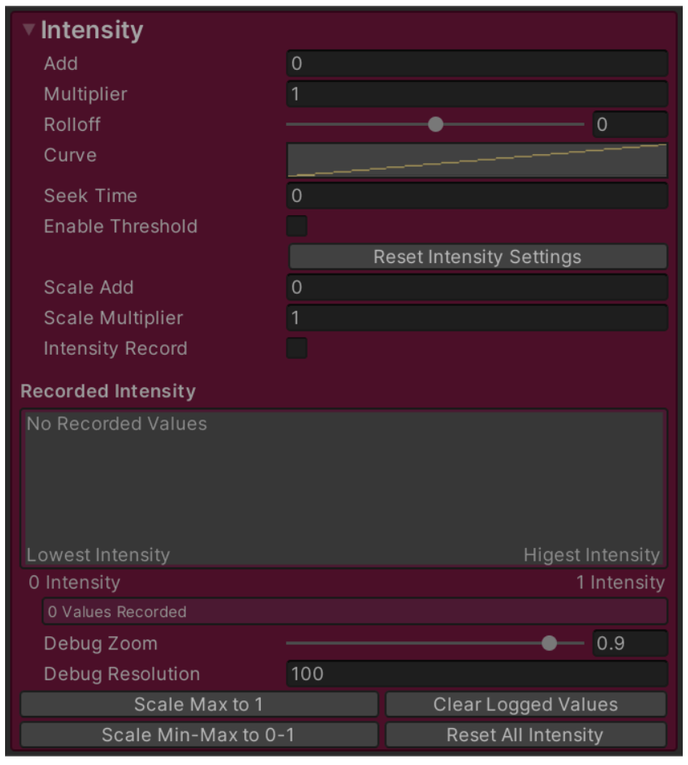
The probability that this SoundEvent should play.

Pass Parameters
If SoundParameters should be passed to sub.
E.g SoundTag, TriggerOnPlay, TriggerOnStop, TriggerOnTail.

Ignore Local Pause
If enabled, the SoundEvent won't be paused unless force pause is used.

Ignore Global Pause
If enabled, the SoundEvent won't be paused when global pause is used.

4.10 SoundEvent Intensity



Intensity contains settings for how SoundParameterIntensity is scaled before it is applied to the enabled intensity options e.g Volume, Pitch etc.

In the example above the sound is played a few times with a SoundParameterIntensity.
The range of the recorded intensity is from 0.5 to 2.5.
When “Scale Min-Max to 0-1” is pressed the following happens:

1. It first finds the Min value which is 0.5 and sets Add to -0.5 so it is 0
2. Then it finds the Max value which now is 2 and sets Multiplier to 0.5 so it is 1

This makes the range into 0 to 1.

The intensity debug log records all intensity values used when this SoundEvent is played.
It is used to scale the SoundParameterIntensity to a 0-1 range.

Add
Adds to the SoundParameterIntensity.

Multiplier
Multiplier of the SoundParameterIntensity.

Curve
Curve of the intensity.
From 0 (soft) to 1 (hard).

Smoothing
The seek time of the SoundParameterIntensity in seconds.
Is calculated using the time scale selected in the SoundManager.

Enable Threshold
If this SoundEvent is played with a SoundParameterIntensity and it is under the threshold it won't be played.

Threshold
The threshold limit after scaling the intensity value.

Scale Add
Adds to the SoundParameterIntensity, used by the automatic scaling.
Is separate from the manual add setting to provide finer control and better multi-editing.

Scale Multiplier
Multiplier of the SoundParameterIntensity, used by the automatic scaling.
Is separate from the manual multiplier setting to provide finer control and better multi-editing.

Intensity Record
If enabled it will record all SoundParameterIntensity used when playing this SoundEvent.

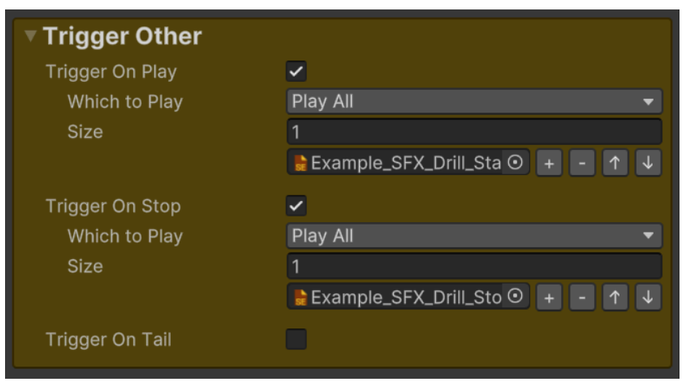
Debug Zoom
Focuses the vertical zoom scale to the values which are within the min and max range.

Debug Resolution
The resolution of the displayed values.

Scale Max to 1
Sets Intensity Add to 0.
Then uses the recorded values to scale the Max value to 1 with Intensity Multiply.
Useful for e.g. impacts where the Min value is 0 and the Max value is unknown.

Scale Min-Max to 0-1
Uses the recorded values to make the Min value 0 with Intensity Add.
Then uses the recorded values to scale the Max value to 1 with Intensity Multiply.
Useful where Min and Max Intensity value ranges are unknown.

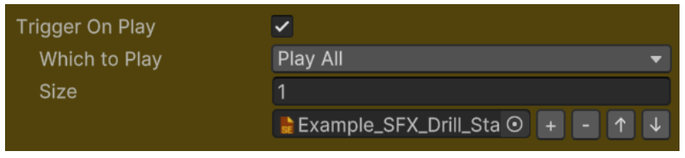
4.11 SoundEvent Trigger Other



Triggers other SoundEvents.

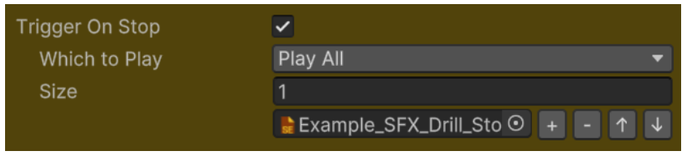
Which to Play
If Play All is selected, then all assigned SoundEvents will be played.
If One Random is selected, then one random of the assigned SoundEvents will be played.
The randomizer uses a pseudo random function remembering which SoundEvents it last played to avoid repetition.

Trigger On Play



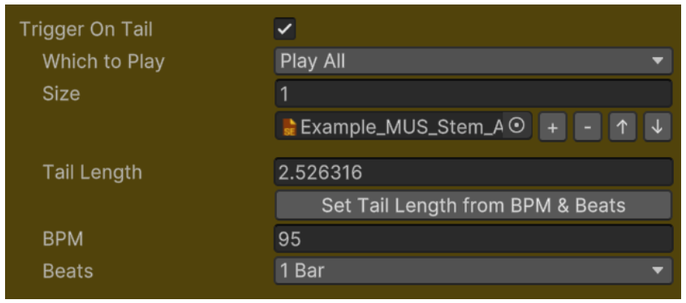
Triggers another SoundEvent when this SoundEvent is played.

Trigger On Stop



Triggers another SoundEvent when this SoundEvent is stopped.

Trigger On Tail



Triggers another SoundEvent "Tail Length" before the end.
It looks at the time of the last played voice on the first SoundContainer.

Useful for music, e.g. if you have an intro with a 2 second tail and a loop you want to play on the tail of the intro.
If you play with SoundManager.PlayMusic, you can stop the next SoundEvent with SoundManager.StopAllMusic without a reference.

If you want it to trigger itself, make sure to set the "Settings" polyphony to 2.
If the trigger timing is not tight enough, try setting the AudioClip "Compression Format" to PCM or ADPCM (Vorbis is less accurate).

Tail Length
How long in seconds before the end of the SoundEvent to trigger the next SoundEvent.

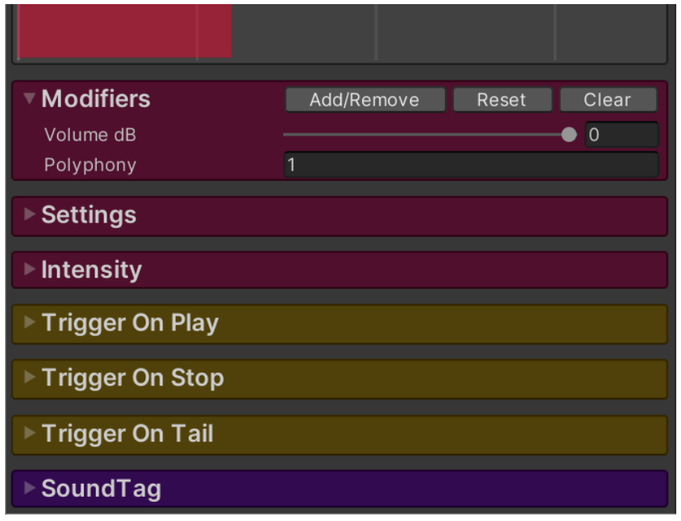
It takes into account pitch when calculating time.
For example, if the tail length is 2 seconds and you pitch it +12 semitones (2x speed) the internal tail length will be 1 seconds.
This is because double the speed with half the duration and vice versa.

Set Tail Length from BPM & Beats
Calculates the Tail Length from the BPM and the Beats settings.

BPM
Beats per minute.

Beats
How long the tail is in beats.

4.15 SoundEvent SoundTag



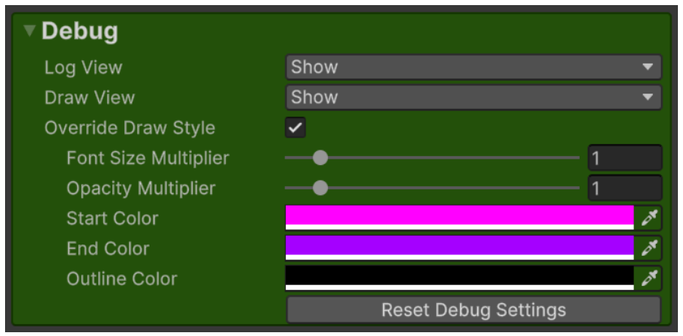
Uses SoundTag to play other SoundEvents and/or change SoundEventModifier.
The SoundTag won't be passed to the SoundEvents of the SoundTag in order to avoid infinite repetitions.

Mode
If local SoundTag is selected you need to pass an SoundTag when playing the SoundEvent.
If global SoundTag is selected you need to set the SoundTag on the SoundManager.

SoundTag
If this SoundTag is selected the SoundEvents below will be played.

Modifiers
If enabled the selected modifiers will be applied to the base SoundEvent and the SoundEvents of the SoundTag.
Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
See Modifiers for more info.

4.16 SoundEvent Debug



Contains settings on how the SoundEvent should be debugged.

Log View
If the SoundEvent should be viewed or hidden from the SoundManager Debug Log SoundEvents.
Force show will make it always debug even if SoundManager Debug Log SoundEvents is disabled.

Draw View
If the SoundEvent should be viewed or hidden from the SoundManager Debug Draw SoundEvents.

Override Draw Style
Overrides the debug draw style set in the SoundManager.

Font Size Multiplier
Multiplier of the font size.

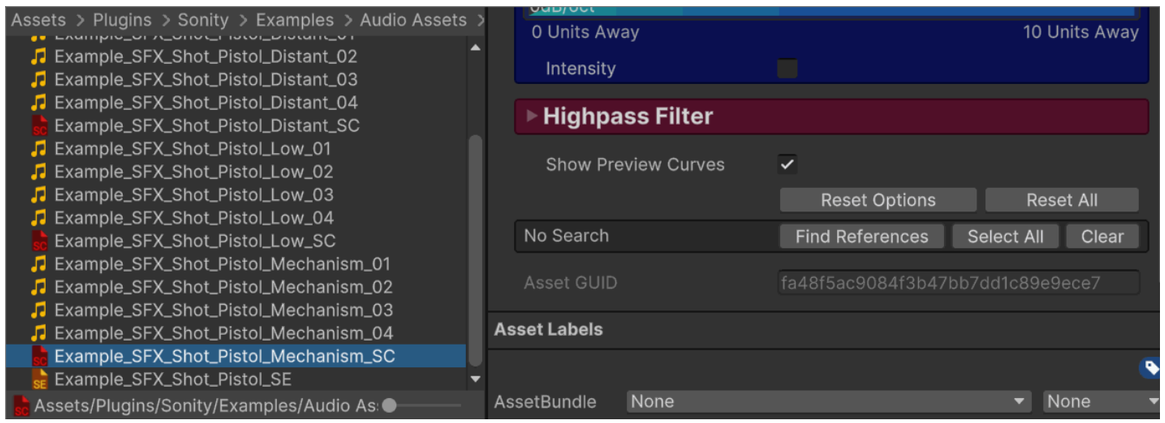
Opacity Multiplier
Multiplier of the opacity of the Debug Draw text.

Start Color
The color the text should have when it starts playing.

End Color
Which color the text should fade to over the lifetime.

Outline Color
The color of the text outline.

4.17 SoundEvent Find References



Find References
Finds all the references to the SoundEvent.

Select All
Selects all the assets with references to the SoundEvent.

Clear
Removes all the found references.

4.18 SoundEvent Asset GUID

The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

5 SoundEvent Functions

The SoundEvent can be played directly from itself instead of playing via the SoundManager.
A SoundManager instance in the scene is required to play SoundEvents.

Example code:

```
using UnityEngine;
using Sonity;

public class PlayStopExample : MonoBehaviour {

    public SoundEvent soundEvent;

    void PlayExample() {
        // Plays the SoundEvent at the position of the transform
        soundEvent.Play(transform);
    }

    void StopExample() {
        // Stops the SoundEvent playing at the transform
        soundEvent.Stop(transform);
    }
}
```

5.1 SoundEvent Play

```
// Plays the SoundEvent at the position of the owner Transform
public void Play(Transform owner)
public void Play(Transform owner, SoundTag localSoundTag)
public void Play(Transform owner, params SoundParameterInternals[] soundParameterInternals)
public void Play(Transform owner, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)
```

Parameters

owner	The owner **Transform**
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

5.2 SoundEvent PlayAtPosition

```
// Plays the SoundEvent at the Vector3 position with another Transform as the owner
public void PlayAtPosition(Transform owner, Vector3 position)
public void PlayAtPosition(Transform owner, Vector3 position, SoundTag localSoundTag)
public void PlayAtPosition(Transform owner, Vector3 position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Transform owner, Vector3 position, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)

// Plays the SoundEvent at the Transform position with another Transform as the owner
```

```
public void PlayAtPosition(Transform owner, Transform position)
public void PlayAtPosition(Transform owner, Transform position, SoundTag localSoundTag)
public void PlayAtPosition(Transform owner, Transform position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Transform owner, Transform position, SoundTag localSoundTag, params SoundParameterInternals[]
soundParameterInternals)
```

Parameters

owner	The owner **Transform**
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played

5.3 SoundEvent Stop

> 💡 **TIP - Stopping Loops**
>
> You really only need to use stop for looping or longer oneshot sounds (or the SoundContainer “Stop if Transform is Null” setting). Use the polyphony setting in the SoundEvent to manage the number of instances playing.

```
// Stops the SoundEvent at the owner Transform
public void Stop(Transform owner, bool allowFadeOut = true)

// Stops the SoundEvent at the position Transform
public void StopAtPosition(Transform position, bool allowFadeOut = true)

// Stops all the SoundEvents at the owner Transform
public void StopAllAtOwner(Transform owner, bool allowFadeOut = true)

// Stops the SoundEvent everywhere
public void StopEverywhere(bool allowFadeOut = true)

// Stops all SoundEvents
public void StopEverything(bool allowFadeOut = true)

// Stop allowing fade out (Useful for UnityEvents because it only has one parameter)
public void StopAllowFadeOut(Transform owner)
public void StopAtPositionAllowFadeOut(Transform position)
public void StopAllAtOwnerAllowFadeOut(Transform owner)

// Stop without fade out (Useful for UnityEvents because it only has one parameter)
public void StopImmediate(Transform owner)
```

```
public void StopAtPositionImmediate(Transform position)
public void StopAllAtOwnerImmediate(Transform owner)
```

Parameters

owner	The owner **Transform**
position	The position **Transform**
allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

5.4 SoundEvent Pause and Unpause

```
// Pauses/unpauses the SoundEvent with the owner Transform locally
public void Pause(Transform owner, bool forcePause = false)
public void Unpause(Transform owner)

// Pauses/unpauses all SoundEvents with the owner Transform locally
public void PauseAllAtOwner(bool forcePause = false)
public void UnpauseAllAtOwner()

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverywhere(bool forcePause = false)
public void UnpauseEverywhere()

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverything(bool forcePause = false)
public void UnpauseEverything()
```

Parameters

owner	The owner **Transform**
forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

5.5 SoundEvent Get State, Length, Time and Contains Loop

```
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState GetSoundEventState(Transform owner)
```

```
// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float GetLastPlayedClipLength(Transform owner, bool pitchSpeed)

// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float GetLastPlayedClipTimeSeconds(Transform owner, bool pitchSpeed)

// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float GetLastPlayedClipTimeRatio(Transform owner)

// Returns the max length (in seconds) of the SoundEvent (calculated from the longest audioClip)
// Is scaled by the pitch of the SoundEvent and SoundContainer
// Does not take into account random, intensity or parameter pitch
public float GetMaxLength()

// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float GetTimePlayed(Transform owner)

// Returns if any SoundContainers in the SoundEvent is set to looping
public bool GetContainsLoop()
```

Parameters

owner	The owner **Transform**
pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

5.6 SoundEvent Get Spectrum Data

```
// Provides a block of spectrum data from AudioSources
public void GetSpectrumData(Transform owner, ref float[] samples, int channel, FFTWindow window, SpectrumDataFrom spectrumDataFrom)
```

Parameters

owner	The owner **Transform**
samples	The array to populate with audio samples. Its length must be a power of 2
channel	The channel to sample from
window	The **FFTWindow** type to use when sampling
spectrumDataFrom	Where to get the spectrum data from

5.7 SoundEvent Get Last Played AudioSource

```
// Returns the last played AudioSource
// Note that the AudioSource might be stolen or reused for different Voices over time
public AudioSource GetLastPlayedAudioSource(Transform owner)
```

Parameters

owner	The owner **Transform**

5.8 SoundEvent Load or Unload Audio Data

```
// Loads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void LoadAudioData()

// Unloads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void UnloadAudioData()
```

5.9 SoundEvent UI

Useful if you want to play SoundEvents without passing an owner.

