



v3.1b

ALMENARA GAMES

ABOUT

Multi Listener Pooling Audio System (MLPAS) is an **Audio System** that was made from the ground up to support multiples **Audio Listeners** and make the process of managing and playing all the sounds of your project a more simple task thanks to the different methods provided to play the sounds, the capacity to plays audio via a Pooling system and also the option to play sounds directly from the **Animator** Component.

This is a fully replacement of the default Unity Audio System and is not only focused for splitscreen games, it can be used for any type of game since it have plenty of tools to set and play your audios.

FEATURES

- **NEW!** Capacity to play sounds directly from the **Animator** Component using our custom **State Machine Behaviour**.
- Support multiple **Audio Listeners** with 3D audio positioning using only one voice per **Audio Source**.
- Automatic audio blending between **Audio Listeners** (*up to 4 listeners*).
- Pooling system using our *easy-to-use* scriptable object "**Audio Object**" to facilitate the task of creating references to **Audio Clips** and setting various configurations of Audio Sources such as: **Volume**, **Pitch**, **Random clips**, **Looping**, **Starting Pitch** and **Volume Alterations**, **Spatial mode**, **Mixer Output**, and other settings.
- Capacity to **Play**, **Stop**, **Pause/Unpause**, **Delay**, **Mute**, **Fade In/Out** the audios.
- Basic system to occlude the sounds through colliders using raycasting.
- Persistent sounds between scenes.
- **Reverb Zones** compatibles with multiple Audio Listeners. (*WebGL not supported*).
- Variety of methods for play the sounds according to your needs.

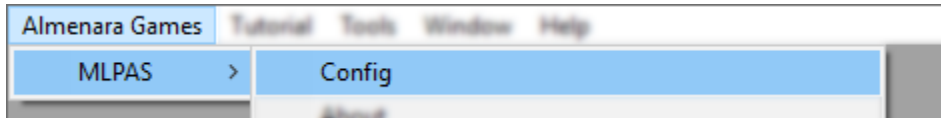
[Check Release Notes](#)

INDEX

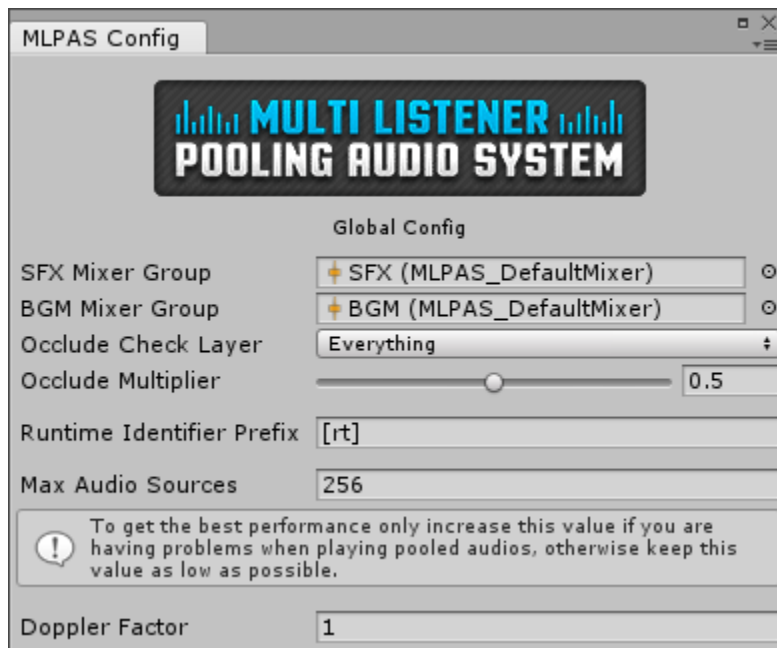
- [Setup](#)
- [Details](#)
- [Audio Object](#)
- [Multi Audio Source](#)
- [Pooled Audio Sources](#)
- [Multi Audio Listener](#)
- [Multi Reverb Zone](#)
- [Multi Audio Manager](#)
- [Identifiers](#)
- [Runtime Identifiers](#)
- [Playing Sounds from the Animator Component](#)
- [MLPAS Animator SFX](#)
- [MLPAS AnimatorSFX Controller](#)

SETUP

1. First open the **Multi Listener Pooling Audio System Config** inside the “Almenara Games” menu item:



2. Setup the **Multi Listener Pooling Audio System Config** the way you need it.



Sfx Mixer Group	Default SFX Mixer Group Output.
Bgm Mixer Group	Default BGM Mixer Group Output.
Occlude Check	Layer Mask used for check whether or not a collider occludes an occludable sound.
Occlude Multiplier	The higher the value, the less the audio is heard when occluded.
Runtime Identifier Prefix	Max pooled Multi Audio Sources.
Max Audio Sources	The prefix used to define Runtime Identifiers.
Doppler Factor	Set how audible the Doppler effect is.

3. You are ready to start using the **Multi Listener Pooling Audio System (MLPAS)**.

DETAILS

The **Multi Listener Pooling Audio System (MLPAS)** works using a custom **Audio Listener** labeled as “**Multi Audio Listener**” and a custom **Audio Source** labeled as “**Multi Audio Source**”, these components have some different options compared to the default ones, additionally the **MLPAS** makes use of a pooling system based on **AudioObjects** and **Channels** to ensure a proper and simple management of the different audios in your project.