SoundEvent UI Play

```
// Plays the SoundEvent with the UI Transform as owner
// Useful to play e.g. UI or other 2D sounds without having to pass a Transform
// To make the sound 2D you still need to disable distance and set spatial blend to 0 in the SoundContainer
public void UIPlay()
public void UIPlay(SoundTag localSoundTag)
public void UIPlay(params SoundParameterInternals[] soundParameterInternals)
public void UIPlay(SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)
```

```
// Plays the SoundEvent at the position with the UI Transform as owner (Useful for UnityEvents because it only has one parameter)
public void UIPlayAtPosition(Vector3 position)
public void UIPlayAtPosition(Transform position)
```

Parameters

localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)

SoundEvent UI Stop

```
// Stops the SoundEvent at the UI Transform
public void UIStop(bool allowFadeOut = true)

// Stops all SoundEvents at the UI Transform
public void UIStopAll(bool allowFadeOut = true)
```

Parameters

allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

SoundEvent UI Pause and Unpause

```
// Pauses/unpauses the SoundEvent at the UI Transform locally
public void UIPause(bool forcePause = false)
public void UIUnpause()

// Pauses/unpauses all SoundEvents at the UI Transform locally
public void UIPauseAll(bool forcePause = false)
public void UIUnpauseAll()
```

Parameters

forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

SoundEvent UI Get State, Length and Time

```
// Uses the UI owner Transform
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
```

```
public SoundEventState UIGetSoundEventState()

// Uses the UI owner Transform
// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float UIGetLastPlayedClipLength(bool pitchSpeed)

// Uses the UI owner Transform
// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float UIGetLastPlayedClipTimeSeconds(bool pitchSpeed)

// Uses the UI owner Transform
// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float UIGetLastPlayedClipTimeRatio()

// Uses the UI owner Transform
// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float UIGetTimePlayed()

// Returns the owner Transform used by UIPlay() etc
public Transform UIGetTransform()
```

Parameters

pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

5.10 SoundEvent Music

SoundEvent Music Play

Useful if you want to play music with automatic stopping of the old music when playing a new song.

```
// Plays the SoundEvent at the SoundManagers music Transform
public void MusicPlay(bool stopAllOtherMusic = true, bool allowFadeOut = true)
```

```
public void MusicPlay(bool stopAllOtherMusic = true, bool allowFadeOut = true, params SoundParameterInternals[] soundParameterInternals)

// Plays allowing fade out (Useful for UnityEvents because it only has one parameter)
public void MusicPlayAllowFadeOut(bool stopAllOtherMusic = true)

// Plays without fade out (Useful for UnityEvents because it only has one parameter)
public void MusicPlayImmediate(bool stopAllOtherMusic = true)
```

Parameters

stopAllOtherMusic	If all other **SoundEvents** played at the **SoundManager** music **Transform** should be stopped
allowFadeOut	If the other stopped **SoundEvent** should be allowed to fade out. Otherwise they are going to be stopped immediately
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played

SoundEvent Music Stop

```
// Stops the SoundEvent playing at the Music Transform
public void MusicStop(bool allowFadeOut = true)

// Stops all the SoundEvents playing at the Music music Transform
public void AllMusicStop(bool allowFadeOut = true)
```

Parameters

allowFadeOut	If the other stopped **SoundEvent** should be allowed to fade out. Otherwise they are going to be stopped immediately

SoundEvent Music Pause and Unpause

```
// Pauses/unpauses the SoundEvent at the Music Transform locally
public void MusicPause(bool forcePause = false)
public void MusicUnpause()

// Pauses/unpauses all SoundEvents at the Music Transform locally
public void AllMusicPause(bool forcePause = false)
public void AllMusicUnpause()
```

Parameters

forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

SoundEvent Music Get State, Length and Time

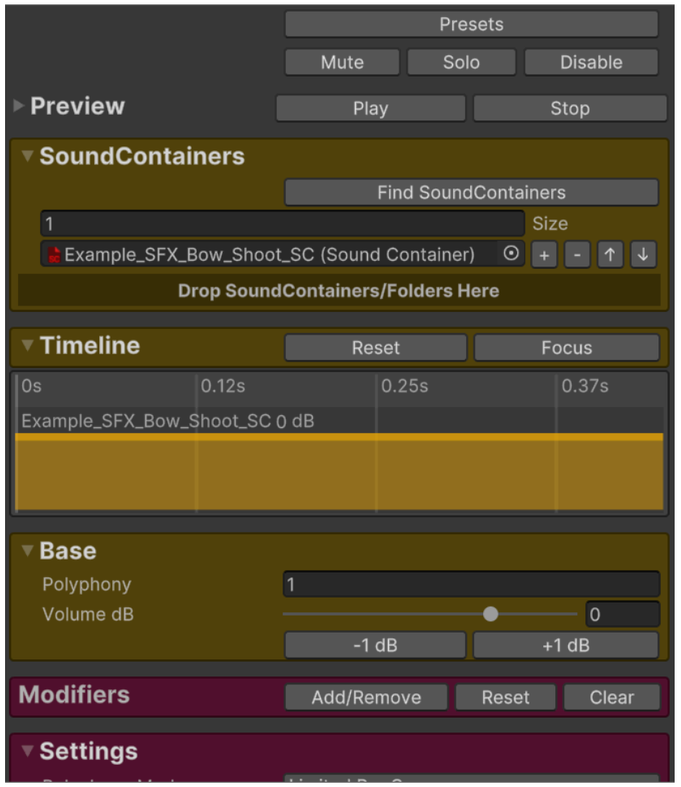
```
// Uses the Music owner Transform
// If playing it returns SoundEventState.Playing
```

```
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState MusicGetSoundEventState()

// Uses the Music owner Transform
// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float MusicGetLastPlayedClipLength(bool pitchSpeed)

// Uses the Music owner Transform
// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float MusicGetLastPlayedClipTimeSeconds(bool pitchSpeed)

// Uses the Music owner Transform
// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float MusicGetLastPlayedClipTimeRatio()

// Uses the Music owner Transform
// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float MusicGetTimePlayed()

// Returns the owner Transform used by PlayMusic() etc
public Transform MusicGetTransform()
```

Parameters

pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

6 Modifiers



Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
They are available on the SoundEvent, SoundTrigger, SoundPicker and SoundMix objects.
They have similar functionality as the SoundParameters, but with visual editing.
Modifiers are only updated once when starting the SoundEvent, not continuously.

Some modifiers use priorities to determine what a final modifier value should be.
The modifier with the highest priority will determine the value.
1st: SoundParameter
2nd: SoundMix
3rd: SoundTrigger/SoundPicker
4th: SoundTag
5th: SoundEvent

Volume dB
Volume offset in decibel.
Note that Modifiers are only updated once when starting the SoundEvent.
If you want realtime volume settings for a loop, use the SoundEvent base, timeline or SoundContainer volume.

Pitch st
Pitch offset in semitones.

Delay
Increase the delay in seconds.

Start Position
Sets the start position, 0 is the start, 1 is the end.
The highest priority added modifier will determine the value.

Reverse
If enabled the AudioClip will be played backwards.
Make sure to set the start position to the end.
Reverse is only supported for AudioClips which are stored in an uncompressed format or will be decompressed at load time.
The highest priority added modifier will determine the value.

Distance Scale
Distance scale multiplier (how far it will be heard).
It is multiplied by the Distance Scale of the SoundManager.

Reverb Zone Mix dB
Reverb Zone Mix volume offset in decibel.

Fade In Length
The length of the fade in.
Is calculated using the time scale selected in the SoundManager.
The highest priority added modifier will determine the value.

Fade In Shape
Shape of the fade in
Negative is exponential, 0 is linear, positive is logarithmic.
The highest priority added modifier will determine the value.

Fade Out Length
The length of the fade out.
Is calculated using the time scale selected in the SoundManager.
The highest priority added modifier will determine the value.

Fade Out Shape
Shape of the fade out
Negative is exponential, 0 is linear, positive is logarithmic.
The highest priority added modifier will determine the value.

Increase 2D
Makes the SoundEvent more 2D (less spatialized).
Useful for first person sounds.

Stereo Pan
Stereo pan offset
-1 is left, 0 is centered, +1 is right.

Intensity
Multiplier of any used SoundParameterIntensity parameter.

Distortion Increase
Increases the distortion.
Distortion needs to be enabled on the SoundContainer for this to have any effect.

Polyphony
How many instances of the SoundEvent that can exist at the same transform.
Modifier polyphony overrides the SoundEvents base Polyphony value.
The highest priority added modifier will determine the value.

Follow Position
If the SoundEvent should follow the given Transform position.
The highest priority added modifier will determine the value.

Bypass Reverb Zones
If enabled all reverb zones will be bypassed.
The highest priority added modifier will determine the value.

Bypass Voice Effects
If enabled all voice effects (lowpass/highpass/distortion) will be bypassed.
The highest priority added modifier will determine the value.

Bypass Listener Effects
If enabled all listener effects will be bypassed.
The highest priority added modifier will determine the value.

7 SoundParameter

SoundParameters are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
They can be passed when playing a SoundEvent and set to update either once or continuously.
They have similar functionality as the Modifiers, but with scripting capabilities.

Example code:

```
using UnityEngine;
using Sonity;

public class ExampleSoundParameter : MonoBehaviour {
```

```
    public SoundEvent soundEvent;
    SoundParameterVolumeDecibel volumeParameter = new SoundParameterVolumeDecibel(0f, UpdateMode.Once);

    void Start() {
        // Sets the volume parameter to a random value between -12 dB and 0 dB
        volumeParameter.VolumeDecibel = Random.Range(-12f, 0f);

        // Plays the SoundEvent with the sound parameter
        soundEvent.Play(transform, volumeParameter);
    }
}
```

Update Modes

The UpdateMode enum determines if the SoundEvent will take the parameter into consideration only once at start or if it will be updated continuously.

```
// The SoundParameter will update only once
UpdateMode.Once

// The SoundParameter will update continuously
UpdateMode.Continuous
```

Parameter Types

The parameter type determines which parameter of the SoundEvent will be controlled.
See Modifiers for more detailed info on the individual parameters.

```
// Volume offset in decibel. Range -Infinity to 0
public class SoundParameterVolumeDecibel(float volumeDecibel = 0f, UpdateMode updateMode = UpdateMode.Once)

// Volume ratio multiplier. Range 0f to 1f
public class SoundParameterVolumeRatio(float volumeRatio = 1f, UpdateMode updateMode = UpdateMode.Once)

// Pitch offset in semitones
public class SoundParameterPitchSemitone(float pitchSemitone = 0f, UpdateMode updateMode = UpdateMode.Once)

// Pitch offset ratio multiplier. Range 0 to Infinity
public class SoundParameterPitchRatio(float pitchRatio = 1f, UpdateMode updateMode = UpdateMode.Once)

// Delay increase. Range 0 to Infinity
public class SoundParameterDelay(float delay = 0f)

// Makes the sound more 2D (less spatialized). Range 0 to 1
public class SoundParameterIncrease2D(float increase2D = 0f, UpdateMode updateMode = UpdateMode.Once)

// Controls the intensity of the SoundContainer
// Go to the SoundContainer to change how intensity will affect the sound
public class SoundParameterIntensity(float intensity = 1f, UpdateMode updateMode = UpdateMode.Once)

// Reverb zone mix offset in decibel. Range -Infinity to 0
public class SoundParameterReverbZoneMixDecibel(float reverbZoneMixDecibel = 0f, UpdateMode updateMode = UpdateMode.Once)

// Reverb zone mix ratio multiplier. Range 0 to 1
public class SoundParameterReverbZoneMixRatio(float reverbZoneMixRatio = 1f, UpdateMode updateMode = UpdateMode.Once)

// Reverb zone mix ratio multiplier. Range 0 to 1
public class SoundParameterReverbZoneMixRatio(float reverbZoneMixRatio = 1f, UpdateMode updateMode = UpdateMode.Once)

// Start position. Range 0 to 1
public class SoundParameterStartPosition(float startPosition = 0f)

// If the sound should be played backwards. If enabled, set the start position to the end.
// Reverse is only supported for AudioClips which are stored in an uncompressed format or will be decompressed at load time.
public class SoundParameterReverse(bool reverse = false)

// The polyphony of the SoundEvent. Range 1 to int.MaxValue
public class SoundParameterPolyphony(int polyphony = 1)

// Distance scale multiplier. Range 0 to Infinity
public class SoundParameterDistanceScale(float distanceScale = 1f)

// Distortion increase. Range 0 to 1
```

```
public class SoundParameterDistortionIncrease(float distortionIncrease = 1f, UpdateMode updateMode = UpdateMode.Once)

// Fade in length. Range 0 to Infinity
public class SoundParameterFadeInLength(float fadeInLength = 0f, UpdateMode updateMode = UpdateMode.Once)

// Fade in shape. Range -16 to 16 (negative is exponential, 0 is linear, positive is logarithmic)
public class SoundParameterFadeInShape(float fadeInShape = 2f, UpdateMode updateMode = UpdateMode.Once)

// Fade out length. Range 0 to Infinity
public class SoundParameterFadeOutLength(float fadeOutLength = 0f, UpdateMode updateMode = UpdateMode.Once)

// Fade out shape. Range -16 to 16 (negative is exponential, 0 is linear, positive is logarithmic)
public class SoundParameterFadeOutShape(float fadeOutShape = -2f, UpdateMode updateMode = UpdateMode.Once)

// If the SoundEvent should follow the Transform position
public class SoundParameterFollowPosition(bool followPosition = true, UpdateMode updateMode = UpdateMode.Once)

// If reverb zones should be bypassed
public class SoundParameterBypassReverbZones(bool bypassReverbZones = false, UpdateMode updateMode = UpdateMode.Once)

// If voice effects (distortion/lowpass/highpass) should be bypassed
public class SoundParameterBypassVoiceEffects(bool bypassEffects = false, UpdateMode updateMode = UpdateMode.Once)

// If listener effects should be bypassed
public class SoundParameterBypassListenerEffects(bool bypassListenerEffects = false, UpdateMode updateMode = UpdateMode.Once)
```

8 SoundParameterIntensity

SoundParameterIntensity is used to pass an intensity value to a SoundContainer when playing a SoundEvent.
This value can be anything you want to affect your sound, e.g. velocity, the size of an explosion, the speed of a car, etc.
The SoundContainer can scale and use the intensity parameter to control intensity options e.g. volume, pitch, distortion etc.

> ❗ *Tip: You can record and debug the intensity at the Intensity in the SoundContainer.*

Physics impacts example code:

```
using UnityEngine;
using Sonity;

public class ExampleIntensityParameterCollision: MonoBehaviour {

    public SoundEvent soundImpact;
    private SoundParameterIntensity soundIntensityParameter = new SoundParameterIntensity(1f, UpdateMode.Once);

    void OnCollisionEnter(Collision collision) {
        // Sets the intensity parameter to the velocity of the collision
        soundIntensityParameter.Intensity = collision.relativeVelocity.magnitude;

        // Plays the SoundEvent with the intensity parameter
        soundImpact.Play(transform, soundIntensityParameter);
    }
}
```

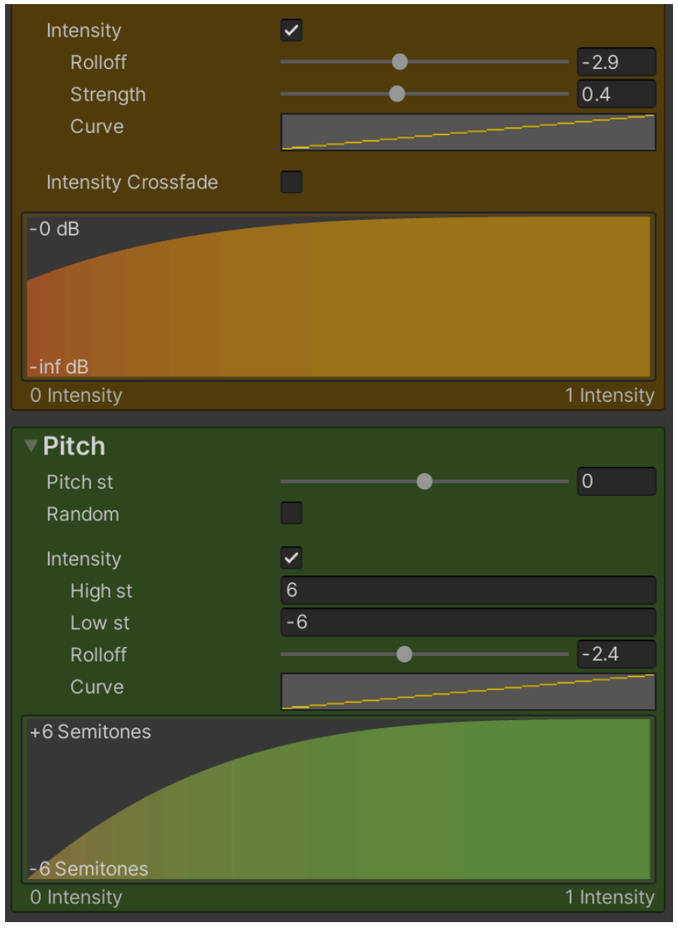
Car engine example code:

```
using UnityEngine;
using Sonity;

public class ExampleIntensityParameterContinuous: MonoBehaviour {

    public SoundEvent soundCarLoop;
    public bool soundCarIsPlaying = false;
    private SoundParameterIntensity soundIntensityParameter = new SoundParameterIntensity(0f, UpdateMode.Continuous);

    public Rigidbody carRigidbody;

    void FixedUpdate() {

        float velocity = carRigidbody.linearVelocity.magnitude;

        // Sets the intensity parameter to the velocity of the car every physics update
        intensityParameter.Intensity = velocity;

        if (!soundCarIsPlaying && velocity > 0.01f) {
            soundCarIsPlaying = true;

            // Plays the SoundEvent with the intensity parameter
            soundCarLoop.Play(transform, soundIntensityParameter);

        } else if (soundCarIsPlaying && velocity <= 0.01f) {
            soundCarIsPlaying = false;
```

```
            soundCarLoop.Stop(transform);
        }
    }
}
```

SoundContainer car engine example

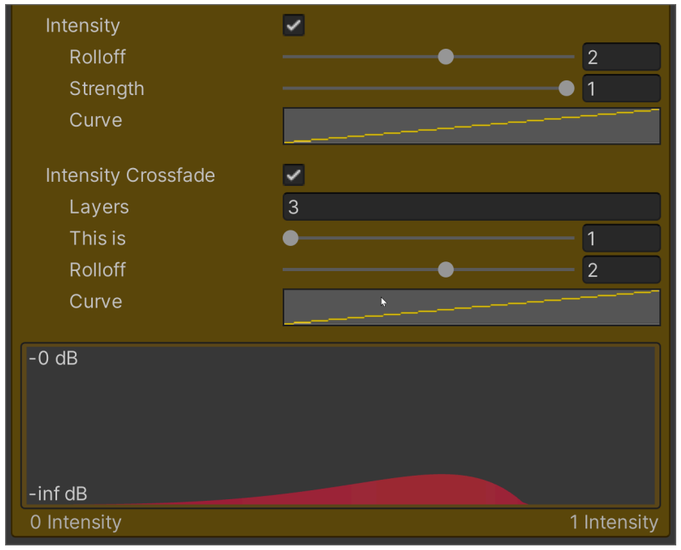
The speed of the car is passed as an continuous intensity parameter which controls the intensity pitch and volume of the SoundContainer.



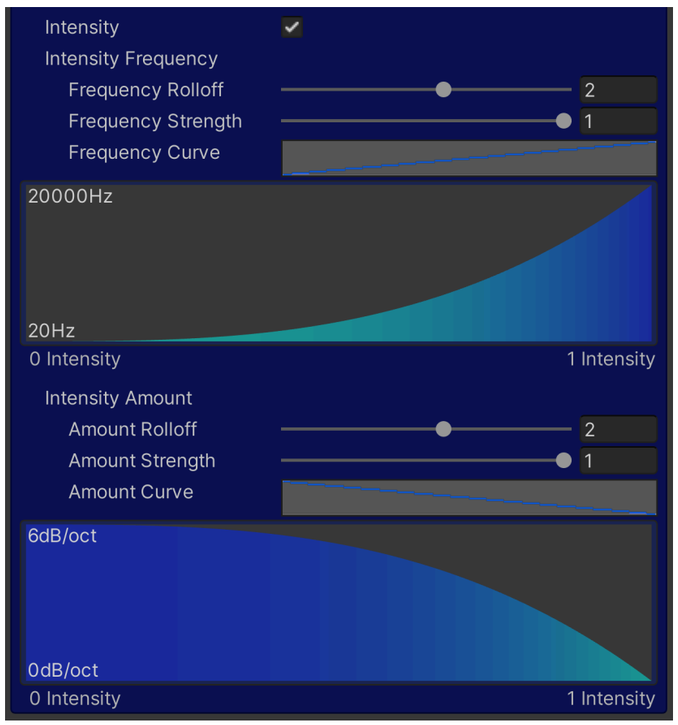
SoundContainer physics impact example

The velocity of the impact is passed once as an intensity parameter which controls the intensity volume and intensity lowpass filter of the SoundContainer.
(Check out the SoundPhysics component for easily playing physics sounds)

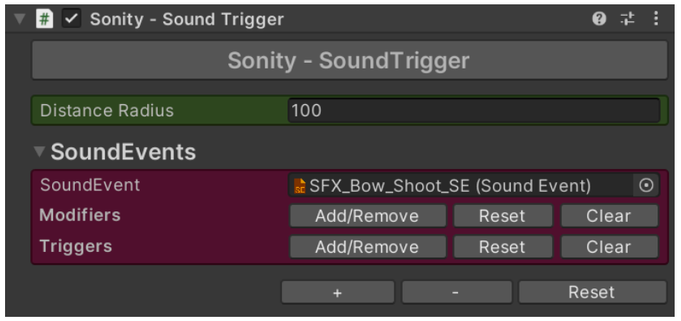
For volume intensity you can crossfade between any number of layers (eg. hard/medium/soft) easily and accurately.



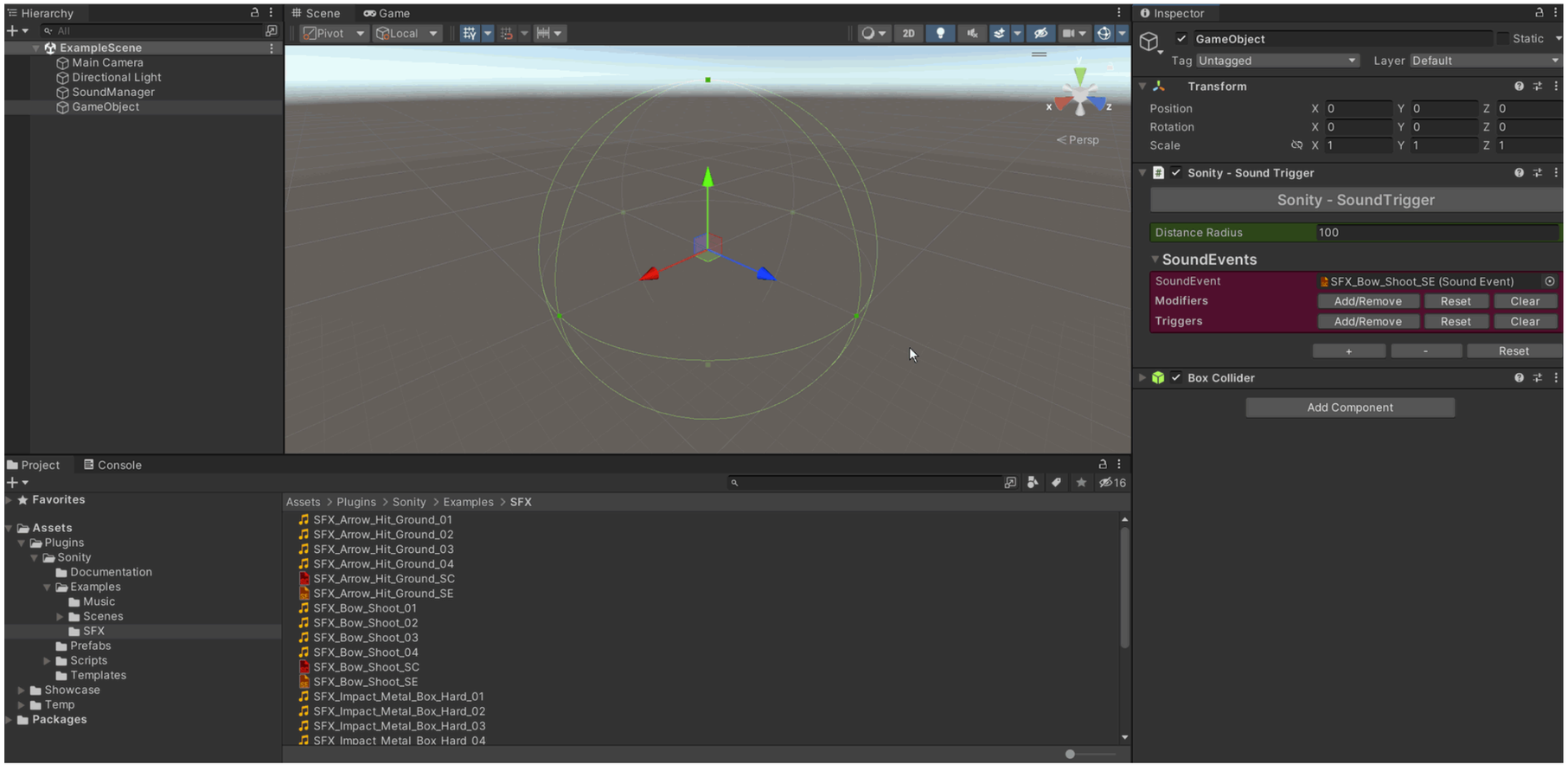
The default settings of the lowpass filter intensity settings make the sound more filtered the lower the intensity value is.
E.g. 0 intensity is maximum filter effect and 1 intensity is unfiltered.



9 SoundTrigger Component



SoundTrigger is a component used for easily playing/stopping SoundEvents on callbacks built into Unity like Enable, Disable, OnCollisionEnter etc.
They contain SoundEvents with modifiers and triggers which decide when it should play or stop.
SoundTriggers also have a radius handle, which is visually editable in the scene viewport for easy adjustment of how far SoundEvents should be heard.
All SoundTrigger components are multi-object editable.

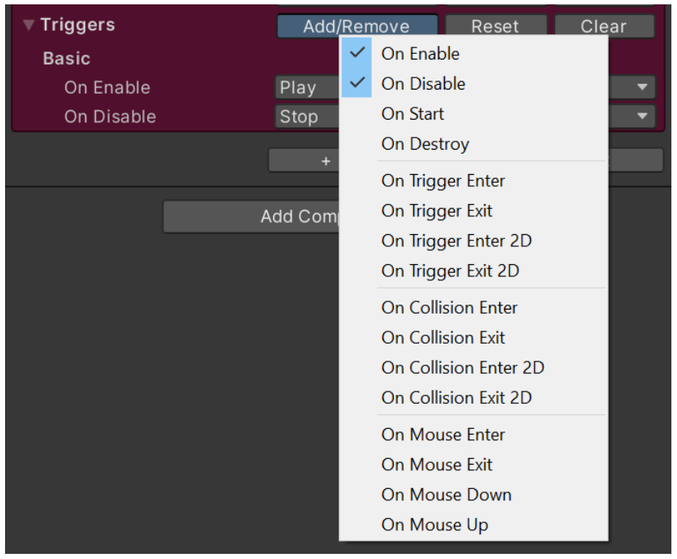


Distance Radius
Distance of the SoundEvent (how far it should be heard).

Modifiers
Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
See Modifiers for more info.

Triggers

Using the triggers you can select when an action happens.

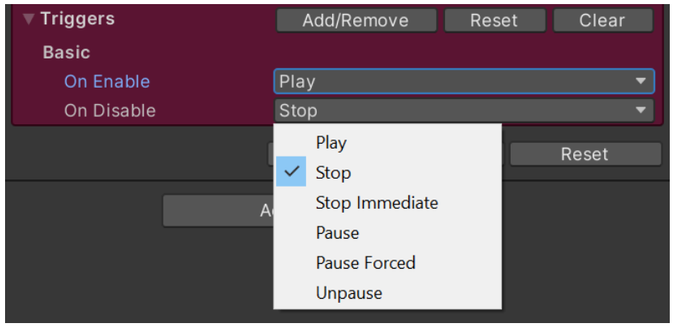


On Trigger and On Collision - Tag
If enabled, the SoundEvent will only play if the triggering object has a tag matching the selected tags.

On Collision - Velocity to Intensity
If enabled, the velocity magnitude will be passed as an intensity parameter.

Actions

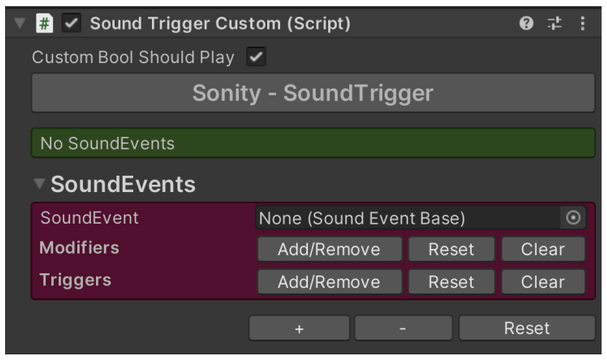
For each trigger you can select what it should do to the SoundEvent.



9.1 SoundTrigger Custom Override

You can override the functionality and editor of the SoundTrigger and extend it with custom functionality.

Example result:



SoundTriggerCustom.cs

```
using Sonity;

public class SoundTriggerCustom : SoundTrigger {

    // Add custom code or functionality here
    public bool customBoolShouldPlay = true;

    // Example override, you can override e.g. OnTriggerEnter etc.
    protected override void OnEnable() {
        if (customBoolShouldPlay) {
            base.OnEnable();
        }
    }
}
```

SoundTriggerCustomEditor.cs

```
using Sonity.Internal;
using UnityEditor;

#if UNITY_EDITOR

[CustomEditor(typeof(SoundTriggerCustom))]
[CanEditMultipleObjects]
public class SoundTriggerCustomEditor : SoundTriggerEditor {

    private SoundTriggerCustom soundTriggerCustom;

    protected override void OnEnable() {
        base.OnEnable();
        // Get the target property
        soundTriggerCustom = (SoundTriggerCustom)target;
    }

    public override void OnInspectorGUI() {

        // Draw the new property for the custom editor
        BeginChange();
        SerializedProperty customBoolShouldPlay = serializedObject.FindProperty(nameof(SoundTriggerCustom.customBoolShouldPlay));
        EditorGUILayout.PropertyField(customBoolShouldPlay);
        EndChange();

        // Draw the original GUI
        base.OnInspectorGUI();
    }
}
#endif
```

10 SoundTrigger Functions

10.1 SoundTrigger Play

```
// Plays the SoundEvent at the position of the owner Transform
public void Play()
public void Play(SoundTag localSoundTag)
public void Play(params SoundParameterInternals[] soundParameterInternals)
public void Play(SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)
```

Parameters

localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

10.2 SoundTrigger PlayAtPosition

```
// Plays the SoundEvent at the Vector3 position with another Transform as the owner
public void PlayAtPosition(Vector3 position)
public void PlayAtPosition(Vector3 position, SoundTag localSoundTag)
public void PlayAtPosition(Vector3 position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Vector3 position, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)

// Plays the SoundEvent at the Transform position with another Transform as the owner
public void PlayAtPosition(Transform position)
public void PlayAtPosition(Transform position, SoundTag localSoundTag)
public void PlayAtPosition(Transform position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Transform position, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)
```

Parameters

owner	The owner **Transform**
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

10.3 SoundTrigger Stop

> 💡 **TIP - Stopping Loops**
>
> You really only need to use stop for looping or longer oneshot sounds (or the SoundContainer “Stop if Transform is Null” setting). Use the polyphony setting in the SoundEvent to manage the number of instances playing.

```
// Stops the SoundEvent at the owner Transform
public void Stop(bool allowFadeOut = true)

// Stops the SoundEvent at the position Transform
public void StopAtPosition(Transform position, bool allowFadeOut = true)
```

Parameters

position	The position **Transform**
allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

10.4 SoundTrigger Pause and Unpause

```
// Pauses/unpauses the SoundEvent with the owner Transform locally
public void Pause(bool forcePause = false)
public void Unpause()

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverywhere(bool forcePause = false)
public void UnpauseEverywhere()
```

Parameters

forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

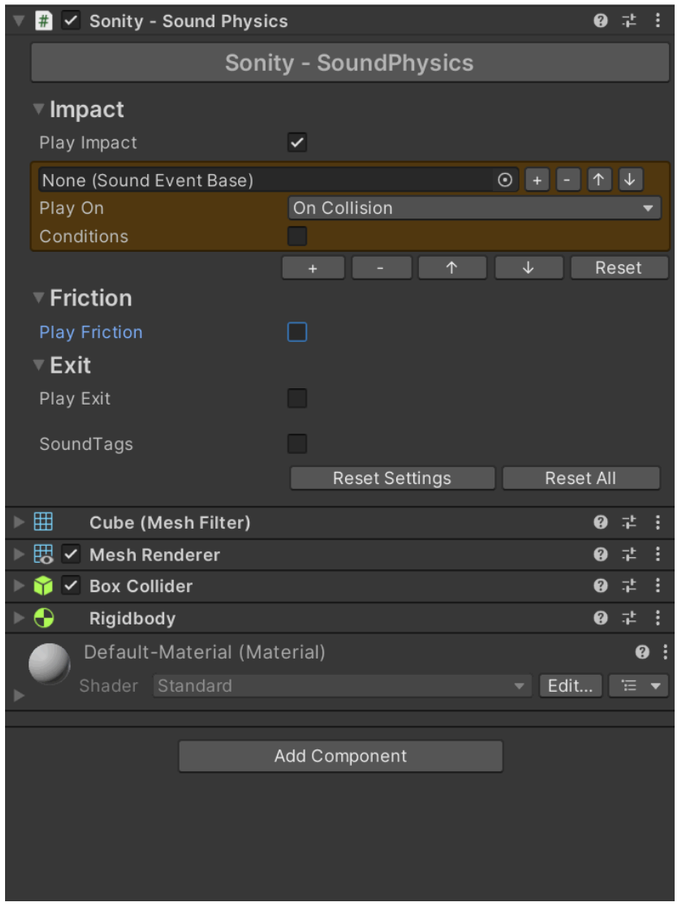
10.5 SoundTrigger Get State

```
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState GetSoundEventState(Transform owner)
```

10.6 SoundTrigger Load or Unload Audio Data

```
// Loads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void LoadAudioData()

// Unloads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void UnloadAudioData()
```

11 SoundPhysics



(SoundPhysics is remade in Sonity 1.0.5 with expanded functionality)

SoundPhysics is a component used for easily playing SoundEvents on physics interactions.
SoundPhysics is split up into 3D/2D versions with and without friction because of performance reasons.
If friction sounds aren't needed and performance is a priority then use the SoundPhysics with no friction.
A Rigidbody/Rigidbody2D is required on this object.
One or several Collider/Collider2D should be placed on this object or its children.
Use intensity record in the SoundEvent for easy scaling of the velocity into a 0 to 1 range.
All SoundPhysics components are multi-object editable.

Impact
Is triggered when the object starts touching a collider.
OnCollision uses velocity from Collision.relativeVelocity.magnitude and the position of the Collision.contacts with the highest impulse.magnitude.
OnTrigger uses velocity from Rigidbody.velocity.magnitude.

Friction
Is triggered when the object is continuously touching a collider.
Uses velocity from Rigidbody.velocity.magnitude.

Exit
Is triggered when the object stops touching a collider.
Uses velocity from Rigidbody.velocity.magnitude.

SoundEvent
The SoundEvents which are to be played.

Play On
Selects if the SoundEvents should be played when a OnCollision and/or OnTrigger event occurs.
OnTrigger is when you have a collider which is set to “Is Trigger”.

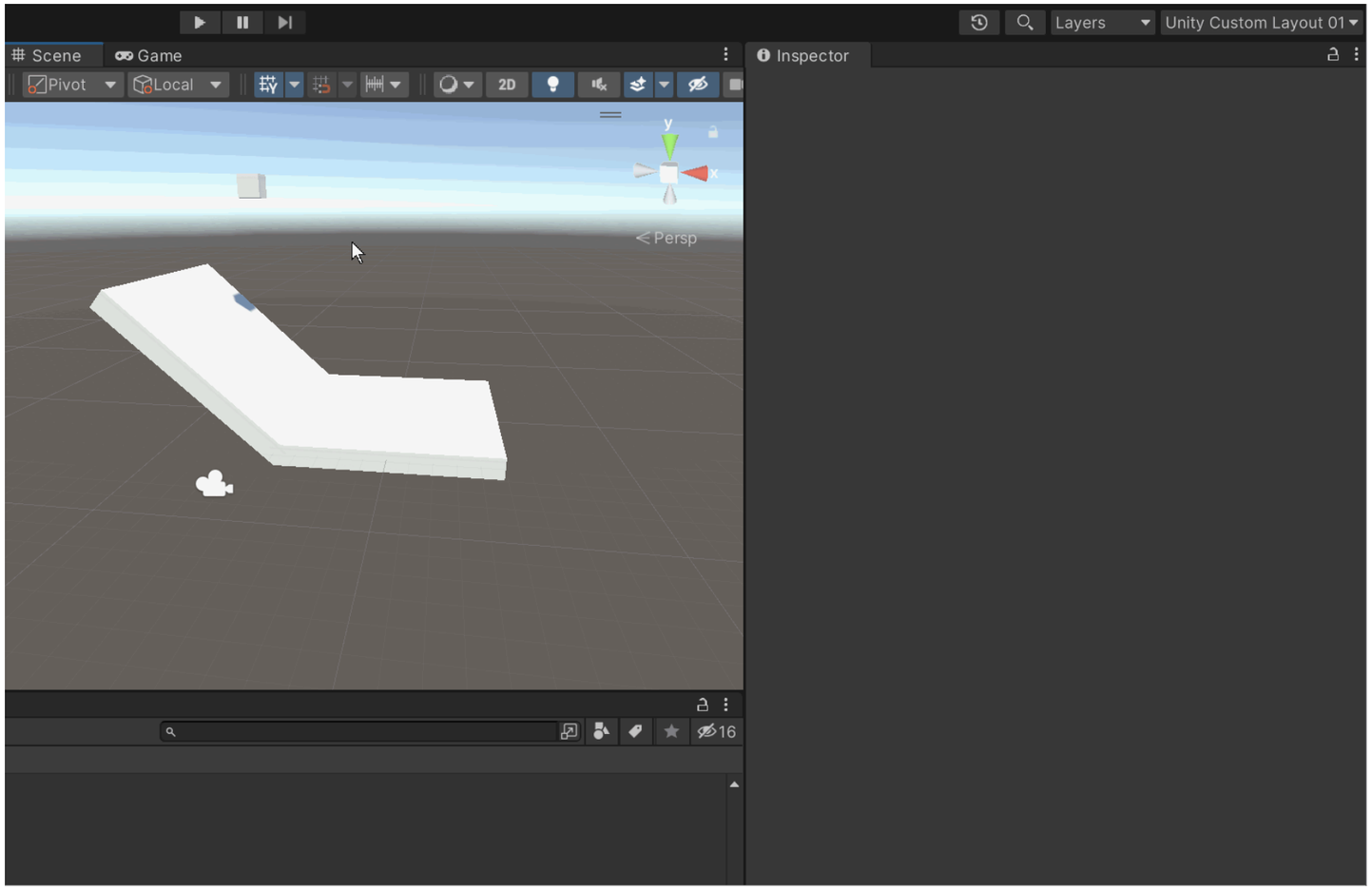
Conditions
If enabled then any SoundPhysicsCondition added will be used to decide if the physics interaction should play a SoundEvent or not.
SoundPhysicsConditions can also be used for playing different sounds with a single SoundEvent by assigning SoundTags in the SoundPhysicsConditions.

SoundTags
If enabled then any SoundTags added will be sent when triggering the SoundEvent.
If a SoundTag is not null it will override any SoundTags which might be sent through the SoundPhysicsConditions.
SoundTags can be used for triggering different sounds with a single SoundEvent.