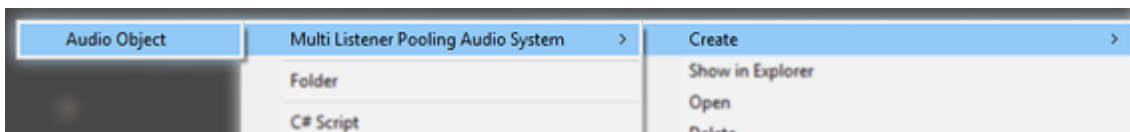
We recommend to see all the **API Details** in order to get the most out of this **Package** and get a better understanding of all the different methods and parameters presents in the diverse components.

Audio Object

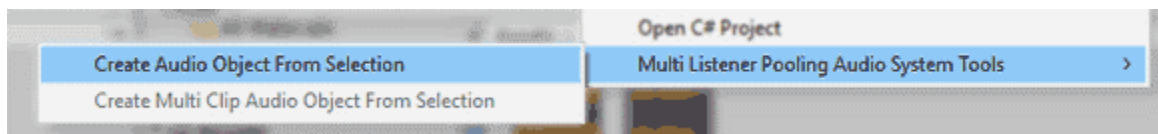
The **Audio Object** is a scriptable object that contains much of the settings of a default **Audio Source** with some other additional things like the capacity of assign multiple **Audio Clips** for play a random clip from the list each time is played or applying a customizable random pitch/volume alteration to the audio... The **Audio Object** is like the **Audio Clip** of the **Multi Audio Source** with the difference that all the settings of the source are inside the clip and not in the source itself, so you can have all the configuration of your sounds per “Audio Clip” instead of per Audio source.

NOTE: An **Audio Object** can be associated with an Identifier or a Runtime Identifier and be played via it identifier.

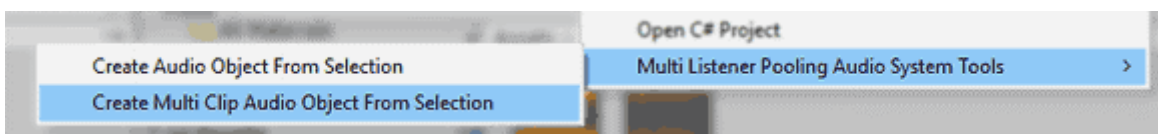
The **Audio Object** can be created from the “Asset” > “Create” menu.



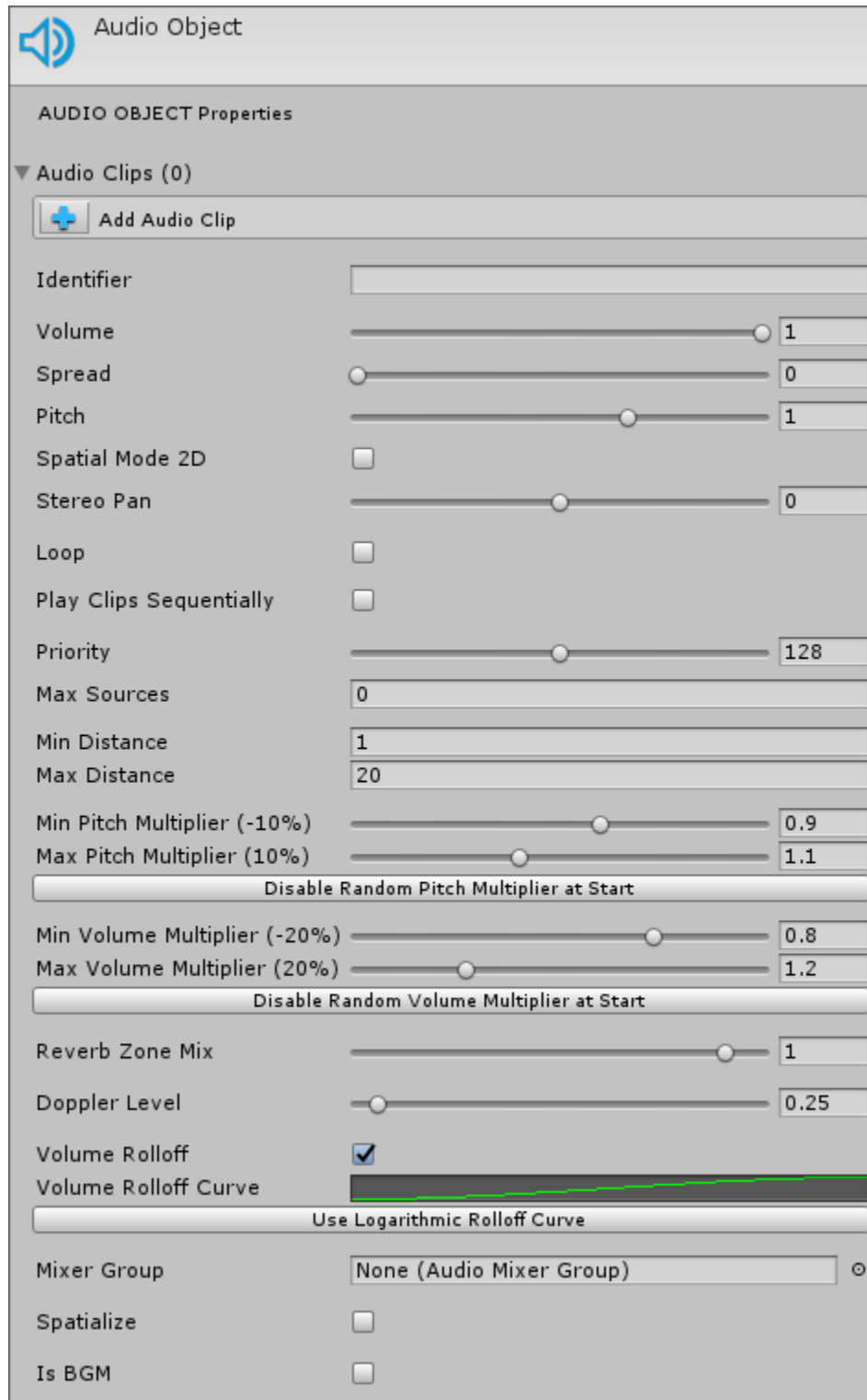
Additionally you can create an **Audio Object** from an **Audio Clip** by right clicking the **Audio Clip** and selecting the option:



Or create a “Multi Clip” **Audio Object** by selecting multiple **Audio Clips** then right clicking one of the **Audio Clips** and selecting the option:



This is how the **Audio Object** looks and all its parameters:



Audio Object

AUDIO OBJECT Properties

▼ Audio Clips (0)

+ Add Audio Clip

Identifier

Volume 1

Spread 0

Pitch 1

Spatial Mode 2D

Stereo Pan 0

Loop

Play Clips Sequentially

Priority 128

Max Sources 0

Min Distance 1

Max Distance 20

Min Pitch Multiplier (-10%) 0.9

Max Pitch Multiplier (10%) 1.1

Disable Random Pitch Multiplier at Start

Min Volume Multiplier (-20%) 0.8

Max Volume Multiplier (20%) 1.2

Disable Random Volume Multiplier at Start

Reverb Zone Mix 1

Doppler Level 0.25

Volume Rolloff ☒

Volume Rolloff Curve

Use Logarithmic Rolloff Curve

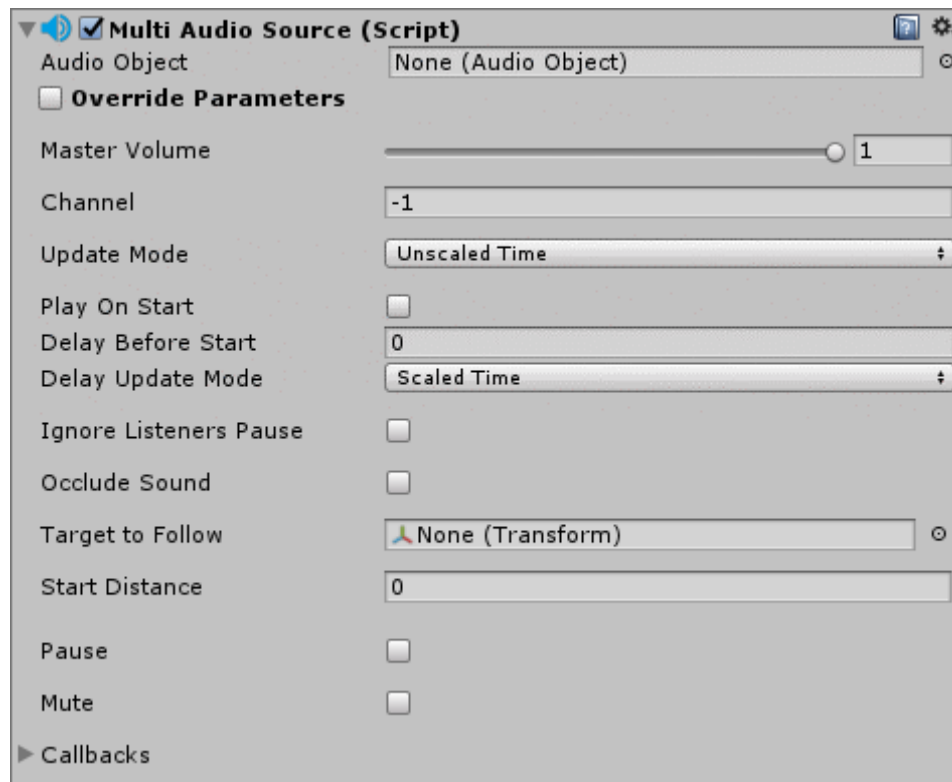
Mixer Group None (Audio Mixer Group)

Spatialize

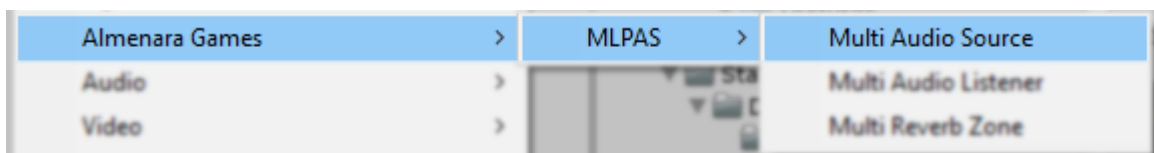
Is BGM

Multi Audio Source

The **Multi Audio Source** is a replace of the default **Audio Source**, unlike the default **Audio Source** the **Multi Audio Source** plays **Audio Objects** and only contains properties to override the assigned **Audio Object** properties and some other basic options such as mute or pause. It was created to be used mainly with the pooling system, however it can also be used individually.



A **Multi Audio Source** can be created from the “GameObject” > “Almenara Games” > “MLPAS” menu or you can add a **Multi Audio Source** component to an already created GameObject from the “Almenara Games” > “MLPAS” component menu.

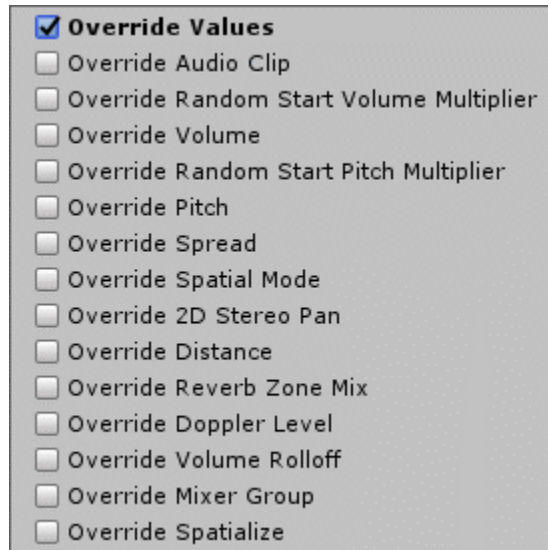


Now you only need to assign an **Audio Object** to the new **Multi Audio Source** and it will be ready to be played using the parameter “Play On Start” or using one of the Play methods of the **Multi Audio Source**.

Override Values

If you need to play an **Audio Object** and change some of its values in runtime or just want to play it a little bit different you can use the **Override Values**.

These are all the override values you can use in the **Multi Audio Source**:



All of the values can be set via inspector or via code.

Pooled Audio Sources

As explained before, the **Multi Audio Source** was created to be used mainly with the pooling system, the **Multi Audio Manager** contains a variety of methods to play the sounds according to your needs without the need to create a **Multi Audio Source** in advance everytime.

If we want more advanced ways to play our sounds we need to understand first what the **Channels** are and know the respective methods that the **Multi Audio Manager** contains to play our sounds.

Channels

The **channels** are used to filter and assign a unique ID to a **Multi Audio Source**, by this way you can later stop, pause or change some properties of an audio already played. A **channel** can play only one **Audio Source** at a time, this can be useful if you want to play the voice of a character because this will go to interrupt the last sentence with the new one played avoiding overlapping voices of the same character. All of the audios are played by default at **channel -1** which is equivalent to not use channels, to use the **channels** you need to play the audios in a **channel** greater than -1.

Override and Set Pooled Audio Source Values

You can override and set values on a **Pooled Audio Source** too using the properties present in the **Multi Audio Source** component.

Example #1

This example plays the **AudioObject** “audioToPlay” at the position “pos”.

```
using UnityEngine;
using AlmenaraGames;

public class TestClass : MonoBehaviour
{
    public AudioObject audioToPlay;
    public Vector3 pos;

    void Start ()
    {
        MultiAudioManager.PlayAudioObject(audioToPlay, pos);
    }
}
```

Example #2

This example plays and overrides/changes some of the values of the **Audio Object** of a **Pooled Audio Source**:

```
using UnityEngine;
using AlmenaraGames;

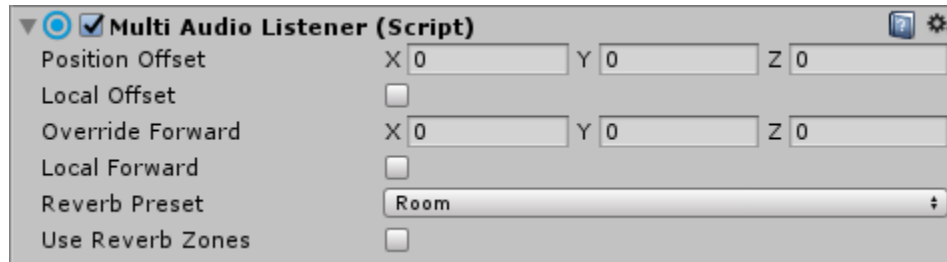
public class TestClass : MonoBehaviour
{
    public AudioObject audioToPlay;
    public Vector3 pos;

    void Start ()
    {
        MultiAudioSource source = MultiAudioManager.PlayAudioObject(audioToPlay, pos);
        source.VolumeOverride = 0.45f;
        source.PlayUpdateMode = MultiAudioManager.UpdateModes.ScaledTime;
    }
}
```

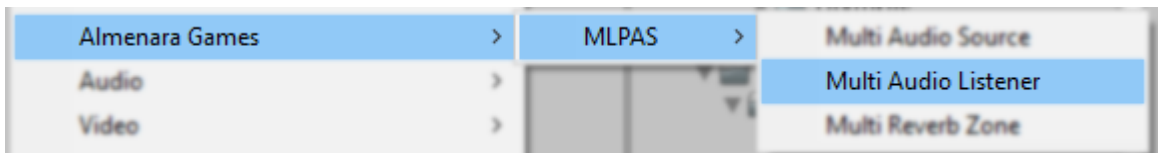
NOTE: The only parameter that can't be overridden in a **Pooled Audio Source** using its properties is: the Audio Clip, for override the Audio Clip you need to use any of the methods in the **Multi Audio Manager** labeled as “Override”.