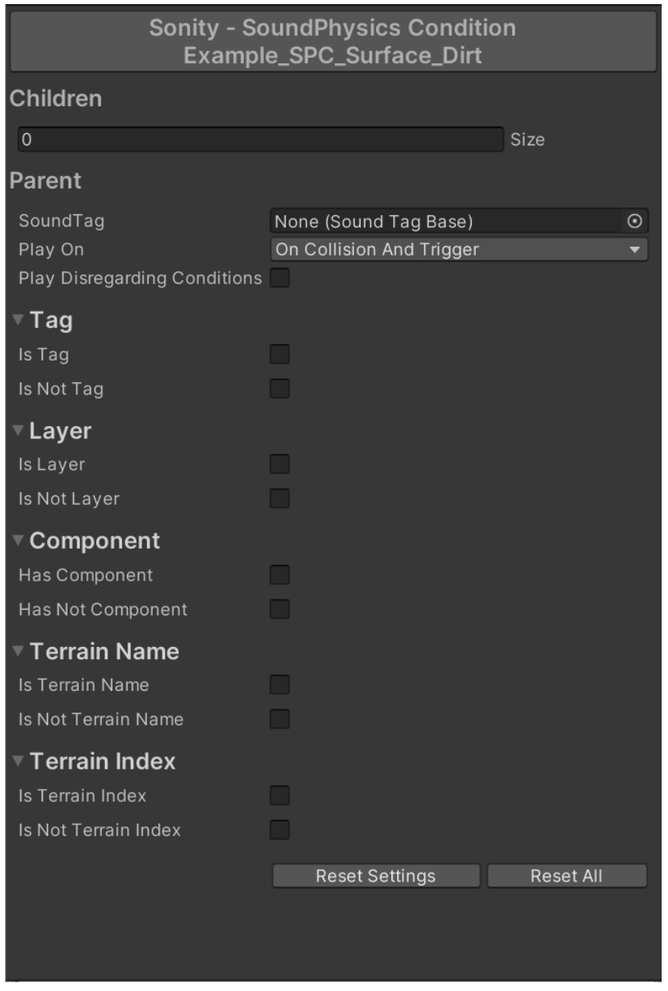
Intensity

All SoundEvents played with SoundPhysics are sent a velocity through the SoundParameterIntensity.
This can be used to control the sound e.g. making it quieter when the velocity is lower.
Read more at SoundParameterIntensity.
For how to debug playing sounds in the scene/game view check out SoundManager Draw SoundEvents.

Example of how to record and debug SoundParameterIntensity.
This is helpful for scaling the incoming velocity values to a 0 to 1 range.


Layers
Unity Custom Layout 01
Scene
Game
Inspector
Pivot
Local
2D
y
x
Persp
16

11.1 SoundPhysicsCondition



SoundPhysicsCondition objects are used in the SoundPhysics component to decide if a physics interaction should play a SoundEvent or not.
They make it easy to manage a large amount of physics objects and linking conditions together with the child nesting feature.
You can also use them for playing different sounds with a single SoundEvent by using SoundTags.
All SoundPhysicsCondition objects are multi-object editable.

Notes
At the top of the SoundPhysicsCondition there is a text field which you can fill with notes.

Children
You can nest SoundPhysicsConditions so you don't have to update all your SoundPhysics components when adding a new SoundPhysicsCondition.
The children of the parent SoundPhysicsCondition are evaluated after the parent.

Parent
The parent SoundPhysicsCondition is evaluated before the children.

SoundTag
By utilizing the SoundTags you can play different sounds with a single SoundEvent (see the example physics assets on how you can set that up).
If no SoundTag is assigned it won't affect the output SoundTag, so if a child has an assigned SoundTag and the parent doesn't, the child one will be used.
They are useful to split into different child SoundPhysicsConditions with different SoundTags.

Play On
Selects if it should be played when a OnCollision and/or OnTrigger event occurs.
OnTrigger is when you have a collider which is set to “Is Trigger”.

Play Disregarding Conditions
If enabled the SoundPhysicsCondition will play regardless of its own conditions.
Useful to combine with child SoundPhysicsConditions with different SoundTags.

Abort All On No Match
If enabled and the assigned condition is not matched it will abort playing and disregard all other conditions.
If not enabled and the assigned condition is not matched another condition which is met can play the sound.

Abort All On Match
If enabled and the assigned condition is matched it will abort playing and disregard all other conditions.
If not enabled and the assigned condition is matched another condition which is met can play the sound.

Tag
Checks if the Tag of the colliding GameObject matches any of the specified Tags.

Layer
Checks if the Layer of the colliding GameObject matches any of the specified Layers.

Terrain Name
Checks if the name of the most dominant Terrain Layer of the colliding Terrain contains any of the specified strings (it is not case sensitive).
E.g. you have multiple Terrain Layers with different grass textures, if the names of the Terrain Layers contain the string it will play the sound.
Terrain conditions are only evaluated if the colliding object has a Terrain component.
The center point of the SoundPhysics object is used to calculate the dominant Terrain Layer because it is more stable than the contacts.

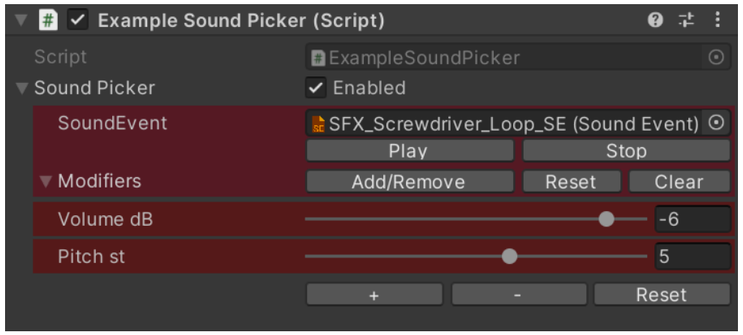
Terrain Index
Checks if the index of the most dominant Terrain Layer of the colliding Terrain is any of the specified indexes.
Terrain conditions are only evaluated if the colliding object has a Terrain component.
The center point of the SoundPhysics object is used to calculate the dominant Terrain Layer because it is more stable than the contacts.

Component
Checks if the Components of the colliding GameObject match any of the specified Components.
The names are case sensitive.

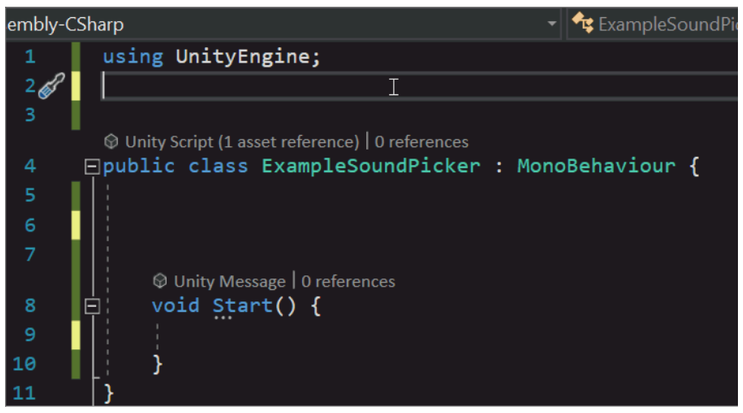
Asset GUID

The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

12 SoundPicker



SoundPicker is a serializable class for easily selecting multiple SoundEvents and modifiers.
Add a serialized or public SoundPicker to a C# script and edit it in the inspector.
The SoundPicker is not nestable in an array or custom class because it is built upon CustomPropertyDrawer.
This is because custom serializable classes do not support inheritance and polymorphism for serialization.
SoundPickers are multi-object editable.



SoundEvent
The SoundEvent to play.

Modifiers
Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
See Modifiers for more info.

Example code:

```
using UnityEngine;
using Sonity;

public class ExampleSoundPicker : MonoBehaviour {

    public SoundPicker soundPicker;

    void Start() {
        // Plays the SoundPicker at the position of the transform
        soundPicker.Play(transform);
    }
}
```

13 SoundPicker Functions

13.1 SoundPicker Play

```
// Plays the SoundEvent at the position of the owner Transform
public void Play(Transform owner)
public void Play(Transform owner, SoundTag localSoundTag)
public void Play(Transform owner, params SoundParameterInternals[] soundParameterInternals)
public void Play(Transform owner, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)
```

Parameters

owner	The owner **Transform**
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

13.2 SoundPicker PlayAtPosition

```
// Plays the SoundEvent at the Vector3 position with another Transform as the owner
public void PlayAtPosition(Transform owner, Vector3 position)
public void PlayAtPosition(Transform owner, Vector3 position, SoundTag localSoundTag)
```

```
public void PlayAtPosition(Transform owner, Vector3 position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Transform owner, Vector3 position, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)

// Plays the SoundEvent at the Transform position with another Transform as the owner
public void PlayAtPosition(Transform owner, Transform position)
public void PlayAtPosition(Transform owner, Transform position, SoundTag localSoundTag)
public void PlayAtPosition(Transform owner, Transform position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(Transform owner, Transform position, SoundTag localSoundTag, params SoundParameterInternals[]
soundParameterInternals)
```

Parameters

owner	The owner **Transform**
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

13.3 SoundPicker Stop

> 💡 **TIP - Stopping Loops**
>
> You really only need to use stop for looping or longer oneshot sounds (or the SoundContainer “Stop if Transform is Null” setting). Use the polyphony setting in the SoundEvent to manage the number of instances playing.

```
// Stops the SoundEvent at the owner Transform
public void Stop(Transform owner, bool allowFadeOut = true)

// Stops the SoundEvent at the position Transform
public void StopAtPosition(Transform position, bool allowFadeOut = true)
```

Parameters

owner	The owner **Transform**
position	The position **Transform**
allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

13.4 SoundPicker Pause and Unpause

```
// Pauses/unpauses the SoundEvent with the owner Transform locally
```

```
public void Pause(Transform owner, bool forcePause = false)
public void Unpause(Transform owner)

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverywhere(bool forcePause = false)
public void UnpauseEverywhere()
```

Parameters

owner	The owner **Transform**
forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

13.5 SoundPicker Get State

```
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState GetSoundEventState(Transform owner)
```

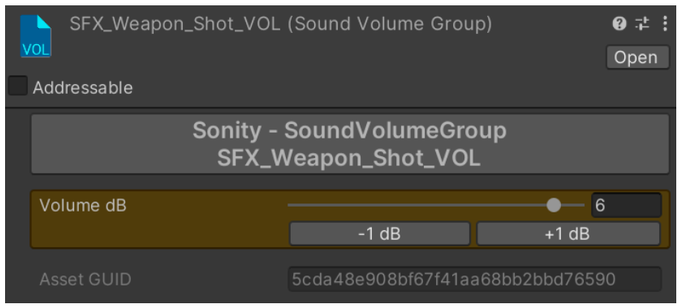
Parameters

owner	The owner **Transform**

13.6 SoundPicker Load or Unload Audio Data

```
// Loads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void LoadAudioData()

// Unloads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void UnloadAudioData()
```

14 SoundVolumeGroup

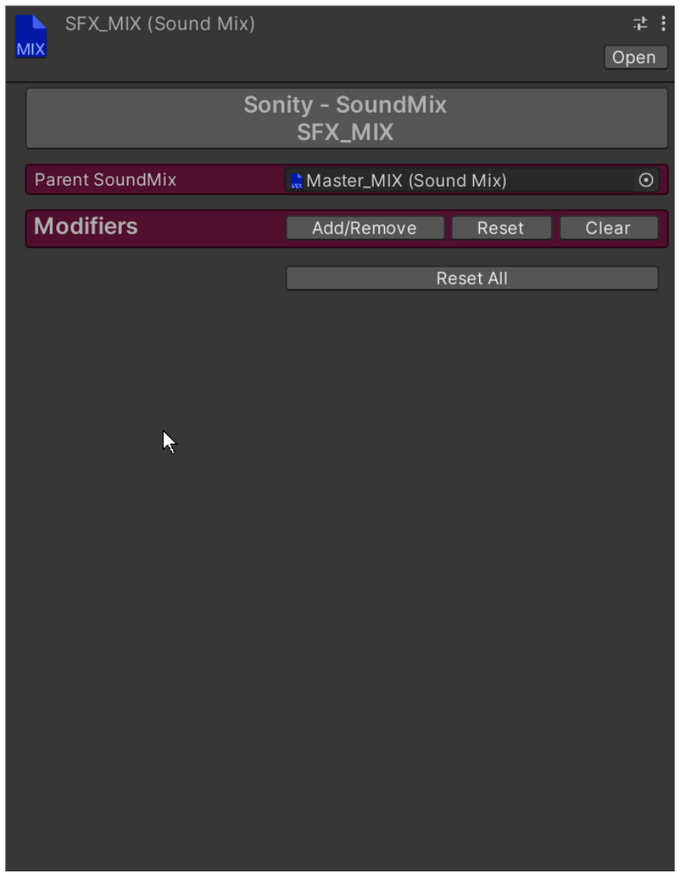


SoundVolumeGroup objects are used for grouped live volume control of multiple SoundEvents.
You can assign them in the SoundEvent settings.
It is perfect for quickly mixing the volume of a large group of sounds.
It also supports +12dB volume if you "Enable Volume Increase" in the SoundManager.
The volume is not meanted to be changed at runtime in builds.
If you want to control the volume in builds, take a look at the TemplateSoundVolumeManager script.
All SoundVolumeGroup objects are multi-object editable.

Example use: Add the same SoundVolumeGroup to all e.g. gunshot SoundEvents so you quickly can change their volumes.

Asset GUID
The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

15 SoundMix



SoundMix objects are used for grouped control of e.g. volume or distance scale for multiple SoundEvents at the same time.
You can assign them in the SoundEvent settings.
Because SoundMix use modifiers they only calculate the values once when the SoundEvent is started.
If you want to have realtime volume control over sounds, use an AudioMixerGroup.
To learn more, take a look at the TemplateSoundVolumeManager script.
All SoundMix objects are multi-object editable.

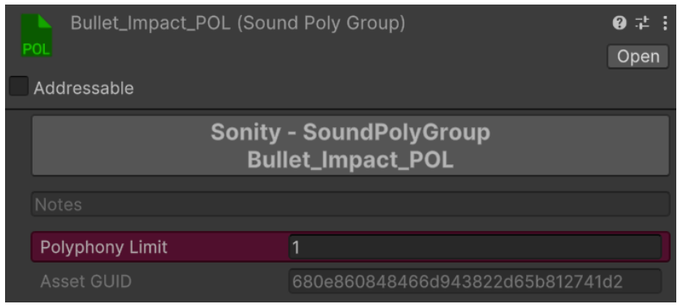
Example use: Set up a “Master_MIX” and a “SFX_MIX” where the Master_MIX is a parent of the SFX_MIX.

Notes
At the top of the SoundMix there is a text field which you can fill with notes.

Modifiers
Modifiers are used to control how SoundEvents are played (e.g. volume, polyphony, fade in length etc).
See Modifiers for more info.

Asset GUID
The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

16 SoundPolyGroup



SoundPolyGroup objects are used to create a polyphony limit shared by multiple different SoundEvents.
You can assign them in the SoundEvent settings.
The priority for voice allocation is calculated by multiplying the priority set in the SoundEvent by the volume of the instance.
A perfect use case would be to have a SoundPolyGroup for all bullet impacts of all the different materials so that when combined, they don’t use too many voices.
If you want simple individual polyphony control, use the polyphony modifier on the SoundEvent.
All SoundPolyGroup objects are multi-object editable.

Notes
At the top of the SoundPolyGroup there is a text field which you can fill with notes.

Polyphony Limit
The maximum number of SoundEvents which can be played at the same time in this SoundPolyGroup.

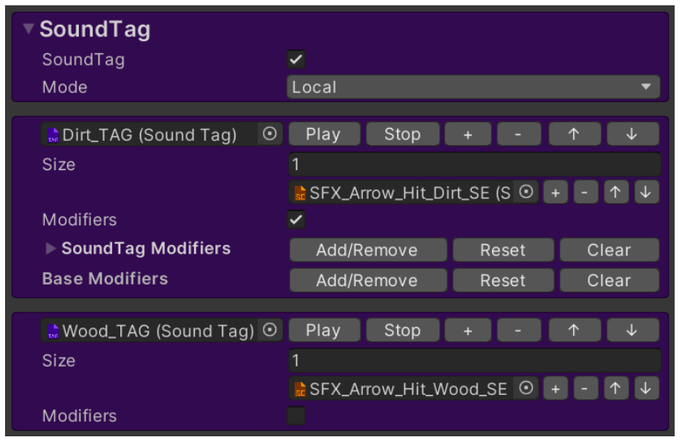
Asset GUID
The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

17 SoundTag

SoundTag objects are passed to modify how a SoundEvent should be played.
You can assign them in the SoundEvents SoundTag section.
You can either pass them when playing a SoundEvent for setting the local SoundTag.
Or you can set the global SoundTag in the SoundManager.

This is useful for e.g; weapon reverb zones.
You can set the SoundTag corresponding to the acoustic space which the listener is in.
And when you play the SoundEvent, your gun reflection layers can correspond to the acoustic space.

Used in a SoundEvent:



Notes
At the top of the SoundTag there is a text field which you can fill with notes.

Asset GUID
The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

18 SoundPreset


Sound Preset (Sound Preset)
Open
Sonity - SoundPreset
SoundPreset
Disable All
Automatic Loop
Preset - MUS
Disable
From SoundContainer
MUS_SC (Sound Container)
From SoundEvent
MUS_SE (Sound Event)
Match
Auto Match
Apply On All
Is Prefix
MUS
Is Not Prefix
Contains
Not Contains
+
-
Reset
Preset - SFX
Disable
From SoundContainer
SFX_SC (Sound Container)
From SoundEvent
SFX_SE (Sound Event)
Match
Auto Match
Apply On All
Is Prefix
SFX
Is Not Prefix
Contains
Not Contains
+
-
Reset
Preset - UI
Disable
From SoundContainer
UI_SC (Sound Container)
From SoundEvent
UI_SE (Sound Event)
Match
Auto Match
Apply On All
Is Prefix
UI
Is Not Prefix
Contains
Not Contains
+
-
Reset
Reset All

Use SoundPreset to apply settings to SoundContainers and SoundEvents automatically by matching the names of your assets.
Its matched and applied when you Create Assets From Selection or manually from the SoundContainer Presets and the SoundEvent Presets toolbars.
Example use: Name UI sounds to “UI_” and use Is Prefix “UI_” to automatically assign settings like disabling distance and setting Spatial Blend to 0.
The settings are applied ordering from the one highest up and going downwards and can match an asset multiple times overwriting previous values.
You can have multiple SoundPreset objects.

Notes
At the top of the SoundPreset there is a text field which you can fill with notes.

Example_SoundPreset
If you want a starting point for your SoundPreset, look at the Example_SoundPreset**.**
Note: The SoundContainers and SoundEvents have volume -12dB so you can increase the volume later.
You can compensate by increasing the volume in the AudioMixer.
All the SoundEvents are also assigned to a AudioMixerGroup in the Example_AudioMixer.

Example Templates

Template Default (Matches to all to catch any stray sounds without prefix)
Default settings
Outputs to SFX

Template AMB (2D Ambience)
Distance disabled
Spatial Blend 0

Template MUS (Music)
Distance disabled
Spatial Blend 0
Never Steal Voice
Never Steal Voice Effects
Priority 1

Template SFX (Sound Effects)
Default settings

Template UI (User Interface)
Distance disabled
Spatial Blend 0

Template VO (Voice/Dialogue)
Random Pitch disabled

Settings

Disable All
If disabled no settings from the SoundPreset will be applied automatically and not available in the preset menus.

Preset

Disable
If disabled this specific preset won’t be available.

Automatic Loop
If enabled and the SoundContainer name contains “loop” then it will automatically enable “Loop”, “Follow Position”, “Stop if Transform is Null” and “Random Start Position”.
This will be applied after all the other settings are copied.
Is not applied if you manually select a preset from the SoundContainer Presets dropdown.

From SoundContainer
The SoundContainer you want to copy the settings from.
Copies everything except:
- AudioClips

From SoundEvent
The SoundEvent you want to copy the settings from.
Copies everything except:
- SoundContainers
- Timeline Settings
- Intensity Scale Add/Multiplier
- TriggerOn SoundEvents
- SoundTag SoundEvents

Match

Auto Match
If enabled then automatic matching will be used when you Create Assets From Selection and in the SoundContainer Presets and the SoundEvent Presets “Auto Match”.
Disable if you just want presets to apply manually to your assets.
Auto matching is not case sensitive.