Multi Audio Listener

The **Multi Audio Listener** is a replace of the default **Audio Listener** that has some options that makes it more customizable. As you can see below, each **Multi Audio Listener** has its own “Reverb Preset”, this replaces the default reverb preset of the listener, if you want to make for example a cave that have some reverb, you can change the “Reverb Preset” of an individual **Multi Audio Listener** when it enters inside a trigger or simply use a **Multi Reverb Zone**. (Due to certain limitations, custom reverb presets can’t be used.)



A **Multi Audio Listener** can be created from the “GameObject” > “Almenara Games” > “MLPAS” menu or you can add a **Multi Audio Listener** component to an already created GameObject from the “Almenara Games” > “MLPAS” component menu.



Unlike the default Unity Audio Listener, the **Multi Audio Listener** supports multiple instances of itself in the scene with automatic blending between **Listeners** (up to 4 listeners) giving you the tools for creating things like, for example, splitscreen games more easily.

Multi Audio Manager

The **Multi Audio Manager** contains a variety of methods for play and manage the sounds according to your needs and is the one that initializes the system using a singleton.

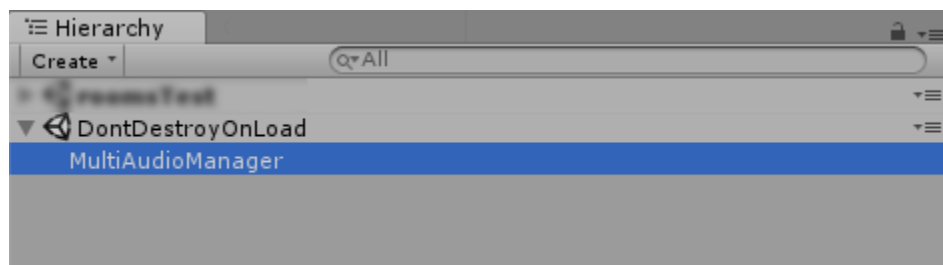
These are some of the methods you will find in the **Multi Audio Manager**:

PlayAudioObject	Plays an AudioObject .
StopAudioSource	Stops the specified MultiAudioSource .
PauseAudioSource	Pauses/Unpauses the specified MultiAudioSource .
FadeInAudioObject	Plays an AudioObject with a fade in.

This is how the **Multi Audio Manager** component looks:

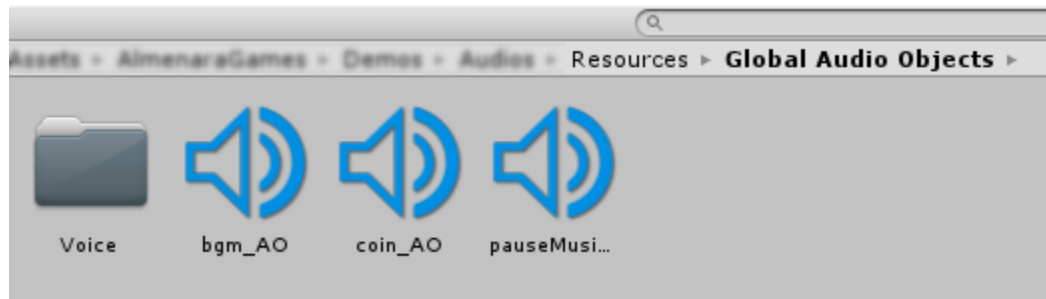


The **Multi Audio Manager** is created automatically by the system and can be found on the hierarchy tab inside the **DontDestroyOnLoad** scene tab.



Identifiers (Resources folder needed)

As mentioned before, an identifier can be applied to an **Audio Object** in order to find it and play it without the need of create a reference of the Audio Object, to assign an Identifier to an **Audio Object** you need to use the Resources folder and create a folder named: “Global Audio Objects” then move the **Audio Objects** you want to be finded by its identifiers.



Runtime Identifiers

A Runtime Identifier works similar to a regular identifier, both works to find and play an **Audio Object** with the difference that a Runtime Identifier don't need to use the Resources folder and needs to be defined via code and it **Audio Object** is assigned via code too. They was created to be used principally with the **MLPASAnimatorSfx** state machine behavior, however they can be handy for some other situations; for example: You can create a Sound Manager that defines and assigns different **Runtime identifiers** with sounds of your game and then you can play the sounds without the need of referencing that Sound Manager, using some method like “MultiAudioManager. **PlayAudioObjectByIdentidier**”.

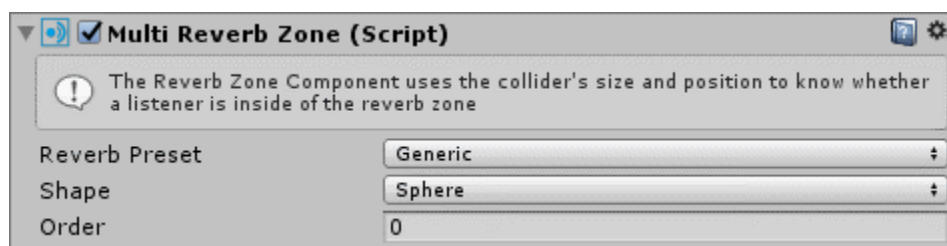
NOTE: If you are playing or finding an **Audio Object** via a **Runtime Identifier** you need to put the “Runtime Identifier Prefix” you has in your MLPAS Config (default: “[rt]”), example: if the runtime identifier is “sound” then is going to be “[rt]sound”.

Multi Reverb Zone

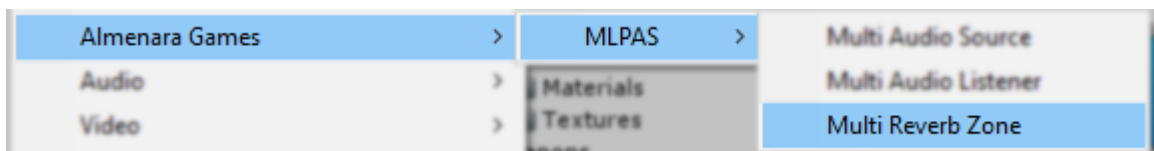
The **Multi Reverb Zone** is a replace of the default **Audio Reverb Zone** that allows Reverb Zones to be easily added in your scenes.

This component automatically adds a **Box Collider/Sphere Collider** to the game object to know whether a **Multi Audio Listener** is inside of the **Multi Reverb Zone**, additionally, the component can't be added to a game object that already contains a **Multi Reverb Zone** component or a **Collider** of any kind.

This is how the **Multi Reverb Zone** component looks:



A **Multi Reverb Zone** can be created from the "GameObject" > "Almenara Games" > "MLPAS" menu or you can add a **Multi Reverb Zone** component to an already created GameObject from the "Almenara Games" > "MLPAS" component menu.



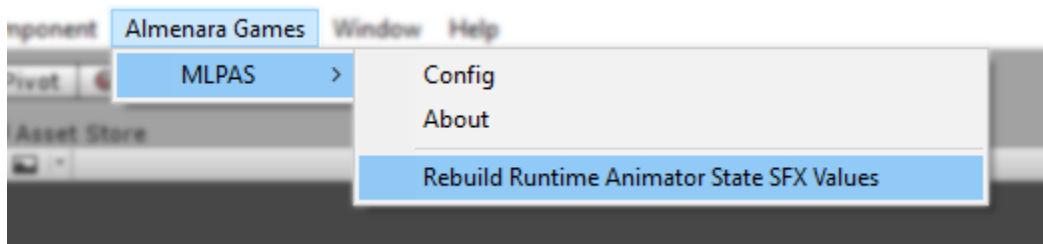
NOTE: The **Multi Reverb Zone** doesn't work on WebGL.

Playing Sounds from the Animator Component

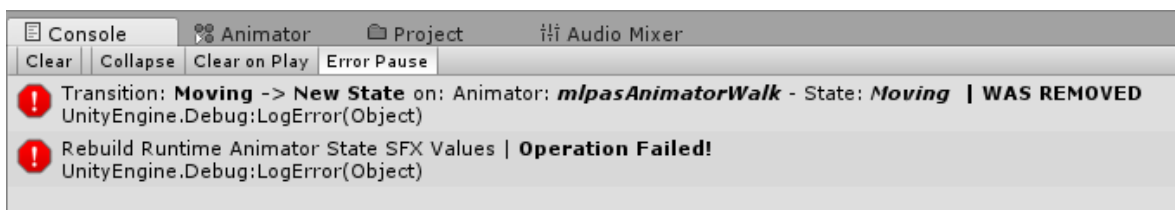
With this package you can play sounds directly from an Animator State or a State Machine using our State Machine Behaviour: **MLPAS Animator SFX** that can be further extended using the component: **MLPAS Animator SFX Controller**.

The way this Animator Sounds system works is by “baking” all the settings of the **MLPAS Animator SFX** State Machine Behaviour during editor time, this is done automatically when changing the Play Mode or when creating a Build, however, is still recommended to manually do the process when creating a build because if there is an error with any of your different **MLPAS Animator SFX** State Machine Behaviours the build process will go to continue ignoring the errors; if you are dealing with **Asset Bundles / Addressables** you may need to manually make the process too.

To manually bake the settings you need to use the “Almenara Games” > “MLPAS” > “Rebuild Runtime Animator State SFX Values” menu option.