Apply On All
If enabled the settings assigned will be applied to all assets on asset creation and “Auto Match”.
Useful to combine with other settings with conditions added later in the chain to assign settings to assets which doesn’t match the names.

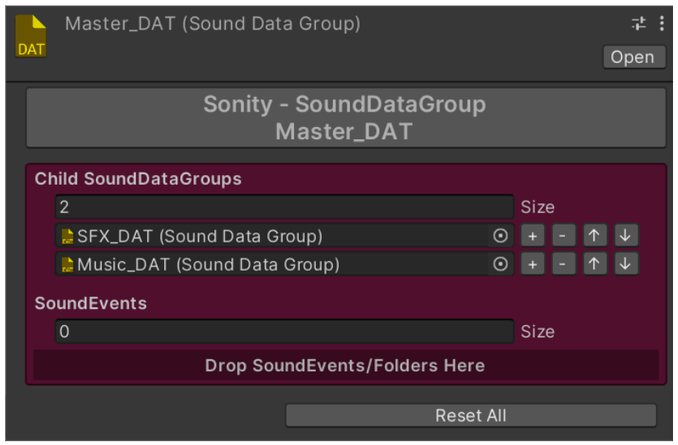
Is Prefix
If the prefix of an asset matches the strings, the settings will be applied.
Example prefixes: “AMB”, “MUS”, “SFX”, “UI”, “VO”.

Is Not Prefix
If the prefix of an asset doesn’t match the strings, the settings will be applied.

Contains
If the name of an asset contains the strings, the settings will be applied.

Not Contains
If the name of an asset doesn’t contain the strings, the settings will be applied.

19 SoundDataGroup



SoundDataGroup objects are used to easily load and unload the audio data of the SoundEvents.
All SoundDataGroup objects are multi-object editable.

Notes
At the top of the SoundDataGroup there is a text field which you can fill with notes.

Child SoundDataGroups
Nesting SoundDataGroups makes it easy to load/unload all audio data or just parts of it.

SoundEvents
The SoundEvent whoms audio data will be loaded or unloaded

Drop SoundEvents/Folders Here
Here you can drag and drop SoundEvents or folders with SoundEvents in them.

Asset GUID
The ID of an asset is cached in the editor because in some cases when using addressables a single scriptable object might be instantiated multiple times.
This might lead to problems where a simple "Object == Object" check will return false, but a "ID == ID" will return true.
If you are upgrading Sonity and use addressable assets you probably want to select all Sonity scriptable objects and run:
"Tools/Sonity 🔊/Set Selected Assets as Dirty (Force Reserialize for Resave)" which will force the assets to cache the GUIDs.
You can search t:SoundEvent, t:SoundContainer, t:SoundTag, t:SoundPolyGroup, t:SoundDataGroup, t:SoundMix, t:SoundPhysicsCondition to find all objects.
If no GUID is cached then it will use the InstanceID in builds.
The free trial version doesn't have full functionality.

Functions Load/Unload

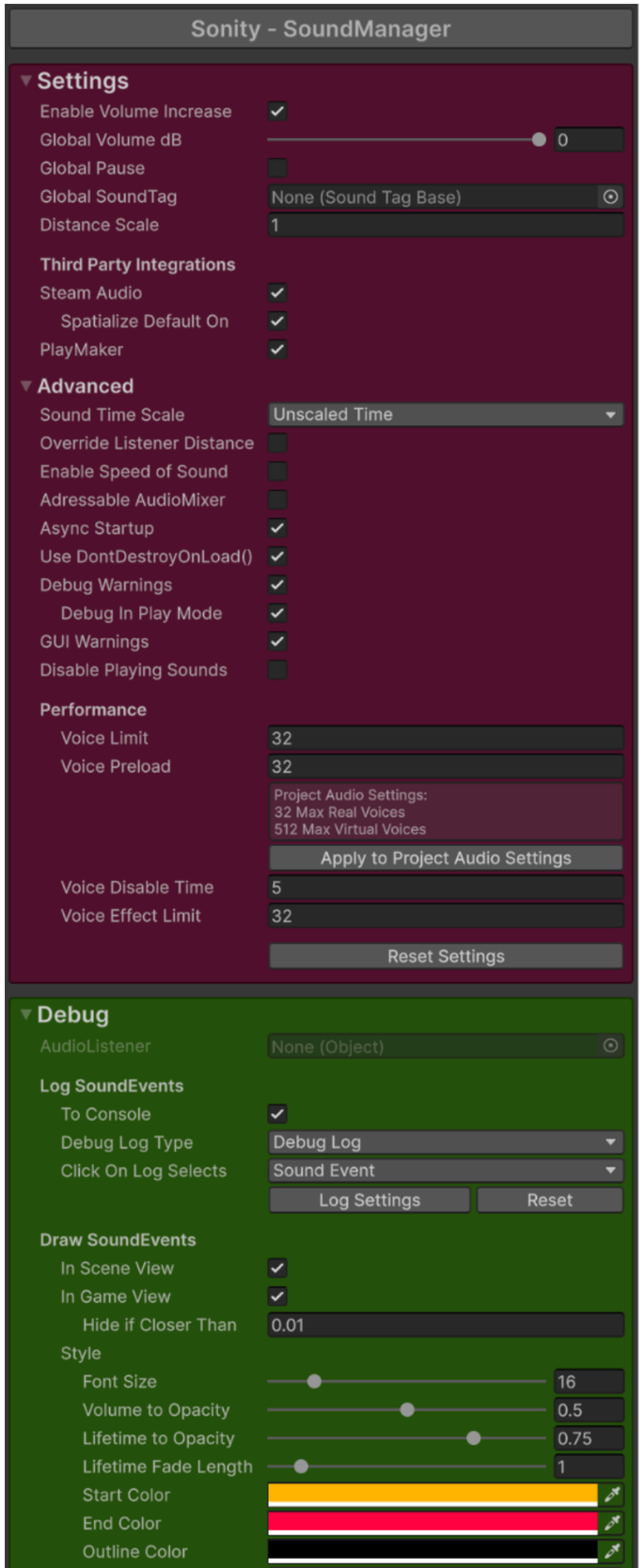
```
// Loads the audio data for the AudioClips of the assigned SoundEvents.
public void LoadAudioData(bool includeChildren)
```

```
// Unloads the audio data for the AudioClips of the assigned SoundEvents.
public void UnloadAudioData(bool includeChildren)
```

Parameters

includeChildren	If to load/unload all the audio data of all the child **SoundDataGroups** also.

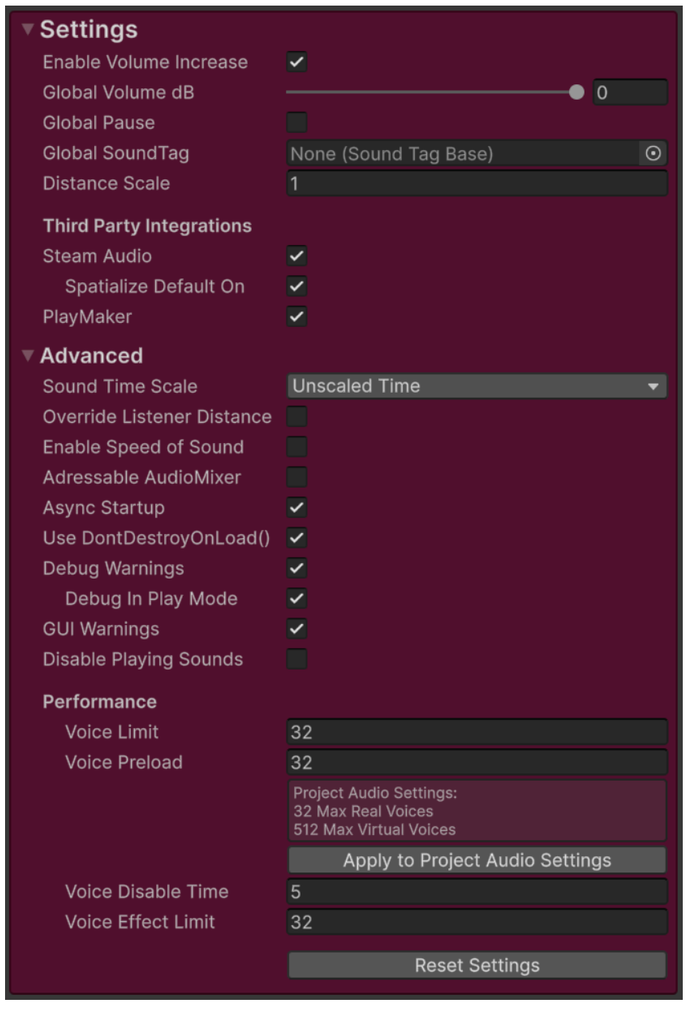
20 SoundManager Component


Sonity - SoundManager
Settings
Enable Volume Increase
Global Volume dB
0
Global Pause
Global SoundTag
None (Sound Tag Base)
Distance Scale
1
Third Party Integrations
Steam Audio
Spatialize Default On
PlayMaker
Advanced
Sound Time Scale
Unscaled Time
Override Listener Distance
Enable Speed of Sound
Adressable AudioMixer
Async Startup
Use DontDestroyOnLoad()
Debug Warnings
Debug In Play Mode
GUI Warnings
Disable Playing Sounds
Performance
Voice Limit
32
Voice Preload
32
Project Audio Settings:
32 Max Real Voices
512 Max Virtual Voices
Apply to Project Audio Settings
Voice Disable Time
5
Voice Effect Limit
32
Reset Settings
Debug
AudioListener
None (Object)
Log SoundEvents
To Console
Debug Log Type
Debug Log
Click On Log Selects
Sound Event
Log Settings
Reset
Draw SoundEvents
In Scene View
In Game View
Hide if Closer Than
0.01
Style
Font Size
16
Volume to Opacity
0.5
Lifetime to Opacity
0.75
Lifetime Fade Length
1
Start Color
End Color
Outline Color

The SoundManager is the master object which is used to play sounds and manage global settings.
An instance of this object is required in the scene in order to play SoundEvents.
You can add the pre-made prefab called “SoundManager” found in “Assets\Plugins\Sonity\Prefabs” to the scene.
Or you can add the “Sonity - Sound Manager” component to an empty GameObject in the scene, it works just as well.

> **💡 TIP - Play On Awake**
>
> If you need to play SoundEvents on Awake() when starting your game you need to edit the Script Execution Order.
> Go to "Project Settings..." -> "Script Execution Order" and add "Sonity.SoundManager".
> Then set it to a negative value (like -50) so it loads before the code which you want to use Awake() to play sounds when starting your game.

20.1 SoundManager Settings



Enable Volume Increase
If enabled you get the ability to raise the volume of SoundContainers and SoundEvents by +24dB each and SoundVolumeGroups by +12dB.
This is done by increasing the AudioListener.volume +60dB and reducing the volume of the AudioSource to compensate.
Note that all other non-Sonity AudioSources will also be affected, you can avoid this by enabling AudioSource.ignoreListenerVolume.
For volume increase to work, you need to set the Scripting Define Symbol "SONITY_ENABLE_VOLUME_INCREASE".
This can be done by pressing the button below (make sure all platforms has this scripting define symbol).
You also need to have a SoundManager in your scene for volume increase to work when previewing and ingame.
The free trial version of Sonity doesn't support this feature because it uses Scripting Define Symbols.

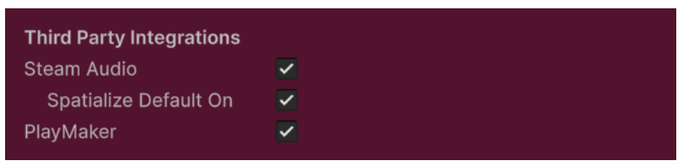
Global Volume
Sets the global volume using AudioListener.volume.
Note that because AudioListener.volume is used, all other non-Sonity AudioSources will also be affected.
This can be remedied with using AudioSource.ignoreListenerVolume.
Can be set with SetGlobalVolumeDecibel() and SetGlobalVolumeRatio() in the SoundManager."

Global Pause
Pauses or unpauses all sound using AudioListener.pause.
Except the SoundEvents which are set to "Ignore Global Pause".
Note that because AudioListener.pause is used, all other non-Sonity AudioSources will also be paused.
This can be remedied with using AudioSource.ignoreListenerPause.
Can be set with SetGlobalPause() and SetGlobalUnpause() in the SoundManager."

Global SoundTag
The selected Global SoundTag.
SoundEvents using global SoundTag are affected by this.

Distance Scale
Global range scale multiplier for all the sounds in Sonity.
Distance is calculated by Unity units of distance.
E.g. if Distance Scale is set to 100, a SoundEvent with the distance multiplier of 1 will be heard up to 100 Unity units away.

Third Party Integrations



20.2 SoundManager Settings - Steam Audio Integration

Enables the integration between Sonity and the third party asset Steam Audio.

Read more about the Steam Audio Sonity Integration here

Spatialize On Default
Makes HRTF Plugin Spatialize in the SoundContainer Settings be enabled by default when creating new assets.
Spatialize makes the sound be affected by Steam Audio with effects like occlusion, reflections and more.
Disable Spatialize for 2D sounds like music etc.

20.3 SoundManager Settings - PlayMaker Integration

Enables the integration between Sonity and the third party asset PlayMaker by adding custom Sonity actions.
Read more about the PlayMaker Sonity Integration here

20.4 SoundManager Settings - Advanced Settings

Sound Time Scale
Change the Time Scale used for sound related calculations ingame.
When the game is not playing in the editor, only Time.realtimeSinceStartup can be used (for preview, etc).
The Time Scale is used to calculate:

- Modifier Delay
- Modifier Fade In/Out Length
- SoundEvent Timeline Delay
- SoundEvent Cooldown Time
- SoundEvent Intensity Seek Time
- SoundContainer Prevent End Clicks Fade
- SoundParameter Delay
- SoundManager Speed of Sound Delay
- SoundManager Voice Disable Time
- SoundManager SoundEvent Debug Live Fade

Override Listener Distance
If enabled an AudioListenerDistance component is required in the scene.
The position of the AudioListenerDistance component will determine all distance based calculations (like volume falloff).
While the AudioListener position will be used for spatialization and Angle to Stereo Pan calculations.

Example of usage in a 3rd person or top down game:
Enable "Override Listener Distance" in the SoundManager.
Put the AudioListener on the main camera and the AudioListenerDistance on the player character.
Try changing the Amount slider to find a nice balance between the different positions.

Amount %
How much weight the AudioListenerDistance position has over the AudioListener position.
The position is linearly interpolated between the two of them.
100% is at the AudioListenerDistance component position.
50% is halfway between them.
0% is at the AudioListener position.

Enable Speed of Sound

Speed of sound is a delay based on the distance between the Audio Listener and a SoundEvent.

Speed of Sound Scale
Global speed of sound delay scale multiplier.
1 equals 430 Unity units per second.
Is calculated using the time scale selected in the SoundManager.

Addressable AudioMixer
If enabled, all AudioMixer references in all SoundEvents and SoundContainersto the selected addressable AudioMixer Reference Asset.
This is to fix problems when the AudioMixer is an addressable asset because it might create both a normal and an addressable instance with different IDs.
Note that when using an addressable AudioMixer you should also enable Async Startup to increase startup speed.
The AudioMixerGroups are matched using FindMatchingGroups using the name of the AudioMixerGroup, so you have to name your groups to something unique.
For builds the AudioMixerGroups are cached and wont be updated once the sound has been played.
When using AudioMixer.SetFloat() etc you need to access this specific AudioMixer instance by using "SoundManager.Instance.GetAddressableAudioMixer();"
For addressable AudioMixer to work, you need to add the package "com.unity.addressables".
You also need to define the Script Define Symbol "SONITY_ENABLE_ADDRESSABLE_AUDIOMIXER".
Asmdef_Sonity.Internal.Runtime references Unity.Addressables and Unity.ResourceManager.
Asmdef_Sonity.Internal.Editor references Unity.Addressables.
The free trial version of Sonity doesn't support this feature because it uses Scripting Define Symbols.

AudioMixer Reference
Assign the AudioMixer reference here, it will be automatically loaded.
Only one AudioMixer per project is supported with this feature.

Async Load In Editor
Enables async loading of the addressable AudioMixer in the editor.
Having it disabled is useful for testing in the editor when you need to access the AudioMixer or play sounds on Awake before async loading is finished.

Async Startup
Makes Sonity initialize using an async loading method.
This is useful for when you use the addressable AudioMixer to speed up load times.
Or when using Steam Audio and you want to preload voices.

Use DontDestroyOnLoad()
Calls DontDestroyOnLoad() at Start for Sonity objects.
Which makes them persistent when switching scenes.
For this to work the parent is set to null, which can move the objects.

Debug Warnings
Makes Sonity output Debug Warnings if anything is wrong.

Debug in Play Mode
Makes Sonity output Debug Warnings if anything is wrong in Play Mode.

GUI Warnings
Makes Sonity show GUI Warnings if anything is wrong in the editor.

Disable Playing Sounds
Disables all the Play functionality.
Useful if you've for example implemented temp sounds and don't want everyone else to hear them.

Performance Settings

Voice Limit
Maximum number of Voices.
If the limit is reached it will steal the Voice with the lowest priority.
If you need extra performance, you could try lowering the real and virtual voices to a lower number.
Voice Limit cannot be lower than Voice Preload.

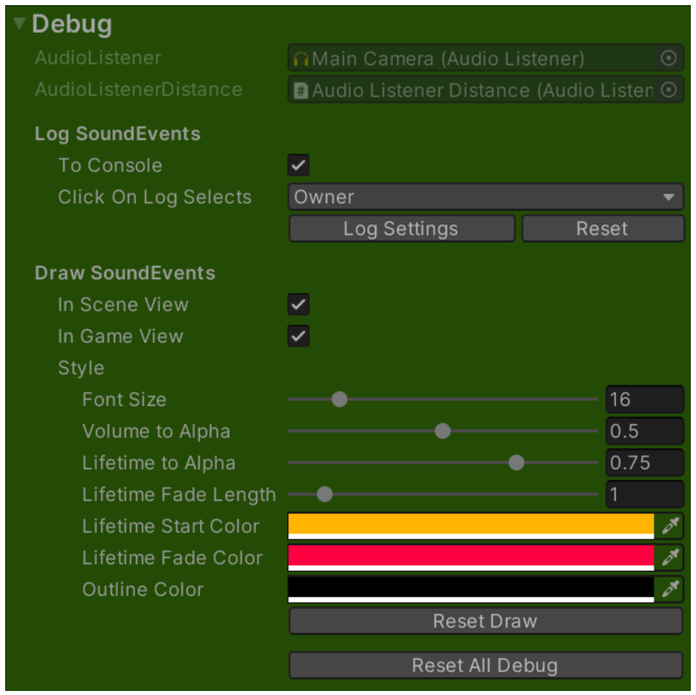
> **Max Real Voices:**
> The maximum number of real (heard) AudioSources that can be played at the same time.
> "Real Voices" should be the same as the "Voice Limit", or more if you play other sounds outside of Sonity.
>
> **Max Virtual Voices:**
> The maximum number of virtual (not heard) AudioSources that can be played at the same time.
> This should always be more than the number of real voices.
> You can change these values manually in:
> "Edit" > "Project Settings" > "Audio"
>
> **Apply to Project Audio Settings**
> Applies the Voice Limit to the Project Audio Settings.
> Sets "Real Voices" to the "Voice Limit".
> You can change these values manually in:
> "Edit" > "Project Settings" > "Audio"

Voice Preload
How many Voices to preload on Awake().
Voice Limit cannot be lower than Voice Preload.

Voice Disable Time
How long in seconds to wait before disabling a Voice when they've stopped playing.
Retriggering a voice which is not disabled is more performant than retriggering a voice which is disabled.
But having a lot of voices enabled which aren't used is also not good for performance, so don't set this value too high.
Is calculated using the time scale selected in the SoundManager.

Voice Effect Limit
Maximum number of Voice Effects which can be used at the same time.
A Voice with any combination of waveshaper/lowpass/highpass counts as one Voice Effect.
If the values of a Voice Effect doesn't have any effect it is disabled automatically (e.g. distortion amount is 0).
If the Voice Effect limit is reached, the Voice Effects are prioritized by the Voices with the highest volume * priority.
Watch out for high load on the audio thread if set too high.
Try setting the buffer size to "Best Performance" in "Edit" > "Project Settings" > "Audio" if you want to run more Voice Effects.

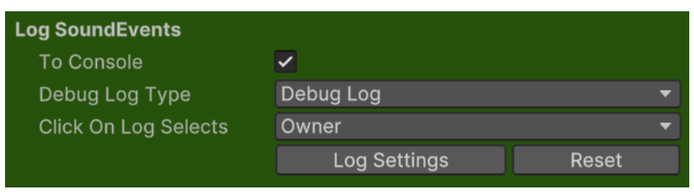
20.5 SoundManager Debug



AudioListener
Here you can see and select the currently cached AudioListener in runtime.
If you click on the field your selection will show you the currently used AudioListener.
The AudioListener component is found and cached automatically and cannot be assigned here.

AudioListenerDistance
Here you can see and select the currently cached AudioListenerDistance in runtime.
If you click on the field your selection will show you the currently used AudioListenerDistance.
The AudioListenerDistance component is found and cached automatically and cannot be assigned here.
Is only shown if “Override Listener Distance” is enabled.

20.6 SoundManager Log SoundEvents

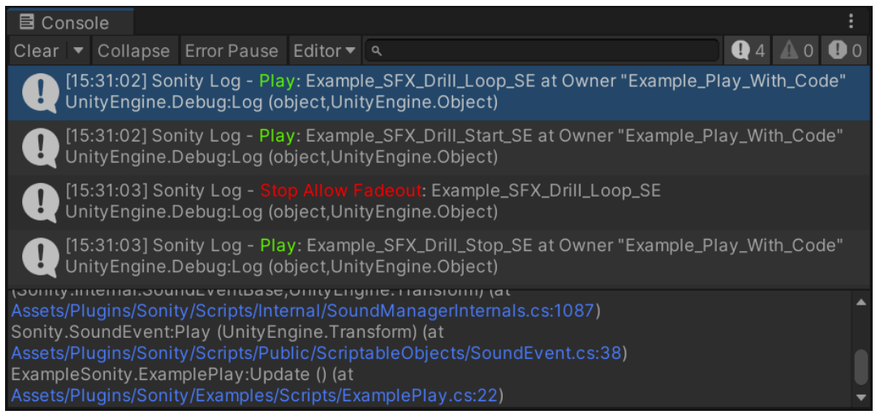


When enabled SoundEvent actions will be logged to the console.
This is useful to debug what happens with SoundEvents during runtime.

It also enables you to see where in the code the SoundEvent is played/stopped from if you enable Stack Tracke Logging “Script Only”.

To Console
If enabled it will log SoundEvent actions to the console.

Example:



Tip: As you can see in the bottom you can also see where the SoundEvent is played in code.

Debug Log Type
Select which kind of Debug.Log you want to use.
Switch between either Debug.Log, Debug.LogWarning or Debug.LogError.

Click On Log Selects
If you click the log in the console, you will be redirected to one of either object:

Owner
Selects the owner transform which is used to play the SoundEvent.

Position
Selects the transform position which is used to play the SoundEvent if its played with PlayAtPosition with a Transform.

SoundEvent
Selects the SoundEvent which is logged.

If an Owner or Position can't be found (for example when logging stop etc) it will redirect to the SoundEvent instead.

Log Settings
Opens a dropdown menu where you can select which events should be logged.

SoundEvent Play
Logs when a SoundEvent is played.

SoundEvent Stop

Logs when a SoundEvent is stopped with a stop command.

SoundEvent Pool
Logs when a SoundEvent is pooled after its stopped.

SoundEvent Pause
Logs when a SoundEvent is paused.

SoundEvent Unpause
Logs when a SoundEvent is unpaused.

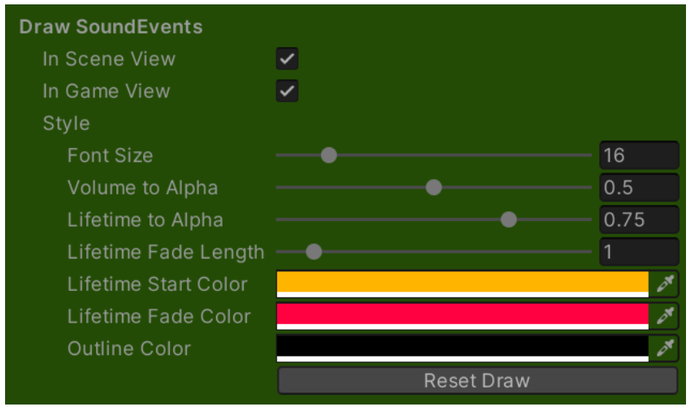
SoundEvent Global Pause
Logs when a SoundEvent is paused using global pause.

SoundEvent Global Unpause
Logs when a SoundEvent is unpaused using global unpause.

SoundParameters Once
Logs any SoundParameter set to UpdateMode Once passed when playing a SoundEvent.

SoundParameters Continuous
Logs any SoundParameter set to UpdateMode Continuous on an active SoundEvent.

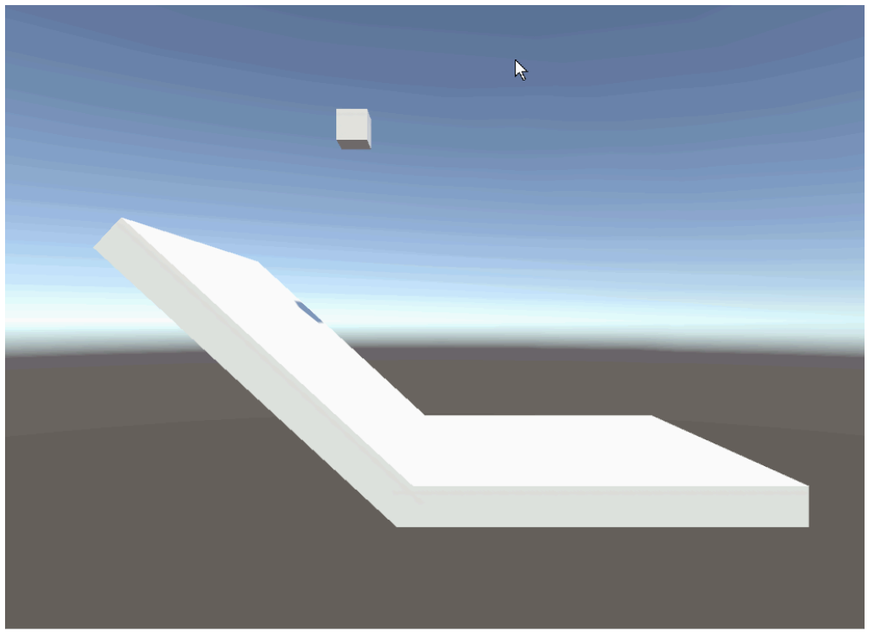
20.7 SoundManager Draw SoundEvents



Draws the names of all currently playing SoundEvent in the scene and/or game view.
Useful for debugging when you want to see what is playing and where.

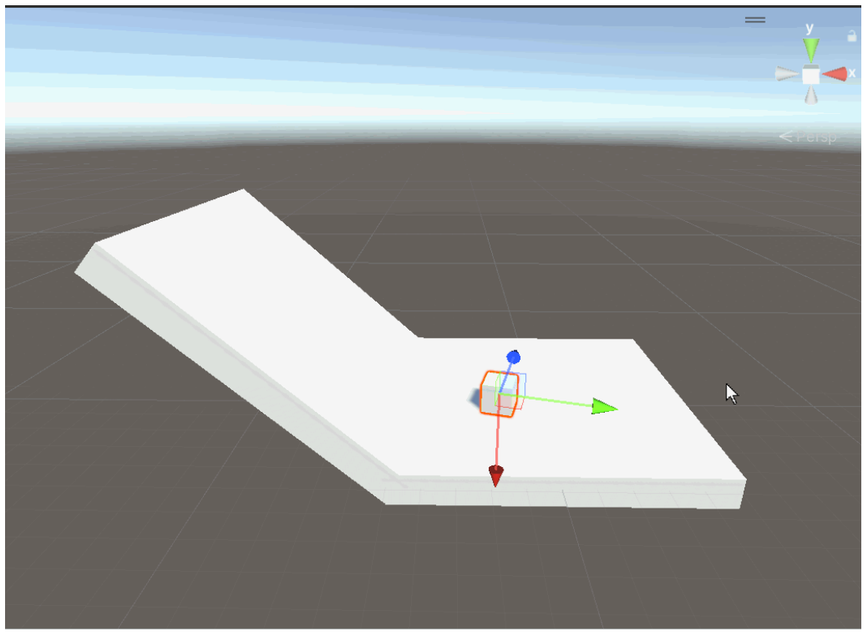
In Scene View
Draws debug names in the Unity scene view.
Doesn't work in Unity versions older than 2019.1.

Example:



In Game View
Draws debug names in the Unity game view.
Only applied in the Unity editor.

Example:



Hide if Closer Than
Hides the debug if its closer to the camera than the allowed value in Game view.
Useful for hiding SoundEvents which are played on the camera which you don't want to see.

Style

Font Size
The font size of the text.

Volume to Opacity
How much of the volume of the SoundEvent will be applied to the transparency of the text.
E.g lower volumes will be more transparent.

Lifetime to Opacity
How much the lifetime of the SoundEvent will affect the transparency of the text.

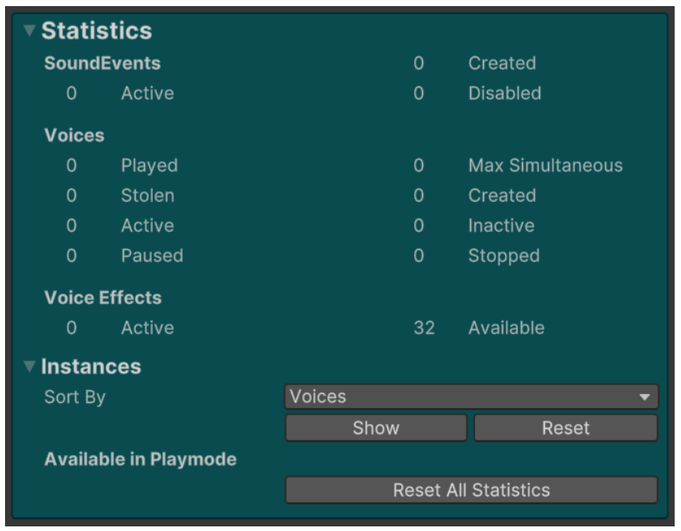
Lifetime Fade Length
How long the fade should be.
Is calculated using the time scale selected in the SoundManager.

Start Color
The color the text should have when it starts playing.

End Color
Which color the text should fade to over the lifetime.

Outline Color
The color of the text outline.

20.8 SoundManager Statistics



SoundEvents
Statistics of SoundEvents.

Created
The number of instantiated SoundEvents.

Active
The number of active SoundEvents.

Disabled
The number of unused and disabled SoundEvents.

Voices
Statistics of Voices.

Played
The number of played Voices since start.

Stolen
The number of stolen Voices since start.

Max Simultaneous
The maximum number of simultaneously playing Voices since start.

Created
The number of Voices in the pool.

Active
The number of Voices playing audio.

Inactive
The number of inactive Voices in the pool.

Paused
The number of paused Voices in the pool.

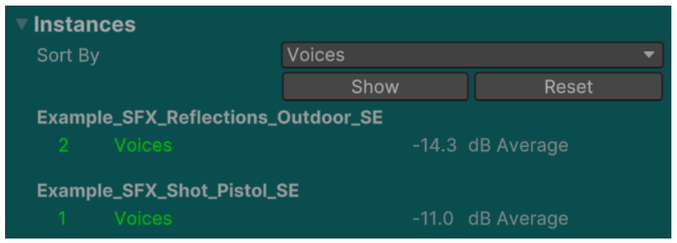
Stopped
The number of stopped Voices in the pool.

Voice Effects
Statistics of Voice Effects.
A Voice with any combination of waveshaper/lowpass/highpass counts as one Voice Effect

Active
The number of active Voice Effects.

Available
How many Voice Effects are available.

Statistics Instances



Real-time statistics per SoundEvent Instance.
Available in Playmode.

Sort By
Which method to sort the list of SoundEvent Instances.

Name
Sorts by alphabetical order

Voices
Sorts by voice count

Plays
Sorts by number of plays

Volume
Sorts by volume

Time
Sorts by last time played

Show
Toggle what information to show about the SoundEvent Instances.

Show Active
How many are currently active.

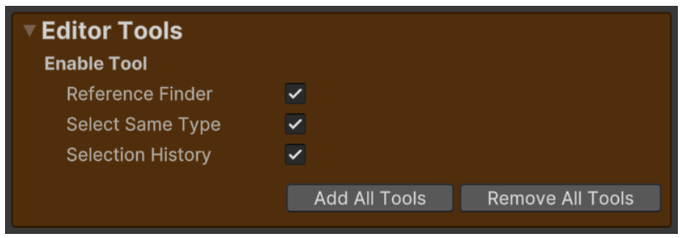
Show Disabled
How many are currently disabled.

Show Voices
How many voices are currently used.

Show Plays
The number of total plays.

Show Volume
The current average volume.

20.9 SoundManager Editor Tools



This is a collection of nifty editor tools which you can use to speed up your workflow.
They are enabled with the use of “Script Define Symbols” in Project Settings -> Player -> Other Settings -> Script Compilation -> Script Define Symbols.
This way they can be bound to default hotkeys and they won’t hog up your toolbars if you don’t use them.

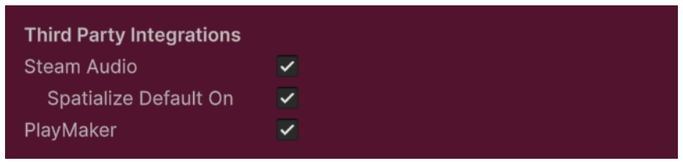
You can read more about them here:
Editor Tool - Reference Finder
Editor Tool - Select Same Type
Editor Tool - Selection History

20.10 AudioListenerDistance Component

If “Override Listener Distance” is enabled in the SoundManager Settings this is a required component in the scene.
The position of the AudioListenerDistance component will determine all distance based calculations (like volume falloff).
While the AudioListener position will be used for spatialization and Angle to Stereo Pan calculations.

Example of usage in a 3rd person or top down game:
Enable "Override Listener Distance" in the SoundManager.
Put the AudioListener on the main camera and the AudioListenerDistance on the player character.
Try changing the Amount slider to find a nice balance between the different positions.

21 Steam Audio - Sonity Integration



The Steam Audio Sonity integration can be enabled in the SoundManager Settings.
You also need to install the Steam Audio Unity asset package.
And add the Script Define Symbol "SONITY_ENABLE_INTEGRATION_STEAM_AUDIO".
The free trial version of Sonity doesn’t support this feature because it uses Scripting Define Symbols.
It is tested and working on Steam Audio version 4.6.1.

Steam Audio adds features like:

- Occlusion of sound sources.
- Reflections and related environmental audio effects.
- Propagation of sound along multiple paths.

When using Steam Audio in combination with Voice Preload you should enable Async Startup because Steam Audio isn't ready on Awake().
Also note that Steam Audio at the moment is not available for all platforms.

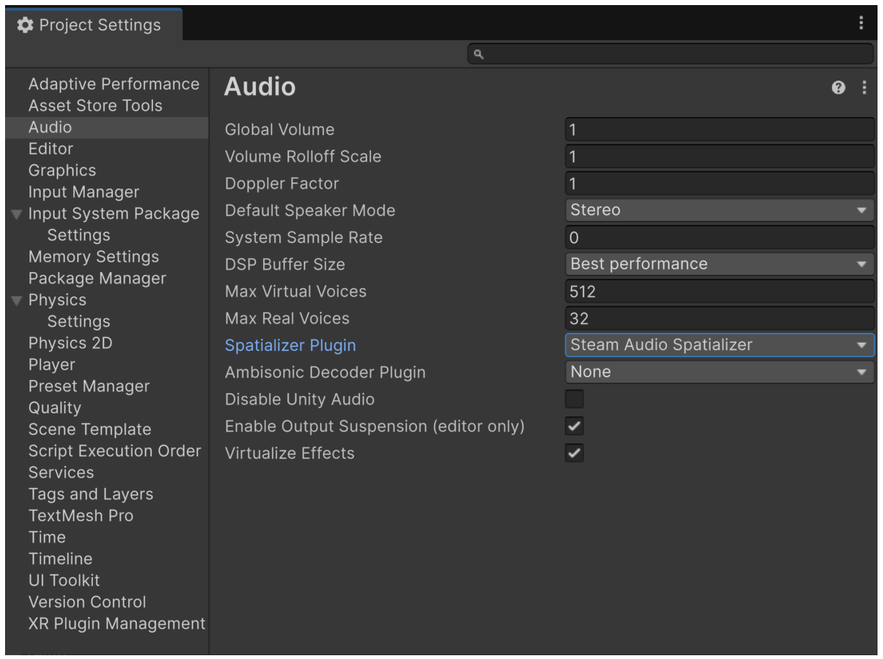
Steam Audio - Getting Started

1. Go to the valve github https://github.com/ValveSoftware/steam-audio/releases
2. Download **steamaudio_unity_X.X.X.zip** and locate the file **SteamAudio.unitypackage**.
3. Drag and drop the Steam Audio unitypackage into Unity and import the contents.
4. Enable “Steam Audio” in the SoundManager Settings.
5. Add the Script Define Symbol "SONITY_ENABLE_INTEGRATION_STEAM_AUDIO" by pressing the “Add Scripting Define Symbol” button.
6. You may need to **restart the Unity project**.

This adds the SoundContainer Settings - Steam Audio tab where you can select how the sound will be played in Steam Audio.

SoundContainer - HRTF Plugin Spatialize
Spatialize makes the sound be affected by Steam Audio with effects like occlusion, reflections and more.
Disable Spatialize for 2D sounds like music etc.