This is an example of how an error is displayed when “Baking” the **MLPAS Animator SFX** values

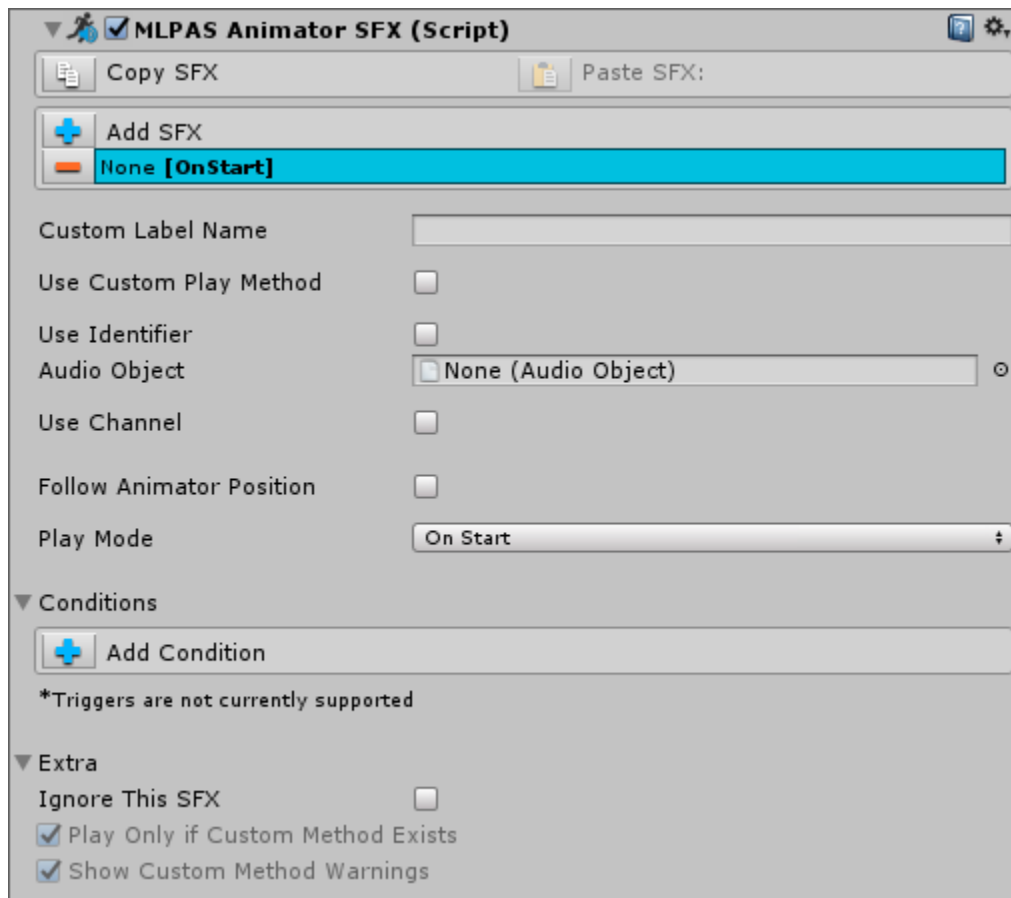


MLPAS Animator SFX

The **MLPAS Animator SFX** is a State Machine Behaviour that allows playing Audio Objects directly from an **Animator State** or **Animator State Machine**.

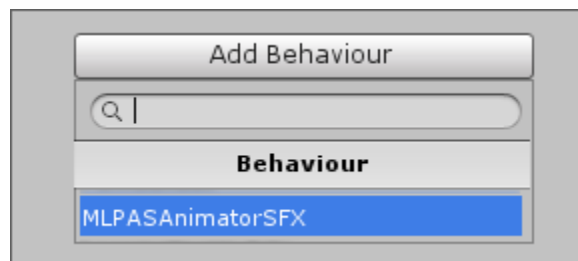
This State Machine Behaviour has a variety of options to ensure that your sounds will going to be played exactly at the moment you want to.

This is how the **MLPAS Animator SFX** state machine behaviour looks:



NOTE: When using a Custom Play Method remember that the method needs to be registered using a **MLPASAnimatorSFXController** next to the Animator Controller containing the State Machine Behaviour, the method can be registered via code or via inspector.

A **MLPAS Animator SFX** can be created from an Animator State/State Machine with the Add Behaviour button.



Custom Play Methods

This is how a Custom Play Method needs to be declared:

```
using UnityEngine;
using AlmenaraGames;

public class TestClass : MonoBehaviour
{
    public void CustomMethodName (MLPASCustomPlayMethodParameters parameters)
    {
        //Method Logic Example
        //Vector3 pos = parameters.Position + transform.position.y * Vector3.up;
        //MultiAudioManager.PlayAudioObject(parameters.AudioObject, pos);
    }
}
```

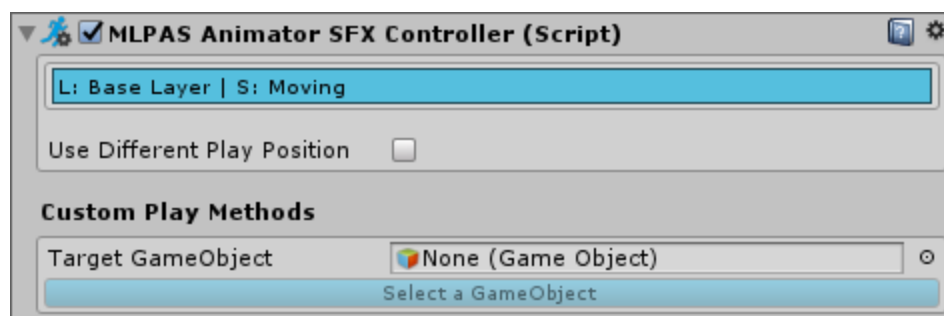
The **MLPASCustomPlayMethodParameters** struct contains information about the specific StateSFX in the Animator State Invoking the corresponding Custom Play Method

MLPAS Animator SFX Controller

The **MLPAS Animator SFX Controller** is a component that works like an extension to the previous mentioned **MLPAS Animator SFX** state machine behavior.

This component allows the assignment of some Scene references (Custom Play Methods or different play positions) that the **MLPAS Animator SFX** is not capable to do because of the nature of the State Machine Behaviours. This component is not always needed.

This is how the **MLPAS Animator SFX Controller** component looks:



The **MLPAS Animator SFX Controller** can be added from the component menu: “Almenara Games” > “MLPAS” > Animator SFX Controller.

Check the DEMO scenes for better understanding of the MLPAS.

V3.1b Changelog:

Changed:

- Now a Non-Pooled Multi Audio Sources can be "**Faded in**" via Inspector.

Fixed:

- Fixed a bug that prevents changing the settings of an Audio Object when setting the ***Time.timeScale*** to 0.
- Fixed a bug with the parameter "**Play On Start**" of Non-Pooled Multi Audio Sources that caused the audio to be played repeatedly.

V3.1 Changelog:

Added:

- Added parameter "**StartDistance**" to the **Multi Audio Source**.
- Added parameter "**DopplerFactor**" to the **MLPAS Config Window**.
- Added **UnityEvent** callback "**OnPlay**" to the **Multi Audio Source**.
- Added **UnityEvent** callback "**OnStop**" to the **Multi Audio Source**.
- Added **UnityEvent** callback "**OnLoop**" to the **Multi Audio Source**.
- Added **UnityEvent** callback "**OnRange**" to the **Multi Audio Source**.
- Added **UnityEvent** callback "**OnOutOfRange**" to the **Multi Audio Source**.

Changed:

- The spatialization of the sounds are now more accurated.
- The **Doppler Effect** now works with a custom implementation.
- The **Multi Audio Source** component has been further optimized.
- The **Multi Audio Source** component now can follows an inactive target.
- The property "**AveragePosition**" now returns the centroid of all listeners.
- Multi Audio Source warnings now points to the correct source.

Fixed:

- Fixed audio stuttering/clicking for sources too far from coords (0, 0, 0).
- Fixed Doppler Effect breaking randomly.

V3.0 Changelog:

Added:

- Added a new state machine behaviour: **MLPAS Animator SFX** that allows playing **Audio Objects** directly from an Animator State or Animator State Machine.
- Added a new component: **MLPAS Animator SFX Controller** that allows the assignment of some Scene references that the **MLPAS Animator SFX** is not capable to do.
- Added a public struct "**MLPASCustomPlayMethodParameters**" to the **AlmenaraGames** namespace to be used with the new **MLPAS Animator SFX** custom play methods.
- Added a public property "**PlaybackTime**" to the **Multi Audio Source**.
- Added **Runtime Identifiers**, now you can avoid the resources folder for the global audio objects and instead you can define runtime identifiers wherever you need them.
- Added a public property "**RuntimeIdentifiers**" to the **Multi Audio Manager**.
- Added a public property "**RuntimeIdentifiersCount**" to the **Multi Audio Manager**.
- Added method **MultiAudioManager.DefineRuntimeIdentifier**.
- Added method **MultiAudioManager.RemoveRuntimeIdentifier**.
- Added method **MultiAudioManager.AssignRuntimeIdentifierAudioObject**.
- Added method **MultiAudioManager.ClearAllRuntimeIdentifiers**.
- Added method **MultiAudioManager.IsRuntimeIdentifierDefined**.
- Added method **MultiAudioSource.IsValueOverridden**.

Changed:

- Now the method **MultiAudioManager.GetAudioObjectByIdentifier** can be implemented with runtime identifiers too.
- Now the method **MultiAudioManager.GetNearestListenerBlendAt** supports a custom minDistance and maxDistance.
- Now the method **MultiAudioManager.GetNearestListenerBlendAt** can be implemented using the reference of a **Multi Audio Source**.
- **Audio Objects** property **loopClipsSequentially** has been renamed to **playClipsSequentially**.
- Now Non-Looped **Audio Objects** can play clips sequentially.
- Now the Channel of Non-Pooled **Multi Audio Sources** can be changed via Inspector.
- Config file is now an asset instead of a Prefab.
- DEMO scenes updated according to the new changes.
- DOCS has been redone.
- Gizmos Icon has been updated.

Fixed:

- Now the properties Master Volume, Pause, Ignore Listener Pause, Update Mode, Delay Update Mode and Occlude Sound can be overridden correctly in **Pooled Audio Sources**.
- Fixed minor bugs and an issue with looped **Audio Objects** that cause some of the parameters of a **Multi Audio Source** to reset when looping.