Audio Spatializer



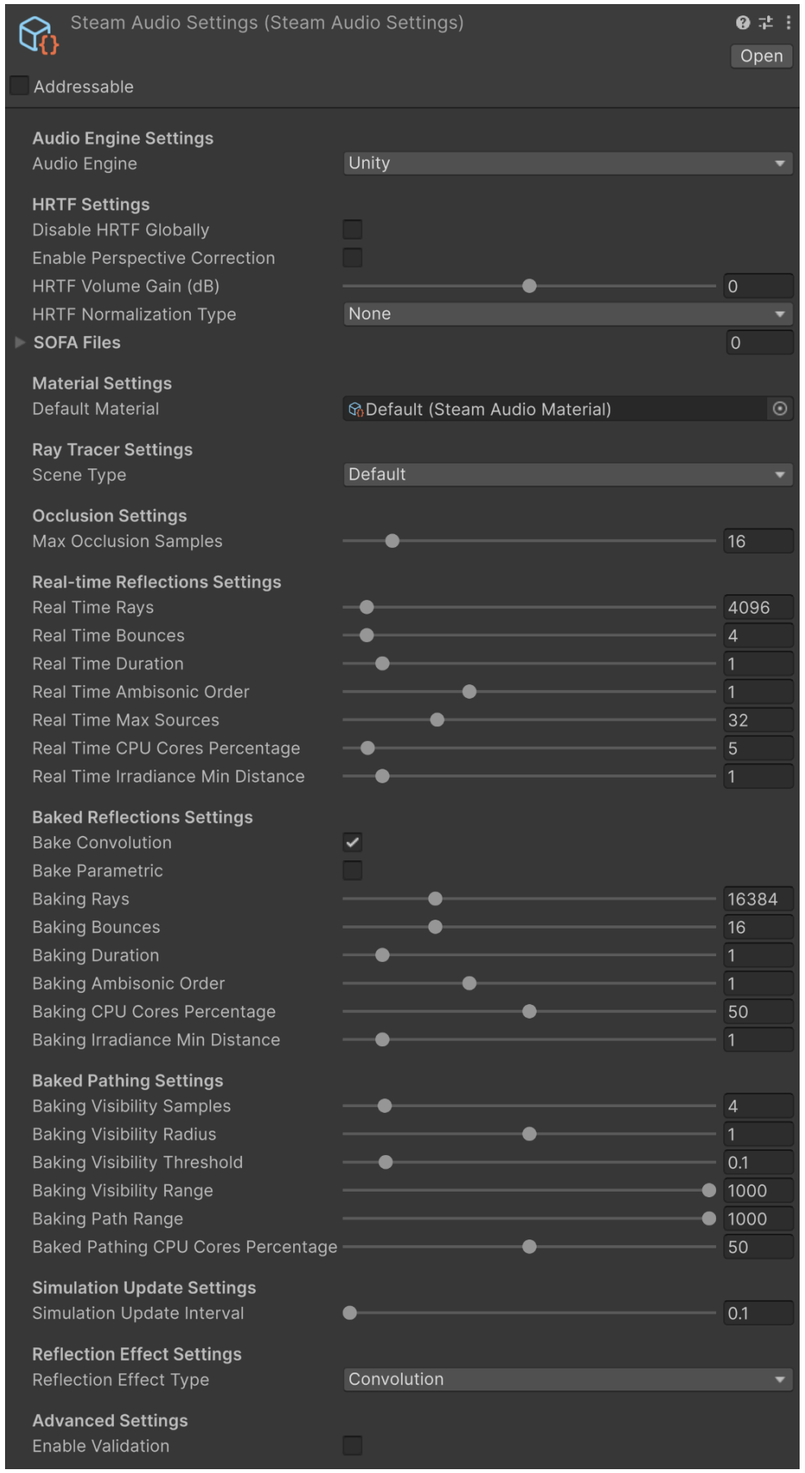
You can also change the Spatializer to Steam Audio Spatializer in the project settings.

Project Settings > Audio > Spatializer Plugin

This changes the spatialization for all the sounds in the project.

You also need to enable **HRTF Plugin Spatialize** in the SoundContaier Settings.

SteamAudioSettings



You can also change settings for Steam Audio in the SteamAudioSettings.asset.

Which is automatically created and located at **Plugins\SteamAudio\Resources**.

Steam Audio Official Valve documentation links:

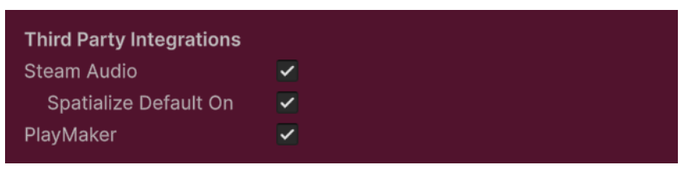
Steam Audio - Basics

- Getting Started
- Reference

Steam Audio - Features

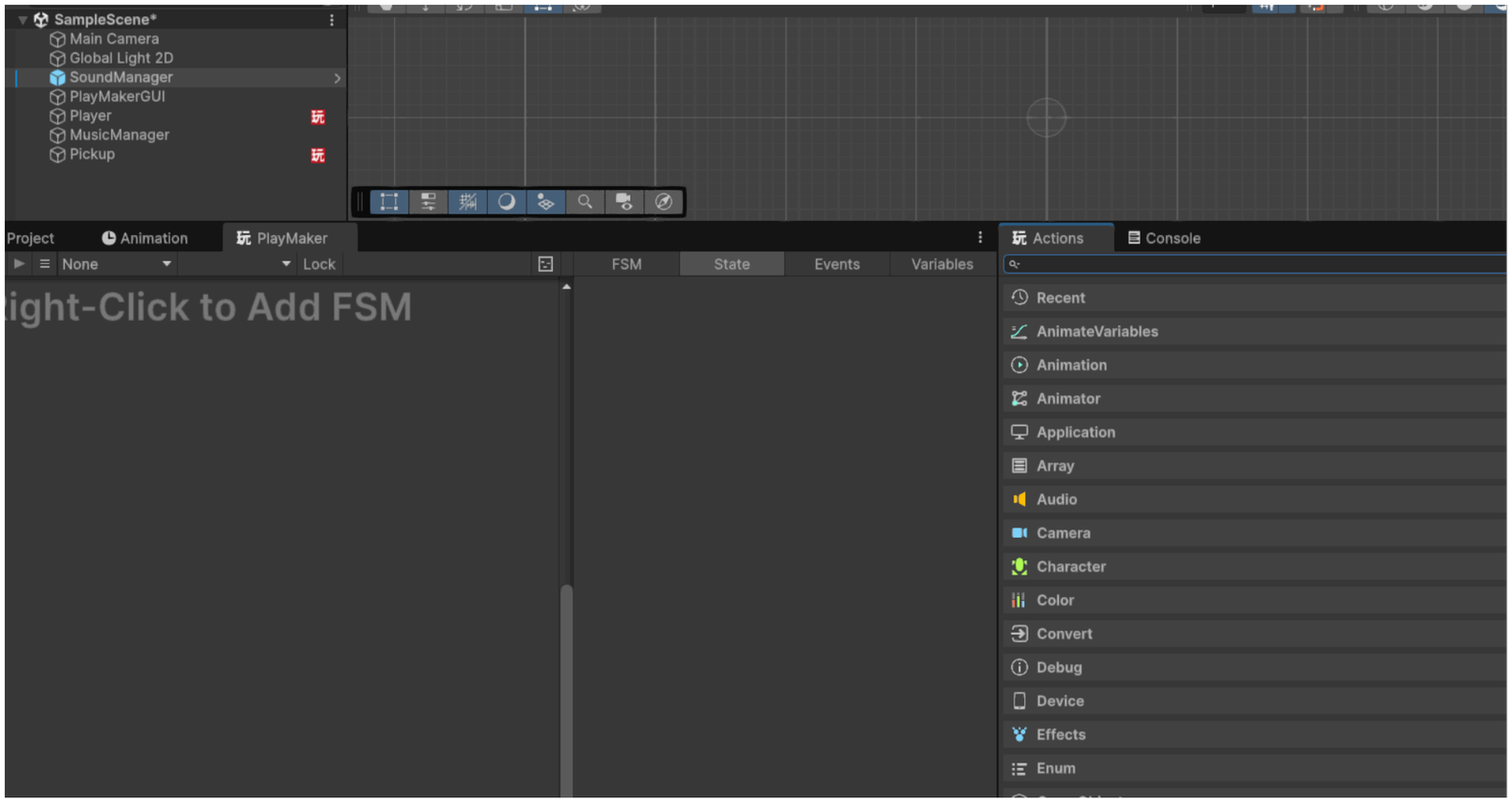
- Tag Acoustic Geometry
- Set Up Dynamic Movable Geometry
- Model Occlusion by Geometry
- Model Transmission Through Geometry
- Model Reflection by Geometry
- Simulate Physics Based Reverb at the Listener Position
- Create Sound Probes for Baked Sound Propagation
- Bake Physics Based Reverb
- Model Sound Paths From a Moving Source to a Moving Listener
- Enable GPU Acceleration

22 PlayMaker - Sonity Integration



The PlayMaker Sonity integration can be enabled in the SoundManager Settings.
You also need to install the PlayMaker Unity asset package.
And add the Script Define Symbol "SONITY_ENABLE_INTEGRATION_PLAYMAKER".
The free trial version of Sonity doesn’t support this feature because it uses Scripting Define Symbols.
It is tested and working on PlayMaker version 1.9.8.
PlayMaker adds easy access to visual scripting.

Example usage:



Below is a list of all available PlayMaker actions for Sonity's third party integration.
Most of this information is also available as tooltips by hovering the Action inside Unity.

22.1 PlayMaker Actions - Sonity Play

Actions

`Sonity Play Sound`	Play the **SoundEvent** at the position of this GameObject
`Sonity Play Sound At Transform`	Plays the **SoundEvent** at the position of input Vector3
`Sonity Play Sound At Vector3`	Plays the **SoundEvent** at the position of the owner Transform

Parameters

`SoundEvent`	The **SoundEvent** to play
`SoundTag`	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
`Vector3`	The Vector3 position to play from.

`Transform`	The Game Object Transform to play the **SoundEvent** from.
`Use Parameters`	Enables the use of **SoundParameters**. For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

Actions

`Sonity Play Sound`	Play the **SoundEvent** at the position of this GameObject
`Sonity Play Sound At Transform`	Plays the **SoundEvent** at the position of input Vector3
`Sonity Play Sound At Vector3`	Plays the **SoundEvent** at the position of the owner Transform

Parameters

`SoundEvent`	The **SoundEvent** to play
`SoundTag`	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
`Vector3`	The Vector3 position to play from.
`Transform`	The Game Object Transform to play the **SoundEvent** from.
`Use Parameters`	Enables the use of **SoundParameters**. For example SoundParameterVolumeDecibel is used to modify how the **SoundEvent** is played

22.2 PlayMaker Actions - Sonity Stop

Actions

`Sonity Stop Sound`	Play the **SoundEvent** at the position of this GameObject
`Sonity Stop At Position`	Stop the **SoundEvent** playing at the GameObject Transform.
`Sonity Stop All At Owner`	Stop all **SoundEvents** playing on this GameObject.
`Sonity Stop Everywhere`	Stop all instances of the **SoundEvent**.
`Sonity Stop Everything`	Stop the **SoundEvent** with fadeout. Disabling this stops the sound immediately.

Parameters

`SoundEvent`	The **SoundEvent** to stop.
`AllowFadeout`	Stop the **SoundEvent** with fadeout. Disabling this stops the sound immediately.
`TargetGameObject`	The GameObject Transform on which to stop the **SoundEvent**.

22.3 PlayMaker Actions - Sonity Pause and Unpause

Actions

`Pause`	Pauses a **SoundEvent** playing on a GameObject.
`Pause All At Owner`	Pauses all **SoundEvents** playing on a GameObject.
`Pause Everything`	Pauses all **SoundEvents** currently playing.
`Pause Everywhere`	Pause all instances of the chosen **SoundEvent**.
`Unpause`	Unpauses a **SoundEvent** playing on a GameObject.
`Unpause All At Owner`	Unpauses all **SoundEvents** playing on a GameObject.
`Unpause Everything`	Unpauses all **SoundEvents** currently playing.
`Unpause Everywhere`	Unpauses all instances of the chosen **SoundEvent**.

Parameters

`SoundEvent`	The **SoundEvent** to pause/unpause.
`TargetGameObject`	The GameObject Transform on which to pause/unpause the **SoundEvent**.
`ForcePause`	Pauses **SoundEvent** even if it is set to "Ignore Local Pause".
`ForceUnpause`	Unpauses **SoundEvent** even if it is set to "Ignore Local Pause".

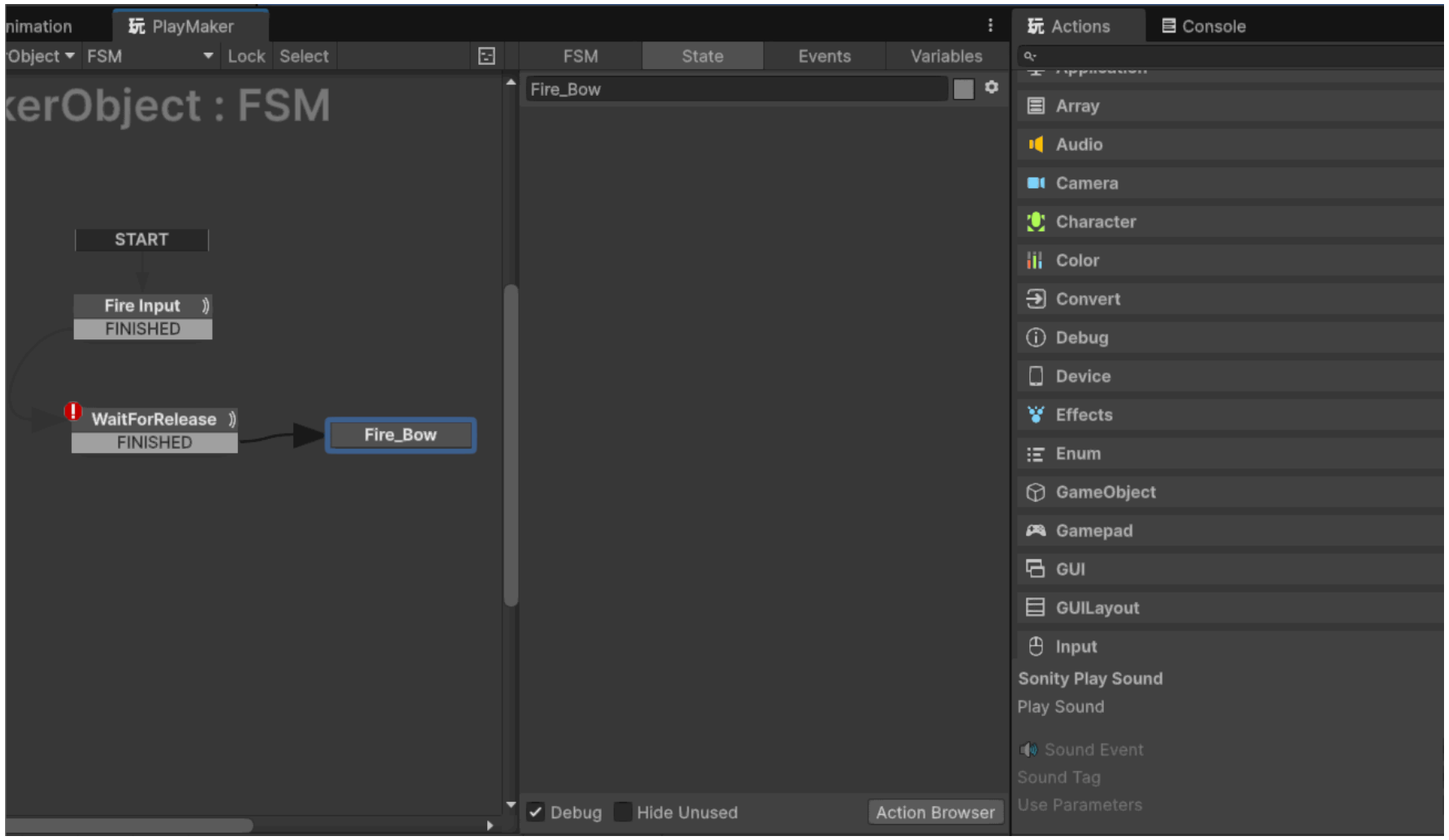
22.4 PlayMaker Actions - Sonity SoundParameters

SoundParameters add an extra layer of control that allows for greater modification of SoundEvents (e.g. volume, polyphony, fade in length etc).
They can be added when first playing a SoundEvent and set to update either once when playback starts or to continuously listen for updates.
They have similar functionality to the SoundContainer Modifiers, but with scripting capabilities and allows for continuous updates to their values.
Read more about SoundParamters.

Enabling SoundParameters

By default, SoundParameters are not enabled on a SoundEvent.
Enabling **Use Parameters** in a Play Action will reveal options to add SoundParameters to that event.
Once enabled, there are several ways of configuring the parameters of a SoundEvent.

> ❗ *Note: You can only use one instance of the same type of each parameter, eg: adding 2 SoundParameterVolumes will only use the last one.*



> **💡 TIP - Updating Parameter Values Continuously**
>
> To create a SoundEvent action with an updatable parameter, set UpdateMode to Continuous and store the ParameterValue in an FSM Variable. When you want to update the value just change the FSM Variables and it will automatically update the SoundEvent.

Options

`Use Parameters`	Enable to show **SoundParameter** options.
`Number of Parameters`	How many SoundParameters are active on this **SoundEvent**. Max amount is 8.
`Show parameters`	Fold-out menu to show or hide **SoundParameters** in the GUI.

Parameters

`Parameter type`	Select which **SoundParameter** to apply to the **SoundEvent**.
`Update mode`	Determines if the **SoundEvent** will take the parameter into consideration only Once at start or if it will be updated Continuously.
`Type` [Read only]	What type of value (float, bool, int) the **SoundParameter** takes. The type cannot be set, only read.

`Parameter value`	Value of the **SoundParameter** when playing the **SoundEvent**. Can be a value or an **FsmVariable**.

22.5 PlayMaker Actions - Sonity SoundEvent Info

Actions

`Get SoundEvent State`	Returns the current **SoundEventState** of the **SoundEvent**.
`Get Last Played Clip Length`	Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource.
`Get Last Played Clip Time Seconds`	Returns the current time (in seconds) of the AudioClip in the last played **AudioSource.** Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Get Last Played Clip Time Ratio`	Returns the progress of playback (in range 0 to 1) of the **AudioClip** in the last played AudioSource**.** Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Get Time Played`	Returns the time (in seconds) since the **SoundEvent** was played. Time is calculated using the time scale selected in the **SoundManager**. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Get Contains Loop`	Returns true if any **SoundContainers** in the **SoundEvent** are set to loop.
`Get Max Length`	Returns the max length (in seconds) of the **SoundEvent** (calculated from the longest AudioClip). The result is automatically scaled by the pitch of the sound.

Parameters

`SoundEvent`	The **SoundEvent** to pause.
`Target GameObject`	The GameObject **Transform** on which to stop the **SoundEvent**.
`Pitch Speed`	Returned value is adjusted by the **SoundEvent** pitch value.
`Store State In`	**FsmVariable** to store the returned value in.

22.6 PlayMaker Actions - Sonity UI

UI SoundEvent Events do not need a GameObject reference.

> 💡 **TIP - UI Play & Music Play**
>
> There are many different actions for starting playback of a SoundEvent.
> The standard way is using the Play action which allows for positional audio inside Unity.
> For sounds that do not need to be positioned, such as UI, there are the UIPlay methods. For music you can use MusicPlay.
> Any action that has “UI” or “Music” included in its name will only work on SoundEvents played by the matching action.

> For example, using the MusicGetSoundEventState Action on a SoundEvent that was started using the normal action Play will not work.

UI Play

Actions

`UI Play`	The **SoundEvent** to play without needing to input an owner GameObject.

Parameters

`SoundEvent`	The **SoundEvent** to play.
`SoundTag`	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent.**
`Use Parameters`	The GameObject Transform to play the **SoundEvent** from.

UI Stop

Actions

`UI Stop`	Stop UI **SoundEvent**.
`UI Stop All`	Stop all playing UI **SoundEvents**.

Parameters

`SoundEvent`	The **SoundEvent** to stop.
`Allow Fadeout`	Stop the **SoundEvent** with fadeout. Disabling this stops the sound immediately.

UI Pause and Unpause

Actions

`UI Pause`	Pause the UI **SoundEvent**.
`UI Pause All`	Pause all playing UI **SoundEvents**.
`UI Unpause`	Unpause the UI **SoundEvent**.
`UI Unpause All`	Unpause all playing UI **SoundEvents**.

Parameters

`SoundEvent`	The UI **SoundEvent** to pause.
`Force Pause`	Pauses the **SoundEvent** even if it is set to "Ignore Local Pause".
`Force Unpause`	Unpauses the **SoundEvent** even if it is set to "Ignore Local Pause".

UI Get Info

Actions

`UI Get Sound Event State`	Returns the current **SoundEventState** of the UI **SoundEvent**.
`UI Get Last Played Clip Length`	Returns length of the last played AudioClip as float seconds. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`UI Get Last Played Clip Time Seconds`	Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`UI Get Last Played Clip Time Ratio`	Returns the current time (in seconds) of the AudioClip in the last played AudioSource. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`UI Get Time Played`	Returns the time (in seconds) since the UI **SoundEvent** was played. Time is calculated using the time scale selected in the **SoundManager**. Returns 0 if the UI **SoundEvent** is not playing or null, and if the Transform is null.
`UI Get Transform`	Returns the owner Transform used when playing a UI **SoundEvent**.

Parameters

`Sound Event`	The **SoundEvent** to get information from.
`Store State In`	Fsm Variable to store the returned value in.
`Pitch Speed`	Returned value is adjusted by the **SoundEvents** pitch value. Eg. -12 semitones will be twice as long.

22.7 PlayMaker Actions - Sonity Music

Music Event Actions are SoundEvents with specific logic related to making music playback easier.
Music SoundEvent Events do not need a GameObject reference.

Music Play

Actions

`Music Play`	Play a Music **SoundEvent**. Currently playing music can optionally be set to stop when this event starts playing.

Parameters

`Sound Event`	The **SoundEvent** to play.
`Stop All Other Music`	Stop all other music events playing when this **SoundEvent** starts playing.
`Allow Fadeout`	Enables or disables fadeout on stopped **SoundEvents**.

Music Stop

Actions

`Music Stop`	Stop a Music **SoundEvent** currently playing.
`Music Stop All`	Stop all Music **SoundEvents** currently playing.

Parameters

`Allow Fadeout`	Enables or disables fadeout on stopped **SoundEvent**.

Music Pause and Unpause

Actions

`Music Pause`	Pause a Music **SoundEvent** currently playing.
`Music Pause All`	Pause all Music **SoundEvents** currently playing.
`Music Unpause`	Unpause a Music **SoundEvent** currently playing.
`Music Unpause All`	Unpause all Music **SoundEvents** currently playing.

Parameters

`Sound Event`	The **SoundEvent** to pause.
`Force Pause`	Pauses the **SoundEvent** even if it is set to "Ignore Local Pause".
`Force Unpause`	Unpauses the **SoundEvent** even if it is set to "Ignore Local Pause".

Music Get Info

Actions

`Music Get Sound Event State`	Returns the current **SoundEventState** of the Music **SoundEvent**.
`Music Get Last Played Clip Length`	Returns length of the last played **AudioClip** as float seconds. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Music Get Last Played Clip Time Seconds`	Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Music Get Last Played Clip Time Ratio`	Returns the current time (in seconds) of the **AudioClip** in the last played AudioSource. Returns 0 if the **SoundEvent** is not playing or null, and if the Transform is null.
`Music Get Time Played`	Returns the time (in seconds) since the Music **SoundEvent** was played. Time is calculated using the time scale selected in the **SoundManager**. Returns 0 if the Music **SoundEvent** is not playing or null, and if the Transform is null.
`Music Get Transform`	Returns the owner Transform used when playing a Music **SoundEvent**.

Parameters

`Sound Event`	The **SoundEvent** to get information from.
`Store State In`	Fsm Variable to store the returned value in.
`Pitch Speed`	Returned value is adjusted by the **SoundEvents** pitch value. Eg. -12 semitones will be twice as long.

22.8 PlayMaker Actions - Sonity Audio Data Load Unload

The Load and Unload Actions lets the user manually load or unload the audio data of a SoundEvent.
By default audio data is loaded when the SoundEvent first plays, unless you have enabled Preload Audio Data in the AudioClip.
Manually loading or unloading can be used if you need to optimize the RAM usage of the audio assets.

Actions

`Load Audio Data`	Loads the audio data of any AudioClips assigned to the **SoundContainers** of the **SoundEvent**.
`Unload Audio Data`	Unloads the audio data of any AudioClips assigned to the **SoundContainers** of the **SoundEvent**.

Parameters

`Sound Event`	The **SoundEvent** which to load or unload audio data from.

23 SoundManager Functions

The SoundManager is the master object which is used to play sounds and manage global settings.
An instance in the scene is required to play SoundEvents.
You can either add the pre-made prefab called “SoundManager” or add the “Sonity - Sound Manager” component to an empty GameObject in the scene.

Example code:

```
using UnityEngine;
using Sonity;

public class PlayStopExample : MonoBehaviour {

    public SoundEvent soundEvent;

    void PlayExample() {
        // Plays the SoundEvent at the position of the transform
        SoundManager.Instance.Play(soundEvent, transform);
    }

    void StopExample() {
        // Stops the SoundEvent playing at the transform
        SoundManager.Instance.Stop(soundEvent, transform);
    }
}
```

23.1 SoundManager.Instance Play

```
// Plays the SoundEvent at the position of the owner Transform
public void Play(SoundEvent soundEvent, Transform owner)
public void Play(SoundEvent soundEvent, Transform owner, SoundTag localSoundTag)
public void Play(SoundEvent soundEvent, Transform owner, params SoundParameterInternals[] soundParameterInternals)
public void Play(SoundEvent soundEvent, Transform owner, SoundTag localSoundTag, params SoundParameterInternals[]
soundParameterInternals)
```

Parameters

soundEvent	The **SoundEvent** to play
owner	The owner **Transform**
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played

23.2 SoundManager.Instance PlayAtPosition

```
// Plays the SoundEvent at the Vector3 position with another Transform as the owner
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Vector3 position)
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Vector3 position, SoundTag localSoundTag)
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Vector3 position, params SoundParameterInternals[] soundParameterInternals)
```

```
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Vector3 position, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)

// Plays the SoundEvent at the Transform position with another Transform as the owner
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Transform position)
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Transform position, SoundTag localSoundTag)
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Transform position, params SoundParameterInternals[] soundParameterInternals)
public void PlayAtPosition(SoundEvent soundEvent, Transform owner, Transform position, SoundTag localSoundTag, params SoundParameterInternals[]
soundParameterInternals)
```

Parameters

soundEvent	The **SoundEvent** to play
owner	The owner **Transform**
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played

23.3 SoundManager.Instance Stop

```
// Stops the SoundEvent at the owner Transform
public void Stop(SoundEvent soundEvent, Transform owner, bool allowFadeOut = true)

// Stops the SoundEvent at the position Transform
public void StopAtPosition(SoundEvent soundEvent, Transform position, bool allowFadeOut =
true)

// Stops all the SoundEvents at the owner Transform
public void StopAllAtOwner(Transform owner, bool allowFadeOut = true)

// Stops the SoundEvent everywhere
public void StopEverywhere(SoundEvent soundEvent, bool allowFadeOut = true)

// Stops all SoundEvents
public void StopEverything(bool allowFadeOut = true)
```

Parameters

soundEvent	The **SoundEvent** to stop
owner	The owner **Transform**
position	The position **Transform**

allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

23.4 SoundManager.Instance Pause and Unpause

```
// Pauses/unpauses the SoundEvent with the owner Transform locally
public void Pause(SoundEvent soundEvent, Transform owner, bool forcePause = false)
public void Unpause(SoundEvent soundEvent, Transform owner)

// Pauses/unpauses all SoundEvents with the owner Transform locally
public void PauseAllAtOwner(Transform owner, bool forcePause = false)
public void UnpauseAllAtOwner(Transform owner)

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverywhere(SoundEvent soundEvent, bool forcePause = false)
public void UnpauseEverywhere(SoundEvent soundEvent)

// Pauses/unpauses the SoundEvent everywhere locally
public void PauseEverything(bool forcePause = false)
public void UnpauseEverything()
```

Parameters

soundEvent	The **SoundEvent** to pause/unpause
owner	The owner **Transform**
forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

23.5 SoundManager.Instance Get State, Length, Time and Contains Loop

```
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState GetSoundEventState(SoundEvent soundEvent, Transform owner)

// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float GetLastPlayedClipLength(SoundEvent soundEvent, Transform owner, bool pitchSpeed)
```

```
// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float GetLastPlayedClipTimeSeconds(SoundEvent soundEvent, Transform owner, bool pitchSpeed)

// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float GetLastPlayedClipTimeRatio(SoundEvent soundEvent, Transform owner)

// Returns the max length (in seconds) of the SoundEvent (calculated from the longest audioClip)
// Is scaled by the pitch of the SoundEvent and SoundContainer
// Does not take into account random, intensity or parameter pitch
// Returns 0 if the SoundEvent is null
public float GetMaxLength(SoundEvent soundEvent)

// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float GetTimePlayed(SoundEvent soundEvent, Transform owner)

// Returns if any SoundContainers in the SoundEvent is set to looping
// Returns false if the SoundEvent is null
public bool GetContainsLoop(SoundEvent soundEvent)

// Returns the Transform of the SoundManager
public Transform GetSoundManagerTransform()
```

Parameters

soundEvent	The **SoundEvent** to get length or time from
owner	The owner **Transform**
pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

23.6 SoundManager.Instance Get Spectrum Data

```
// Provides a block of spectrum data from AudioSources
public void GetSpectrumData(SoundEvent soundEvent, Transform owner, ref float[] samples, int channel, FFTWindow window, SpectrumDataFrom
spectrumDataFrom)
```

Parameters

soundEvent	The **SoundEvent** to get the spectrum data from
owner	The owner **Transform**
samples	The array to populate with audio samples. Its length must be a power of 2
channel	The channel to sample from
window	The **FFTWindow** type to use when sampling
spectrumDataFrom	Where to get the spectrum data from

23.7 SoundManager.Instance Get Last Played AudioSource

```
// Returns the last played AudioSource
// Note that the AudioSource might be stolen or reused for different Voices over time
public AudioSource GetLastPlayedAudioSource(SoundEvent soundEvent, Transform owner)
```

Parameters

soundEvent	The **SoundEvent** to get the **AudioSource** from
owner	The owner **Transform**

23.8 SoundManager.Instance Load or Unload Audio Data

```
// Loads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void LoadAudioData(SoundEvent soundEvent)

// Unloads the audio data of any AudioClips assigned to the SoundContainers of the SoundEvent
public void UnloadAudioData(SoundEvent soundEvent)
```

23.9 SoundManager.Instance UI

Useful if you want to play SoundEvents without passing an owner.

SoundManager.Instance UI Play

```
// Plays the SoundEvent with the UI Transform as owner
// Useful to play e.g. UI or other 2D sounds without having to pass a Transform
// To make the sound 2D you still need to disable distance and set spatial blend to 0 in the SoundContainer
public void UIPlay(SoundEvent soundEvent)
```

```
public void UIPlay(SoundEvent soundEvent, SoundTag localSoundTag)
public void UIPlay(SoundEvent soundEvent, params SoundParameterInternals[] soundParameterInternals)
public void UIPlay(SoundEvent soundEvent, SoundTag localSoundTag, params SoundParameterInternals[] soundParameterInternals)

// Plays the SoundEvent at the position with the UI Transform as owner (Useful for UnityEvents because it only has one parameter)
public void UIPlayAtPosition(Vector3 position)
public void UIPlayAtPosition(Transform position)
```

Parameters

soundEvent	The **SoundEvent** to play
localSoundTag	The **SoundTag** which will determine the Local **SoundTag** of the **SoundEvent**
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played
position	The **Transform** or **Vector3** where is should play at (**Transform** can follow position)

SoundManager.Instance UI Stop

```
// Stops the SoundEvent at the UI Transform
public void UIStop(SoundEvent soundEvent, bool allowFadeOut = true)

// Stops all SoundEvents at the UI Transform
public void UIStopAll(Transform owner, bool allowFadeOut = true)
```

Parameters

soundEvent	The **SoundEvent** to stop
allowFadeOut	If the **SoundEvent** should be allowed to fade out. Otherwise it is going to be stopped immediately

SoundManager.Instance UI Pause and Unpause

```
// Pauses/unpauses the SoundEvent at the UI Transform locally
public void UIPause(SoundEvent soundEvent, bool forcePause =
false)
public void UIUnpause(SoundEvent soundEvent)

// Pauses/unpauses all SoundEvents at the UI Transform locally
public void UIPauseAll(bool forcePause = false)
public void UIUnpauseAll()
```

Parameters

soundEvent	The **SoundEvent** to pause/unpause

forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

SoundManager.Instance UI Get State, Length and Time

```
// Uses the UI owner Transform
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState UIGetSoundEventState(SoundEvent soundEvent)

// Uses the UI owner Transform
// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float UIGetLastPlayedClipLength(SoundEvent soundEvent, bool pitchSpeed)

// Uses the UI owner Transform
// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float UIGetLastPlayedClipTimeSeconds(SoundEvent soundEvent, bool pitchSpeed)

// Uses the UI owner Transform
// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float UIGetLastPlayedClipTimeRatio(SoundEvent soundEvent)

// Uses the UI owner Transform
// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float UIGetTimePlayed(SoundEvent soundEvent)

// Returns the owner Transform used by UIPlay() etc
public Transform UIGetTransform()
```

Parameters

soundEvent	The **SoundEvent** to get length or time from

pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

23.10 SoundManager.Instance Music

Useful if you want to play music with automatic stopping of the old music when playing a new song.

SoundManager.Instance Music Play

```
// Plays the SoundEvent at the SoundManagers music Transform
public void MusicPlay(SoundEvent soundEvent, bool stopAllOtherMusic = true, bool allowFadeOut = true)
public void MusicPlay(SoundEvent soundEvent, bool stopAllOtherMusic = true, bool allowFadeOut = true, params SoundParameterInternals[] soundParameterInternals)
```

Parameters

soundEvent	The **SoundEvent** to play
stopAllOtherMusic	If all other **SoundEvents** played at the **SoundManager** music **Transform** should be stopped
allowFadeOut	If the other stopped **SoundEvent** should be allowed to fade out. Otherwise they are going to be stopped immediately
soundParameterInternals	For example **SoundParameterVolumeDecibel** is used to modify how the **SoundEvent** is played

SoundManager.Instance Music Stop

```
// Stops the SoundEvent playing at the Music Transform
public void MusicStop(SoundEvent soundEvent, bool allowFadeOut = true)

// Stops all the SoundEvents playing at the Music Transform
public void MusicStopAll(bool allowFadeOut = true)
```

Parameters

soundEvent	The **SoundEvent** to stop
allowFadeOut	If the other stopped **SoundEvent** should be allowed to fade out. Otherwise they are going to be stopped immediately

SoundManager.Instance Music Pause and Unpause

```
// Pauses/unpauses the SoundEvent at the Music Transform locally
public void MusicPause(SoundEvent soundEvent, bool forcePause = false)
public void MusicUnpause(SoundEvent soundEvent)

// Pauses/unpauses all SoundEvents at the Music Transform locally
public void MusicPauseAll(bool forcePause = false)
public void MusicUnpauseAll()
```

Parameters

soundEvent	The **SoundEvent** to pause/unpause
forcePause	If the **SoundEvent** should be paused even if it is set to "Ignore Local Pause"

SoundManager.Instance Music Get State, Length and Time

```
// Uses the Music owner Transform
// If playing it returns SoundEventState.Playing
// If paused either locally or globally it returns SoundEventState.Paused
// If not playing, but it is delayed it returns SoundEventState.Delayed
// If not playing and it is not delayed it returns SoundEventState.NotPlaying
// If the SoundEvent or Transform is null it returns SoundEventState.NotPlaying
public SoundEventState MusicGetSoundEventState(SoundEvent soundEvent)

// Uses the Music owner Transform
// Returns the length (in seconds) of the AudioClip in the last played AudioSource
// Returns Mathf.Infinity if the InstanceSoundEvent is not playing
// Returns Mathf.Infinity if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float MusicGetLastPlayedClipLength(SoundEvent soundEvent, bool pitchSpeed)

// Uses the Music owner Transform
// Returns the current time (in seconds) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
// pitchSpeed determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long
public float MusicGetLastPlayedClipTimeSeconds(SoundEvent soundEvent, bool pitchSpeed)

// Uses the Music owner Transform
// Returns the current time (in range 0 to 1) of the AudioClip in the last played AudioSource
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float MusicGetLastPlayedClipTimeRatio(SoundEvent soundEvent)

// Uses the Music owner Transform
// Returns the time (in seconds) since the SoundEvent was played
// Is calculated using the time scale selected in the SoundManager
// Returns 0 if the InstanceSoundEvent is not playing
// Returns 0 if the SoundEvent or Transform is null
public float MusicGetTimePlayed(SoundEvent soundEvent)

// Returns the owner Transform used by MusicPlay() etc
public Transform MusicGetTransform()
```

Parameters

soundEvent	The **SoundEvent** to get length or time from
pitchSpeed	Determines if it should be scaled by pitch. E.g. -12 semitones will be twice as long

23.11 SoundManager.Instance Global Pause and Unpause

```
// Pauses the all SoundEvents except those who are set to "Ignore Global Pause"
public void SetGlobalPause()
// Unpauses the all SoundEvents except those who are set to "Ignore Global Pause"
public void SetGlobalUnpause()

// Returns if Global Pause is enabled
public bool GetGlobalPaused()
```

23.12 SoundManager.Instance Global Volume

```
// Sets a global volume of all sounds in decibels using AudioListener.volume, ranging from NegativeInfinity dB to 0 dB
public void SetGlobalVolumeDecibel(float volumeDecibel)

// Returns the global volume in decibels, ranging from NegativeInfinity dB to 0 dB
public void SetGlobalVolumeRatio(float volumeRatio)

// Sets a global volume of all sounds in linear scale using AudioListener.volume, ranging from 0 to 1
public float GetGlobalVolumeDecibel()

// Returns the global volume in linear scale, ranging from 0 to 1
public float GetGlobalVolumeRatio()
```

23.13 SoundManager.Instance Global SoundTag

```
// Sets the global SoundTag
public void SetGlobalSoundTag(SoundTag soundTag)

// Returns the global SoundTag
public SoundTag GetGlobalSoundTag()
```

23.14 SoundManager.Instance Global Distance Scale

```
// Sets the global distance scale (default is a scale of 100 units)
public void SetGlobalDistanceScale(float distanceScale)

// Returns the global distance scale
public float GetGlobalDistanceScale()
```

23.15 SoundManager.Instance Speed Of Sound

```
// Set if speed of sound should be active
public void SetSpeedOfSoundEnabled(bool speedOfSoundEnabled)

// Set the speed of sound scale
// The default is a multiplier of 1 (by the base value of 340 unity units per second)
public void SetSpeedOfSoundScale(float speedOfSoundScale)

// Returns the speed of sound scale
public float GetSpeedOfSoundScale()
```

23.16 SoundManager.Instance Voice Limit

```
// Sets the Voice limit
public void SetVoiceLimit(int voiceLimit)

// Returns the Voice limit
public int GetVoiceLimit()

// Sets the VoiceEffect limit
public void SetVoiceEffectLimit(int voiceEffectLimit)

// Returns the VoiceEffect limit
public int GetVoiceEffectLimit()
```

23.17 SoundManager.Instance Disable Playing Sounds

```
// Disables/enables all the Play functionality
public void SetDisablePlayingSounds(bool disablePlayingSounds)
```

```
// If the Play/PlayAtPosition functionality is disabled
public bool GetDisablePlayingSounds()
```

23.18 SoundManager.Instance Get Addressable AudioMixer

```
// If you have addressable AudioMixer enabled in the SoundManager you need to use this AudioMixer instance for e.g. AudioMixer.SetFloat().
// This is to fix problems when the AudioMixer is an addressable asset because it might create both a normal and an addressable instance with different IDs.
// Note that when using an addressable AudioMixer you should also enable Async Startup to increase startup speed.
// For addressable AudioMixer to work, you need to add the package "com.unity.addressables".
// You also need to define the Script Define Symbol "SONITY_ENABLE_ADDRESSABLE_AUDIOMIXER".
// Asmdef_Sonity.Internal.Runtime references Unity.Addressables and Unity.ResourceManager.
// Asmdef_Sonity.Internal.Editor references Unity.Addressables.
public AudioMixer GetAddressableAudioMixer()
```